ตัวสร้างสตับและไดร์เวอร์สำหรับการทดสอบรวมของคลาสจากแผนภาพลำดับและแผนภาพคลาส

นายพีรวุฒิ เหลืองเรืองโรจน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

Stubs and Drivers Generator for Class Integration Testing from Sequence and Class Diagrams

Mr. Peerawut Luengruengroj

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2019

หัวข้อวิทยานิพนธ์	ตัวสร้างสตับและไดร์เวอร์สำหรับการทดสอบรวมของคลาสจากแผนภาพลำดับและแผนภาพคลาส
โดย	นายพีรวุฒิ เหลืองเรืองโรจน์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการ
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์)

พีรวุฒิ เหลืองเรืองโรจน์ : ตัวสร้างสตับและไดร์เวอร์สำหรับการทดสอบรวมของคลาสจากแผนภาพลำดับและแผนภาพคลาส. (Stubs and Drivers Generator for Class Integration Testing from Sequence and Class Diagrams) อ.ที่ปรึกษาหลัก : รศ. ดร.ธาราทิพย์ สุวรรณศาสตร์

ในการพัฒนาซอฟต์แวร์ นักเขียนโปรแกรมแต่ละคนอาจจะเขียนโปรแกรมแต่ละส่วนเสร็จไม่พร้อมกัน หากต้องรอนักเขียนโปรแกรมทุกคนเขียนโปรแกรมเสร็จสิ้นจึงเริ่มการทดสอบจะเป็นการเสียเวลา การทดสอบซอฟต์แวร์จึงเริ่มทันทีเมื่อรหัสต้นฉบับบางส่วนพัฒนาเสร็จสิ้น ซึ่งสตับและไดร์เวอร์จะถูกนำมาใช้แทนมอดูลที่ยังพัฒนาไม่เสร็จ อย่างไรก็ตามสตับและไดร์เวอร์เป็นรหัสต้นฉบับเสียเปล่าที่ถูกสร้างเพื่อใช้เพียงครั้งเดียวและไม่สามารถนำมาใช้กับโครงการอื่น ๆ ได้ การสร้างสตับและไดร์เวอร์จึงควรใช้ความพยายามในการพัฒนาให้น้อยที่สุด

งานวิจัยก่อนหน้าได้นำเสนอตัวสร้างสตับและไดร์เวอร์โดยใช้ข้อมูลจากแผนภาพลำดับและแผนภาพคลาส ซึ่งมีข้อจำกัดด้านการสร้างสตับและไดร์เวอร์สำหรับคลาสนามธรรม คลาสภายใน และอินเตอร์เฟส รวมทั้งสามารถเลือกคลาสภายใต้การทดสอบได้เพียงคลาสเดียวและไม่สามารถสร้างกรณีทดสอบได้ โดยงานวิจัยนี้จะพัฒนาตัวสร้าง สตับและไดร์เวอร์ที่แก้ไขข้อจำกัดของงานวิจัยดังกล่าว ผู้ทดสอบสามารถสร้างสตับและไดร์เวอร์สำหรับการทดสอบรวมของคลาสโดยนำเข้าแผนภาพลำดับและแผนภาพคลาสในรูปเอกซ์เอ็มแอล จากนั้นตัวสร้างจะประมวลผลแผนภาพและสร้างกราฟการเรียกใช้งานขึ้นมา ผู้ทดสอบสามารถเลือกกคลาสภายใต้การทดสอบได้ตั้งแต่หนึ่งคลาสขึ้นไป และตัวสร้างจะรวบรวมสตับและไดร์เวอร์ที่ต้องใช้ทดสอบคลาสที่เลือกและสร้างรหัสต้นฉบับของสตับและไดร์เวอร์ขึ้นมาเพื่อทดสอบคลาสเหล่านั้น นอกจากนี้ตัวสร้างจะสุ่มค่าของข้อมูลทดสอบได้แก่ข้อมูลนำเข้าและผลลัพธ์ที่คาดหวังลงไปในไดร์เวอร์ จากนั้นตัวสร้างจะส่งออกไฟล์รหัสต้นฉบับเพื่อการทดสอบรวมคลาสดังกล่าว ซึ่งสตับและไดร์เวอร์ที่สร้างขึ้นมาจะนำไปทดสอบรวมกับกรณีศึกษา 3 กรณี ซึ่งพบว่าสตับและไดร์เวอร์ที่ถูกสร้างขึ้นสามารถแทนที่คลาสที่ยังพัฒนาไม่เสร็จได้

สาขาวิชา	วิศวกรรมซอฟต์แวร์	ลายมือชื่อนิสิต .....
ปีการศึกษา	2562	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6170947621 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Software Testing, Object-Oriented Programming, Sequence Diagram, Class Diagram, Stub, Driver, Test Case, Class Integration Testing

Peerawut Luengruengroj : Stubs and Drivers Generator for Class Integration Testing from Sequence and Class Diagrams. Advisor: Assoc. Prof. TARATIP SUWANNASART, Ph.D.

During software development, each programmer may not finish coding simultaneously. Meanwhile, waiting until all programmers complete their work is a wasting of time, so the testing process should be started right after some of the source code is completed. Stubs and drivers are used for replacing unfinished modules. Since they are throwaway code—in other words, they are non-reusable code, it takes a significant effort to produce.

In our previous research, the stub-and-driver generators using sequence and class diagrams was proposed; however, there are some limitations regarding generating stubs and drivers for abstract classes, inner classes, and interfaces. Testers are also allowed to select just a single class under test at a time, and the generator is only capable of generating stubs and drivers. Another point is that test input values are not generated. Therefore, this research aims to solve those limitations. To generate stubs-and-drivers source code, first, the sequence and class diagrams are imported in XML format to create a call graph. Next, testers select a group of classes under test. Then, all required stubs and drivers are identified and generated from those selected classes. Test input values and expected results for the drivers are randomly generated. Finally, these stubs and drivers are exported as source code files and sent to perform class integration testing with three case studies. The result shows that these stubs can replace those unfinished classes.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2019 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาช่วยเหลือ แนะนำ และให้คำปรึกษาในการทำวิจัย ตลอดจนตรวจทานและปรับปรุงแก้ไขข้อบกพร่องต่าง ๆ ด้วยความเอาใจใส่อย่างดียิ่งจาก รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ ประธานกรรมการสอบ รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ และผู้ช่วยศาสตราจารย์ ดร.ภัทรชัย ลลิตโรจน์วงศ์ กรรมการสอบวิทยานิพนธ์ ที่ได้เสียสละเวลาและให้คำแนะนำเกี่ยวกับงานวิจัย เพื่อให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์และครบถ้วนยิ่งขึ้น

ขอขอบคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยที่ได้มอบวิชาความรู้ทางวิชาการในด้านต่าง ๆ อีกทั้งบุคลากรในภาควิชาทุกท่านที่ช่วยประสานงาน และให้ความช่วยเหลือระหว่างที่ผู้วิจัยกำลังศึกษาตลอดจนสอบวิทยานิพนธ์สำเร็จลุล่วง

ขอขอบคุณรุ่นพี่ และเพื่อน ๆ นิสิตร่วมรุ่นในหลักสูตรวิศวกรรมซอฟต์แวร์ ที่ให้คำแนะนำช่วยเหลือ และเป็นกำลังใจให้เสมอมา

สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา และญาติพี่น้องของผู้วิจัย ที่ให้การสนับสนุนและเป็นกำลังใจให้ผู้วิจัยเสมอมาตลอดจนจบการศึกษา

พีรวุฒิ เหลืองเรืองโรจน์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การทดสอบซอฟต์แวร์เป็นขั้นตอนหนึ่งในการพัฒนาซอฟต์แวร์ โดยมีวัตถุประสงค์เพื่อค้นหาข้อผิดพลาดของซอฟต์แวร์ที่ยังคงอยู่ [1] และเพื่อประกันคุณภาพของซอฟต์แวร์ให้เป็นที่ยอมรับได้ การทดสอบซอฟต์แวร์จึงเป็นขั้นตอนหนึ่งที่สำคัญในกระบวนการพัฒนาซอฟต์แวร์

ในกระบวนการพัฒนาซอฟต์แวร์ด้วยวีโมเดล (V-Model) [2] มีขั้นตอนการออกแบบเชิงละเอียด (Detailed Design) ซึ่งเป็นขั้นตอนที่ออกแบบมอดูล (Module) ในเชิงลึกเพื่อนำไปสู่การพัฒนาเป็นรหัสต้นฉบับ (Source Code) ในขั้นตอนนี้จะมีการใช้แผนภาพยูเอ็มแอล (UML – Unified Modelling Language) จำลองการออกแบบเพื่อให้เห็นรายละเอียดของซอฟต์แวร์มากขึ้น โดยใช้แผนภาพคลาส (Class Diagram) แสดงโครงสร้างของแต่ละคลาส (Class) และความสัมพันธ์ระหว่างคลาส นอกจากนี้ยังมีการใช้แผนภาพลำดับ (Sequence Diagram) ในการแสดงการเรียกใช้งานเมท็อด (Method) ระหว่างคลาสเชิงลำดับของการเรียกใช้งาน ในขั้นตอนนี้ยังมีกระบวนการที่ทำควบคู่กันคือขั้นตอนการเตรียมการทดสอบส่วนย่อย (Unit Test) เพื่อทดสอบรหัสต้นฉบับหลังจากการพัฒนาเสร็จสิ้น

สำหรับการทดสอบส่วนย่อยของซอฟต์แวร์ที่พัฒนาด้วยการพัฒนาซอฟต์แวร์เชิงวัตถุนั้น การทดสอบจะเป็นการทดลองเรียกใช้เมท็อดของแต่ละคลาส เพื่อให้เมท็อดเหล่านั้นทำงานและตรวจสอบผลลัพธ์ที่ได้จากการทำงาน (Actual Result) ว่าตรงกับผลลัพธ์ที่คาดหวัง (Expected Result) หรือไม่ [3] ซึ่งในความเป็นจริงนั้น ซอฟต์แวร์ที่อยู่ภายใต้การทดสอบ (Software under test) อาจจะประกอบด้วยคลาสจำนวนมาก และแต่ละเมท็อดอาจมีการเรียกใช้เมท็อดอื่น ๆ ในคลาสเดียวกันหรือคลาสอื่น ๆ ด้วย ในการพัฒนาซอฟต์แวร์บ่อยครั้งนักเขียนโปรแกรม (Programmer) เข้ามาร่วมทีมกันต้องเข้ามาร่วมมือกัน แต่ในระหว่างการพัฒนา นักเขียนโปรแกรมแต่ละคนอาจจะเขียนโปรแกรมแต่ละส่วนเสร็จไม่พร้อมกัน หากต้องรอนักเขียนโปรแกรมทุกคนเขียนโปรแกรมเสร็จสิ้น จึงเริ่มการทดสอบส่วนย่อยจะเป็นการเสียเวลา การทดสอบจึงเริ่มทันทีเมื่อรหัสต้นฉบับบางส่วนพัฒนาเสร็จสิ้น หากคลาสภายใต้การทดสอบ (Class under test) มีการเรียกใช้งานเมท็อดของคลาสอื่นแต่คลาสเหล่านั้นยังพัฒนาไม่เสร็จจะมีการสร้างรหัสต้นฉบับของคลาสเหล่านั้นที่มีโครงสร้างเหมือนกันแต่ไม่มีฟังก์ชันการทำงานใด ๆ มีเพียงการรับค่าและส่งค่ากลับเท่านั้น คลาสเหล่านั้นเรียกว่าสตับ (Stub) [1] และในขณะเดียวกันหากเมท็อดในคลาสภายใต้การทดสอบถูกเรียกใช้โดยคลาสอื่นและคลาสเหล่านั้นยังถูกพัฒนาไม่เสร็จสิ้นจะมีการสร้างรหัสต้นฉบับของคลาสเหล่านั้นเพื่อมาเรียกใช้เมท็อดของคลาสภายใต้การทดสอบ คลาสเหล่านั้นเรียกว่าไดร์เวอร์ (Driver) [1] อย่างไรก็ตามรหัส

ต้นฉบับของคลาสเหล่านี้ขึ้นมาเพื่อการทดสอบเป็นรหัสต้นฉบับเสียเปล่า (Throwaway Code) [1] ซึ่งต้องใช้ความพยายาม (Effort) ในการสร้างขึ้นมาแต่ถูกใช้เพียงครั้งเดียวและไม่สามารถนำไปใช้ในโครงการอื่น ๆ ได้ การพัฒนารหัสต้นฉบับดังกล่าวจึงควรใช้ความพยายามในการพัฒนาให้น้อยที่สุด

จากงานวิจัย Stub and Driver Generator for Object-Oriented Software Testing Using Sequence and Class Diagrams [4] ผู้วิจัยได้เสนอตัวสร้างสตับและไดร์เวอร์ (Stub and Driver Generator) สำหรับการทดสอบโปรแกรมที่เขียนจากภาษาเชิงวัตถุจากแผนภาพคลาสและแผนภาพลำดับ โดยตัวสร้าง (Generator) จะสร้างรหัสต้นฉบับของสตับและไดร์เวอร์สำหรับทดสอบคลาสภายใต้การทดสอบที่ผู้ทดสอบเลือก โดยวิเคราะห์หาเมท็อดที่ถูกคลาสภายใต้การทดสอบเรียกใช้และเมท็อดที่เรียกใช้เมท็อดภายในคลาสภายใต้การทดสอบจากกราฟการเรียกใช้งาน และตัวสร้างจะสร้างรหัสต้นฉบับโดยอ้างอิงจากเมท็อดซิกเนเจอร์ (Method signature) ของเมท็อดเหล่านั้นในแผนภาพคลาส อย่างไรก็ตามตัวสร้างที่งานวิจัยดังกล่าวนำเสนอมีข้อจำกัดในการสร้างสตับและไดร์-เวอร์สำหรับคลาสนามธรรม (Abstract class) คลาสภายใน (Inner class) และอินเตอร์เฟส (Interface) นอกจากนี้ตัวสร้างดังกล่าวรองรับการสร้างสตับและไดร์เวอร์สำหรับทดสอบคลาสภายใต้การทดสอบเพียงคลาสเดียวเท่านั้นและตัวสร้างดังกล่าวเป็นเพียงตัวสร้างสตับและไดร์เวอร์เท่านั้น ไม่สามารถสร้างกรณีทดสอบได้

จากการศึกษางานวิจัยที่เกี่ยวข้อง [5-13] พบว่าสามารถสร้างสตับและไดร์เวอร์จากคลาสนามธรรม คลาสภายใน และอินเตอร์เฟสได้โดยไม่ต้องพึ่งพารหัสต้นฉบับ นอกจากนี้ยังสามารถสร้างกรณีทดสอบเพื่อใช้ทดสอบคลาสดังกล่าวได้จากข้อมูลที่มีอย่างจำกัดจากแผนภาพคลาสและแผนภาพลำดับ

งานวิจัยนี้มีจุดประสงค์เพื่อพัฒนาตัวสร้างสตับและไดร์เวอร์ที่มีการแก้ไขข้อจำกัดของงานวิจัยก่อนหน้า [4] โดยพัฒนาตัวสร้างให้รองรับการทดสอบคลาสภายใต้การทดสอบที่เป็นคลาสนามธรรม คลาสภายใน และอินเตอร์เฟส รวมถึงพัฒนาตัวสร้างให้รองรับการทดสอบคลาสภายใต้การทดสอบมากกว่าหนึ่งคลาสและสามารถสร้างข้อมูลทดสอบเพื่อใช้ทดสอบคลาสภายใต้การทดสอบดังกล่าวได้

1.2 วัตถุประสงค์ประสงค์ของงานวิจัย

1. เพื่อนำเสนอตัวสร้างสตับ ไดร์เวอร์ และกรณีทดสอบจากแผนภาพลำดับและแผนภาพคลาส

2. เพื่อพัฒนาตัวสร้างสตับ ไดร์เวอร์ และกรณีทดสอบจากแผนภาพลำดับและแผนภาพคลาส

1.3 ขอบเขตการดำเนินงาน

1. แผนภาพลำดับและแผนภาพคลาสที่เป็นข้อมูลนำเข้าจะอยู่ในรูปของไฟล์เอกซ์เอ็มแอลที่สร้างจากโปรแกรม Visual Paradigm โดยแผนภาพลำดับและแผนภาพคลาสต้องมีคุณสมบัติดังต่อไปนี้

1.1 แผนภาพลำดับและแผนภาพคลาสต้องออกแบบถูกต้องตามหลักการออกแบบเชิงวัตถุ [2]

1.2 แผนภาพลำดับและแผนภาพคลาสมีความสอดคล้องกัน

1.3 คลาสในแผนภาพลำดับต้องเป็นหนึ่งเดียว (Unique) ภายในแผนภาพคลาส

1.4 เมสเสจตอบกลับภายในแผนภาพลำดับต้องมีชื่อไม่ซ้ำกัน

1.5 การ์ดคอนดิชันต้องมีเป็นการเปรียบเทียบค่าระหว่างตัวแปรกับค่าคงที่เท่านั้น เช่น n > 0 โดยตัวแปรดังกล่าวต้องเป็นเมสเสจส่งกลับที่ปรากฏในแผนภาพลำดับนั้น

1.6 การ์ดคอนดิชันต้องไม่เป็นเงื่อนไขประกอบ (Composite condition) เช่น n>0 && n<100

2. ตัวสร้างมีความสามารถดังต่อไปนี้

2.1 ผู้ทดสอบสามารถนำเข้าไฟล์เอกซ์เอ็มแอลของแผนภาพลำดับและแผนภาพคลาสได้ทางส่วนต่อประสานผู้ใช้ (User Interface)

2.2 ผู้ทดสอบสามารถเลือกแผนภาพลำดับที่ต้องการทดสอบ แผนภาพคลาสที่เกี่ยวข้อง และคลาสภายใต้การทดสอบภายในแผนภาพลำดับที่เลือกได้ผ่านส่วนต่อประสานผู้ใช้

2.3 ตัวสร้างสามารถแสดงรหัสต้นฉบับที่สร้างผ่านส่วนต่อประสานผู้ใช้

2.4 ผู้ทดสอบสามารถแก้ไขรหัสต้นฉบับผ่านส่วนต่อประสานผู้ใช้ได้

3. ตัวสร้างสามารถสร้างสตับและไดร์เวอร์โดยมีเงื่อนไขดังต่อไปนี้

3.1 ผู้ทดสอบสามารถเลือกคลาสภายใต้การทดสอบได้ตั้งแต่ 1 คลาสขึ้นไปโดยต้องเลือกภายใต้แผนภาพลำดับเดียวเท่านั้น

3.2 ในกรณีที่ผู้ทดสอบเลือกคลาสภายใต้การทดสอบที่มีการอ้างอิงถึงแผนภาพลำดับอื่น ผู้ทดสอบต้องนำเข้าแผนภาพลำดับดังกล่าวสู่ตัวสร้างด้วย

3.3 การสร้างสตับและไดร์เวอร์จะพิจารณาจากแผนภาพลำดับภายใต้การทดสอบเท่านั้น โดยไม่พิจารณาถึงแผนภาพลำดับอื่น

3.4 ตัวสร้างจะรองรับการสร้างรหัสต้นฉบับสตับและไดร์เวอร์เป็นภาษาจาวา

4. งานวิจัยประเมินผลโดยการทดสอบสตับและไดร์เวอร์กับระบบอย่างน้อยสามระบบเพื่อทดสอบว่าสตับและไดร์เวอร์ที่สร้างจากตัวสร้างสามารถทำงานแทนคลาสที่ยังพัฒนาไม่เสร็จได้

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษางานวิจัยต้นแบบ [4] เพื่อวิเคราะห์หาความสามารถ ข้อจำกัดและจุดที่ต้องปรับปรุงของเครื่องมือที่พัฒนาในงานวิจัยดังกล่าว

2. ศึกษาแนวทางในการทดสอบคลาสนามธรรม คลาสภายใน และอินเตอร์เฟส

3. ศึกษาแนวทางในการสร้างกรณีทดสอบจากแผนภาพลำดับ

4. กำหนดแนวคิดและขอบเขตของงานวิจัย

5. วิเคราะห์และออกแบบฟังก์ชันการใช้งานเพื่อปรับปรุงตัวสร้าง รวมถึงส่วนต่อประสานผู้ใช้ที่ต้องปรับปรุงหรือสร้างเพิ่มเติม

6. พัฒนาตัวสร้างต่อจากเวอร์ชันก่อนหน้าโดยการเพิ่มความสามารถและฟังก์ชันการใช้งานตามที่ออกแบบไว้

7. ทดสอบตัวสร้างที่ถูกพัฒนาเพิ่ม ทั้งการทดสอบฟังก์ชันที่สร้างใหม่ และการทำทดสอบแบบถดถอย (Regression Test) และแก้ไขข้อผิดพลาดที่พบระหว่างการทดสอบ

8. สรุปและประเมินผลการวิจัยและข้อเสนอแนะ

9. จัดทำรูปเล่มวิทยานิพนธ์และบทความทางวิชาการ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาสที่รองรับการทดสอบคลาสภายใต้การทดสอบที่มากกว่าหนึ่งคลาส รวมถึงคลาสที่เป็นคลาสนามธรรม คลาสภายใน และอินเตอร์เฟส

2. แนวทางการสร้างกรณีทดสอบโดยใช้ข้อมูลจากแผนภาพลำดับและแผนภาพคลาสเท่านั้น

3. แนวทางการสร้างกรณีทดสอบจากการ์ดคอนดิชันภายในแผนภาพลำดับ

1.6 บทความทางวิชาการที่ได้รับการตีพิมพ์

งานวิจัยนี้ได้รับคัดเลือกให้ตีพิมพ์เป็นบทความทางวิชาการเรื่อง “Stubs and Drivers Generator for Class Integration Testing using Sequence and Class Diagrams” โ ด ย พีรวุฒิ เหลืองเรืองโรจน์ และธาราทิพย์ สุวรรณศาสตร์ ในการประชุมวิชาการ “2019 The 3rd International Conference on Software and e-Business (ICSEB 2019)” ระหว่างวันที่ 9 -11 ธันวาคม 2562 ณ มหาวิทยาลัยวาเซดะ วิทยาเขตนิชิ-วาเซดะ เมืองโตเกียว ประเทศญี่ปุ่น

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

เพื่อให้ทราบถึงแนวทางการดำเนินงานของวิทยานิพนธ์ที่สามารถทำได้ จำเป็นต้องศึกษาวิธีการ หรือทฤษฎีที่เกี่ยวข้องกับโครงการ ดังต่อไปนี้

2.1.1 แผนภาพคลาส

แผนภาพคลาสเป็นแผนภาพยูเอ็มแอลที่ใช้ในการแสดงโครงสร้างของซอฟต์แวร์แบบเชิงวัตถุในรูปแบบของความสัมพันธ์ระหว่างคลาส [2] โดยระบุถึงคุณสมบัติ (Attribute) และพฤติกรรมหรือเมท็อดของคลาส รวมถึงความสัมพันธ์ระหว่างคลาสด้วย

2.1.2 แผนภาพลำดับ

แผนภาพลำดับเป็นแผนภาพยูเอ็มแอลที่ใช้ในการอธิบายปฏิสัมพันธ์สัมพันธ์ของอ็อบเจกต์ (Object) หนึ่งกับอ็อบเจกต์อื่น ๆ เป็นลำดับขั้นของการดำเนินการ [2] โดยแผนภาพลำดับหนึ่งแผนภาพจะใช้อธิบายกิจกรรมหนึ่งของการดำเนินการในซอฟต์แวร์

2.1.3 เอกซ์เอ็มแอล (XML – Extensible Markup Language)

เอกซ์เอ็มแอล [14] เป็นภาษาที่ใช้แสดงข้อมูลและข้อมูลรูปแบบเข้าด้วยกันในรูปแบบของตัวอักษร พัฒนาโดย W3C (World Wide Web Consortium) ซึ่งเป็นองค์กรระหว่างประเทศที่คอยควบคุมและจัดระบบมาตรฐานที่ใช้งานบนเวิลด์ไวด์เว็บ (WWW หรือ W3) โดยภาษานี้ออกแบบให้มีรูปแบบที่สามารถอ่านด้วยมนุษย์ และคอมพิวเตอร์อ่านเข้าใจ

2.1.4 กราฟการเรียกใช้งาน (Call Graph)

กราฟการเรียกใช้งาน [15] เป็นกราฟควบคุมการไหล (Control Flow Graph) แบบหนึ่งที่ใช้แสดงความสัมพันธ์ระหว่างกระบวนการย่อย (Subroutine) ภายในโปรแกรม โดยแต่ละโหนด (Node) ของกราฟจะแสดงถึงกระบวนการทำงาน และเส้นเชื่อมแต่ละเส้น (Edge) จะแสดงถึงการเรียกใช้งานจากจุดหนึ่งไปถึงจุดหนึ่ง ซึ่งเส้นเชื่อมจะมีทิศทางระบุเสมอ

2.1.5 สตับ

สตับ [1] เป็นรหัสต้นฉบับที่สร้างขึ้นเพื่อจำลองโครงสร้างของมอดูลใด ๆ สำหรับให้มอดูลภายใต้การทดสอบเรียกใช้งานในการทดสอบแบบซอฟต์แวร์ เมื่อมอดูลที่มอดูลภายใต้การทดสอบเรียกใช้ยังสร้างไม่เสร็จหรือในกรณีที่ต้องการทดสอบแบบรวม (Integration Test) เพื่อทดสอบว่ามอดูลภายใต้การทดสอบสามารถทำงานร่วมกับมอดูลอื่นผ่านส่วนต่อประสานโปรแกรม (Application Programming Interface - API) ได้อย่างถูกต้อง รูปที่ 2-1 แสดงตัวอย่างรหัสต้นฉบับ

ภาษาจาวา (Java) ของสตับ โดยเมท็อดภายในจะสตับจะแสดงค่าของพารามิเตอร์ที่รับเข้ามาออกทางคอนโซล (Console) ด้วยคำสั่ง println และส่งค่าคงที่กลับ

```
public class Stub{
    public int absolute(int n){
        System.out.println(n);
        return 0;
    }
}
```

รูปที่ 2-1 ตัวอย่างรหัสต้นฉบับภาษาจาวาของสตับ

2.1.6 ไดร์เวอร์

ไดร์เวอร์ [1] เป็นรหัสต้นฉบับที่จำลองการเรียกใช้งานมอดูลภายใต้การทดสอบ สำหรับทดสอบการทำงานของมอดูลภายใต้การทดสอบ เมื่อมอดูลที่เรียกใช้งานมอดูลภายใต้การทดสอบยังสร้างไม่เสร็จหรือในกรณีที่ต้องการทดสอบแบบรวมเพื่อทดสอบการเรียกใช้งานของมอดูลภายใต้การทดสอบผ่านส่วนต่อประสานโปรแกรม รูปที่ 2-2 แสดงรหัสต้นฉบับภาษาจาวาของไดร์เวอร์ที่เรียกใช้งานเมท็อดชื่อ absolute ของโปรแกรมหนึ่งที่พัฒนาเสร็จแล้ว โดยรหัสต้นฉบับของโปรแกรมดังกล่าวแสดงในรูปที่ 2-3

```
public Class Driver{
    public static void main(){
        absolute(5);
    }
}
```

รูปที่ 2-2 ตัวอย่างรหัสต้นฉบับภาษาจาวาของไดร์เวอร์

```
public Class A{
    public static int absolute(int n){
        if(n<0){
            return n*(-1);
        }
        return n;
    }
}
```

รูปที่ 2-3 ตัวอย่างรหัสต้นฉบับภาษาจาวาของโปรแกรมที่พัฒนาเสร็จแล้ว

2.1.7 JUnit

Junit [16] เป็นส่วนต่อประสานโปรแกรมที่ใช้ในการทดสอบส่วนย่อยในภาษาจาวา โดยการทำงานของ JUnit จะทำงานแบบ Runnable Class กล่าวคือเป็นคลาสที่สามารถทำงานได้ด้วยตัวเอง โดยไม่ต้องมีการเรียกใช้งานจากคลาสอื่นโดยที่ตัวคลาสเองไม่ใช่คลาสหลัก (Main Class) ของโปรแกรม ซึ่ง JUnit จะมีฟังก์ชันเกี่ยวกับการเปรียบเทียบค่าที่ส่งกลับจากเมท็อดกับผลลัพธ์ที่คาดหวังภายในตัวด้วย รูปที่ 2-4 แสดงรหัสต้นฉบับภาษาจาวาของกรณีทดสอบที่ใช้งาน JUnit

```
package driver.enrollment;

import enrollment.Grader;
import org.junit.Assert;
import org.junit.Test;

public class GraderDriver {
    @Test
    public void testGetGPAX(){
        double returnValue = Grader.getGPAX("1233264");
        Assert.assertEquals(165.45,returnValue,15.66);
    }
}
```

รูปที่ 2-4 รหัสต้นฉบับภาษาจาวาของกรณีทดสอบที่ใช้งาน Junit

2.2 งานวิจัยที่เกี่ยวข้อง

ในการทำวิทยานิพนธ์นี้ได้ศึกษางานวิจัยอื่น ๆ ที่คาดว่าจะเป็นประโยชน์ต่อวิทยานิพนธ์ หรือสามารถนำมาประยุกต์ใช้กับวิทยานิพนธ์ได้ โดยเลือกงานวิจัยมาดังต่อไปนี้

2.2.1 งานวิจัย “SeDiTeC-testing based on sequence diagrams”

งานวิจัยโดย Falk Fraikin และ Thomas Leonhardt (2002:261-266) [5] นำเสนอเครื่องมือทดสอบโปรแกรมที่เขียนจากภาษาเชิงวัตถุ จากแผนภาพลำดับชื่อว่า SeDiTec โดยใช้ข้อมูลที่ระบุอยู่ในแผนภาพลำดับเป็นข้อมูลในการสร้างกรณีทดสอบ โดยผู้วิจัยได้กำหนดคุณสมบัติของแผนภาพลำดับที่เครื่องมือสามารถสร้างการทดสอบได้ขึ้นมา เรียกว่าแผนภาพลำดับที่ทดสอบได้ (Testable Sequence Diagrams) โดยแผนภาพลำดับดังกล่าวมีคุณสมบัติดังต่อไปนี้

1. แผนภาพลำดับมีผู้กระทำ (Actor) เพียงอันเดียวเท่านั้น
2. แผนภาพลำดับต้องมีอ็อบเจกต์ ที่ไม่รวมผู้กระทำอย่างน้อย 1 อ็อบเจกต์
3. ส่วนต่อประสานของทุกอ็อบเจกต์ ต้องมีการระบุซิกเนเจอร์

4. ทุกเมท็อดของทุกอ็อบเจกต์ ต้องมีชื่อไม่ซ้ำกัน (Unique name) และไม่เป็นเมท็อดเชิงสถิต (Static Method)

5. การเรียกเมท็อดแรกของแผนภาพต้องถูกเรียกโดยผู้กระทำเท่านั้น

6. ทุกการเรียกเมท็อดต้องสัมพันธ์กับชื่อเมท็อดที่ระบุไว้ในคลาสปลายทาง

7. การเรียกทุกเมท็อดต้องเป็นการเรียกใช้ครั้งเดียว (Single thread of execution)

หากแผนภาพลำดับที่ต้องการให้เครื่องมือดังกล่าวสร้างกรณีทดสอบไม่เป็นไปตามคุณสมบัติข้างต้น ต้องมีการปรับแต่ง (Refine) ก่อนให้อยู่ในลักษณะของแผนภาพลำดับทดสอบได้ จากนั้นเครื่องมือจะสร้างรหัสต้นฉบับของเมท็อดภายในแผนภาพลำดับในรูปแบบของรหัสสตับ (Stub code) ที่มีเพียงเมท็อดซิกเนเจอร์เท่านั้น

จากแนวคิดข้างต้นจะเห็นว่าเครื่องมือมีข้อจำกัดที่สำคัญคือต้องมีการปรับแต่งแผนภาพลำดับก่อนใช้เครื่องมือ และรหัสต้นฉบับที่สร้างขึ้นเป็นเพียงรหัสสตับเท่านั้นทำให้ทดสอบได้เพียงการส่งข้อมูลระหว่างส่วนต่อประสานและลำดับของการเรียกเมท็อดเท่านั้น อย่างไรก็ตามแนวคิดของงานวิจัยดังกล่าวสามารถนำมาต่อยอดงานวิจัยนี้ได้

2.2.2 งานวิจัย “Automatic Test Case Generation from UML Sequence Diagram”

งานวิจัยโดย Monarisa Sarma, Debasish Kundu และ Rajib Mall (2015:657-687) [6] นำเสนอแนวคิดในการสร้างกรณีทดสอบจากแผนภาพลำดับโดยเปลี่ยนแผนภาพลำดับให้เป็นกราฟแผนภาพลำดับ (Sequence Diagram Graph; SDG) และใช้ข้อมูลเพิ่มเติมจากยูสเคสต้นแบบ (Use Case Template) แผนภาพคลาส และพจนานุกรมข้อมูล (Data Dictionary) ซึ่งผู้วิจัยเขียนอยู่ในลักษณะของ OCL 2.0 ซึ่งเป็นภาษารูปนัยในการอธิบายภาษายูเอ็มแอลแบบหนึ่งที่กำหนดรูปแบบการเขียนที่ตามตัวและเป็นแบบแผน โดยเครื่องมือจะสร้างกรณีทดสอบที่ทำให้โปรแกรมทำงานตามเส้นทางของกราฟแผนภาพลำดับด้วยความครอบคลุมระดับประพจน์ (Statement Coverage)

จากแนวคิดการวิจัยข้างต้น พบว่าการสร้างกรณีทดสอบต้องพึ่งพาข้อมูลจากแหล่งที่มาจำนวนมาก และต้องแปลงข้อมูลดังกล่าวให้อยู่ในรูปของ OCL 2.0 ก่อนป้อนให้เครื่องมือไปใช้ในการสร้างกรณีทดสอบจากแผนภาพลำดับ อย่างไรก็ตามเครื่องมือดังกล่าวสามารถใช้สร้างกรณีทดสอบจากแผนภาพลำดับใด ๆ ก็ได้โดยไม่ต้องจำกัดเงื่อนไขและคุณสมบัติของแผนภาพลำดับ

2.2.3 งานวิจัย “Intra-Class Testing of Abstract Class Features”

งานวิจัยโดย Peter J. Clark และคณะ (2007:191-200) [9] นำเสนอวิธีการทดสอบคลาสนามธรรมโดยปรับปรุงจากวิธีการทดสอบแบบดั้งเดิมคือการทดสอบโดยสร้างคลาสรูปธรรมขึ้นมาจาก

คลาสนามธรรม และทดสอบคลาสรูปธรรมนั้น ซึ่งงานวิจัยนี้ได้นำเสนอขั้นตอนการทดสอบคลาสนามธรรม 4 ขั้นตอน

1. คัดกรองและจัดกลุ่มคลาสตามคลาสนามธรรมและคลาสรูปธรรมที่สืบทอดคลาสนามธรรมแต่ละคลาส โดยระบุเมท็อดที่สืบทอดมาด้วย

2. สร้างกราฟการเรียกใช้เมท็อดภายในแบบปรับปรุง (Modified inter-method call graph) เพื่อแสดงการสืบทอดของคลาสที่ได้จากการแบ่งกลุ่มในขั้นตอนที่ 1

3. ใช้ขั้นตอนวิธีแบบละโมบ (Greedy Algorithm) ในการจัดกลุ่มของเมท็อดที่ต้องสร้างเพื่อทดสอบให้เล็กที่สุด

4. สร้างลำดับในการทดสอบแบบรวมของคลาสจากกราฟในข้อที่ 2 เพื่อใช้สตับให้น้อยที่สุด

จากแนวคิดของงานวิจัยข้างต้น สามารถนำแนวคิดเรื่องการทดสอบคลาสนามธรรมและการทดสอบรวมของคลาสมาประยุกต์ใช้กับงานวิจัยนี้ได้

บทที่ 3
ตัวสร้างสตับและไดร์เวอร์

ในบทนี้จะอธิบายแนวคิดของตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาส

3.1 ภาพรวมของตัวสร้างสตับและไดร์เวอร์

ภาพรวมของตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาสแสดงในรูปที่ 3-1 ซึ่งประกอบไปด้วยขั้นตอนการทำงาน 7 ขั้นตอน โดยมีรายละเอียดดังต่อไปนี้

3.1.1 ขั้นตอนการนำเข้าไฟล์แผนภาพลำดับและแผนภาพคลาส

ผู้ทดสอบนำเข้าไฟล์แผนภาพลำดับชื่อ SD1 แสดงตัวอย่างในรูปที่ 3-2 และแผนภาพคลาสชื่อ CD1 แสดงตัวอย่างในรูปที่ 3-3 ในรูปแบบของไฟล์เอกซ์เอ็มแอล โดยไฟล์เอกซ์เอ็มแอลสร้างได้จากเครื่องมือสร้างแผนภาพยูเอ็มแอล เช่น Visual Paradigm เมื่อตัวสร้างนำเข้าไฟล์เสร็จสิ้น ตัวสร้างจะสร้างรหัสอ้างอิงแผนภาพลำดับหรือรหัสอ้างอิงแผนภาพคลาสตามประเภทของไฟล์และบันทึกข้อมูลของไฟล์ประกอบด้วย รหัสอ้างอิงแผนภาพ ชื่อไฟล์ และที่อยู่ของไฟล์ ลงในฐานข้อมูลของตัวสร้างเพื่อเตรียมการประมวลผลแผนภาพ ตารางที่ 3-1 แสดงตัวอย่างการเก็บไฟล์แผนภาพลำดับ และตารางที่ 3-2 แสดงตัวอย่างการเก็บไฟล์แผนภาพคลาส แผนภาพลำดับและแผนภาพคลาสดังกล่าวจะใช้เป็นตัวอย่างสำหรับการอธิบายขั้นตอนต่าง ๆ ภายในบทนี้

3.1.2 ขั้นตอนการประมวลผลแผนภาพลำดับ

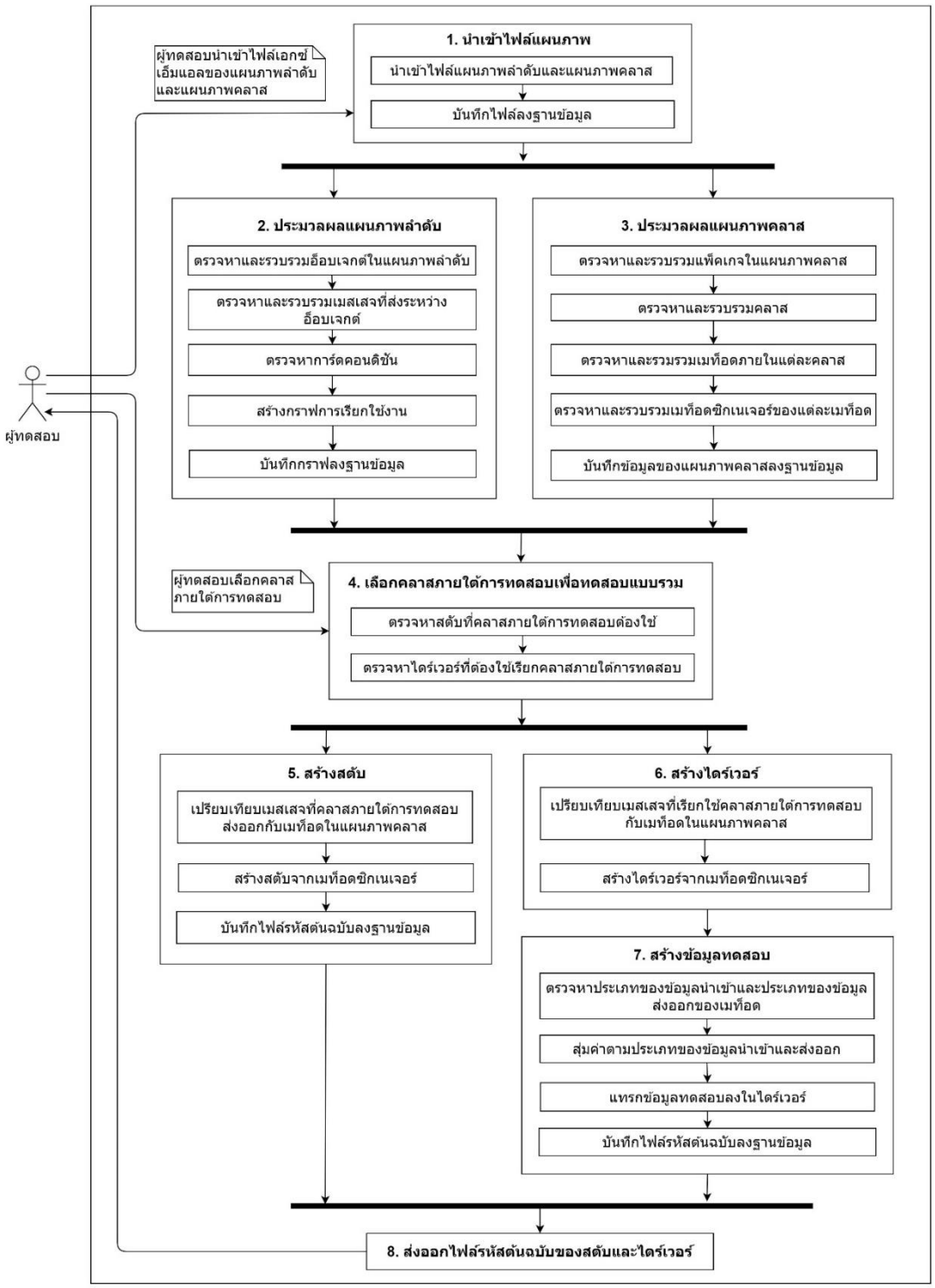
ตัวสร้างจะอ่านไฟล์แผนภาพลำดับตามที่อยู่ของไฟล์แสดงในตารางที่ 3-1 และประมวลผลแผนภาพลำดับที่นำเข้ามาเพื่อสร้างเป็นกราฟการเรียกใช้งาน โดยขั้นตอนการประมวลผลแผนภาพลำดับประกอบด้วยขั้นตอนย่อย ๆ 4 ขั้นตอน โดยมีรายละเอียดดังต่อไปนี้

1. ตรวจหาและรวบรวมอ็อบเจกต์ภายในแผนภาพลำดับ

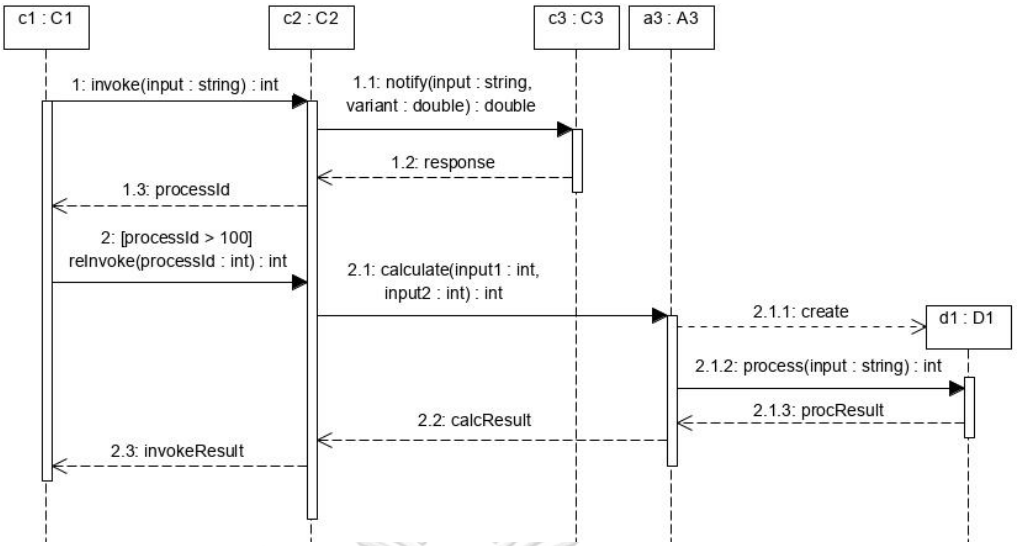
ตัวสร้างจะอ่านแผนภาพลำดับและตรวจหาอ็อบเจกต์ทั้งหมดที่ปรากฏในแผนภาพลำดับ แสดงตัวอย่างจากรูปที่ 3-2 แผนภาพลำดับประกอบไปด้วยอ็อปเจกต์ทั้งหมด 5 อ็อบเจกต์ ได้แก่ C1, C2, C3, A3 และ D1 ตัวสร้างจะรวบรวมและจัดเก็บข้อมูลของอ็อปเจกต์ดังกล่าวได้แก่ รหัสอ้างอิงแผนภาพลำดับจากตารางที่ 3-1 รหัสอ้างอิงอ็อบเจกต์ซึ่งสร้างโดยตัวสร้าง ชื่ออ็อปเจกต์ และคลาสที่สร้างเป็นอ็อบเจกต์นั้น ดังตารางที่ 3-3

2. ตรวจหาและรวบรวมเมสเสจที่ส่งระหว่างอ็อบเจกต์

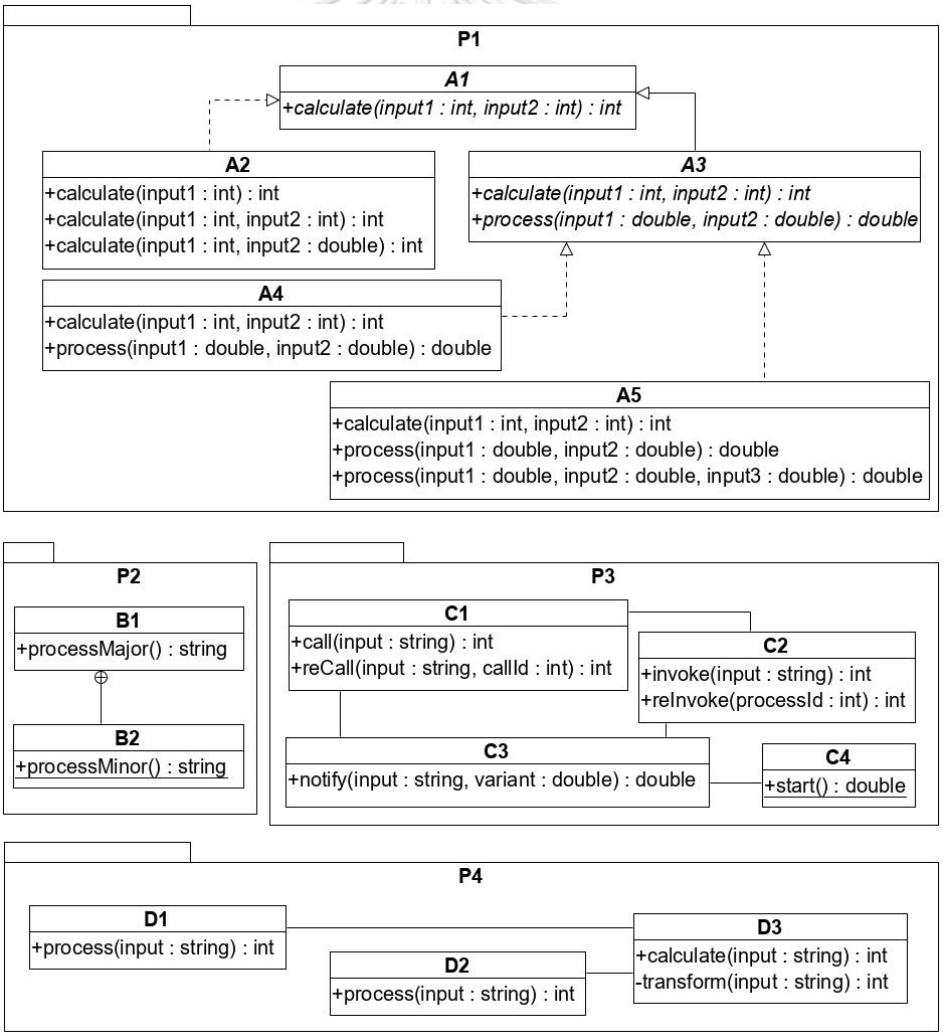
หลังจากตัวสร้างตรวจหาอ็อบเจกต์ทั้งหมดที่ปรากฏในแผนภาพลำดับ ตัวสร้างจะพิจารณาอ็อบเจกต์แต่ละอ็อบเจกต์ว่าส่งเมสเสจชื่ออะไร ส่งหาอ็อบเจกต์อะไร และเป็นเมสเสจประเภทใด โดยประเภทของเมสเสจแบ่งออกเป็น 3 ประเภทได้แก่



รูปที่ 3-1 ภาพรวมของตัวสร้างสตับและไดร์เวอร์



รูปที่ 3-2 แผนภาพลำดับชื่อ SD1



รูปที่ 3-3 แผนภาพคลาสชื่อ CD1

ตารางที่ 3-1 ตัวอย่างการเก็บข้อมูลของแผนภาพลำดับ

รหัสอ้างอิงแผนภาพลำดับ	ชื่อไฟล์แผนภาพลำดับ	ที่อยู่ของไฟล์แผนภาพ
SEQ001	SD1	../SeqDiagramRepo/SD1.xml

ตารางที่ 3-2 ตัวอย่างการเก็บข้อมูลของแผนภาคคลาส

รหัสอ้างอิงแผนภาพคลาส	ชื่อไฟล์แผนภาพคลาส	ที่อยู่ของไฟล์แผนภาพ
CLS001	CD1	../ClassDiagramRepo/CD1.xml

ตารางที่ 3-3 ตัวอย่างการจัดเก็บข้อมูลของอ็อบเจ็คในแผนภาพลำดับ

รหัสอ้างอิงแผนภาพลำดับ		SEQ001
รหัสอ้างอิงอ็อบเจกต์	**ชื่ออ็อบเจกต์**	**คลาสที่สร้างเป็นอ็อบเจกต์**
OBJ01	c1	C1
OBJ02	c2	C2
OBJ03	c3	C3
OBJ04	a3	A3
OBJ05	d1	D1

1. CALL เป็นเมสเสจประเภทการเรียกใช้งาน
2. RETURN เป็นเมสเสจประเภทการตอบกลับ
3. CREATE เป็นเมสเสจประเภทการสร้างอ็อบเจกต์

ตัวอย่างการตรวจหาเมสเสจ จากรูปที่ 3-2 แผนภาพลำดับประกอบไปด้วยอ็อบ-เจกต์ 5 อ็อบเจกต์ โดยแต่ละอ็อบเจกต์มีการส่งเมสเสจดังต่อไปนี้

- อ็อบเจกต์ c1 ส่งเมสเสจหาอ็อบเจกต์ c2 ด้วยเมสเสจประเภท CALL ชื่อ invoke และ reinvoke

- อ็อบเจกต์ c2 ส่งเมสเสจหาอ็อบเจกต์ c3 ด้วยเมสเสจประเภท CALL ชื่อ notify และส่งเมสเสจประเภท CALL ชื่อ calculate หาอ็อบเจกต์ a3 นอกจากนี้อ็อบเจกต์ c2 ยังส่งเมสเสจประเภท RETURN ชื่อ processId และ invokeResult หาอ็อบเจกต์ c1 ด้วย

- อ็อบเจคต์ c3 ส่งเมสเสจประเภท RETURN ชื่อ response หาอ็อบเจกต์ c2

- อ็อบเจกต์ a3 ส่งเมสเสจประเภท CREATE ชื่อ create เพื่อสร้างอ็อบเจกต์ d1 จากคลาส D1 จากนั้นส่งเมสเสจประเภท CALL ชื่อ process หาอ็อบเจกต์ d1 และส่งเมสเสจประเภท RETURN ชื่อ calcResult หาอ็อปเจกต์ c3

- อ็อบเจกต์ d1 ส่งเมสเสจประเภท RETURN ชื่อ procResult หาอ็อบเจกต์ a3

ตารางที่ 3-4 แสดงตัวอย่างการเก็บข้อมูลของเมสเสจในแผนภาพลำดับ ประกอบด้วย รหัสอ้างอิงแผนภาพลำดับจากตารางที่ 3-1 รหัสอ้างอิงเมสเสจซึ่งสร้างโดยตัวสร้าง รหัสอ้างอิงอ็อบเจกต์ต้นทางจากตารางที่ 3-3 ชื่อเมสเสจ ประเภทของเมสเสจ (CALL, RETURN, CREATE) และรหัสอ้างอิงอ็อบเจกต์ปลายทางจากตารางที่ 3-3

ตารางที่ 3-4 ตัวอย่างการจัดเก็บข้อมูลของเมสเสจในแผนภาพลำดับ

รหัสอ้างอิงแผนภาพลำดับ			SEQ001	
รหัสอ้างอิงเมสเสจ	**รหัสอ้างอิงอ็อบเจกต์ต้นทาง**	**ชื่อเมสเสจ**	**ประเภทเมสเสจ**	**รหัสอ้างอิงอ็อบเจกต์ปลายทาง**
MSG01	OBJ01	invoke	CALL	OBJ02
MSG02	OBJ01	reinvoke	CALL	OBJ02
MSG03	OBJ02	notify	CALL	OBJ03
MSG04	OBJ02	calculate	CALL	OBJ04
MSG05	OBJ02	processId	RETURN	OBJ01
MSG06	OBJ02	invokeResult	RETURN	OBJ01
MSG07	OBJ03	response	RETURN	OBJ02
MSG08	OBJ04	create	CREATE	OBJ05
MSG09	OBJ04	process	CALL	OBJ05
MSG10	OBJ04	calcResult	RETURN	OBJ02
MSG11	OBJ05	procResult	RETURN	OBJ03

3. ตรวจหาการ์ดคอนดิชัน (Guard Condition)

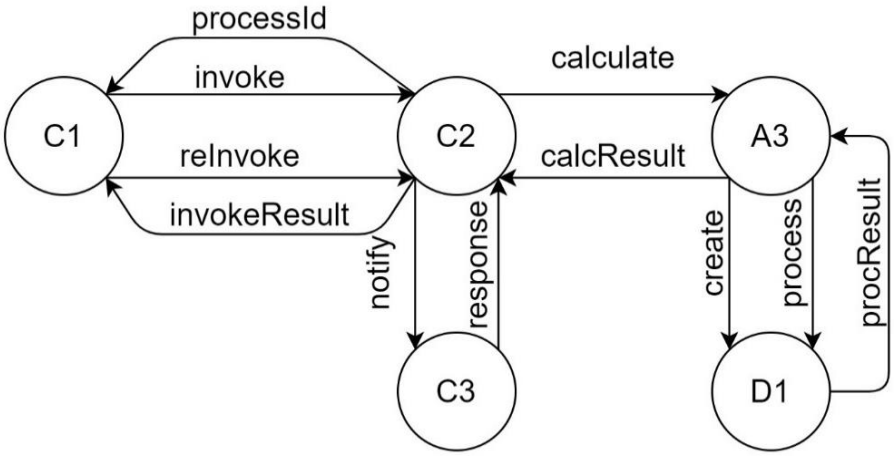
ในกรณีที่เมสเสจถูกกำกับด้วยการ์ดคอนดิชัน แสดงตัวอย่างในรูปที่ 3-2 เมสเสจชื่อ reinvoke ซึ่งส่งโดยอ็อบเจกต์ c1 หาอ็อบเจกต์ c2 ถูกกำกับด้วยการ์ดคอนดิชันที่มีรายละเอียดว่า processId > 100 ซึ่งหมายความว่า อ็อบเจกต์ c1 ส่งเมสเสจ reinvoke หาอ็อบเจกต์ c2 เมื่อ processId ที่เป็นอาร์กิวเมนต์ของเมสเสจ reinvoke มีค่ามากกว่า 100 ตัวสร้างจะจัดเก็บการ์ด-คอนดิชันดังกล่าวไว้ดังแสดงในตารางที่ 3-5 ประกอบด้วยรหัสอ้างอิงแผนภาพลำดับจากตารางที่ 3-1 รหัสอ้างอิงเมสเสจจากตารางที่ 3-4 และรายละเอียดของการ์ดคอนดิชัน

ตารางที่ 3-5 ตัวอย่างการจัดเก็บข้อมูลของการ์ดคอนดิชัน

รหัสอ้างอิงแผนภาพลำดับ	SEQ001
รหัสอ้างอิงเมสเสจ	**รายละเอียดของการ์ดคอนดิชัน**
MSG02	processId > 100

4. สร้างกราฟการเรียกใช้งาน

หลังจากตัวสร้างรวบรวมอ็อบเจกต์และเมสเสจที่ส่งระหว่างแต่ละ อ็อบเจกต์ ตัวสร้างจะสร้างกราฟการเรียกใช้งานโดยโหนดของกราฟจะแสดงถึงคลาสที่สร้างอ็อบเจกต์ในแผนภาพลำดับ และเส้นเชื่อมของกราฟจะแสดงถึงเมสเสจ รูปที่ 3-4 แสดงกราฟการเรียกใช้งานที่สร้างจากแผนภาพลำดับรูปที่ 3-2 กราฟประกอบด้วยโหนดที่แสดงถึงคลาสที่สร้างอ็อบเจกต์ในแผนภาพลำดับแสดงในตารางที่ 3-3 และเส้นเชื่อมแสดงเมสเสจที่ส่งระหว่างอ็อบเจกต์แสดงในตารางที่ 3-4 จากนั้นตัวสร้างจะบันทึกข้อมูลของกราฟการเรียกใช้งานและการ์ดคอนดิชันลงฐานข้อมูล



รูปที่ 3-4 กราฟการเรียกใช้งานของแผนภาพลำดับรูปที่ 3-2

3.1.3 ขั้นตอนการประมวลผลแผนภาพคลาส

ตัวสร้างอ่านไฟล์แผนภาพคลาสตามที่อยู่ของไฟล์ดังแสดงในตารางที่ 3-2 และประมวลผลแผนภาพคลาสเพื่อรวบรวมเมท็อดซิกเนเจอร์ของทุกเมท็อดของทุกคลาสในแผนภาพคลาส ขั้นตอนการประมวลผลแผนภาพคลาสประกอบไปด้วย 4 ขั้นตอนย่อย โดยมีรายละเอียดดังต่อไปนี้

1. ตรวจหาและรวบรวมแพ็คเกจในแผนภาพคลาส

ตัวสร้างจะตรวจหาแพ็คเกจภายในแผนภาพคลาสทั้งหมดและรวบรวมชื่อแพ็คเกจและเนมสเปซ (Namespace) ของแพ็คเกจ แผนภาพคลาสในรูปที่ 3-3 ประกอบไปด้วย 4 แพ็คเกจ ได้แก่ แพ็คเกจ P1, P2, P3, และ P4 โดยตัวสร้างจะเก็บข้อมูลชองแพ็คเกจดังตารางที่ 3-6 โดยประกอบด้วย รหัสอ้างอิงแผนภาพคลาสจากตารางที่ 3-2 รหัสอ้างอิงแพ็คเกจซึ่งถูกสร้างโดยตัวสร้าง ชื่อแพ็คเกจ และเนมสเปซ

ตารางที่ 3-6 ตัวอย่างการจัดเก็บข้อมูลของแพ็คเกจในแผนภาพคลาส

รหัสอ้างอิงแผนภาพคลาส	CLS001	
รหัสอ้างอิงแพ็คเกจ	**ชื่อแพ็คเกจ**	**เนมสเปซ**
PKG01	P1	P1
PKG02	P2	P2
PKG03	P3	P3
PKG04	P4	P4

2. ตรวจหาและรวบรวมคลาสในแต่ละแพ็คเกจ

ตัวสร้างจะตรวจหาคลาสที่ปรากฏในแต่ละแพ็คเกจ โดยรวบรวมข้อมูลของคลาสได้แก่ชื่อคลาส และประเภทของคลาส โดยประเภทของคลาสสามารถแบ่งได้เป็น 3 ประเภทได้แก่

1. CONCRETE คือคลาสรูปธรรม ซึ่งรวมถึงคลาสภายใน
2. ABSTRACT คือคลาสนามธรรม
3. INTERFACE คืออินเตอร์เฟส

จากแผนภาพคลาสรูปที่ 3-3 ขอยกตัวอย่างการตรวจหาและรวบรวมคลาสภายในแพ็คเกจ P1 โดยจากรูป แพ็คเกจ P1 ประกอบไปด้วยคลาสทั้งหมด 5 คลาสได้แก่คลาส A1, A2, A3, A4, และ A5 คลาส A1 เป็นคลาสนามธรรม โดยมีคลาส A2 ซึ่งเป็นคลาสรูปธรรมและคลาส A3 ซึ่งเป็นคลาสนามธรรมสืบทอดมา และคลาส A3 มีคลาส A4 และคลาส A5 ที่เป็นคลาสรูปธรรมสืบทอดมา ตัวสร้างจะเก็บข้อมูลของคลาสข้างต้นดังแสดงในตารางที่ 3-7 ประกอบด้วยรหัสอ้างอิงแผนภาพ

คลาสจากตารางที่ 3-2 รหัสอ้างอิงแพ็คเกจจากตารางที่ 3-6 รหัสอ้างอิงคลาสซึ่งถูกสร้างโดยตัวสร้าง ชื่อคลาส และประเภทของคลาส (CONCRETE, ABSTRACT, INTERFACE)

ตารางที่ 3-7 ตัวอย่างการจัดเก็บข้อมูลของคลาสในแพ็คเกจ P1

รหัสอ้างอิงแผนภาพคลาส	CLS001	
รหัสอ้างอิงแพ็คเกจ	PKG01	
รหัสอ้างอิงคลาส	**ชื่อคลาส**	**ประเภทของคลาส**
CLASS01	A1	ABSTRACT
CLASS02	A2	CONCRETE
CLASS03	A3	ABSTRACT
CLASS04	A4	CONCRETE
CLASS05	A5	CONCRETE

นอกจากนี้ตัวสร้างยังรวบรวมข้อมูลการสืบทอดในกรณีที่คลาสเป็นคลาสนามธรรมหรืออินเตอร์เฟส โดยเก็บรหัสอ้างอิงคลาสแม่ (คลาสที่ถูกสืบทอด) และรหัสอ้างอิงคลาสลูก (คลาสที่สืบทอด) โดยตารางที่ 3-8 แสดงตัวอย่างการเก็บข้อมูลของการสืบทอดของคลาสประกอบด้วย รหัสอ้างอิงแผนภาพคลาสจากตารางที่ 3-2 รหัสอ้างอิงคลาสแม่ และรหัสอ้างอิงคลาสลูก ซึ่งเป็นรหัสอ้างอิงคลาสจากตารางที่ 3-7

ตารางที่ 3-8 ตัวอย่างการจัดเก็บข้อมูลของการสืบทอดของคลาสในแผนภาพคลาสรูปที่ 3-3

รหัสอ้างอิงแผนภาพคลาส	CLS001
รหัสอ้างอิงคลาสแม่	**รหัสอ้างอิงคลาสลูก**
CLASS01	CLASS02
CLASS01	CLASS03
CLASS03	CLASS04
CLASS03	CLASS05

3. ตรวจหาและรวมรวมเมท็อดในแต่ละคลาส

ตัวสร้างจะตรวจหาเมท็อดในแต่ละคลาสเพื่อรวบรวมรายการของเมท็อดของแต่ละคลาส ขอยกตัวอย่างการตรวจหาและรวบรวมเมท็อดของคลาสภายในแพ็คเกจ P1 ของแผนภาพคลาสรูปที่ 3-3 แพ็คเกจ P1 ประกอบไปด้วยคลาสทั้งหมด 5 คลาสดังแสดงรายละเอียดในตารางที่ 3-7 โดยแต่ละคลาสประกอบด้วยเมท็อดดังต่อไปนี้

- คลาส A1 ประกอบด้วยเมท็อดหนึ่งเมท็อดชื่อ calculate
- คลาส A2 ประกอบด้วยเมท็อด 3 เมท็อดชื่อ calculate ทั้ง 3 เมท็อดซึ่งเป็นคุณสมบัติของการโอเวอร์โหลดดิ้ง (Overloading) [2] เมท็อดทั้งสามถึงแม้ว่าจะมีชื่อเหมือนกันแต่ถือว่าเป็นคนละเมท็อดกันเนื่องจากมีเมท็อดซิกเนเจอร์ที่ไม่เหมือนกัน
- คลาส A3 ประกอบด้วยเมท็อด 2 เมท็อดชื่อ calculate และ process
- คลาส A4 ประกอบด้วยเมท็อด 2 เมท็อดชื่อ calculate และ process
- คลาส A5 ประกอบด้วยเมท็อด 3 เมท็อดชื่อ calculate และ process โดยเมท็อด process มีการโอเวอร์โหลดดิ้งออกเป็น 2 เมท็อด

ตารางที่ 3-9 แสดงตัวอย่างการเก็บข้อมูลของเมท็อดของคลาสในแพ็คเกจ P1 โดยประกอบด้วยรหัสอ้างอิงแผนภาพคลาสจากตารางที่ 3-2 รหัสอ้างอิงแพ็คเกจจากตารางที่ 3-6 รหัสอ้างอิงคลาสอ้างอิงจากตารางที่ 3-7 รหัสอ้างอิงเมท็อดซึ่งถูกสร้างโดยตัวสร้าง และชื่อเมท็อด

ตารางที่ 3-9 ตัวอย่างการจัดเก็บข้อมูลของเมท็อดของแต่ละคลาสในแพ็คเกจ P1

รหัสอ้างอิงแผนภาพคลาส	CLS001	
รหัสอ้างอิงแพ็คเกจ	PKG01	
รหัสอ้างอิงคลาส	**รหัสอ้างอิงเมท็อด**	**ชื่อเมท็อด**
CLASS01	MT01	calculate
CLASS02	MT02	calculate
CLASS02	MT03	calculate
CLASS02	MT04	calculate
CLASS03	MT05	calculate
CLASS03	MT06	process
CLASS04	MT07	calculate
CLASS04	MT08	process

ตารางที่ 3-9 ตัวอย่างการจัดเก็บข้อมูลของเมท็อดของแต่ละคลาสในแพ็คเกจ (ต่อ)

รหัสอ้างอิงคลาส	รหัสอ้างอิงเมท็อด	ชื่อเมท็อด
CLASS05	MT09	calculate
CLASS05	MT10	process
CLASS05	MT11	process

4. ตรวจหาและรวบรวมเมท็อดซิกเนเจอร์ของเมท็อดแต่ละเมท็อด

ตัวสร้างจะตรวจหาและรวบรวมเมท็อดซิกเนเจอร์ของเมท็อดแต่ละเมท็อด โดยเมท็อดซิกเนเจอร์ประกอบไปด้วย

1. ชื่อเมท็อด
2. ระดับการมองเห็น (Visibility) เช่น public, private, protected
3. ประเภทของข้อมูลส่งออก
4. ประเภทของเมท็อด
5. รายการของพารามิเตอร์และชนิดข้อมูลของพารามิเตอร์

สำหรับประเภทของเมท็อด สามารถแบ่งได้ออกเป็นสามประเภทได้แก่

1. CONCRETE คือเมท็อดรูปธรรม
2. ABSTRACT คือเมท็อดนามธรรม
3. STATIC คือเมท็อดเชิงสถิต

ตัวอย่างการรวบรวมเมท็อดซิกเนเจอร์ ขอยกตัวอย่างเมท็อดชื่อ calculate ในคลาส A1 แสดงในแผนภาพคลาสรูปที่ 3-3 เมท็อด calculate เป็นเมท็อดนามธรรม สังเกตได้จากการใช้ตัวเอียง [2] และมีระดับการมองเห็นเป็น public สังเกตได้จากเครื่องหมาย + หน้าชื่อเมท็อด [2] เมท็อดมีประเภทของข้อมูลส่งออกเป็นจำนวนเต็มหรือ int และมีพารามิเตอร์สองตัวชื่อ input1 และ input2 โดยพารามิเตอร์ทั้งสองมีประเภทของข้อมูลเป็นจำนวนเต็ม ตารางที่ 3-10 แสดงตัวอย่างการจัดเก็บข้อมูลของเมท็อดซิกเนเจอร์ของเมท็อด calculate ของคลาส A1 โดยประกอบด้วยรหัสอ้างอิงคลาสจากตารางที่ 3-7 รหัสอ้างอิงเมท็อดจากตารางที่ 3-9 ชื่อเมท็อด ระดับการมองเห็น ประเภทของเมท็อด ประเภทของข้อมูลส่งออก และข้อมูลของพารามิเตอร์แต่ละตัวประกอบด้วย ลำดับพารามิเตอร์ ชื่อพารามิเตอร์ และประเภทของข้อมูล

ตารางที่ 3-10 ตัวอย่างการจัดเก็บข้อมูลของเมท็อดซิกเนเจอร์ของเมท็อด calculate ของคลาส A1

รหัสอ้างอิงคลาส		CLASS01
รหัสอ้างอิงเมท็อด		MT01
ชื่อเมท็อด		calculate
ระดับการมองเห็น		public
ประเภทของเมท็อด		ABSTRACT
ประเภทของข้อมูลส่งออก		int
ลำดับพารามิเตอร์	**ชื่อพารามิเตอร์**	**ประเภทของข้อมูล**
1	input1	int
2	input2	int

3.1.4 ขั้นตอนการเลือกคลาสภายใต้คลาสภายใต้การทดสอบ

หลังจากผู้ทดสอบนำเข้าแผนภาพลำดับและแผนภาพคลาส ผู้ทดสอบจะเลือกคลาสภายใต้การทดสอบโดยเลือกกราฟการเรียกใช้งานและแผนภาพคลาสที่เกี่ยวข้องกับกราฟการเรียกใช้งานดังกล่าว จากนั้นผู้ทดสอบจะเลือกคลาสในกราฟการเรียกใช้งานเป็นคลาสภายใต้การทดสอบ หากผู้ทดสอบเลือกคลาสภายใต้การทดสอบมากกว่าหนึ่งคลาส จะเรียกคลาสที่เลือกว่า “กลุ่มของคลาสภายใต้การทดสอบ” เมื่อผู้ทดสอบเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบเสร็จสิ้น ตัวสร้างจะวิเคราะห์กราฟการเรียกใช้งานเพื่อตรวจหาและรวบรวมรายการของสตับและไดร์เวอร์ที่ต้องสร้างเพื่อทดสอบคลาสหรือกลุ่มของคลาสภายใต้การทดสอบข้างต้น การเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบสามารถเกิดรูปแบบการเลือกได้ 5 รูปแบบ โดยแต่ละรูปแบบมีรายละเอียดการเลือกและการตรวจหาสตับและไดร์เวอร์ที่ต้องใช้ดังต่อไปนี้

1. กลุ่มของคลาสภายใต้การทดสอบที่อยู่ติดกัน

กลุ่มของคลาสภายใต้การทดสอบที่อยู่ติดกันเป็นรูปแบบการเลือกกลุ่มของคลาสภายใต้การทดสอบที่มีเส้นทางถึงกันโดยตรง ยกตัวอย่างเช่น ผู้ทดสอบเลือกคลาส C2 และ C3 ในกราฟการเรียกใช้งานรูปที่ 3-4 เป็นกลุ่มของคลาสภายใต้การทดสอบ จากกราฟการเรียกใช้งานพบว่าคลาส C2 มีเส้นทางถึงคลาส C3 โดยตรง ทำให้คลาส C2 กับ C3 เป็นกลุ่มของคลาสภายใต้การทดสอบที่อยู่ติดกัน

การตรวจหาสตับที่ต้องใช้ทดสอบกลุ่มของคลาสภายใต้การทดสอบ ตัวสร้างจะพิจารณาสตับที่ต้องใช้ทดสอบของแต่ละคลาสในกลุ่มโดยมีรายละเอียดดังต่อไปนี้

1. คลาส C2 จากรูปที่ 3-4 คลาส C2 ส่งเมสเสจหาทั้งหมด 4 เมสเสจได้แก่ notify, calculate, processId, และ invokeResult ตัวสร้างจะพิจารณาเฉพาะเมสเสจประเภท CALL และ CREATE ว่าถูกส่งไปหาคลาสใดเนื่องจากเมสเสจประเภท RETURN เป็นการส่งค่ากลับไม่ใช่การเรียกใช้งาน ดังนั้นตัวสร้างจะพิจารณาเฉพาะเมสเสจ notify และ calculate ซึ่งมีรายละเอียดดังนี้

1.1 เมสเสจ notify จากรูปที่ 3-4 เมสเสจ notify ถูกส่งหาคลาส C3 เพราะฉะนั้นต้องสร้างสตับของคลาส C3 เพื่อใช้ทดสอบคลาส C2

1.2 เมสเสจ calculate จากรูปที่ 3-4 เมสเสจ calculate ถูกส่งหาคลาส A3 เพราะฉะนั้นต้องสร้างสตับของคลาส A3 เพื่อใช้ทดสอบคลาส C2

2 คลาส C3 จากรูปที่ 3-4 คลาส C3 ส่งเมสเสจชื่อ response หาคลาส C2 เพียงเมสเสจเดียว แต่เนื่องจากเมสเสจ response เป็นเมสเสจประเภท RETURN จึงไม่ถูกนำมาพิจารณา ดังนั้นจึงไม่ต้องสร้างสตับเพื่อทดสอบคลาส C3

จากการพิจารณากราฟการเรียกใช้งานข้างต้น พบว่าต้องสร้างสตับของคลาส C3 และ A3 เพื่อทดสอบคลาส C2 และ C3 แต่เนื่องจากคลาส C3 ถูกเลือกเป็นคลาสภายใต้การทดสอบจึงไม่จำเป็นต้องสร้างสตับของคลาส C3 ขึ้นมา ดังนั้นสตับที่ต้องใช้ในการทดสอบกลุ่มของคลาสภายใต้การทดสอบดังกล่าวได้แก่สตับของคลาส A3 เพียงคลาสเดียว

สำหรับการตรวจหาไดร์เวอร์ที่ต้องใช้ทดสอบกลุ่มของคลาสภายใต้การทดสอบ ตัวสร้างจะพิจารณาไดร์เวอร์ที่ต้องใช้ทดสอบของแต่ละคลาสในกลุ่มโดยมีรายละเอียดดังต่อไปนี้

1. คลาส C2 จากรูปที่ 3-4 คลาส C2 ได้รับเมสเสจชื่อ invoke, reinvoke, response, และ calcResult แต่เนื่องจากเมสเสจ response และ calcResult เป็นเมสเสจประเภท RETURN ตัวสร้างจึงไม่พิจารณาเมสเสจดังกล่าว เหลือพิจารณาเพียงเมสเสจ invoke และ reinvoke โดยรายละเอียดการพิจารณาเมสเสจทั้งสองมีดังต่อไปนี้

1.1 เมสเสจ invoke จากรูปที่ 3-4 เมสเสจ invoke ถูกส่งโดยคลาส C1 เพราะฉะนั้นต้องสร้างไดร์เวอร์เพื่อเป็นตัวแทนคลาส C1 สำหรับทดสอบคลาส C2

1.2 เมสเสจ reinvoke จากรูปที่ 3-4 เมสเสจ reInvoke ถูกส่งโดย คลาส C1 เช่นเดียวกันกับเมสเสจ invoke จึงต้องใช้ไดร์เวอร์ที่เป็นตัวแทนคลาส C1 เช่นกัน

2. คลาส C3 จากรูปที่ 3-4 คลาส C3 ได้รับเมสเสจชื่อ notify และ procResult แต่เนื่องจากเมสเสจ procResult เป็นเมสเสจประเภท RETURN ตัวสร้างจึงพิจารณาเฉพาะเมสเสจ notify เพียงเมสเสจเดียว ซึ่งเมสเสจ notify ถูกส่งจากคลาส C2 เพราะฉะนั้นต้องสร้างไดร์เวอร์เพื่อเป็นตัวแทนคลาส C2 สำหรับทดสอบคลาส C3

จากการพิจารณากราฟการเรียกใช้งานพบว่าต้องสร้างไดร์เวอร์ที่เป็นตัวแทนคลาส C1 และ C2 เพื่อทดสอบคลาส C2 และ C3 แต่เนื่องจากคลาส C2 ถูกเลือกเป็นคลาสภายใต้การทดสอบ จึงไม่จำเป็นต้องสร้างไดร์เวอร์เพื่อเป็นตัวแทนคลาส C2 ขึ้นมา ดังนั้นไดร์เวอร์ที่ต้องใช้ในการทดสอบกลุ่มของคลาสภายใต้การทดสอบดังกล่าวได้แก่ไดร์เวอร์ที่เป็นตัวแทนของคลาส C1 เพียงคลาสเดียว

จากตัวอย่างข้างต้นจะได้รายการของสตับที่ต้องใช้ได้แก่สตับของคลาส A3 และรายการของไดร์เวอร์ที่ต้องใช้ได้แก่ไดร์เวอร์ตัวแทนคลาส C1

2. การเลือกกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางหนึ่งคลาส

การเลือกกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางหนึ่งคลาสเป็นการเลือกคลาสภายใต้การทดสอบที่เส้นทางจากคลาสที่เลือกคลาสหนึ่งถึงอีกคลาสหนึ่งต้องผ่านคลาสไม่ได้เลือกเป็นคลาสภายใต้การทดสอบหนึ่งคลาสเท่านั้น ยกตัวอย่างเช่น ผู้ทดสอบเลือกคลาส C1 และ C3 ในกราฟการเรียกใช้งานรูปที่ 3-4 เป็นกลุ่มของคลาสภายใต้การทดสอบ จากกราฟการเรียกใช้งานคลาส C1 มีเส้นทางถึงคลาส C3 แต่ต้องผ่านคลาส C2 หนึ่งโหนดซึ่งคลาส C2 ไม่ได้เป็นถูกเลือกเป็นคลาสภายใต้การทดสอบ ทำให้คลาส C1 และ C3 เป็นกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางหนึ่งคลาส

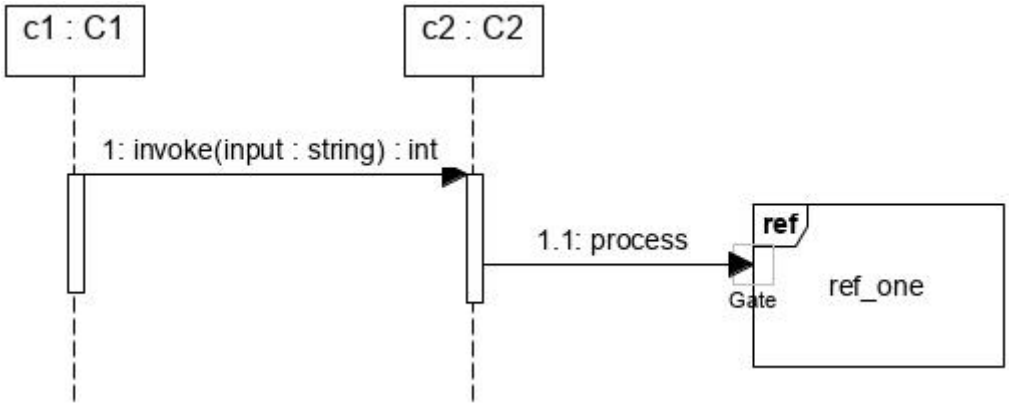
การตรวจหาสตับเพื่อทดสอบกลุ่มของคลาสภายใต้การทดสอบข้างต้น ตัวสร้างพิจารณาคลาสในกลุ่มทีละคลาส ซึ่งจากรูปที่ 3-4 พบว่าคลาส C1 มีการส่งเมสเสจประเภท CALL หาคลาส C2 ในขณะที่คลาส C3 ไม่มีการส่งเมสเสจประเภท CALL หรือ CREATE หาอ็อบเจกต์อื่น ดังนั้นต้องสร้าง สตับของคลาส C2 เพื่อทดสอบกลุ่มของคลาสดังกล่าว

สำหรับการตรวจหาไดร์เวอร์ ตัวสร้างพิจารณาคลาสในกลุ่มทีละคลาส ซึ่งจากรูปที่ 3-4 พบว่าคลาส C1 ไม่ได้รับเมสเสจประเภท CALL หรือ CREATE (ประเภทของเมสเสจแสดงในตารางที่ 3-4) จากคลาสอื่น ๆ ในขณะที่คลาส C3 รับเมสเสจประเภท CALL จากคลาส C2 ดังนั้นต้องสร้างไดร์เวอร์ตัวแทนคลาส C2 เพื่อทดสอบกลุ่มของคลาสดังกล่าว

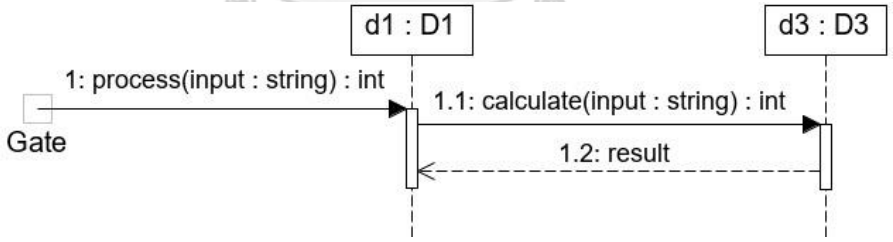
จากการพิจารณากราฟการเรียกใช้งานเพื่อรวบรวมรายการของสตับและไดร์เวอร์เพื่อทดสอบกลุ่มของคลาสภายใต้การทดสอบข้างต้นพบว่าต้องสร้างสตับของคลาส C2 แต่ในขณะเดียวกันก็ต้องสร้างไดร์เวอร์ขึ้นมาเป็นตัวแทนของคลาส C2 เช่นกัน ซึ่งในทางปฏิบัติมีความเป็นไปได้ที่จะสร้างมอดูลตัวหนึ่งที่เป็นทั้งสตับและไดร์เวอร์ที่มีพื้นฐานมาจากคลาส C2 แต่เนื่องจากแนวคิดการออกแบบเชิงวัตถุ [2] และแนวคิดของสตับและไดร์เวอร์ [1] สตับและไดร์เวอร์ควรมีหน้าที่เพียงอย่างเดียวเท่านั้น ตัวสร้างจึงสร้างสตับของคลาส C2 และไดร์เวอร์ตัวแทนคลาส C2 แยกกัน

3. การเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบที่มีการอ้างอิงแผนภาพลำดับอื่น

การเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบที่มีการอ้างอิงแผนภาพลำดับอื่นเป็นการเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสใด ๆ ที่เลือกส่งเมสเสจหาแผนภาพลำดับอื่น ยกตัวอย่างเช่น ผู้ทดสอบเลือกคลาส C2 ในแผนภาพลำดับรูปที่ 3-5 เป็นคลาสภายใต้การทดสอบ จากรูปที่ 3-5 อ็อบเจกต์ c1 ซึ่งสร้างจากคลาส C1 ส่งเมสเสจประเภท CALL ชื่อ invoke หาอ็อบเจกต์ c2 ซึ่งสร้างจากคลาส C2 และอ็อบเจกต์ c2 ส่ง เมสเสจประเภท CALL ชื่อ process หาแผนภาพลำดับชื่อว่า ref_one โดยแผนภาพลำดับ ref_one แสดงในรูปที่ 3-6



รูปที่ 3-5 แผนภาพลำดับที่มีการส่งเมสเสจหาแผนภาพลำดับอื่น



รูปที่ 3-6 แผนภาพลำดับ ref_one

เมื่อพิจารณาคลาส C2 ที่ถูกเลือกเป็นคลาสภายใต้การทดสอบ คลาส C2 ส่งเมสเสจชื่อ process หาแผนภาพลำดับ ref_one แสดงว่าต้องสร้างสตับที่เป็นตัวแทนของแผนภาพลำดับ ref_one ซึ่งจากรูปที่ 3-6 อ็อบเจกต์แรกที่ปรากฏในแผนภาพคืออ็อบเจกต์ d1 ซึ่งหมายความว่าเมสเสจที่ถูกส่งมาจาก คลาส C2 หาแผนภาพลำดับ ref_one ต้องถูกส่งหาอ็อบเจกต์ d1 ของแผนภาพลำดับ ref_one เพราะฉะนั้นตัวสร้างจะสร้างสตับของคลาส D1 ซึ่งเป็นคลาสที่สร้างอ็อบเจกต์ d1 เป็นตัวแทนของแผนภาพลำดับ ref_one เพื่อใช้ทดสอบคลาส C2

ดังนั้นจากตัวอย่างนี้จะได้รายการของสตับและไดร์เวอร์ที่ต้องสร้างเพื่อทดสอบคลาส C2 ได้แก่สตับของคลาส D1 และไดร์เวอร์ตัวแทนคลาส C1

4. การเลือกกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางสองคลาสขึ้นไป

การเลือกกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางสองคลาสขึ้นไป เป็นรูปแบบการเลือกกลุ่มของคลาสภายใต้การทดสอบที่เส้นทางจากคลาสที่เลือกคลาสหนึ่งถึงอีกคลาสหนึ่งต้องผ่านคลาสอื่นที่ไม่ได้เลือกตั้งแต่สองคลาสขึ้นไป ยกตัวอย่างเช่น ผู้ทดสอบเลือกคลาส C1 และคลาส D1 ในกราฟการเรียกใช้งานรูปที่ 3-4 เป็นกลุ่มของคลาสภายใต้การทดสอบ จากกราฟการเรียกใช้งานจะเห็นว่าเส้นทางจากคลาส C1 ถึงคลาส D1 ต้องผ่านคลาส C2 และคลาส A3 ทำให้คลาส C1 และ D1 เป็นกลุ่มของคลาสภายใต้การทดสอบที่มีคลาสอื่นขั้นกลางสองคลาส

จากหลักการการทดสอบแบบรวม [1] การทดสอบรวมของคลาสทั้งสองไม่สมเหตุสมผลเนื่องจากคลาสทั้งสองเป็นอิสระต่อกัน ทำให้ในกรณีนี้ตัวสร้างจะไม่สร้างสตับและไดร์-เวอร์เพื่อทดสอบคลาสทั้งสองและแจ้งเตือนผู้ทดสอบว่าคลาสทั้งสองไม่เกี่ยวข้องกัน

5. การเลือกกลุ่มของคลาสภายใต้การทดสอบที่ไม่มีเส้นทางถึงกัน

การเลือกกลุ่มของคลาสภายใต้การทดสอบที่ไม่มีเส้นทางถึงกัน เป็นรูปแบบการเลือกคลาสภายใต้การทดสอบที่จากคลาสหนึ่งถึงอีกคลาสหนึ่งมีการเส้นทางถึงกัน เมื่อพิจารณาเฉพาะเส้นเชื่อมที่เป็นเมสเสจประเภท CALL หรือ CREATE ตัวอย่างเช่น ผู้ทดสอบเลือกคลาส C3 และคลาส D1 ในกราฟการเรียกใช้งานรูปที่ 3-4 จากรูปพบว่าคลาส C3 มีเส้นทางออกหนึ่งเส้นทางคือเมสเสจ response แต่จากตารางที่ 3-4 เมสเสจ response เป็นเมสเสจประเภท RETURN ซึ่งไม่ถูกนำมาพิจารณา ทำให้ไม่มีเส้นทางออกจากคลาส C3 ในทางกลับกัน เมื่อพิจารณาคลาส D1 พบว่ามีเส้นทางออกหนึ่งเส้นทางชื่อ procResult แต่จากตารางที่ 3-4 เมสเสจ procResult เป็นเมสเสจประเภท RETURN ทำให้ไม่มีเส้นทางออกจากคลาส D1 เช่นกัน ดังนั้นสามารถสรุปได้ว่าไม่มีเส้นทางจากคลาส C3 ไปถึงคลาส D1 และในทางกลับกัน ก็ไม่มีเส้นทางจากคลาส D1 ไปถึงคลาส C3 ทำให้การเลือกคลาส C3 และ D1 เป็นคลาสภายใต้การทดสอบจึงเป็นการเลือกกลุ่มของคลาสภายใต้การทดสอบที่ไม่มีเส้นทางถึงกันจากหลักการการทดสอบแบบรวม [1] การทดสอบรวมของคลาสทั้งสองไม่สมเหตุสมผลเนื่องจากคลาสทั้งสองเป็นอิสระต่อกัน ทำให้ในกรณีนี้ตัวสร้างจะไม่สร้างสตับและไดร์เวอร์เพื่อทดสอบคลาสทั้งสองและแจ้งเตือนผู้ทดสอบว่าคลาสทั้งสองไม่เกี่ยวข้องกัน

3.1.5 ขั้นตอนการสร้างรหัสต้นฉบับของสตับ

เมื่อได้รายการของสตับที่ต้องสร้าง ตัวสร้างจะสร้างรหัสต้นฉบับของสตับของแต่ละคลาสเป็นภาษาจาวา โดยการสร้างสตับประกอบไปด้วยขั้นตอนย่อย 2 ขั้นตอนโดยมีรายละเอียดดังต่อไปนี้

1. เปรียบเทียบเมสเสจกับเมท็อดในแผนภาพคลาส

ขั้นตอนการเปรียบเทียบเมสเสจกับเมท็อดในแผนภาพคลาส เป็นการตรวจหาว่าเมสเสจนั้นเป็นเมท็อดใดของคลาสเพื่อนำเมท็อดนั้นไปค้นหาเมท็อดซิกเนเจอร์ ยกตัวอย่างเช่น เมส-เสจ invoke ในรูปที่ 3-2 เป็นเมสเสจที่ถูกส่งหาคลาส C2 ซึ่งสำหรับแผนภาพลำดับชื่อเมสเสจต้องเป็นชื่อเมท็อดหนึ่งของคลาสที่รับเมสเสจนั้น [2] จากแผนภาพคลาสรูปที่ 3-3 พบว่าเมสเสจชื่อ invoke คือเมท็อด invoke ในคลาส C2 ดังนั้น เมสเสจ invoke เป็นการเรียกใช้งานเมท็อด invoke ของคลาส C2 และจากแผนภาพคลาสเมท็อด invoke มีเมท็อดซิกเนเจอร์แสดงในตารางที่ 3-11

ตารางที่ 3-11 เมท็อดซิกเนเจอร์ของเมท็อด invoke ของคลาส C2

ชื่อเมท็อด		invoke
ระดับการมองเห็น		public
ประเภทของเมท็อด		CONCRETE
ประเภทของข้อมูลส่งออก		int
ลำดับพารามิเตอร์	**ชื่อพารามิเตอร์**	**ประเภทของข้อมูล**
1	input	string

2. สร้างรหัสต้นฉบับจากเมท็อดซิกเนเจอร์

ขั้นตอนการสร้างรหัสต้นฉบับของสตับจากเมท็อดซิกเนเจอร์ เป็นขั้นตอนการสร้างไฟล์รหัสต้นฉบับของสตับในรูปแบบจาวาคลาส (Java Class) ซึ่งมีเมท็อดที่มีซิกเนเจอร์เหมือนกับที่แสดงไว้ในแผนภาพคลาส แต่มีฟังก์ชันการทำงานเพียงสองอย่างได้แก่

1. แสดงค่าของพารามิเตอร์ผ่านคอนโซล (Console) ด้วยคำสั่ง System.out.println หากเมท็อดไม่มีพารามิเตอร์ เมท็อดในสตับจะไม่มีฟังก์ชันนี้ รูปที่ 3-7 แสดงตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่มีการแสดงค่าของพารามิเตอร์ชื่อ input ด้วยคำสั่ง System.out.println

```
public void printValue(String input){
    System.out.println(input);
}
```

รูปที่ 3-7 ตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่มีการแสดงค่าของพารามิเตอร์

2. การส่งค่ากลับตามประเภทของข้อมูลส่งออก ตัวสร้างจะส่งข้อมูลออกด้วยคำสั่ง return ตามประเภทของข้อมูลส่งออก โดยฟังก์ชันนี้จะมีรูปแบบการทำงานตามประเภทของข้อมูลส่งออกดังนี้

2.1 ส่งออกข้อมูลแบบสุ่ม ในกรณีที่ข้อมูลส่งออกมีประเภทของข้อมูลเป็นประเภทข้อมูลพื้นฐานเช่น int (จำนวนเต็ม), double (ทศนิยม) หรือเป็นข้อมูลประเภทสายอักขระ(string) ตัวสร้างจะสุ่มค่าตามประเภทของข้อมูลนั้นและส่งออกด้วยคำสั่ง return รูปที่ 3-8 แสดงตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่มีการส่งค่ากลับแบบสุ่มประเภทจำนวนเต็ม

```
public int returnRandomValue(){
    return 12345;
}
```

รูปที่ 3-8 ตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่มีการส่งออกค่าแบบสุ่ม

2.2 ส่งออกค่านัล (Null) ในกรณีที่ข้อมูลส่งออกประเภทอื่นนอกจากประเภทข้อมูลพื้นฐานและสายอักขระ หรือเป็นโครงสร้างข้อมูล (Data structure) เช่น อาเรย์ (Array) ตัวสร้างจะส่งข้อมูลออกเป็นค่านัลด้วยคำสั่ง return รูปที่ 3-9 แสดงตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่ส่งออกค่านัลเนื่องจากเมท็อดมีข้อมูลส่งออกเป็นอ็อบเจกต์ชื่อ Object1

```
public Object1 returnNullValue(){
    return null;
}
```


รูปที่ 3-9 ตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่มีการส่งออกค่านัล

2.3 ไม่ส่งข้อมูลออก ในกรณีที่ข้อมูลส่งออกเป็นวอยด์ (void) ซึ่งหมายความว่าเมท็อดนั้นไม่มีการส่งค่ากลับ เมท็อดในสตับจะไม่มีการส่งค่ากลับด้วยคำสั่ง return รูปที่ 3-10 แสดงตัวอย่างรหัสต้นฉบับของเมท็อดในสตับที่ไม่มีการส่งออกค่าใด ๆ

```
public void returnNothing(){

}
```

รูปที่ 3-10 ตัวอย่างรหัสต้นฉบับของเมท็อดในสตับไม่มีการส่งค่าออก

การสร้างรหัสต้นฉบับของสตับสามารถเกิดรูปแบบการสร้างได้ 3 รูปแบบได้แก่การสร้างสตับของคลาสรูปธรรม การสร้างสตับของคลาสนามธรรมหรืออินเตอร์เฟส และการสร้างสตับของคลาสภายใน โดยมีรายละเอียดของแต่ละรูปแบบดังต่อไปนี้

1. การสร้างสตับของคลาสรูปธรรม

การสร้างสตับของคลาสรูปธรรมเป็นรูปแบบการสร้างสตับโดยทั่วไป โดยตัวสร้างจะสร้างรหัสต้นฉบับของจาวาคลาสที่มีชื่อเหมือนกับคลาสต้นแบบแต่มีคำว่า Stub ต่อท้ายเช่น ถ้าสร้างสตับของคลาส A จาวาคลาสของสตับจะมีชื่อว่า AStub เพื่อสื่อว่าเป็นสตับของคลาส A และสร้างเมท็อดที่มีเมท็อดซิกเนเจอร์เหมือนกับที่ระบุไว้ในแผนภาพคลาส แต่จะมีฟังก์ชันการทำงานดังที่กล่าวไว้ข้างต้น ยกตัวอย่างการสร้างสตับรูปแบบนี้เช่น ผู้ทดสอบเลือกคลาส C1 ในกราฟการเรียกใช้งานรูปที่ 3-4 เป็นคลาสภายใต้การทดสอบ ซึ่งจากกราฟการใช้งานพบว่าต้องสร้างสตับของคลาส C2 เพื่อทดสอบคลาส C1 ตัวสร้างจะสร้างจาวาคลาสของสตับของคลาส C2 ชื่อว่า C2Stub แสดงในรูปที่ 3-11 รหัสต้นฉบับประกอบด้วยเมท็อดสองเมท็อดชื่อ invoke และ reInvoke ซึ่งมีฟังก์ชันแสดงค่าของพารามิเตอร์และการส่งค่ากลับด้วยค่าสุ่มตามประเภทของข้อมูลส่งออก

```
package stub.P3;
public class C2Stub{
    public int invoke(String input){
        System.out.println(input);
        return 952118730;
    }
    public int reInvoke(int processId){
        System.out.println(processId);
        return 414145724;
    }
}
```

รูปที่ 3-11 รหัสต้นฉบับภาษาจาวาของสตับของคลาส C2

ในกรณีเมท็อดมีการโอเวอร์โหลดดิ้งเช่นเมท็อด calculate ของคลาส A2 แสดงในแผนภาพคลาสรูปที่ 3-3 ตัวสร้างจะสร้างรหัสต้นฉบับของเมท็อดดังกล่าวทุกรูปแบบของการโอเวอร์โหลดดิ้ง เนื่องจากข้อมูลในแผนภาพลำดับไม่สามารถระบุได้ว่าเมสเสจเปรียบเทียบได้กับเมท็อดใดของการโอเวอร์โหลดดิ้ง รูปที่ 3-12 แสดงรหัสต้นฉบับของสตับของคลาส A2 เมื่อมีการเรียกใช้เมท็อด calculate

```java
package stub.P1;
public class A2Stub{
    public int calculate(int input1){
        System.out.println(input1);
        return 1664814506;
    }
    public int calculate(int input1,int input2){
        System.out.println(input1);
        System.out.println(input2);
        return 1573003440;
    }
    public int calculate(int input1,double input2){
        System.out.println(input1);
        System.out.println(input2);
        return 1735795417;
    }
}
```

รูปที่ 3-12 รหัสต้นฉบับของสตับของคลาส A2

อย่างไรก็ตามตัวสร้างจะสร้างรหัสต้นฉบับของเมท็อดเฉพาะเมท็อดที่ถูกเรียกในกราฟการเรียกใช้งานเท่านั้น เมท็อดอื่น ๆ ในแผนภาพคลาสจะยังไม่ถูกสร้างจนกว่าจะมีการทดสอบกราฟการเรียกใช้งานอื่น ๆ ที่มีการเรียกเมท็อดเหล่านั้นเพื่อให้สตับมีเฉพาะเมท็อดที่จำเป็นเท่านั้น

2. การสร้างสตับของคลาสนามธรรม

ในกรณีที่ต้องมีการสร้างสตับของคลาสที่เป็นคลาสนามธรรม ซึ่งคลาสนามธรรมจะไม่สามารถสร้างเป็นอ็อบเจกต์ได้โดยตรง ต้องทดสอบจากคลาสรูปธรรมที่สืบทอดมา [9] ดังนั้นตัวสร้างจะค้นหาคลาสรูปธรรมที่สืบทอดคลาสดังกล่าว และสร้างสตับของคลาสเหล่านั้นแทน ยกตัวอย่างการสร้างสตับรูปแบบนี้เช่น ผู้ทดสอบเลือกคลาส C2 ในกราฟการเรียกใช้งานรูปที่ 3-4 ซึ่งมีคลาส A3 เป็นหนึ่งในรายการของสตับที่ต้องสร้าง จากแผนภาพคลาสรูปที่ 3-3 คลาส A3 เป็นคลาสนามธรรมที่มีคลาส A4 และคลาส A5 สืบทอด ดังนั้นตัวสร้างจะสร้างรหัสต้นฉบับของสตับของคลาส A4 และ A5 แทนสตับของคลาส A3 รูปที่ 3-13 แสดงรหัสต้นฉบับของสตับของคลาส A4 และรูปที่ 3-14 รหัสต้นฉบับของคลาส A5

```
package stub.P1;
public class A4Stub{
    public int calculate(int input1,int input2){
        System.out.println(input1);
        System.out.println(input2);
        return 370929578;
    }
}
```

รูปที่ 3-13 รหัสต้นฉบับของสตับของคลาส A4

```
package stub.P1;
public class A5Stub{
    public int calculate(int input1,int input2){
        System.out.println(input1);
        System.out.println(input2);
        return 85220770;
    }
}
```

รูปที่ 3-14 รหัสต้นฉบับของสตับของคลาส A5

3. การสร้างสตับของคลาสภายใน

ในกรณีที่ต้องมีการสร้างสตับของคลาสภายใน ตัวสร้างจะสร้างสตับของคลาสนั้นเสมือนว่าเป็นคลาสรูปธรรม

หลังจากสร้างรหัสต้นฉบับของสตับเสร็จสิ้นตัวสร้างจะสร้างรหัสอ้างอิงไฟล์และบันทึกไฟล์รหัสต้นฉบับของสตับลงฐานข้อมูล

3.1.6 ขั้นตอนการสร้างรหัสต้นฉบับของไดร์เวอร์

เมื่อได้รายการของไดร์เวอร์ที่ต้องสร้าง ตัวสร้างจะสร้างรหัสต้นฉบับของไดร์เวอร์ของแต่ละคลาสเป็นภาษาจาวาโดยการสร้างไดร์เวอร์ประกอบไปด้วยขั้นตอนย่อย 2 ขั้นตอนโดยมีรายละเอียดดังต่อไปนี้

1. เปรียบเทียบเมสเสจกับเมท็อดในแผนภาพคลาส

ขั้นตอนการเปรียบเทียบเมสเสจกับเมท็อดในแผนภาพคลาส เป็นขั้นตอนการค้นหาเป็นการตรวจหาว่าเมสเสจนั้นเป็นเมท็อดใดของคลาสเพื่อนำเมท็อดนั้นไปค้นหาเมท็อดซิกเนเจอร์

2. สร้างรหัสต้นฉบับของไดร์เวอร์

ตัวสร้างจะสร้างรหัสต้นฉบับของไดร์เวอร์เป็นภาษาจาวา ในรูปแบบของ JUnit โดยการสร้างไดร์เวอร์สามารถแบ่งออกได้เป็น 4 กรณีโดยมีรายละเอียดดังต่อไปนี้

1. การสร้างไดร์เวอร์เพื่อเรียกเมท็อดประเภท CONCRETE

การสร้างไดร์เวอร์เพื่อเรียกเมท็อดประเภท CONCRETE เป็นการสร้างไดร์เวอร์เพื่อเรียกใช้งานคลาสภายใต้การทดสอบที่เป็นคลาสรูปธรรม และเมท็อดเป็นเมท็อดประเภท CONCRETE เช่นเมท็อด invoke ของคลาส C2 แสดงในตารางที่ 3-11 ตัวอย่างรหัสต้นฉบับของไดร์เวอร์ที่ใช้ทดสอบคลาส C2 แสดงในรูปที่ 3-15 ซึ่งเป็นไดร์เวอร์ของคลาส C1 ที่เรียกคลาส C2 ตามกราฟการเรียกใช้งานรูปที่ 3-4 โดยรหัสต้นฉบับมีรายละเอียดดังต่อไปนี้

```
1  package driver.P3;
2  /*--- AUTO IMPORT START HERE ---*/
3  import P3.C2;
4  /*--- AUTO IMPORT END HERE ---*/
5  public class C1Driver{
6      @Test
7      public void testInvokeInC2(){
8          C2 c2 = new C2();
9          int actualResult = c2.invoke("TVe");
10         assertEquals(1203664961,actualResult);
11     }
12     @Test
13     public void testReInvokeInC2(){
14         C2 c2 = new C2();
15         int actualResult = c2.reInvoke(942724724);
16         assertEquals(1454521054,actualResult);
17     }
18 }
```

รูปที่ 3-15 ตัวอย่างรหัสต้นฉบับภาษาจาวาของไดร์เวอร์ตัวแทนคลาส C1

- บรรทัดที่ 1 เป็นการประกาศแพ็คเกจของไดร์เวอร์
- บรรทัดที่ 3 เป็นการนำเข้าเนมสเปซคลาส C2 เพื่อให้ JUnit สามารถเรียกใช้งานคลาส C2 ได้

- บรรทัดที่ 5 เป็นการประกาศชื่อจาวาคลาสของไดร์เวอร์ โดยสร้างจากชื่อของคลาสที่ไดร์เวอร์เป็นตัวแทนตามด้วยคำว่า Driver เพื่อสื่อว่าไดร์เวอร์นี้เป็นตัวแทนของคลาสดังกล่าว

- บรรทัดที่ 7 เป็นการประกาศเมท็อดชื่อว่า testInvokeInC2 ซึ่งเป็นเมท็อดที่ทำหน้าที่เรียกใช้งานเมท็อด invoke ในคลาส C2

- บรรทัดที่ 8 เป็นการสร้างอ็อบเจกต์ของคลาส C2 ชื่อ c2 เนื่องจากเมท็อดซิกเนเจอร์ของเมท็อด invoke แสดงในตารางที่ 3-11 เป็นเมท็อดประเภทประเภท CONCRETE ซึ่งไม่สามารถเรียกใช้งานโดยตรงจากคลาสได้ ต้องเรียกใช้จากอ็อบเจกต์ที่สร้างจากคลาสนั้น

- บรรทัดที่ 9 เป็นการเรียกใช้งานเมท็อด invoke ของคลาส C2 ผ่านอ็อบเจกต์ c2 โดยมีการประกาศตัวแปรชื่อ actualResult มารับค่าที่ได้จากการเรียกเมท็อดดังกล่าว โดยค่าอาร์กิวเมนต์ที่ส่งให้เมท็อด invoke เป็นค่าสุ่มโดยจะกล่าวรายละเอียดในหัวข้อที่ 3.1.7

- บรรทัดที่ 10 เป็นการเปรียบเทียบผลลัพท์ของการเรียกเมท็อด invoke ด้วยฟังก์ชัน assertEqual ซึ่งเป็นฟังก์ชันหนึ่งของ JUnit โดยจะกล่าวรายละเอียดในหัวข้อที่ 3.1.7

- บรรทัดที่ 13-16 เป็นเมท็อดที่ใช้เรียกใช้งานเมท็อด reinvoke ของคลาส C2

2. การสร้างไดร์เวอร์เพื่อเรียกเมท็อดประเภท STATIC

การสร้างไดร์เวอร์เพื่อเรียกเมท็อดประเภท STATIC เป็นการสร้างไดร์เวอร์เพื่อเรียกใช้งานคลาสภายใต้การทดสอบที่เป็นคลาสรูปธรรมซึ่งเมท็อดเป็นเมท็อดเชิงสถิต ซึ่งเมท็อดเชิงสถิตสามารถเรียกใช้งานได้จากคลาสโดยตรง โดยไม่ต้องสร้างอ็อบเจกต์จากคลาสนั้นก่อน ดังนั้นไดร์เวอร์ที่เรียกใช้งานเมท็อดรูปแบบนี้จะไม่มีการประกาศการสร้างอ็อบเจกต์และเรียกใช้เมท็อดนั้นโดยตรง ตัวอย่างเช่น การสร้างไดร์เวอร์เพื่อเรียกเมท็อด start ของคลาส C4 ในแผนภาพลำดับรูปที่ 3-3 ซึ่งเมท็อด start เป็นเมท็อดเชิงสถิตทำให้สามารถเรียกเมท็อด start ได้จากคลาส C4 โดยตรงโดยไม่ต้องสร้างอ็อบเจกต์ของคลาส C4 ก่อน รูปที่ 3-16 แสดงตัวอย่างของรหัสต้นฉบับบางส่วนของไดร์เวอร์ที่ใช้เรียกเมท็อด start ของคลาส C4

```
@Test
public void testStartInC4(){
    double actualResult = C4.start();
    assertEquals(849742164.616,actualResult,3165);
}
```

รูปที่ 3-16 ตัวอย่างรหัสต้นฉบับบางส่วนของไดร์เวอร์ที่เรียกใช้งานเมท็อดเชิงสถิต

3. การสร้างไดร์เวอร์เพื่อเรียกใช้งานคลาสนามธรรมหรืออินเตอร์เฟส

ในกรณีที่ต้องสร้างไดร์เวอร์เพื่อเรียกคลาสภายใต้การทดสอบที่เป็นคลาสนามธรรมหรืออินเตอร์เฟส ตัวสร้างจะไม่เรียกใช้งานคลาสดังกล่าวโดยตรง แต่จะเรียกคลาสรูปธรรมที่สืบทอดคลาสดังกล่าวแทน แสดงตัวอย่างเช่น การสร้างไดร์เวอร์เพื่อทดสอบคลาส A3 แสดงในกราฟการเรียกใช้งานรูปที่ 3-4 ไดร์เวอร์ที่เป็นตัวแทนคลาส C2 จะเรียกใช้งานคลาส A4 และ A5 ซึ่งสืบทอดมาจากคลาส A3 (แสดงในแผนภาพคลาสรูปที่ 3-3) แทนคลาส A3 รูปที่ 3-17 แสดงรหัสต้นฉบับของไดร์เวอร์ตัวแทนคลาส C2 ที่เรียกใช้งานคลาส A4 และ A5 แทนคลาส A3

```
package driver.P3;
/*--- AUTO IMPORT START HERE ---*/
import P1.A4;
import P1.A5;
/*--- AUTO IMPORT END HERE ---*/
public class C2Driver{
    @Test
    public void testCalculateInA4(){
        A4 a4 = new A4();
        int actualResult = a4.calculate(184160841,635143819);
        assertEquals(1259243256,actualResult);
    }
    @Test
    public void testCalculateInA5(){
        A5 a5 = new A5();
        int actualResult = a5.calculate(531914021,1591676053);
        assertEquals(849742164,actualResult);
    }
}
```

รูปที่ 3-17 รหัสต้นฉบับของไดร์เวอร์ตัวแทนคลาส C2 สำหรับเรียกใช้งานคลาส A3

4. การสร้างไดร์เวอร์เพื่อเป็นตัวแทนคลาสนามธรรมหรืออินเตอร์เฟส

ในกรณีที่ต้องสร้างไดร์เวอร์เพื่อเป็นตัวแทนของคลาสนามธรรมหรืออินเตอร์เฟส ตัวสร้างจะไม่สร้างไดร์เวอร์ที่เป็นตัวแทนของคลาสนั้นโดยตรง ตัวสร้างจะสร้างไดร์-เวอร์ตัวแทนจากคลาสรูปธรรมที่สืบทอดคลาสนั้นมาเพื่อเป็นตัวแทนของคลาสนั้นเช่น จากรูปที่ 3-4 หากต้องสร้างไดร์เวอร์ที่เป็นตัวแทนของคลาส A3 เพื่อทดสอบคลาส D1 ตัวสร้างจะสร้างไดร์เวอร์ตัวแทนจากคลาส A4 และ A5 เพื่อเรียกเมท็อด process ในคลาส D1

หลังจากสร้างรหัสต้นฉบับของไดร์เวอร์เสร็จสิ้นตัวสร้างจะสร้างรหัสอ้างอิงไฟล์และบันทึกไฟล์รหัสต้นฉบับของไดร์เวอร์ลงฐานข้อมูล

3.1.7 ขั้นตอนการสร้างข้อมูลทดสอบ

ขั้นตอนการสร้างข้อมูลทดสอบเป็นขั้นตอนในการสร้างค่าสุ่มในไดร์เวอร์สำหรับใช้เป็นค่าอาร์กิวเมนต์ที่ส่งให้เมท็อดของคลาสภายใต้การทดสอบและค่าของผลลัพธ์ที่คาดหวังที่ได้จากการเรียกใช้งานเมท็อด ซึ่งจะทำให้ไดร์เวอร์มีคุณสมบัติเหมือนกรณีทดสอบกล่าวคือ มีข้อมูลนำเข้า มีค่าของผลลัพธ์ที่คาดหวัง และมีการเปรียบเทียบผลลัพธ์ที่เกิดขึ้นจริงกับผลลัพธ์ที่คาดหวัง [1] โดยการสร้างค่าสุ่มแบ่งออกเป็น 3 กรณีได้แก่

1. การสร้างค่าสุ่มแบบทั่วไป

การสุ่มแบบทั่วไปเป็นการสุ่มค่าตามประเภทของข้อมูล โดยข้อมูลต้องเป็นข้อมูลประเภทพื้นฐานเช่น จำนวนเต็ม หรือเป็นข้อมูลประเภทสายอักขระ แสดงตัวอย่างในรูปที่ 3-15 ในบรรทัดที่ 9 เมท็อดชื่อ invoke มีพารามิเตอร์ชื่อ input ซึ่งเป็นข้อมูลประเภท string หรือสายอักขระ (แสดงในตารางที่ 3-11) ตัวสร้างจะสุ่มค่าของสายอักขระรวมถึงความยาวของสายอักขระ เพื่อเป็นค่าอาร์กิวเมนต์ของเมท็อดนี้

จากรูปที่ 3-15 ในบรรทัดที่ 10 เป็นการเปรียบเทียบผลลัพธ์ที่คาดหวังซึ่งได้จากการสุ่มและผลลัพธ์ของการเรียกใช้งานเมท็อด invoke ด้วยฟังก์ชัน assertEqual ของ JUnit เมท็อด invoke ส่งค่ากลับเป็นจำนวนเต็ม (แสดงในตารางที่ 3-11) ดังนั้นตัวสร้างจึงสร้างค่าสุ่มเป็นจำนวนเต็ม

2. การสร้างค่าสุ่มตามการ์ดคอนดิชัน

ในกรณีที่เมสเสจที่เรียกคลาสภายใต้การทดสอบถูกกำกับด้วยการ์ดคอนดิชันดังแสดงในแผนภาพลำดับรูปที่ 3-2 เมสเสจชื่อ reinvoke ถูกกำกับด้วยการ์ดคอนดิชัน processId > 100 หมายความว่าเมสเสจชื่อ reinvoke จะถูกส่งเมื่อ processId มีค่ามากกว่า 100 ดังนั้นตัวสร้างจะสุ่มค่าของ processId ให้มากกว่า 100 ดังแสดงในรูปที่ 3-15 บรรทัดที่ 15

3. การสร้างค่านัลแทนค่าสุ่ม

ในกรณีที่พารามิเตอร์ของเมท็อดหรือข้อมูลส่งออกเป็นโครงสร้างข้อมูลหรือเป็นข้อมูลประเภทอื่น ๆ นอกจากข้อมูลพื้นฐานหรือสายอักขระ ตัวสร้างจะใช้ค่านัลแทนค่าสุ่ม

3.1.8 ขั้นตอนการส่งออกไฟล์รหัสต้นฉบับของสตับและไดร์เวอร์

หลังจากตัวสร้างสร้างไฟล์รหัสต้นฉบับของสตับและไดร์เวอร์ที่มีข้อมูลทดสอบเสร็จสิ้น และบันทึกไฟล์ลงฐานข้อมูล ตัวสร้างจะส่งออกไฟล์รหัสต้นฉบับเพื่อให้ผู้ทดสอบนำไฟล์รหัสต้นฉบับไปใช้ทดสอบรหัสต้นฉบับของคลาสภายใต้การทดสอบ

บทที่ 4
การออกแบบและพัฒนาตัวสร้าง

ในบทนี้ผู้วิจัยได้นำแนวคิดของตัวสร้างสตับและไดร์เวอร์ดังที่อธิบายในบทที่ 3 มาพัฒนาเป็นตัวสร้าง สตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาส โดยตัวสร้างถูกออกแบบด้วยการโปรแกรมเชิงวัตถุ โดยใช้ยูเอ็มแอลเพื่ออธิบายฟังก์ชัน โครงสร้าง และพฤติกรรมของตัวสร้าง นอกจากนี้ผู้วิจัยได้กำหนดสภาพแวดล้อมที่ใช้ในการพัฒนาตัวสร้างและออกแบบโครงสร้างของส่วนต่อประสานผู้ใช้ ซึ่งมีรายละเอียดดังต่อไปนี้

4.1 การออกแบบตัวสร้าง

ในหัวข้อนี้จะกล่าวถึงการออกแบบตัวสร้างสตับและไดร์เวอร์ประกอบด้วยแบบจำลองการวิเคราะห์ระบบได้แก่ แบบจำลองเชิงฟังก์ชัน (Functional Model) แบบจำลองเชิงโครงสร้าง (Structural Model) และแบบจำลองเชิงพฤติกรรม (Behavioral Model) รวมทั้งการออกแบบโครงสร้างงานข้อมูล และสถาปัตยกรรมของตัวสร้างโดยมีรายละเอียดดังต่อไปนี้

4.1.1 แบบจำลองเชิงฟังก์ชัน

สำหรับแบบจำลองเชิงฟังก์ชัน ผู้วิจัยได้เลือกใช้แผนภาพยูสเคส (Use Case Diagram) และแผนภาพกิจกรรม (Activity Diagram) ในการอธิบายฟังก์ชันของตัวสร้างสตับและไดร์เวอร์โดยมีรายละเอียดดังต่อไปนี้

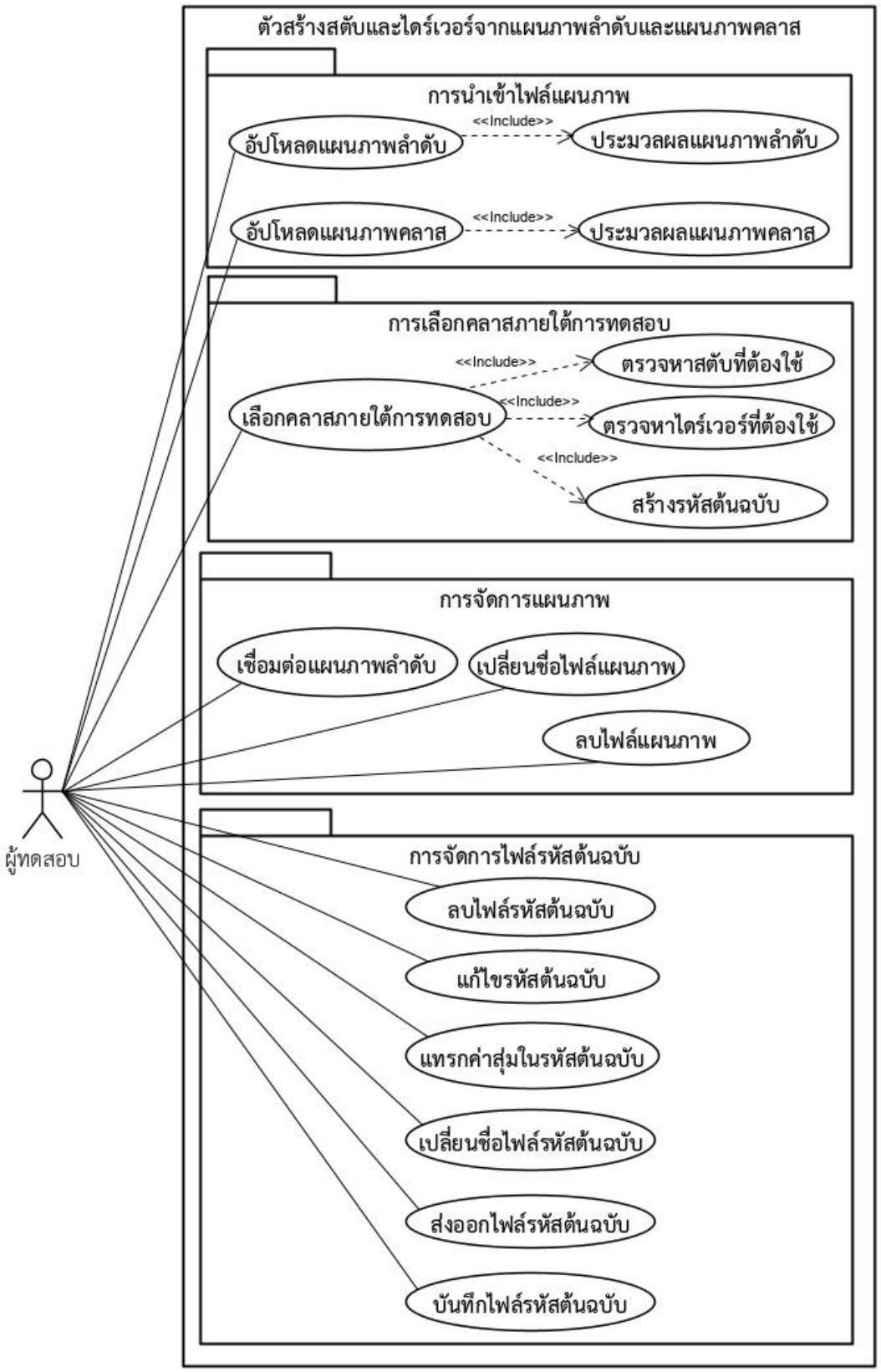
4.1.1.1 แผนภาพยูสเคส

แผนภาพยูสเคสของตัวสร้าง แสดงในรูปที่ 4-1 โดยยูสเคส (Use case) ถูกแบ่งออกเป็น 4 แพ็คเกจตามฟังก์ชันการใช้งานดังต่อไปนี้

1. การนำเข้าไฟล์แผนภาพ ประกอบด้วยการนำเข้าไฟล์แผนภาพลำดับและการนำเข้าแผนภาพคลาส รวมถึงการประมวลผลแผนภาพลำดับเพื่อสร้างเป็นกราฟการเรียกใช้งาน และการประมวลผลแผนภาพคลาสเพื่อรวบรวมข้อมูลสำหรับการสร้างรหัสต้นฉบับ

2. การเลือกคลาสภายใต้การทดสอบ ประกอบด้วยยูสเคสที่ผู้ทดสอบเลือกคลาสภายใต้การทดสอบ การตรวจหาสตับและไดร์เวอร์ที่ต้องใช้ และการสร้างรหัสต้นฉบับของสตับและไดร์เวอร์

3. การจัดการแผนภาพ ประกอบด้วยการเชื่อมต่อแผนภาพลำดับที่มีการอ้างอิงแผนภาพลำดับอื่นเข้ากันกับแผนภาพลำดับที่ถูกอ้างอิง และการจัดการแผนภาพลำดับและแผนภาพคลาสประกอบด้วย การลบแผนภาพออกจากตัวสร้างและการเปลี่ยนชื่อแผนภาพ



รูปที่ 4-1 แผนภาพยูสเคสของตัวสร้างสตับและไดร์เวอร์

4. การจัดการรหัสต้นฉบับ ประกอบด้วยยูสเคสที่เกี่ยวข้องกับการแก้ไขไฟล์ การแทรกค่าสุ่มลงในรหัสต้นฉบับ การบันทึกความเปลี่ยนแปลงจากการแก้ไขไฟล์รหัสต้นฉบับ และการจัดการไฟล์รหัสต้นฉบับประกอบด้วย การเปลี่ยนชื่อไฟล์ การลบไฟล์ออกจากตัวสร้าง และการส่งออกไฟล์

สำหรับคำอธิบายยูสเคส (Use case description) ของแต่ละยูสเคสถูกแสดงไว้ในภาคผนวก ก.

4.1.1.2 แผนภาพกิจกรรม

แผนภาพกิจกรรมหลักของตัวสร้างสตับและไดร์เวอร์ประกอบด้วยแผนภาพกิจกรรม 4 แผนภาพซึ่งมีรายละเอียดดังต่อไปนี้

1. แผนภาพกิจกรรมของการนำเข้าไฟล์แผนภาพลำดับและการประมวลผลแผนภาพลำดับ

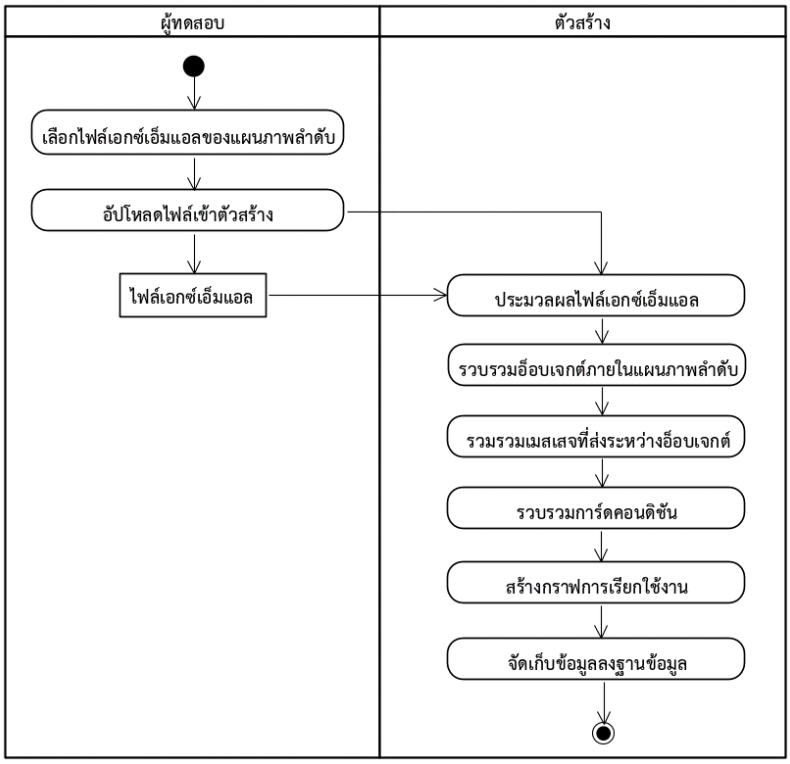
แผนภาพกิจกรรมของการนำเข้าไฟล์แผนภาพลำดับและการประมวลผลแผนภาพลำดับแสดงในรูปที่ 4-2 โดยเริ่มต้นจากผู้ทดสอบเลือกไฟล์แผนภาพลำดับในรูปแบบไฟล์เอกซ์เอ็มแอลและ อัปโหลดไฟล์เข้าตัวสร้าง จากนั้นตัวสร้างจะประมวลผลไฟล์เอกซ์เอ็มแอล เพื่อรวบรวมอ็อบเจกต์ เมสเสจที่ส่งระหว่างอ็อบเจกต์ และการ์ดคอนดิชันภายในแผนภาพ เพื่อสร้างเป็นกราฟการเรียกใช้งาน และจัดเก็บกราฟการเรียกใช้งานลงฐานข้อมูล

2. แผนภาพกิจกรรมการนำเข้าแผนภาพคลาสและการประมวลผลแผนภาพคลาส

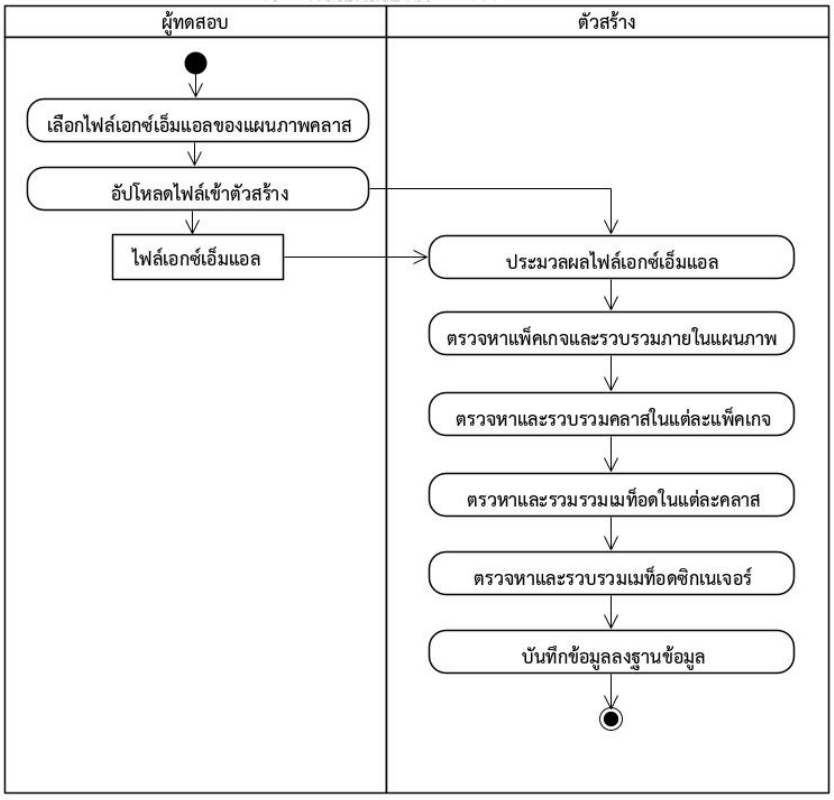
แผนภาพกิจกรรมการนำเข้าแผนภาพคลาสและการประมวลผลแผนภาพคลาส แสดงในรูปที่ 4-3 โดยเริ่มต้นจากผู้ทดสอบเลือกไฟล์แผนภาพคลาสในรูปแบบไฟล์เอกซ์เอ็มแอลและอัปโหลดไฟล์เข้าตัวสร้าง จากนั้นตัวสร้างจะประมวลผลไฟล์เอกซ์เอ็มแอล เพื่อรวบรวมแพ็คแกจในแผนภาพคลาส รวบรวมคลาสในแต่ละแพ็คเกจ รวบรวมเมท็อดในแต่ละคลาส รวบรวมเมท็อดซิกเนเจอร์ของแต่ละเมท็อด และจัดเก็บข้อมูลแผนภาพคลาสลงฐานข้อมูล

3. แผนภาพกิจกรรมของการเลือกคลาสภายใต้การทดสอบ

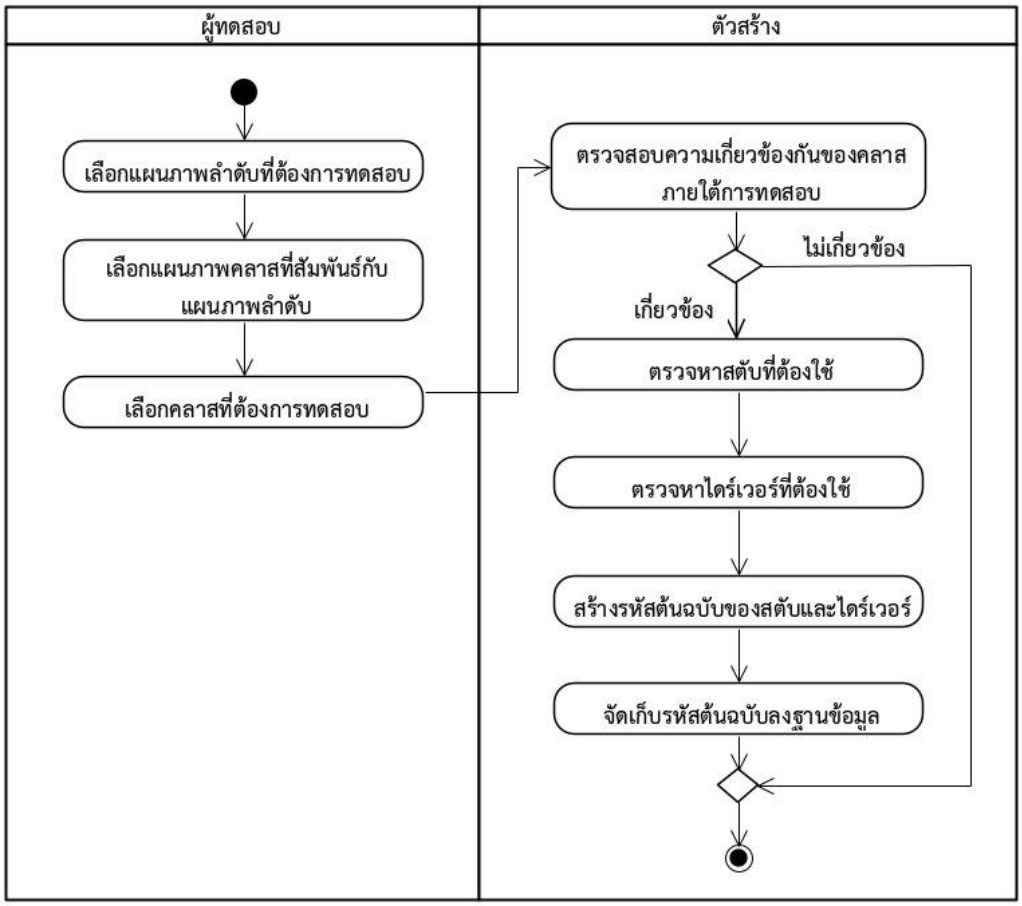
แผนภาพกิจกรรมของการเลือกคลาสภายใต้การทดสอบครอบคลุมตั้งแต่การเลือกแผนภาพลำดับที่ต้องการทดสอบ การเลือกแผนภาพคลาสที่สอดคล้องกับแผนภาพลำดับที่ต้องการทดสอบ การเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบ การตรวจหาสตับและไดร์เวอร์ที่ต้องใช้ในการทดสอบคลาสภายใต้การทดสอบ และการสร้างรหัสต้นฉบับของสตับและไดร์เวอร์ดังกล่าว โดยแผนภาพกิจกรรมแสดงในรูปที่ 4-4



รูปที่ 4-2 แผนภาพกิจกรรมการอัปโหลดไฟล์แผนภาพลำดับและการประมวลผลแผนภาพลำดับ



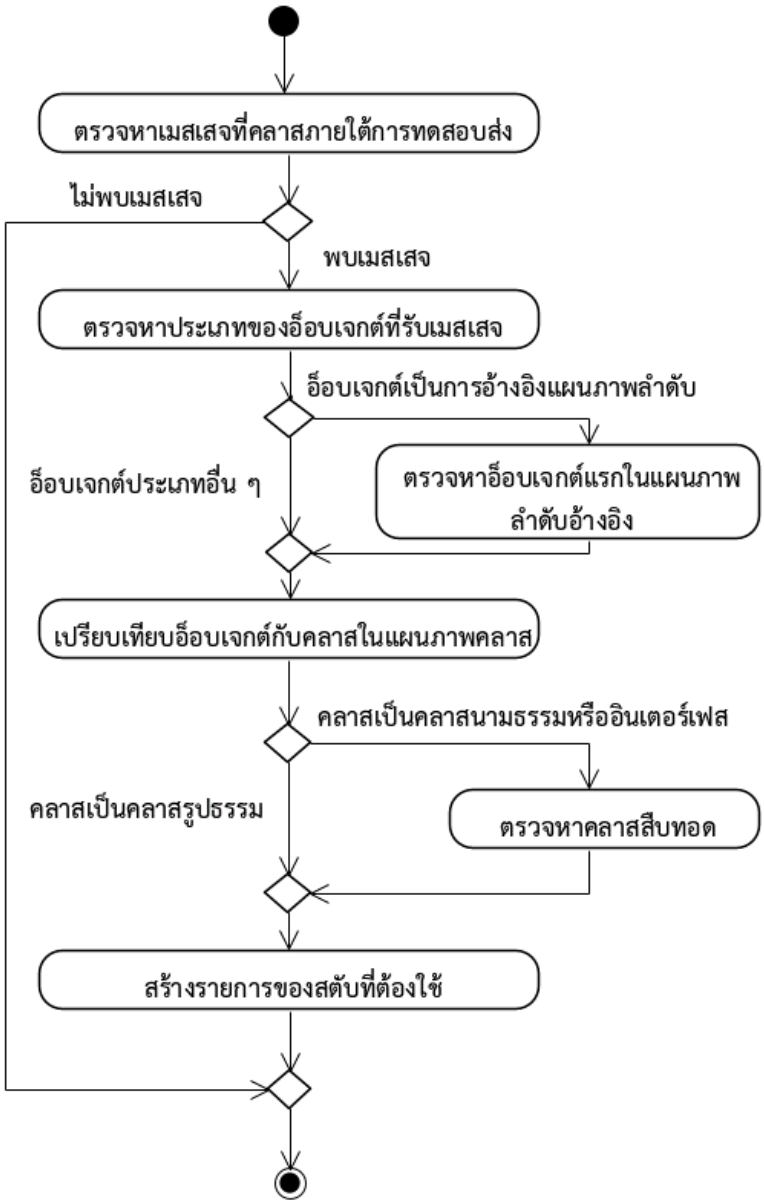
รูปที่ 4-3 แผนภาพกิจกรรมการอัปโหลดไฟล์แผนภาพคลาสและการประมวลผลแผนภาพคลาส



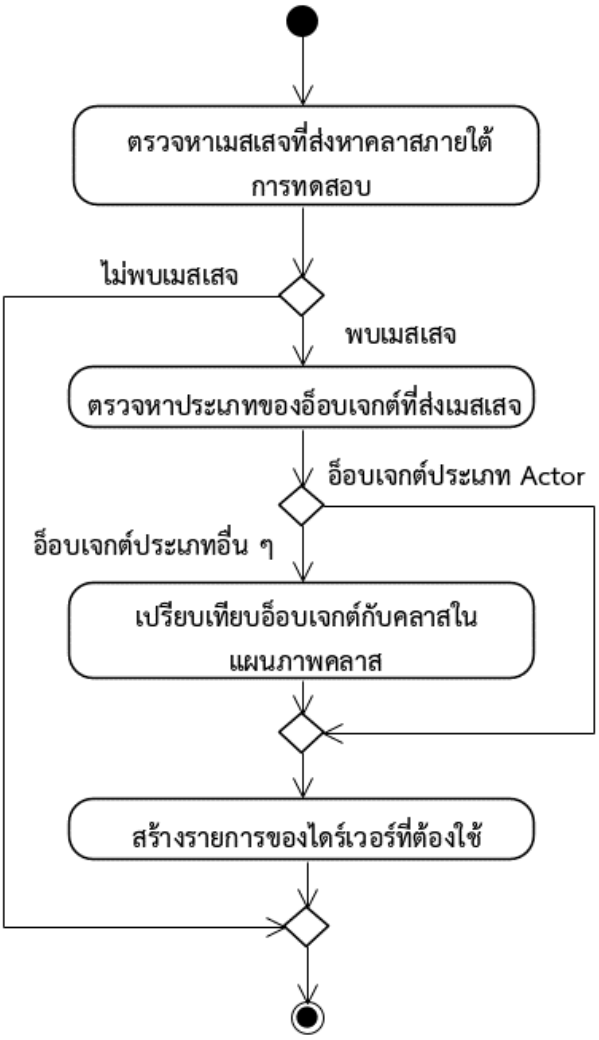
รูปที่ 4-4 แผนภาพกิจกรรมของการเลือกคลาสภายใต้การทดสอบ

เริ่มต้นผู้ทดสอบจะเลือกแผนภาพลำดับที่ต้องการทดสอบ และเลือกแผนภาพคลาสที่สอดคล้องกับแผนภาพลำดับที่เลือก จากนั้นผู้ทดสอบจะเลือกคลาสหรือกลุ่มของภายใต้การทดสอบจากแผนภาพลำดับดังกล่าว และตัวสร้างจะวิเคราะห์คลาสหรือกลุ่มของคลาสภายใต้การทดสอบว่ามีความสัมพันธ์กันหรือไม่ โดยเกณฑ์การพิจารณาได้กล่าวไว้ในบทที่ 3 หัวข้อที่ 3.1.4 หากคลาสภายใต้การทดสอบมีความสัมพันธ์กัน ตัวสร้างจะตรวจหาสตับที่ต้องใช้ในการทดสอบคลาสภายใต้การทดสอบดังกล่าวด้วยการค้นหาเมสเสจที่คลาสภายใต้การทดสอบส่งหาอ็อบเจกต์ อื่นจากกราฟการเรียกใช้งาน จากนั้นตรวจสอบประเภทของอ็อบเจกต์ ว่าเป็นการอ้างอิงแผนภาพลำดับอื่น ๆ หรือไม่ ถ้าใช่ ตัวสร้างจะวิเคราะห์แผนภาพลำดับที่ถูกอ้างอิงว่าอ็อบเจกต์แรกของแผนภาพลำดับปลายทางเป็นอ็อบเจกต์ใด จากนั้นนำอ็อบเจกต์ ไปเทียบกับคลาสแผนภาพคลาสว่าเป็นคลาสใดเพื่อที่จะสร้างสตับที่เป็นตัวแทนของคลาสนั้น ในกรณีที่พบว่าคลาสในแผนภาพคลาสเป็นคลาสนามธรรมหรืออินเตอร์เฟส ตัวสร้างจะค้นหาต่อว่าคลาสรูปธรรมที่สืบทอดคลาสดังกล่าวเป็นคลาสใดและสร้างสตับเป็นตัวแทนของคลาสเหล่านั้นแทน โดยแผนภาพกิจกรรมของการตรวจหาสตับที่ต้องใช้

แสดงในรูปที่ 4-5 เมื่อได้รายการของสตับที่ต้องใช้แล้ว ตัวสร้างตรวจหาไดร์เวอร์ที่ต้องใช้โดยการวิเคราะห์กราฟการเรียกใช้งานว่าคลาสภายใต้การทดสอบถูกอ็อบเจกต์ใดส่งเมสเสจหาและตรวจสอบว่าอ็อบเจกต์ดังกล่าว เป็นอ็อบเจกต์ประเภทใด หากไม่เป็นอ็อบเจกต์ประเภท Actor ตัวสร้างจะนำอ็อบเจกต์ ดังกล่าวไปค้นหาในแผนภาพคลาสว่าเป็นคลาสใดและสร้างไดร์เวอร์ที่จำลองการเรียกใช้คลาสภายใต้การทดสอบจากคลาสดังกล่าว โดยแผนภาพกิจกรรมของการตรวจหาไดร์เวอร์แสดงในรูปที่ 4-6 เมื่อได้รายการของสตับและไดร์เวอร์แล้ว ตัวสร้างจะสร้างรหัสต้นฉบับของ สตับและไดร์-เวอร์ดังกล่าวจากนั้นตัวสร้างจะบันทึกไฟล์ของรหัสต้นฉบับลงฐานข้อมูล



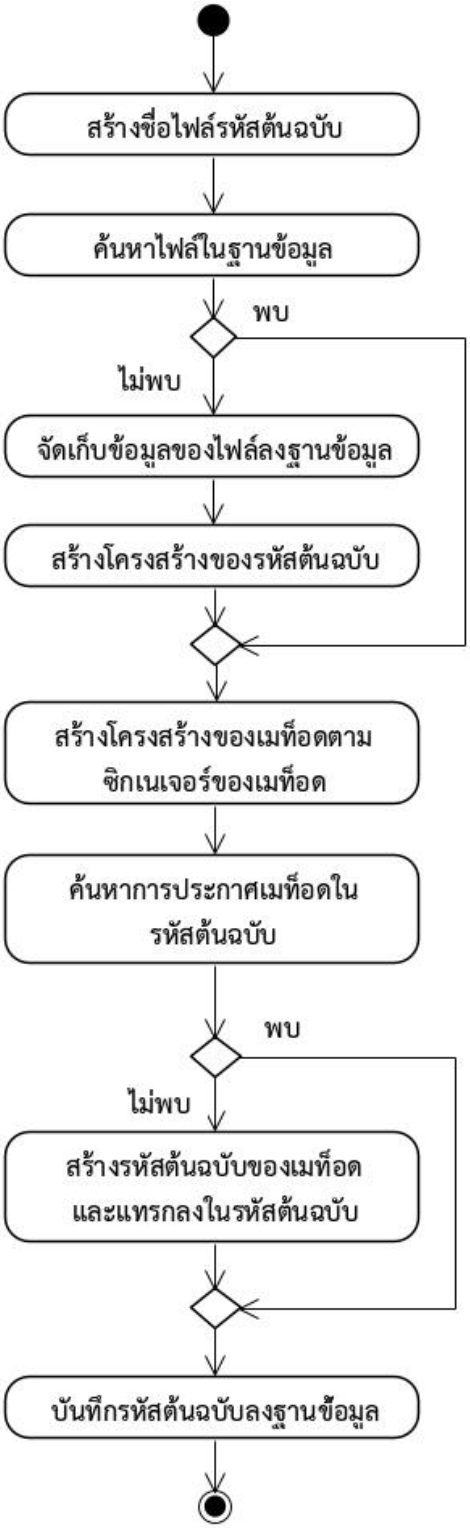
รูปที่ 4-5 แผนภาพกิจกรรมของการตรวจหาสตับที่ต้องใช้



รูปที่ 4-6 แผนภาพกิจกรรมของการตรวจหาไดร์เวอร์ที่ต้องใช้

4. แผนภาพกิจกรรมของการสร้างรหัสต้นฉบับของสตับและไดร์เวอร์

เมื่อผู้ทดสอบเลือกคลาสภายใต้การทดสอบ ตัวสร้างตรวจหารายการของสตับและไดร์เวอร์ที่ต้องใช้แล้ว ตัวสร้างจะสร้างรหัสต้นฉบับของสตับและไดร์เวอร์จากรายการดังกล่าว แผนภาพกิจกรรมของการสร้างรหัสต้นฉบับแสดงในรูปที่ 4-7 โดยตัวสร้างจะค้นหาในฐานข้อมูลว่าเคยสร้างรหัสต้นฉบับของสตับและไดร์เวอร์ดังกล่าวมาก่อนหรือไม่ หากไม่เคยสร้าง ตัวสร้างจะสร้างไฟล์ใหม่ขึ้นมาและบันทึกลงฐานข้อมูล หากมีไฟล์อยู่แล้วจะข้ามขั้นตอนดังกล่าวไป จากนั้นตัวสร้างจะสร้างโครงสร้างของเมท็อดตามเมท็อดซิกเนเจอร์ตามที่ระบุไว้ในแผนภาพคลาสและตรวจสอบว่ามีเมท็อดดังกล่าวอยู่ในไฟล์อยู่แล้วหรือไม่ ถ้าไม่มีเมท็อดดังกล่าว ตัวสร้างจะแทรกเมท็อดดังกล่าวลงไฟล์และสร้างเนื้อหาของเมท็อดขึ้นมาก่อนบันทึกไฟล์ลงฐานข้อมูล



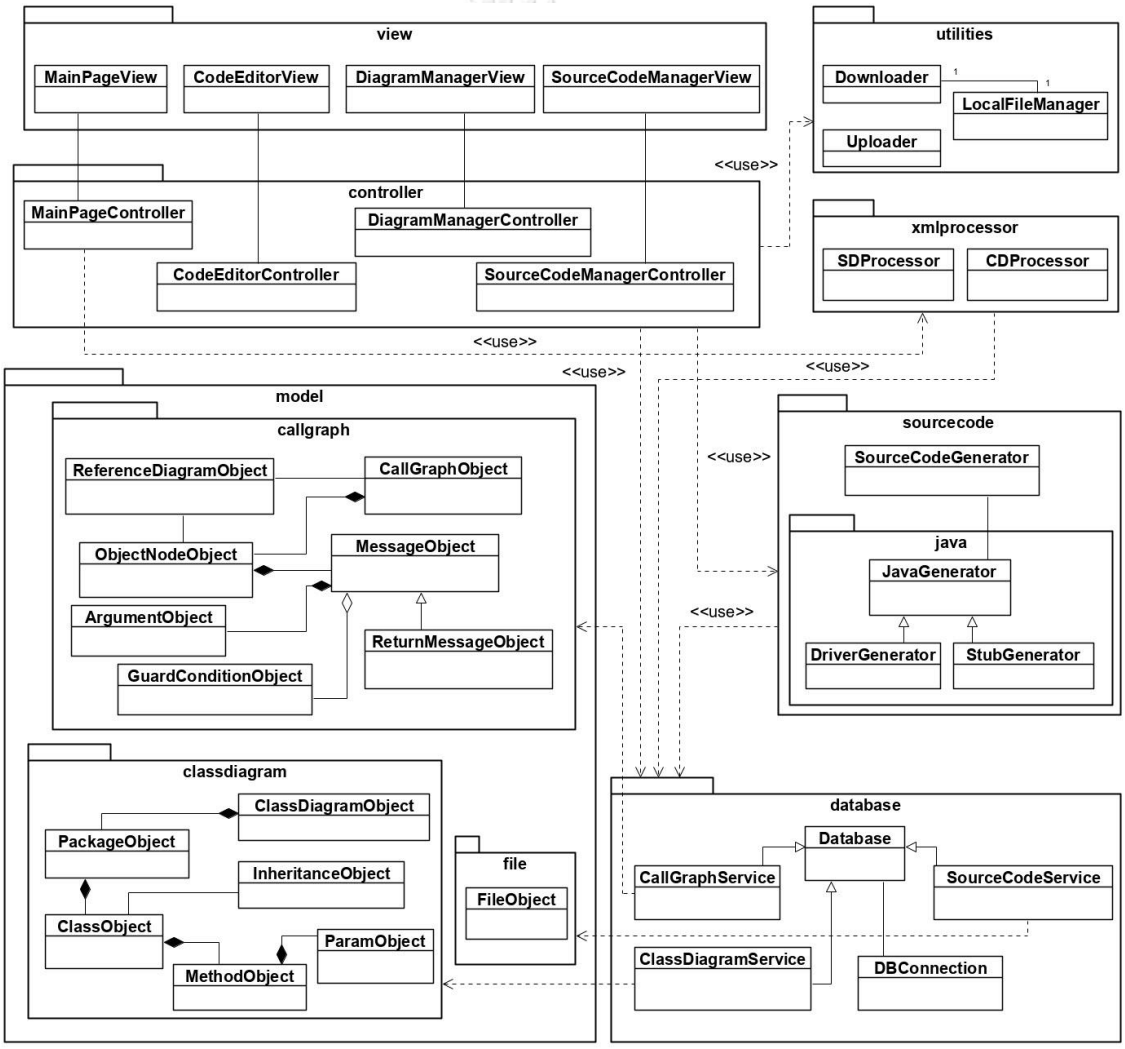
รูปที่ 4-7 แผนภาพกิจกรรมของการสร้างรหัสต้นฉบับ

4.1.2 แบบจำลองเชิงโครงสร้าง

สำหรับแบบจำลองเชิงโครงสร้างผู้วิจัยได้ใช้แผนภาพคลาสในการอธิบายโครงสร้างของตัวสร้างสตับและไดร์เวอร์ โดยมีรายละเอียดดังต่อไปนี้

4.1.2.1 แผนภาพคลาส

แผนภาพคลาสของตัวสร้างสตับและไดร์เวอร์แสดงในรูปที่ 4-8 โดยแผนภาพลำดับดังกล่าวออกแบบด้วยรูปแบบเอ็มวีซี (MVC – Model-View-Controller Pattern) [17] และประกอบด้วย 7 แพ็คเกจซึ่งมีรายละเอียดดังต่อไปนี้

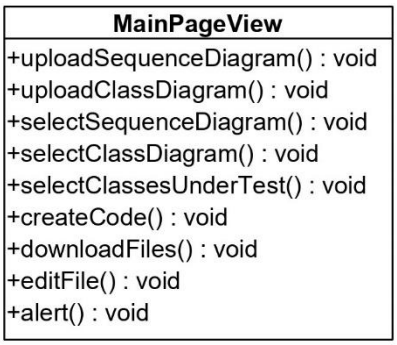


รูปที่ 4-8 แผนภาพคลาสของตัวสร้างสตับและไดร์เวอร์

1. แพ็คเกจ view

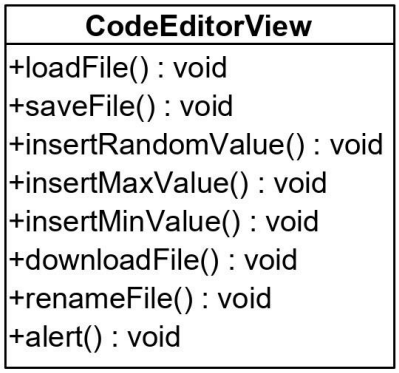
แพ็คเกจ view ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มวิว (View) ตามรูปแบบเอ็มวีซี รายละเอียดของแต่ละคลาสมีดังต่อไปนี้

1.1 คลาส MainPageView เป็นคลาสส่วนต่อประสานผู้ใช้ของหน้าต่างหลักของตัวสร้างซึ่งรองรับการอัปโหลดไฟล์แผนภาพและการเลือกคลาสภายใต้การทดสอบ นอกจากนี้คลาสนี้ยังทำหน้าที่นำทางไปหน้าต่างส่วนต่อประสานผู้ใช้อื่น ๆ ด้วย รูปที่ 4-9 แสดงรายละเอียดของคลาส MainPageView



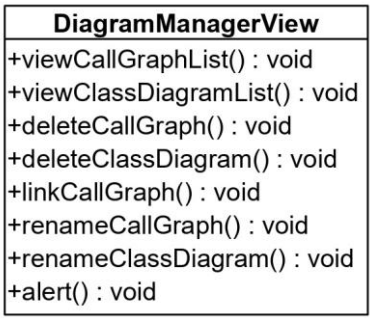
รูปที่ 4-9 คลาส MainPageView

1.2 คลาส CodeEditorView เป็นคลาสส่วนต่อประสานผู้ใช้ของหน้าต่างการแก้ไขรหัสต้นฉบับ ซึ่งมีฟังก์ชันการเปลี่ยนชื่อไฟล์ การบันทึกไฟล์ การส่งออกไฟล์ การแทรกค่าสุ่ม การแทรกค่าสูงสุดและค่าต่ำสุดตามประเภทของข้อมูล รูปที่ 4-10 แสดงรายละเอียดของคลาส CodeEditiorView



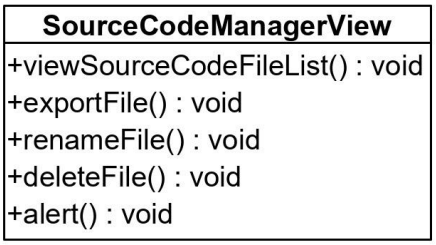
รูปที่ 4-10 คลาส CodeEditorView

1.3 คลาส DiagramManagerView เป็นคลาสส่วนต่อประสานผู้ใช้ของหน้าต่างการจัดการแผนภาพได้แก่ การลบแผนภาพ การเปลี่ยนชื่อแผนภาพ รวมถึงการเชื่อมต่อแผนภาพลำดับที่มีการอ้างอิงแผนภาพลำดับอื่น รูปที่ 4-11 แสดงรายละเอียดของคลาส DiagramManagerView



รูปที่ 4-11 คลาส DiagramManagerView

1.4 คลาส SourceCodeManagerView เป็นคลาสส่วนต่อประสานผู้ใช้ของหน้าต่างการจัดการไฟล์รหัสต้นฉบับได้แก่ การลบไฟล์ การส่งออกไฟล์ และการเปลี่ยนชื่อไฟล์ รูปที่ 4-12 แสดงรายละเอียดของคลาส SourceCodeManagerView

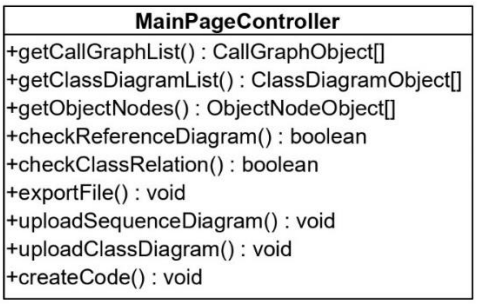


รูปที่ 4-12 คลาส SourceCodeManagerView

2. แพ็คเกจ controller

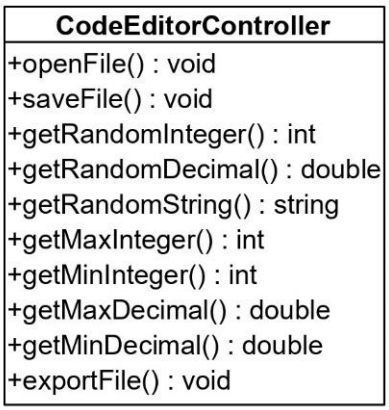
แพ็คเกจ controller ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มคอลโทลเลอร์ (Controller) ตามรูปแบบเอ็มวีซี โดยแต่ละคลาสมีรายละเอียดดังต่อไปนี้

2.1 คลาส MainPageController เป็นคลาสคอลโทลเลอร์ของหน้าต่างหลักของตัวสร้างซึ่งทำหน้าที่ปฏิบัติฟังก์ชันการใช้งานที่ได้รับจากคลาส MainPageView รูปที่ 4-13 แสดงรายละเอียดของคลาส MainPageController



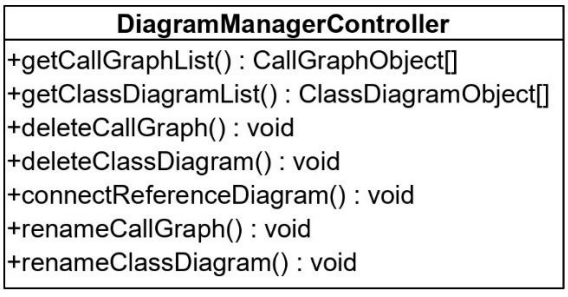
รูปที่ 4-13 คลาส MainPageController

2.2 คลาส CodeEditorController เป็นคลาสคอลโทลเลอร์ของหน้าต่างการแก้ไขรหัสต้นฉบับของตัวสร้างซึ่งทำหน้าที่ปฏิบัติฟังก์ชันการใช้งานที่ได้รับจากคลาส CodeEditorView รูปที่ 4-14 แสดงรายละเอียดของคลาส CodeEditorController



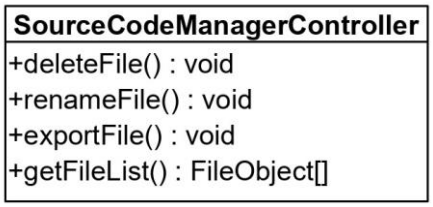
รูปที่ 4-14 คลาส CodeEditorController

2.3 คลาส DiagramManagerController เป็นคลาสคอลโทลเลอร์ของหน้าต่างการจัดการแผนภาพของตัวสร้างซึ่งทำหน้าที่ปฏิบัติฟังก์ชันการใช้งานที่ได้รับจากคลาส Diagram-ManagerView รูปที่ 4-15 แสดงรายละเอียดของคลาส DiagramManagerController



รูปที่ 4-15 คลาส DiagramManagerController

2.4 คลาส SourceCodeManagerController เป็นคลาสคอลโทลเลอร์ของหน้าต่างการจัดการไฟล์รหัสต้นฉบับของตัวสร้างซึ่งทำหน้าที่ปฏิบัติฟังก์ชันการใช้งานที่ได้รับจากคลาส SourceCodeManagerView รูปที่ 4-16 แสดงรายละเอียดของคลาส SourceCodeManager-Controller



รูปที่ 4-16 คลาส SourceCodeManagerController

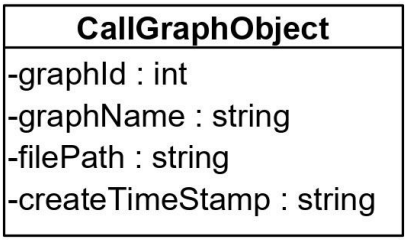
3. แพ็คเกจ model

แพ็คเกจ model ประกอบด้วยคลาสที่ทำหน้าที่เป็นคลาสกลุ่มโมเดล (Model) ตามรูปแบบเอ็มวีซี แพ็คเกจนี้ประกอบด้วยแพ็คเกจย่อย 3 แพ็คเกจซึ่งมีรายละเอียดดังต่อไปนี้

3.1 แพ็คเกจ callgraph

แพ็คเกจ callgraph ประกอบด้วยคลาสประเภทโมเดลที่แสดงถึงโครงสร้างของกราฟการเรียกใช้งาน โดยรายละเอียดของแต่ละคลาสมีดังต่อไปนี้

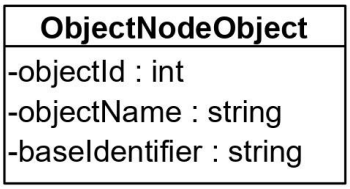
3.1.1 คลาส CallGraphObject เป็นคลาสที่เก็บข้อมูลของกราฟการเรียกใช้งานที่สร้างจากแผนภาพลำดับได้แก่ รหัสอ้างอิงของกราฟ ชื่อกราฟ ที่อยู่ของไฟล์ และวันเวลาที่สร้างกราฟ รูปที่ 4-17 แสดงรายละเอียดของคลาส CallGraphObject



รูปที่ 4-17 คลาส CallGraphObject

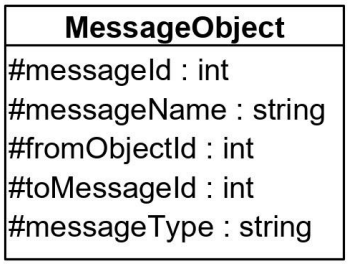
3.1.2 คลาส ObjectNodeObject เป็นคลาสที่เก็บข้อมูลของอ็อบเจกต์ภายในกราฟการเรียกใช้งานได้แก่ รหัสอ้างอิงของอ็อบเจกต์ ชื่อของอ็อบเจกต์ และคลาสพื้นฐานของอ็อบเจกต์ โดยคลาสพื้นฐานของอ็อบเจกต์ จะบอกว่าอ็อบเจกต์ ดังกล่าวมีพื้นฐานมาจากคลาส

ใด หรือเป็น Actor หรือเป็นแผนภาพลำดับอ้างอิง รูปที่ 4-18 แสดงรายละเอียดของคลาส ObjectNodeObject



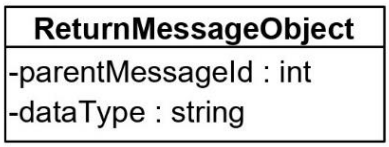
รูปที่ 4-18 คลาส ObjectNodeObject

3.1.3 คลาส MessageObject เป็นคลาสที่เก็บข้อมูลของเมสเสจที่ส่งหากันระหว่างอ็อบเจกต์ โดยเก็บรหัสอ้างอิง ชื่อของเมสเสจ และระบุว่าถูกส่งจากอ็อบเจกต์ ต้นทางใดและส่งถึงอ็อบเจกต์ ปลายทางใด รวมทั้งระบุว่าเป็นเมสเสจประเภทใดระหว่าง การเรียกใช้ (Call) การส่งค่ากลับ (Return) หรือการสร้างอ็อบเจกต์ (Create) รูปที่ 4-19 แสดงรายละเอียดของคลาส MessageObject



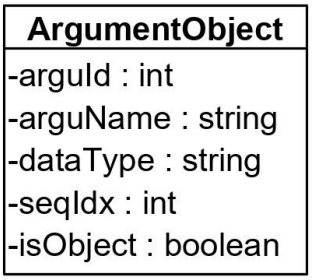
รูปที่ 4-19 คลาส MessageObject

3.1.4 คลาส ReturnMessageObject เป็นคลาสที่เก็บข้อมูลของเมสเสจประเภทการส่งค่ากลับ โดยคลาสนี้ขยาย (Extend) มาจากคลาส MessageObject เพื่อเก็บข้อมูลเพิ่มเติมว่าเมสเสจส่งกลับนี้เป็นการตอบกลับเมสเสจใดและมีประเภทของข้อมูลส่งกลับเป็นชนิดใด โดยข้อมูลดังกล่าวจะนำไปใช้ในการวิเคราะห์การ์ดคอนดิชัน รูปที่ 4-20 แสดงรายละเอียดของคลาส ReturnMessageObject



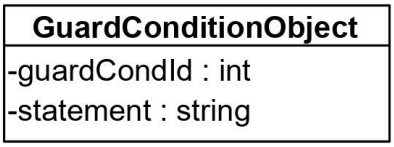
รูปที่ 4-20 คลาส ReturnMessageObject

3.1.5 คลาส ArgumentObject เป็นคลาสที่เก็บข้อมูลของอาร์กิวเมนต์ (Argument) ของเมสเสจประกอบด้วยรหัสอ้างอิง ชื่ออาร์กิวเมนต์ ประเภทของข้อมูล ลำดับของอาร์กิวเมนต์ และตัวบ่งชี้ (Indicator) ของอาร์กิวเมนต์ว่าเป็นอ็อบเจกต์ หรือไม่ ซึ่งข้อมูลดังกล่าวจะนำไปใช้ในการวิเคราะห์การ์ดคอนดิชัน รูปที่ 4-21 แสดงรายละเอียดของคลาส ArgumentObject



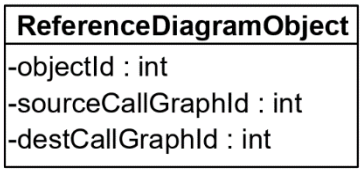
รูปที่ 4-21 คลาส ArgumentObject

3.1.6 คลาส GuardConditionObject เป็นคลาสที่เก็บข้อมูลของการ์ดคอนดิชัน ประกอบด้วยรหัสอ้างอิง เนื้อหาของเงื่อนไข และเมสเสจที่ถูกการ์ดคอนดิชันกำกับอยู่ รูปที่ 4-22 แสดงรายละเอียดของคลาส GuardConditionObject



รูปที่ 4-22 คลาส GuardConditionObject

3.1.7 คลาส ReferenceDiagramObject เป็นคลาสที่ทำหน้าที่เก็บข้อมูลของการอ้างอิงแผนภาพลำดับ โดยเก็บรหัสอ้างอิงของอ็อบเจกต์ รหัสอ้างอิงของแผนภาพลำดับต้นทาง และรหัสอ้างอิงของแผนภาพลำดับที่ถูกอ้างอิง รูปที่ 4-23 แสดงรายละเอียดของคลาส ReferenceDiagramObject

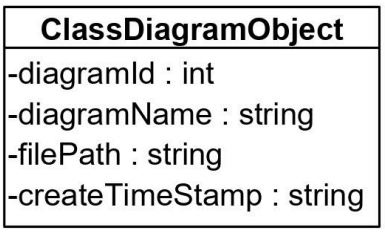


รูปที่ 4-23 คลาส ReferenceDiagramObject

3.2 แพ็คเกจ classdiagram

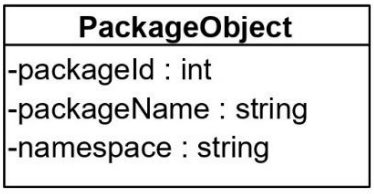
แพ็คเกจ classdiagram ประกอบด้วยคลาสประเภทโมเดลที่แสดงถึงโครงสร้างของแผนภาพคลาส โดยรายละเอียดของแต่ละคลาสมีดังต่อไปนี้

3.2.1 คลาส ClassDiagramObject เป็นคลาสที่เก็บข้อมูลของแผนภาพคลาสได้แก่ รหัสอ้างอิง ชื่อแผนภาพ ที่อยู่ของไฟล์ และวันเวลาที่นำเข้าแผนภาพ รูปที่ 4-24 แสดงรายละเอียดของคลาส ClassDiagramObject



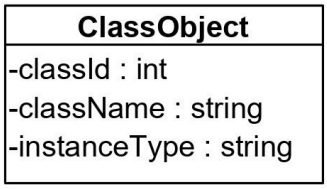
รูปที่ 4-24 คลาส ClassDiagramObject

3.2.2 คลาส PackageObject เป็นคลาสที่เก็บข้อมูลของแพ็คเกจในแผนภาพลำดับได้แก่ รหัสอ้างอิง ชื่อแพ็คเกจ และเนมสเปซของแพ็คเกจ รูปที่ 4-25 แสดงรายละเอียดของคลาส PackageObject



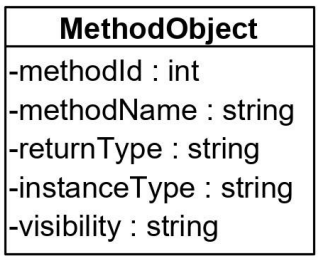
รูปที่ 4-25 คลาส PackageObject

3.2.3 คลาส ClassObject เป็นคลาสที่เก็บข้อมูลของคลาสในแผนภาพคลาสได้แก่ รหัสอ้างอิง ชื่อคลาส และประเภทของคลาสว่าเป็นคลาสรูปธรรม คลาสนามธรรม หรืออินเตอร์เฟส รูปที่ 4-26 แสดงรายละเอียดของคลาส ClassObject



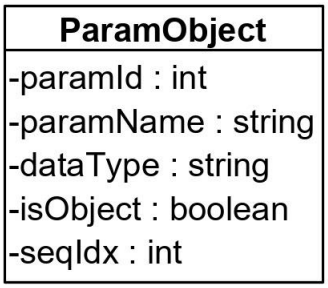
รูปที่ 4-26 คลาส ClassObject

3.2.4 คลาส MethodObject เป็นคลาสที่เก็บข้อมูลของเมท็อดได้แก่ รหัสอ้างอิง ชื่อเมท็อด ประเภทของข้อมูลส่งกลับ ประเภทของเมท็อด (รูปธรรม นามธรรม หรือ เชิงสถิต) และระดับการมองเห็น ซึ่งข้อมูลข้างต้นเป็นเมท็อดซิกเนเจอร์ที่จะใช้ในการสร้างรหัสต้นฉบับ รูปที่ 4-27 แสดงรายละเอียดของคลาส MethodObject



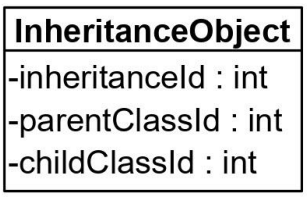
รูปที่ 4-27 คลาส MethodObject

3.2.5 คลาส ParamObject เป็นคลาสที่เก็บข้อมูลของพารามิเตอร์ของแต่ละเมท็อดได้แก่ รหัสอ้างอิง ชื่อพารามิเตอร์ ประเภทของข้อมูล ตัวบ่งชี้ว่าเป็นข้อมูลประเภทอ็อบเจกต์หรือไม่ และลำดับของพารามิเตอร์ ซึ่งข้อมูลดังกล่าวเป็นส่วนหนึ่งของเมท็อดซิกเนเจอร์ที่จะนำไปสร้างรหัสต้นฉบับ รูปที่ 4-28 แสดงรายละเอียดของคลาส ParamObject



รูปที่ 4-28 คลาส ParamObject

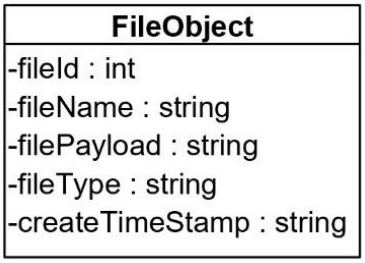
3.2.6 คลาส InheritanceObject เป็นคลาสที่เก็บข้อมูลการสืบทอดของคลาสภายในแผนภาพลำดับโดยระบุรหัสอ้างอิงของคลาสแม่และรหัสอ้างอิงของคลาสที่สืบทอด รูปที่ 4-29 แสดงรายละเอียดของคลาส InheritanceObject



รูปที่ 4-29 คลาส InheritanceObject

3.3 แพ็คเกจ file

แพ็คเกจ file ประกอบด้วยคลาสโมเดลที่แสดงถึงโครงสร้างของไฟล์รหัสต้นฉบับของสตับและไดร์เวอร์ โดยแพ็คเกจดังกล่าวประกอบด้วยคลาสหนึ่งคลาสได้แก่ คลาส FileObject ซึ่งเป็นคลาสที่ใช้เก็บข้อมูลของไฟล์รหัสต้นฉบับได้แก่ รหัสอ้างอิง ชื่อไฟล์ เนื้อหาของไฟล์ ประเภทของไฟล์ (สตับหรือไดร์เวอร์) และวันเวลาที่สร้างไฟล์ รูปที่ 4-30 แสดงรายละเอียดของคลาส FileObject

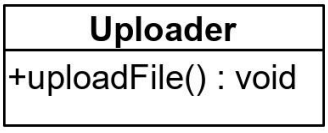


รูปที่ 4-30 คลาส FileObject

4. แพ็คเกจ utilities

แพ็คเกจ utilities เป็นแพ็คเกจของคลาสที่ใช้ในการสนับสนุนกิจกรรมทั่วไปภายในตัวสร้าง แพ็คเกจนี้ประกอบด้วยคลาส 3 คลาสโดยมีละเอียดดังต่อไปนี้

4.1 คลาส Uploader เป็นคลาสที่ทำหน้าที่ช่วยในการนำเข้าไฟล์แผนภาพ รูปที่ 4-31 แสดงรายละเอียดของคลาส Uploader



รูปที่ 4-31 คลาส Uploader

4.2 คลาส Downloader เป็นคลาสที่ทำหน้าที่ช่วยในการส่งออกไฟล์รหัสต้นฉบับ รูปที่ 4-32 แสดงรายละเอียดของคลาส Downloader

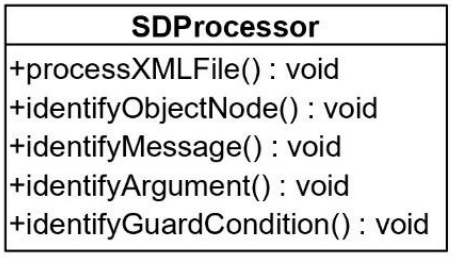
รูปที่ 4-32 คลาส Downloader

4.3 คลาส LocalFileManager เป็นคลาสที่ทำหน้าที่ช่วยในการจัดการไฟล์ของตัวสร้าง โดยคลาสนี้มีฟังก์ชันในการสร้างไฟล์บีบอัดของไฟล์รหัสต้นฉบับเมื่อส่งออกไฟล์หลาย ๆ ไฟล์พร้อมกัน รูปที่ 4-33 แสดงรายละเอียดของคลาส LocalFileManager

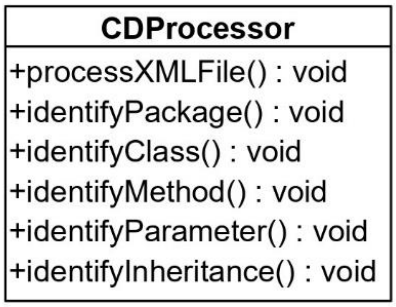
รูปที่ 4-33 คลาส LocalFileManager

5. แพ็คเกจ xmlprocessor

แพ็คเกจ xmlprocessor เป็นแพ็คเกจที่ประกอบด้วยคลาสที่ใช้ในการประมวลผลไฟล์เอกซ์เอ็มแอลซึ่งได้แก่ ไฟล์แผนภาพลำดับและไฟล์แผนภาพคลาส โดยไฟล์แผนภาพลำดับจะถูกประมวลผลด้วยคลาส SDProcessor ซึ่งแสดงรายละเอียดในรูปที่ 4-34 และไฟล์แผนภาพคลาสถูกประมวลผลด้วยคลาส CDProcessor ซึ่งแสดงรายละเอียดในรูปที่ 4-35



รูปที่ 4-34 คลาส SDProcessor

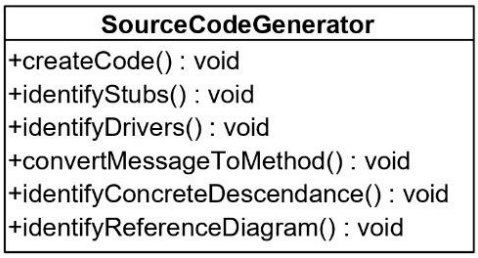


รูปที่ 4-35 คลาส CDProcessor

6. แพ็คเกจ sourcecode

แพ็คเกจ sourcecode ประกอบด้วยคลาสที่ทำหน้าที่สร้างรหัสต้นฉบับของสตับและไดร์เวอร์ โดยแพ็คเกจดังกล่าวประกอบด้วยคลาสหนึ่งคลาสและแพ็คเกจหนึ่งแพ็คเกจโดยมีรายละเอียดดังต่อไปนี้

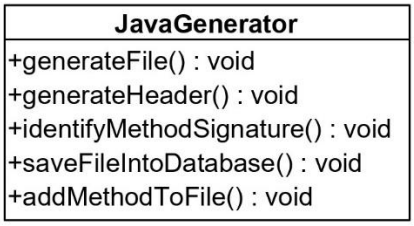
6.1 คลาส SourceCodeGenerator เป็นคลาสที่ทำหน้าที่ตรวจหาสตับและไดร์เวอร์ที่คลาสภายใต้การทดสอบต้องใช้ ค้นหาคลาสที่เป็นตัวแทนของแผนภาพลำดับที่ถูกอ้างอิง และค้นหาคลาสรูปธรรมที่สืบทอดมาจากคลาสนามธรรมเมื่อคลาสนามธรรมนั้นต้องถูกสร้างเป็นสตับ รูปที่ 4-36 แสดงรายละเอียดของคลาส SourceCodeGenerator



รูปที่ 4-36 คลาส SourceCodeGenerator

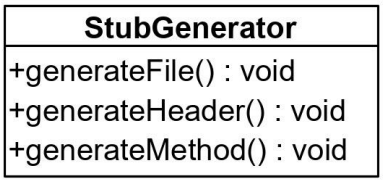
6.2 แพ็คเกจ java เป็นแพ็คเกจที่ประกอบด้วยคลาสที่ทำหน้าที่สร้างไฟล์รหัสต้นฉบับภาษาจาวา ซึ่งประกอบด้วยคลาส 3 คลาสที่มีรายละเอียดดังต่อไปนี้

6.2.1 คลาส JavaGenerator เป็นคลาสที่ทำหน้าที่ควบคุมการสร้างไฟล์รหัสต้นฉบับภาษาจาวา รูปที่ 4-37 แสดงรายละเอียดของคลาส JavaGenerator



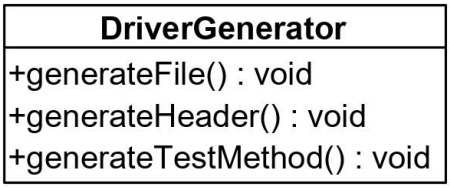
รูปที่ 4-37 คลาส JavaGenerator

6.2.2 คลาส StubGenerator เป็นคลาสที่ขยายมาจากคลาส JavaGenerator ทำหน้าที่สร้างรหัสต้นฉบับของสตับ รูปที่ 4-38 แสดงรายละเอียดของคลาส StubGenerator



รูปที่ 4-38 คลาส StubGenerator

6.2.3 คลาส DriverGenerator เป็นคลาสที่ขยายมาจากคลาส SourceCodeGenerator ทำหน้าที่สร้างรหัสต้นฉบับของไดร์เวอร์ รูปที่ 4-39 แสดงรายละเอียดของคลาส DriverGenerator

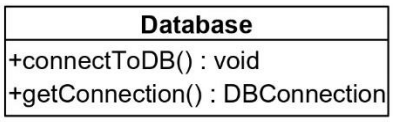


รูปที่ 4-39 คลาส DriverGenerator

7. แพ็คเกจ database

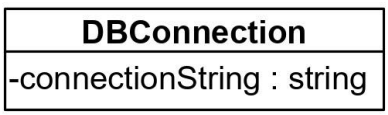
แพ็คเกจ database เป็นแพ็คเกจที่รวบรวมคลาสที่ทำหน้าที่เชื่อมต่อกับฐานข้อมูล โดยรายละเอียดของแต่ละคลาสมีดังต่อไปนี้

7.1 คลาส Database คือ คลาสที่ทำหน้าที่สร้างการเชื่อมต่อกับฐานข้อมูล รูปที่ 4-40 แสดงรายละเอียดของคลาส Database



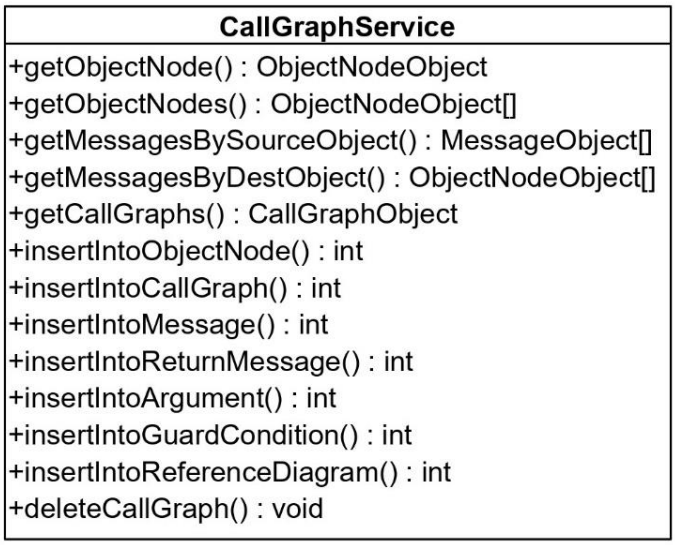
รูปที่ 4-40 คลาส Database

7.2 คลาส DBConnection เป็นคลาสที่ทำหน้าที่เป็นตัวเชื่อมต่อระหว่างตัวสร้างกับฐานข้อมูล รูปที่ 4-41 แสดงรายละเอียดของคลาส DBConnection



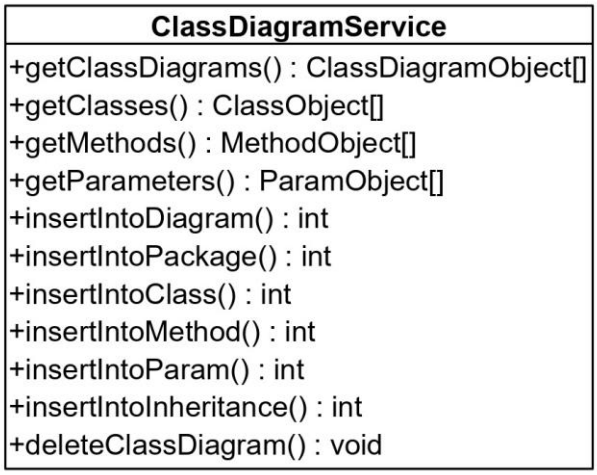
รูปที่ 4-41 คลาส DBConnection

7.3 คลาส CallGraphService คือ คลาสที่ทำหน้าที่รวบรวมฟังก์ชันกลุ่มซีอาร์ยูดี (CRUD - Create, Read, Update, Delete) [18] บนฐานข้อมูลของกราฟการเรียกใช้งาน รูปที่ 4-42 แสดงรายละเอียดของคลาส CallGraphService



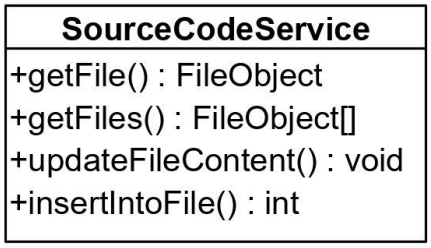
รูปที่ 4-42 คลาส CallGraphService

7.4 คลาส ClassDiagramService เป็นคลาสที่ทำหน้าที่รวบรวมฟังก์ชันกลุ่มซีอาร์ยูดีบนฐานข้อมูลของแผนภาพคลาส รูปที่ 4-43 แสดงรายละเอียดของคลาส ClassDiagramService



รูปที่ 4-43 คลาส ClassDiagramService

7.5 คลาส SourceCodeService คือ คลาสที่ทำหน้าที่รวบรวมฟังก์ชันกลุ่มซีอาร์ยูดีบนฐานข้อมูลของไฟล์รหัสต้นฉบับ รูปที่ 4-44 แสดงรายละเอียดของคลาส SourceCodeService



รูปที่ 4-44 คลาส SourceCodeService

4.1.3 แบบจำลองเชิงพฤติกรรม

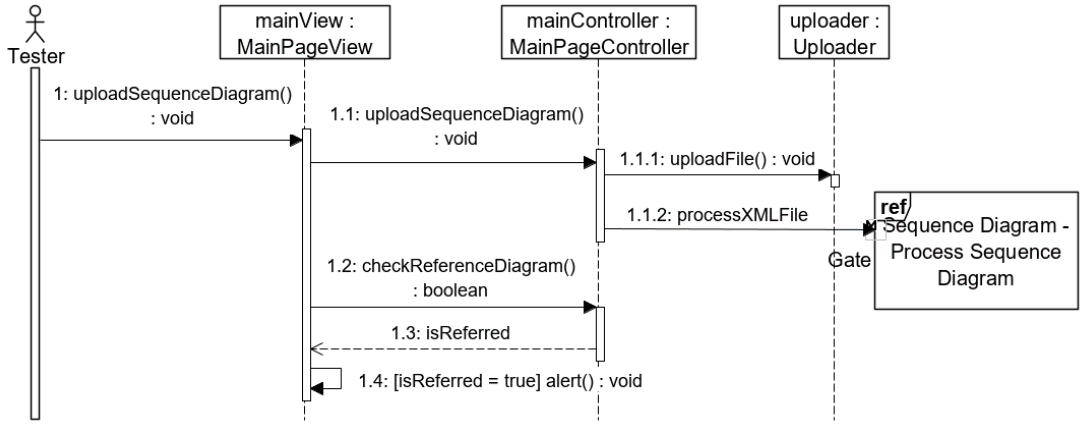
สำหรับแบบจำลองเชิงพฤติกรรมผู้วิจัยได้ใช้แผนภาพลำดับในการอธิบายพฤติกรรมของตัวสร้างสตับและไดร์เวอร์ โดยมีรายละเอียดดังต่อไปนี้

4.1.3.1 แผนภาพลำดับ

แผนภาพลำดับของฟังก์ชันหลักของตัวสร้างสตับและไดร์เวอร์ประกอบด้วยแผนภาพลำดับ 6 แผนภาพ ดังต่อไปนี้

1. แผนภาพลำดับการนำเข้าไฟล์แผนภาพลำดับ

แผนภาพลำดับของการนำเข้าไฟล์แผนภาพลำดับแสดงในรูปที่ 4-45 เริ่มต้นจาก ผู้ทดสอบหรือ Tester ในแผนภาพลำดับเรียกใช้ฟังก์ชัน uploadSequenceDiagram ของคลาส MainPageView และคลาส MainPageView ส่งเมสเสจเรียกใช้ฟังก์ชัน upload-SequenceDiagram ของคลาส MainPageController จากนั้นคลาส MainPageController จะเรียกใช้ฟังก์ชัน uploadFile ของคลาส Uploader เพื่อนำเข้าไฟล์แผนภาพลำดับ และคลาส MainPageController จะส่งเมสเสจ processXMLFile เพื่อประมวลผลไฟล์แผนภาพลำดับที่นำเข้ามา หลังจากประมวลผลไฟล์แผนภาพลำดับเสร็จ คลาส MainPageView จะส่งเมสเสจหาคลาส MainPageController เพื่อให้ตรวจสอบว่าในแผนภาพลำดับที่นำเข้ามีอ็อบเจกต์ ที่อ้างอิงถึงแผนภาพลำดับอื่นหรือไม่ ถ้ามีการอ้างอิงคลาส MainPageView จะแสดงเมสเสจเตือนว่ามีการอ้างอิงถึงแผนภาพลำดับอื่นให้ผู้ทดสอบนำเข้าไฟล์แผนภาพลำดับที่ถูกอ้างอิงและเชื่อมต่อเข้ากับแผนภาพลำดับข้างต้น



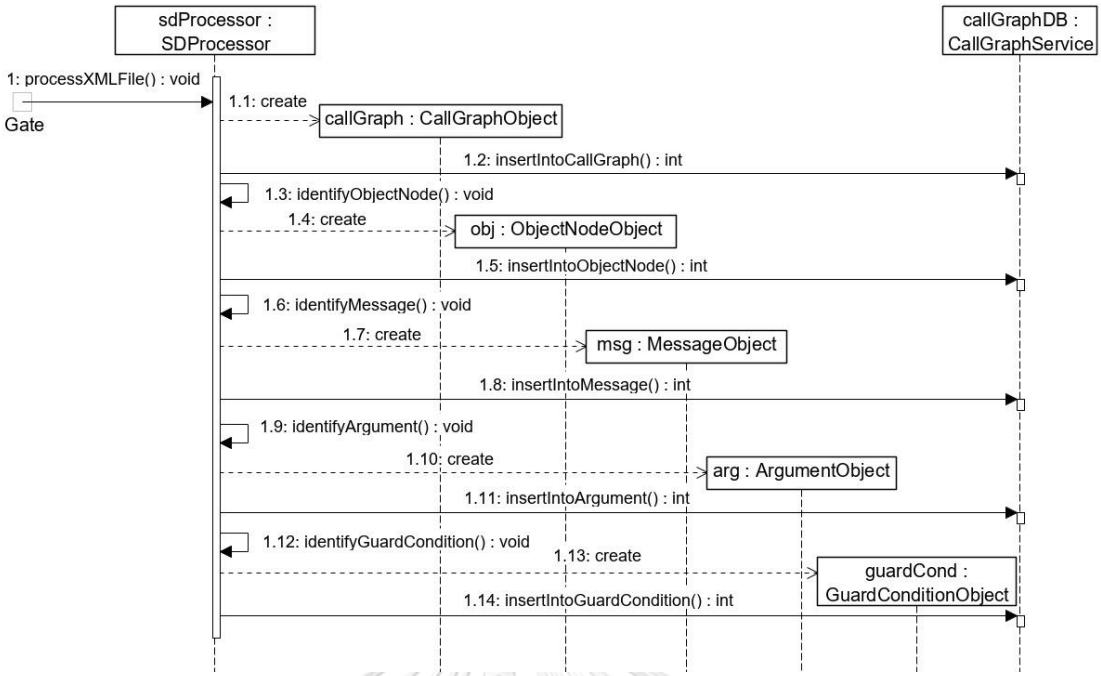
รูปที่ 4-45 แผนภาพลำดับการนำเข้าไฟล์แผนภาพลำดับ

2. แผนภาพลำดับการประมวลผลแผนภาพลำดับ

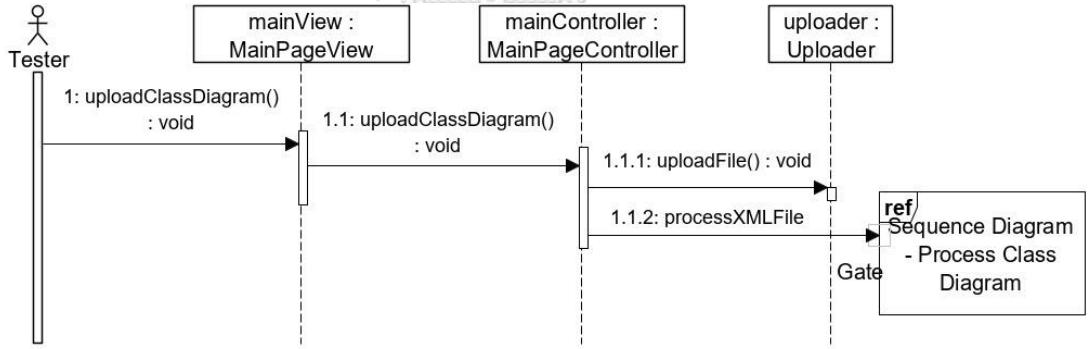
แผนภาพลำดับของการประมวลผลแผนภาพลำดับแสดงในรูปที่ 4-46 เมื่อคลาส SDProcessor ได้รับเมสเสจให้ประมวลผลแผนภาพลำดับจากคลาส MainPageController ในรูปที่ 4-45 คลาส SDProcessor จะเริ่มประมวลผลไฟล์เอกซ์เอ็มแอลเพื่อรวบรวมข้อมูลของแผนภาพลำดับ สร้างอ็อบเจกต์ จากคลาส CallGraphObject เพื่อเก็บข้อมูลของแผนภาพลำดับและบันทึกลงฐานข้อมูลผ่านคลาส CallGraphService ถัดไป คลาส SDProcessor จะตรวจหาอ็อบเจกต์ ภายในแผนภาพลำดับ สร้างอ็อบเจกต์ จากคลาส ObjectNodeObject เพื่อเก็บข้อมูลของอ็อบเจกต์ ในแผนภาพลำดับ และบันทึกลงฐานข้อมูลผ่านทางคลาส CallGraph-Service จากนั้นตัวสร้างจะตรวจหาเมสเสจ อาร์กิวเมนต์ของแต่ละเมสเสจ และการ์ดคอนดิชัน และสร้างอ็อบเจกต์ จากคลาส MessageObject ArgumentObject และ GuardConditionObject ตามลำดับเพื่อเก็บข้อมูลข้างต้น และบันทึกข้อมูลข้างต้นลงฐานข้อมูลผ่านทางคลาส CallGraphService

3. แผนภาพลำดับการนำเข้าไฟล์แผนภาพคลาส

แผนภาพลำดับการนำเข้าไฟล์แผนภาพคลาสแสดงในรูปที่ 4-47 ผู้ทดสอบหรือ Tester ในแผนภาพลำดับเรียกใช้ฟังก์ชัน uploadClassDiagram ของคลาส MainPageView และคลาส MainPageView จะเรียกใช้ฟังก์ชัน uploadClassDiagram ของคลาส MainPage-Controller ถัดไป คลาส MainPageController จะเรียกใช้ฟังก์ชัน uploadFile ของคลาส Uploader เพื่อนำเข้าไฟล์แผนภาพคลาส และคลาส MainPageController จะส่งเมสเสจ processXMLFile เพื่อประมวลผลไฟล์แผนภาพคลาสที่นำเข้ามา



รูปที่ 4-46 แผนภาพลำดับการประมวลผลแผนภาพลำดับ

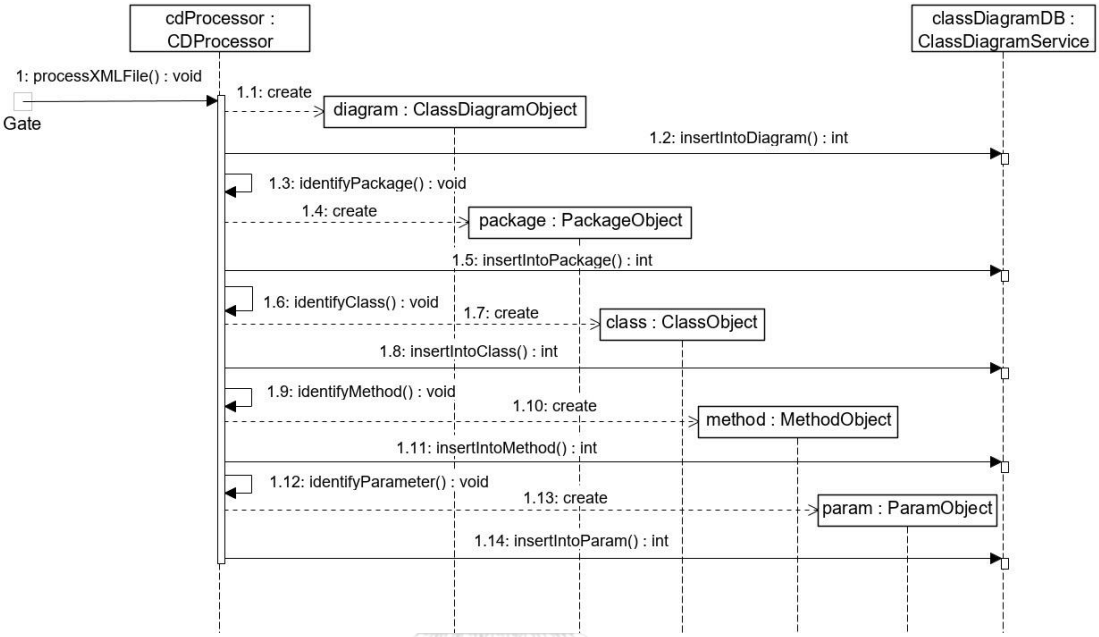


รูปที่ 4-47 แผนภาพลำดับการนำเข้าแผนภาพคลาส

4. แผนภาพลำดับการประมวลผลแผนภาพคลาส

แผนภาพลำดับของการประมวลผลแผนภาพคลาสแสดงในรูปที่ 4-48 เมื่อคลาส CDProcessor ได้รับเมสเสจให้ประมวลผลแผนภาพคลาส คลาส CDProcessor จะเริ่มการประมวลผลไฟล์เอกซ์เอ็มแอลเพื่อรวบรวมข้อมูลของแผนภาพคลาส สร้างอ็อบเจกต์ จากคลาส ClassDiagramObject เพื่อเก็บข้อมูลของแผนภาพคลาส และบันทึกลงฐานข้อมูลผ่านคลาส ClassDiagramService ถัดไป คลาส CDProcessor จะตรวจหาแพ็คเกจภายในแผนภาพคลาส

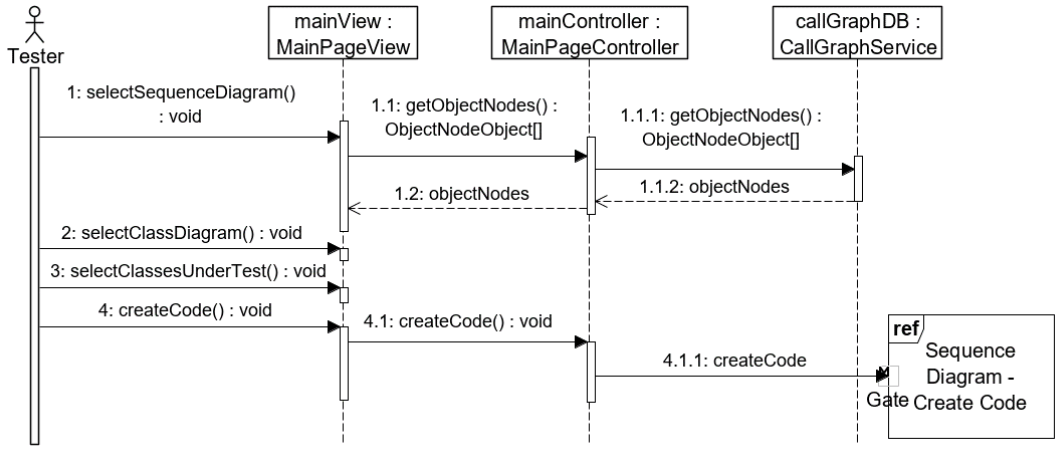
สร้างอ็อบเจกต์ จากคลาส PackageObject เพื่อเก็บข้อมูลของแพ็คเกจ และบันทึกลงฐานข้อมูลผ่านทางคลาส ClassDiagramService จากนั้นตัวสร้างจะตรวจหาคลาสในแต่ละแพ็คเกจ เมท็อดของแต่ละคลาส พารามิเตอร์ของแต่ละเมท็อด และสร้างอ็อบเจกต์ จากคลาส ClassObject MethodObject และ ParamObject ตามลำดับเพื่อเก็บข้อมูลข้างต้น และบันทึกข้อมูลลงฐานข้อมูลผ่านทางคลาส ClassDiagramService



รูปที่ 4-48 แผนภาพลำดับการประมวลผลแผนภาพคลาส

5. แผนภาพลำดับการเลือกคลาสภายใต้การทดสอบ

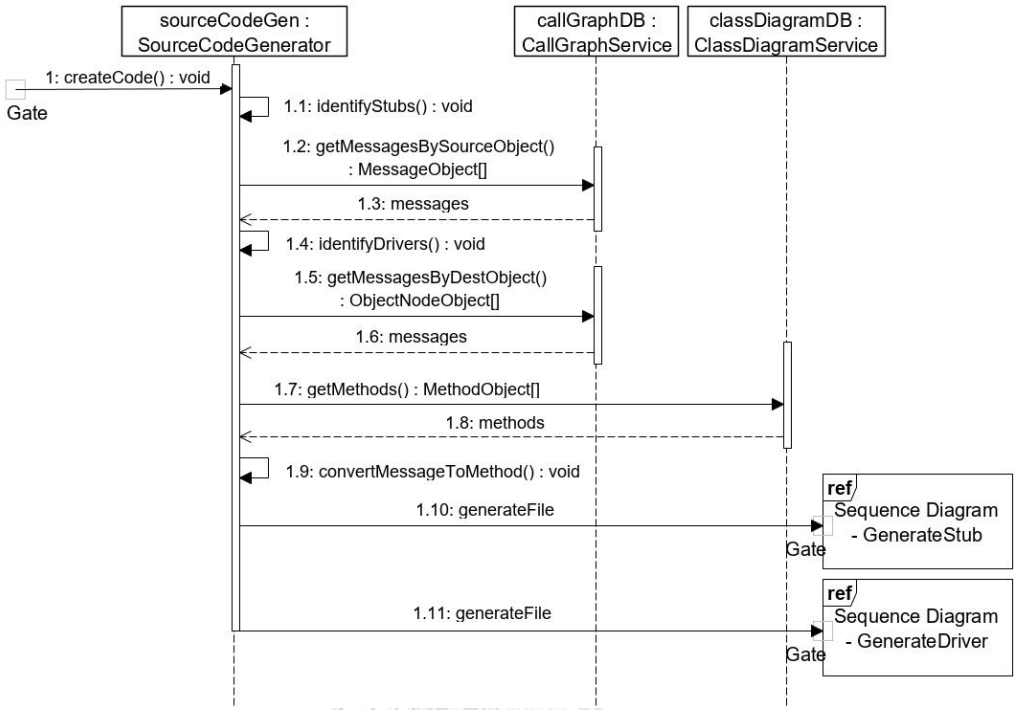
แผนภาพลำดับของการเลือกคลาสภายใต้การทดสอบแสดงในรูปที่ 4-49 เริ่มต้นจาก ผู้ทดสอบ หรือ Tester ในแผนภาพลำดับเรียกใช้ฟังก์ชัน selectSequenceDiagram ของคลาส MainPageView และคลาส MainPageView จะเรียกใช้ฟังก์ชัน getObjectNodes ของคลาส MainPageController จากนั้นคลาส MainPageController จะส่งเรียกใช้ฟังก์ชัน getObjectNodes ของคลาส CallGraphService เพื่อเตรียมรายการของอ็อบเจกต์ ภายในแผนภาพลำดับที่ผู้ทดสอบเลือก ผู้ทดสอบเลือกแผนภาพคลาสที่สอดคล้องกับแผนภาพลำดับข้างต้นผ่านฟังก์ชัน selectClassDiagram และเลือกคลาสภายใต้การทดสอบผ่านฟังก์ชัน select-ClassUnderTest ของคลาส MainPageView จากนั้นผู้ทดสอบจะสร้างรหัสต้นฉบับโดยเรียกใช้ฟังก์ชัน createCode ของคลาส MainPageView และคลาส MainPageView จะเรียกใช้ฟังก์ชัน createCode ของคลาส MainPageController สุดท้ายคลาส MainPageController จะส่งเมสเสจ createCode เพื่อเริ่มการสร้างรหัสต้นฉบับของสตับและไดร์เวอร์



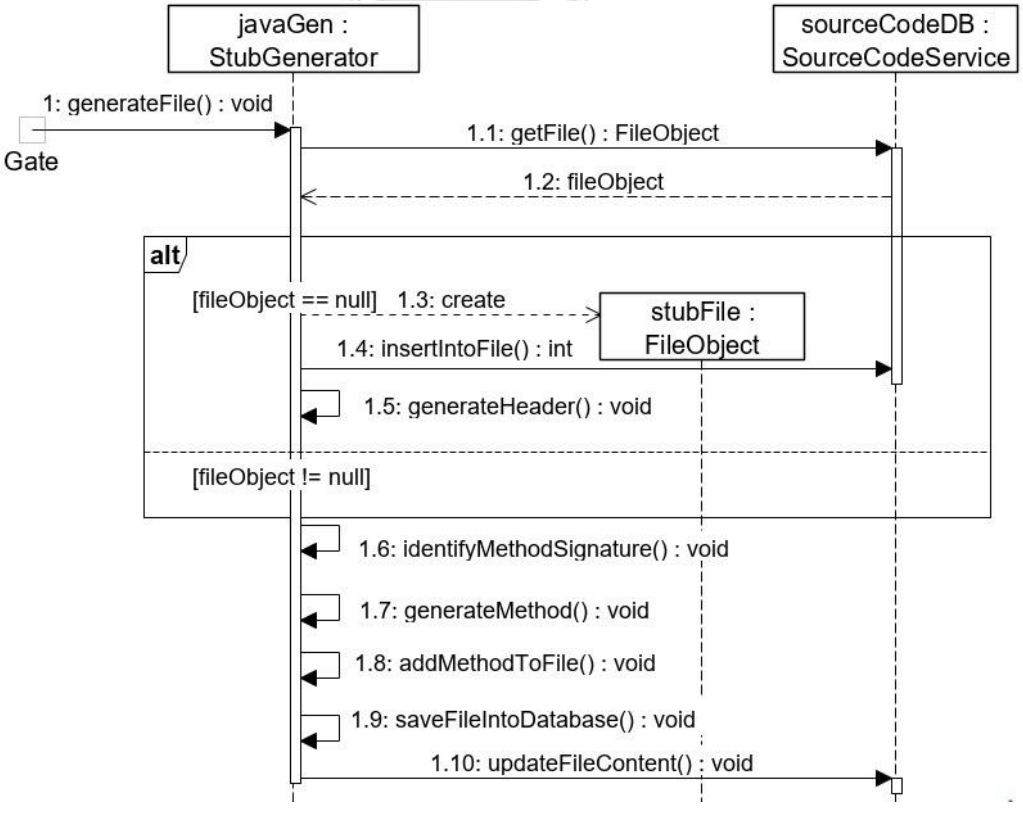
รูปที่ 4-49 แผนภาพลำดับการเลือกคลาสภายใต้การทดสอบ

6. แผนภาพลำดับการสร้างรหัสต้นฉบับ

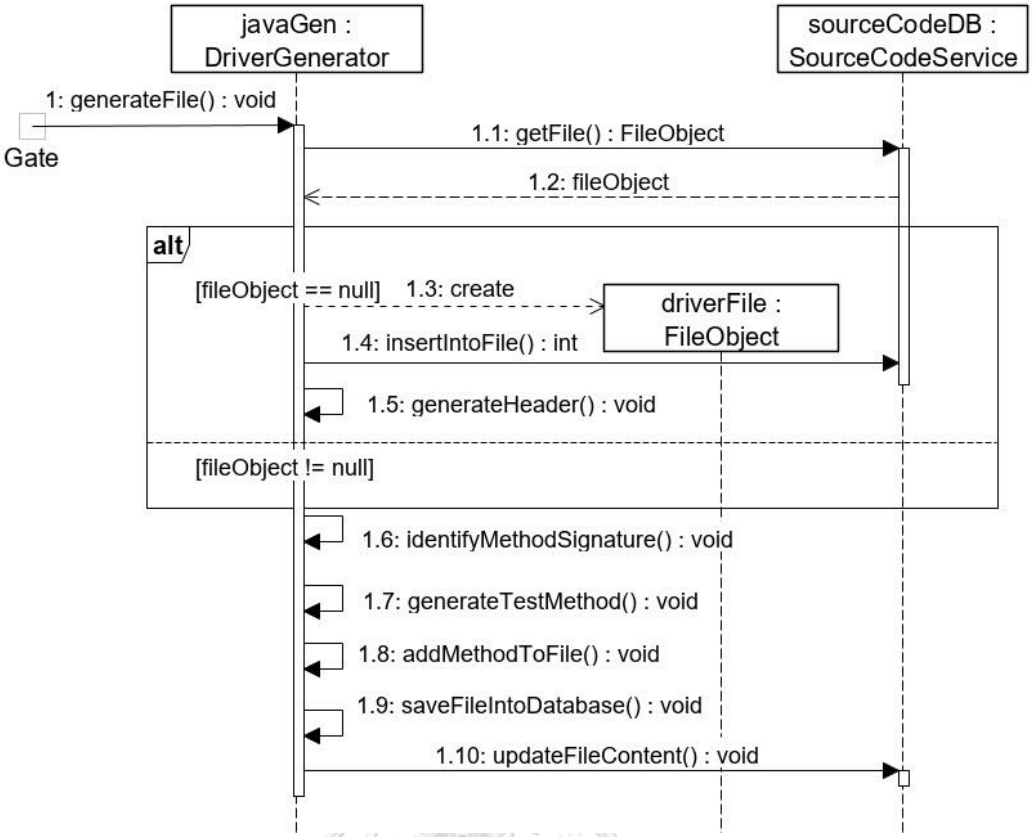
แผนภาพลำดับการสร้างรหัสต้นฉบับแสดงในรูปที่ 4-50 เมื่อคลาส SourceCodeGenerator ได้รับเมสเสจให้สร้างรหัสต้นฉบับ คลาส SourceCodeGenerator จะตรวจหาสตับที่ต้องใช้ด้วยฟังก์ชัน identifyStubs และเรียกใช้ฟังก์ชัน getMessage-BySourceObject ของคลาส CallGraphService เพื่อรวบรวมข้อมูลของเมสเสจดังกล่าว ในขณะเดียวกันคลาส SourceCodeGenerator จะตรวจหาไดร์เวอร์ที่ต้องใช้ด้วยฟังก์ชัน identifyDrivers และเรียกใช้ฟังก์ชัน getMessageByDestObject ของคลาส CallGraphService เพื่อรวบรวมข้อมูลของเมสเสจข้างต้น เมื่อรวบรวมเมสเสจเสร็จ คลาส SourceCode-Generator จะเรียกใช้ฟังก์ชัน getMethods ของคลาส ClassDiagramService เพื่อนำข้อมูลของเมท็อดซิกเนเจอร์มาเปลี่ยนจากเมสเสจให้กลายเป็นเมท็อด และเรียกใช้ฟังก์ชัน generateFile ของคลาส Stub-Generator และ DriverGenerator เพื่อสร้างรหัสต้นฉบับของสตับและไดร์เวอร์ตามลำดับ โดยขั้นตอนการสร้างไฟล์รหัสต้นฉบับของสตับแสดงด้วยแผนภาพลำดับรูปที่ 4-51 และขั้นตอนการสร้างไดร์เวอร์แสดงด้วยแผนภาพลำดับรูปที่ 4-52



รูปที่ 4-50 แผนภาพลำดับการสร้างรหัสต้นฉบับ



รูปที่ 4-51 แผนภาพลำดับของการสร้างรหัสต้นฉบับของสตับชื่อ GenerateStub



รูปที่ 4-52 แผนภาพลำดับของการสร้างรหัสต้นฉบับของไดร์เวอร์ชื่อ GenerateDriver

จากรูปที่ 4-51 เมื่อคลาส StubGenerator ได้รับเมสเสจให้สร้างสตับ คลาส StubGenerator จะตรวจสอบก่อนว่าเคยสร้างไฟล์ของสตับดังกล่าวขึ้นมาก่อนหรือไม่ โดยเรียกใช้ฟังก์ชัน getFile ของคลาส SourceCodeService เพื่อค้นหาไฟล์ดังกล่าว หากไม่เคยสร้างมาก่อน คลาส StubGenerator จะสร้างอ็อบเจกต์ ของไฟล์จากคลาส FileObject เพื่อเก็บข้อมูลของไฟล์ และบันทึกลงฐานข้อมูลด้วยฟังก์ชัน insertIntoFile ของคลาส SourceCodeService ก่อนสร้างโครงของไฟล์ขึ้นมาด้วยฟังก์ชัน generateHeader จากนั้นรวบรวมเมท็อดซิกเนเจอร์ด้วยฟังก์ชัน identifyMethodSignature และสร้างรหัสต้นฉบับของเมท็อดดังกล่าวด้วยฟังก์ชัน generateMethod แทรกลงไฟล์ด้วยฟังก์ชัน addMethodToFile และเมื่อสร้างเมท็อดเสร็จสิ้น คลาส StubGenerator จะปิดไฟล์และเตรียมบันทึกไฟล์ลงฐานข้อมูลด้วยฟังก์ชัน saveFileIntoDatabase และอัปเดตเนื้อหาของไฟล์ลงฐานข้อมูลด้วยเมท็อด updateFileContent ของคลาส SourceCodeService

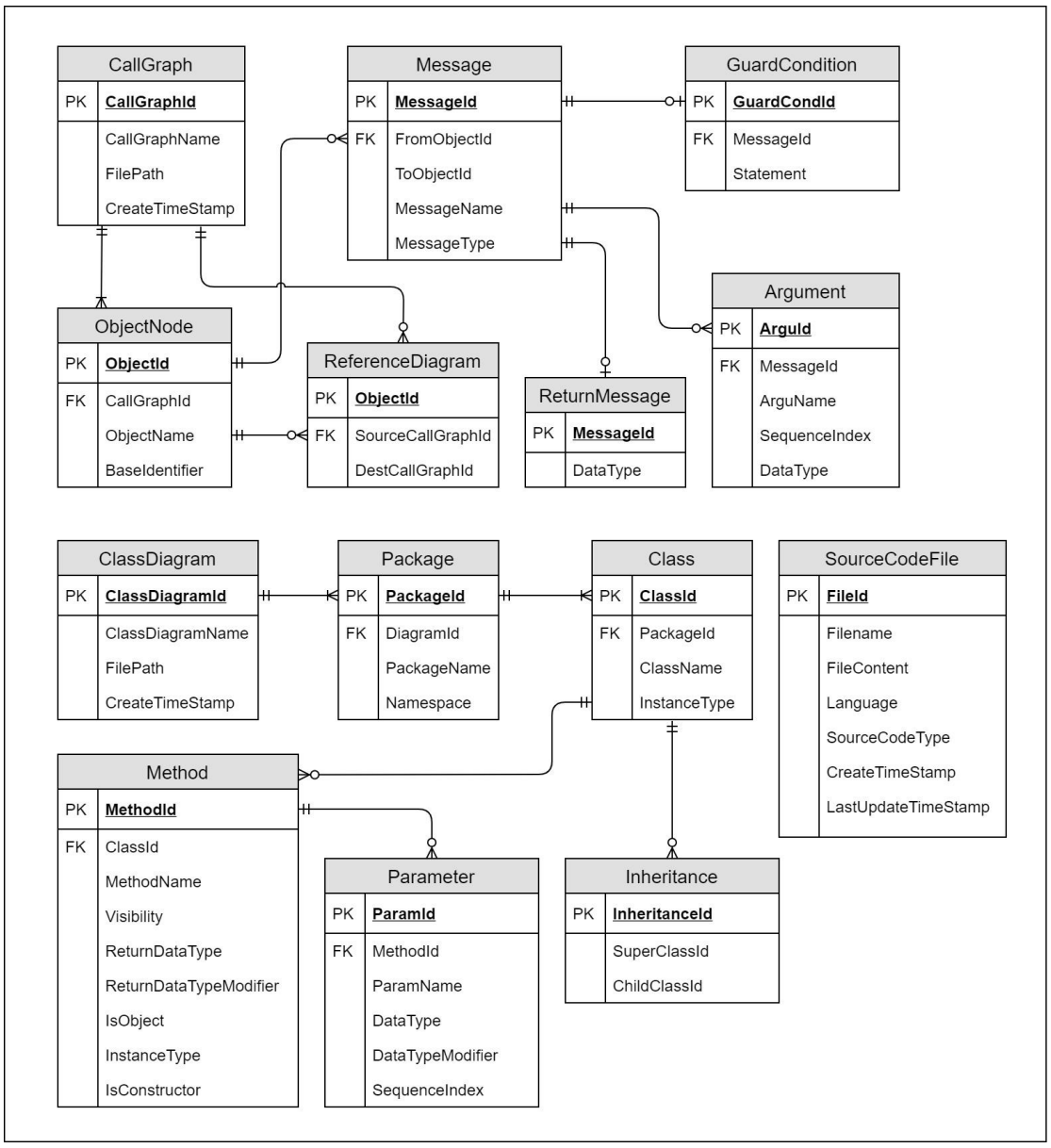
จากรูปที่ 4-52 เมื่อคลาส DriverGenerator ได้รับเมสเสจให้สร้างไดร์เวอร์ คลาส DriverGenerator จะตรวจสอบก่อนว่าเคยสร้างไฟล์ของไดร์เวอร์ดังกล่าวขึ้นมาก่อนหรือไม่

โดยเรียกใช้ฟังก์ชัน getFile ของคลาส SourceCodeService เพื่อค้นหาไฟล์ดังกล่าว หากไม่เคยสร้างมาก่อน คลาส DriverGenerator จะสร้างอ็อบเจกต์ ของไฟล์จากคลาส FileObject เพื่อเก็บข้อมูลของไฟล์ และบันทึกลงฐานข้อมูลด้วยฟังก์ชัน insertIntoFile ของคลาส SourceCodeService ก่อนสร้างโครงของไฟล์ขึ้นมาด้วยฟังก์ชัน generateHeader จากนั้นรวบรวมเมท็อดซิกเนเจอร์ด้วยฟังก์ชัน identifyMethodSignature และสร้างรหัสต้นฉบับของเมท็อดดังกล่าวด้วยฟังก์ชัน generateTestMethod แทรกลงไฟล์ด้วยฟังก์ชัน addMethodToFile และเมื่อสร้างเมท็อดเสร็จ คลาส DriverGenerator จะปิดไฟล์และเตรียมบันทึกไฟล์ลงฐานข้อมูลด้วยฟังก์ชัน saveFileIntoDatabase และอัปเดตเนื้อหาของไฟล์ลงฐานข้อมูลด้วยเมท็อด updateFileContent ของคลาส SourceCodeService

4.1.4 โครงสร้างของฐานข้อมูล

ผู้วิจัยได้ออกแบบโครงสร้างของฐานข้อมูลแสดงด้วยแผนภาพอีอาร์ (ER – Entity Relationship Diagram) แสดงในรูปที่ 4-53 โดยรายละเอียดของแต่ละเอนทิตี้ (Entity) มีดังต่อไปนี้

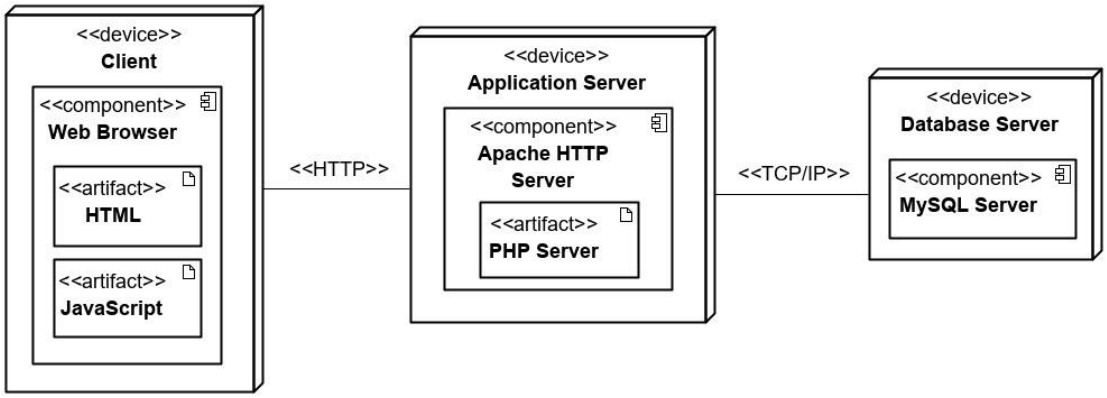
1. เอนทิตี้ CallGraph ใช้เก็บข้อมูลของกราฟการเรียกใช้งาน
2. เอนทิตี้ ObjectNode ใช้เก็บข้อมูลของอ็อบเจกต์ ในกราฟการเรียกใช้งาน
3. เอนทิตี้ Message ใช้เก็บข้อมูลของเมสเสจที่ส่งระหว่างอ็อบเจกต์
4. เอนทิตี้ ReturnMessage ใช้เก็บข้อมูลของเมสเสจประเภทเมสเสจตอบกลับ
5. เอนทิตี้ Argument ใช้เก็บข้อมูลของอาร์กิวเมนต์ของเมสเสจ
6. เอนทิตี้ GuardCondition ใช้เก็บข้อมูลของการ์ดคอนดิชัน
7. เอนทิตี้ ReferenceDiagram ใช้เก็บข้อมูลของการอ้างอิงแผนภาพลำดับของอ็อบเจกต์ประเภทการอ้างอิง
8. เอนทิตี้ ClassDiagram ใช้เก็บข้อมูลของแผนภาพคลาส
9. เอนทิตี้ Package ใช้เก็บข้อมูลของแพ็คเกจในแผนภาพคลาส
10. เอนทิตี้ Class ใช้เก็บข้อมูลของคลาสในแต่ละแพ็คเกจ
11. เอนทิตี้ Method ใช้เก็บข้อมูลของเมท็อดในแต่ละคลาส
12. เอนทิตี้ Param ใช้เก็บข้อมูลของพารามิเตอร์ของแต่ละเมท็อด
13. เอนทิตี้ Inheritance ใช้เก็บข้อมูลของการสืบทอด
14. เอนทิตี้ SourceCodeFile ใช้เก็บข้อมูลไฟล์รหัสต้นฉบับ



รูปที่ 4-53 แผนภาพอีอาร์ของตัวสร้าง

4.1.5 สถาปัตยกรรมของตัวสร้าง

ตัวสร้างสตับและไดร์เวอร์ออกแบบโดยใช้สถาปัตยกรรมแบบ 3 เทียร์ ไคลเอนต์-เซิร์ฟเวอร์ (3-Tier Client-Server) [18] โดยประกอบด้วย ไคลเอนต์ (Client) แอปพลิเคชันเซิร์ฟเวอร์ (Application Server) และเซิร์ฟเวอร์ฐานข้อมูล (Database Server) โดยสถาปัตยกรรมของตัวสร้างแสดงด้วยแผนภาพดีพลอยเมนต์ (Deployment Diagram) ในรูปที่ 4-54



รูปที่ 4-54 แผนภาพดีพลอยเมนต์ของตัวสร้าง

4.2 การพัฒนาตัวสร้างสตับและไดร์เวอร์

ในส่วนนี้จะกล่าวถึงการพัฒนาตัวสร้างสตับและไดร์เวอร์ประกอบด้วย สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนาตัวสร้าง และการพัฒนาส่วนต่อประสานผู้ใช้ โดยมีรายละเอียดดังต่อไปนี้

4.2.1 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

สภาพแวดล้อมที่และเครื่องมือที่ใช้ในการพัฒนามีรายละเอียดดังต่อไปนี้

4.2.1.1 ฮาร์ดแวร์ (Hardware)

เครื่องคอมพิวเตอร์ที่ใช้ในการพัฒนาซอฟต์แวร์มีข้อกำหนด (Specification) ดังต่อนี้

- หน่วยประมวลผล Intel Core i5-4590 ความเร็ว 3.3 กิกะเฮร์ตซ์ (GHz)
- หน่วยความจำหลัก (RAM) 16 กิกะไบต์ (GB)
- ฮาร์ดดิสก์ความจุ 512 กิกะไบต์
- หน่วยประมวลผลกราฟฟิก Nvidia GeForce GTX 750 Ti
- ระบบปฏิบัติการ Windows 10 Pro

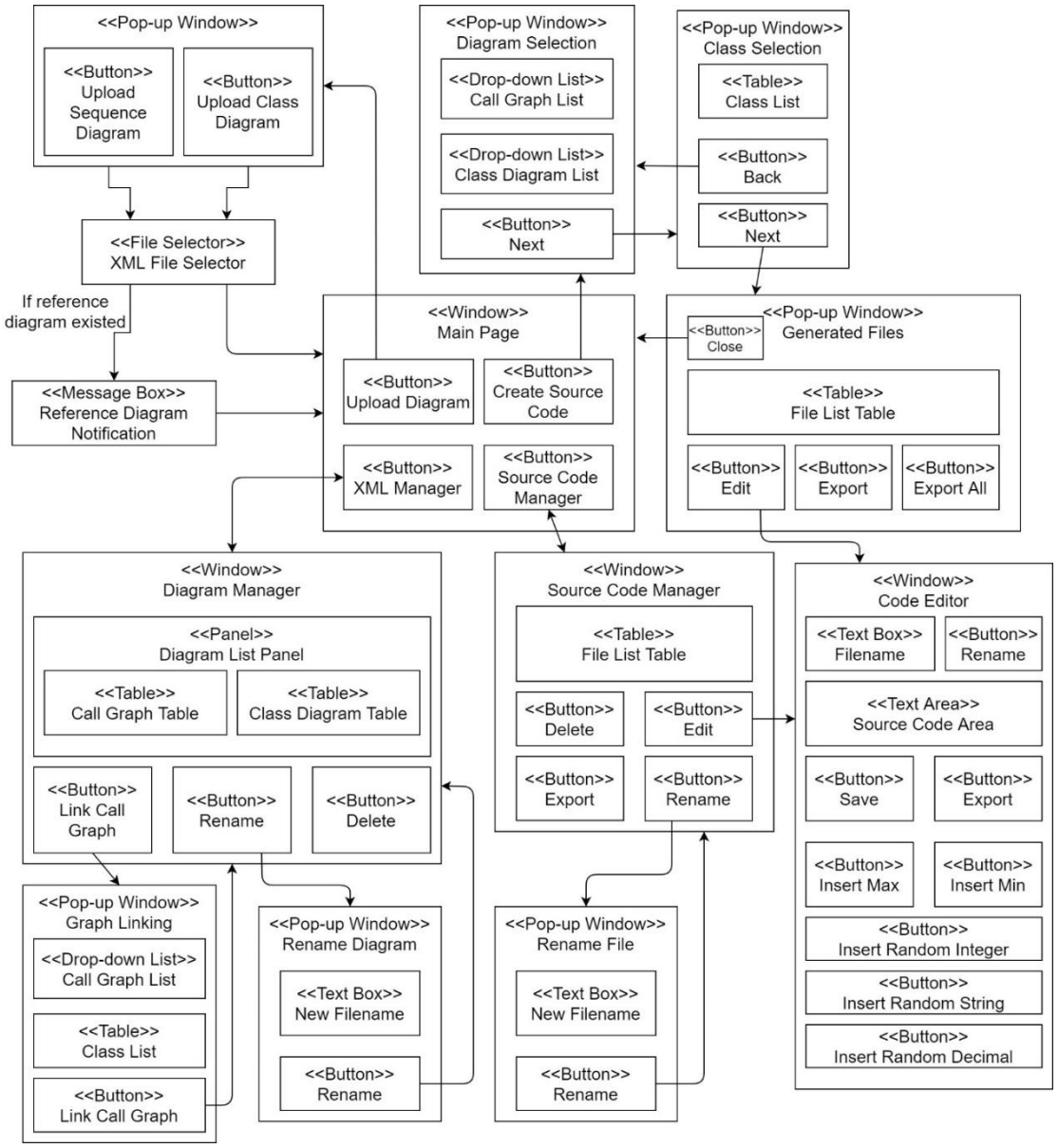
4.2.1.2 ซอฟต์แวร์ (Software)

ในการพัฒนาตัวสร้าง ผู้วิจัยได้ใช้เครื่องมือดังต่อไปนี้

- Eclipse PHP สำหรับพัฒนารหัสต้นฉบับภาษา PHP
- Microsoft Visual Studio Code สำหรับพัฒนารหัสต้นฉบับภาษา JavaScript และพัฒนาหน้าเว็บด้วย HTML และ CSS
- MAMP สำหรับการจำลองสภาพแวดล้อม โดยช่วยจำลอง Apache HTTP Server, PHP Server และ MySQL Server

4.2.2 การพัฒนาส่วนต่อประสานผู้ใช้

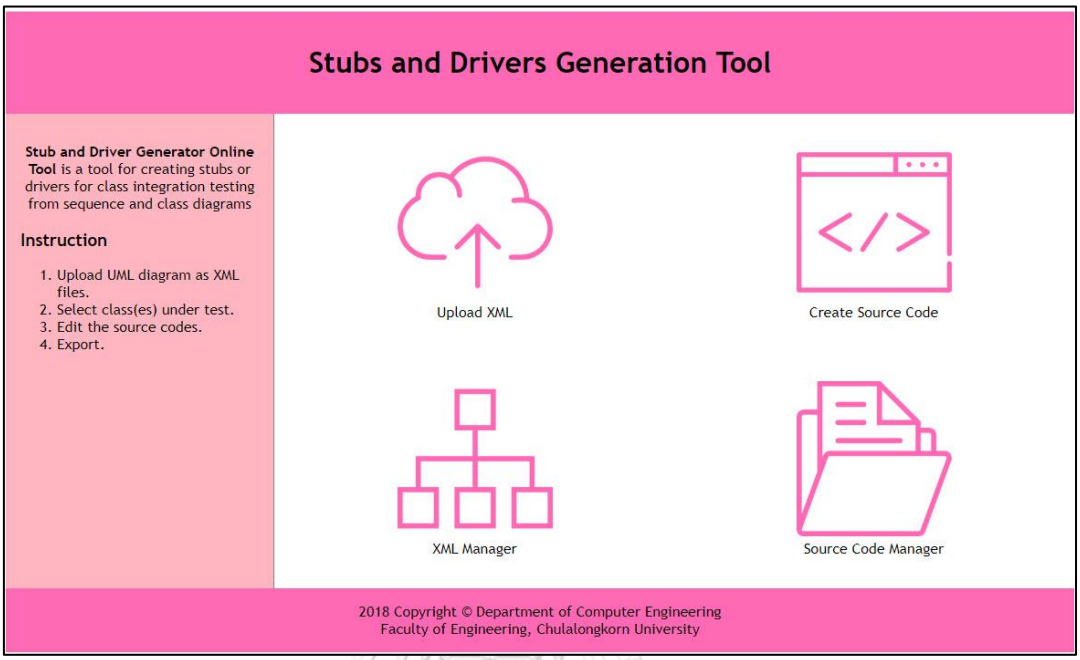
โครงสร้างส่วนต่อประสานผู้ใช้ของตัวสร้างสตับและไดร์เวอร์แสดงด้วยแผนภาพวินโดว์เนวิเกชัน (Window Navigation Diagram) ในรูปที่ 4-55 ซึ่งอธิบายความสัมพันธ์ระหว่างส่วนต่อประสานผู้ใช้ สำหรับรายละเอียดของแต่ละหน้าต่างมีดังต่อไปนี้



รูปที่ 4-55 แผนภาพวินโดว์เนวิเกชันของตัวสร้าง

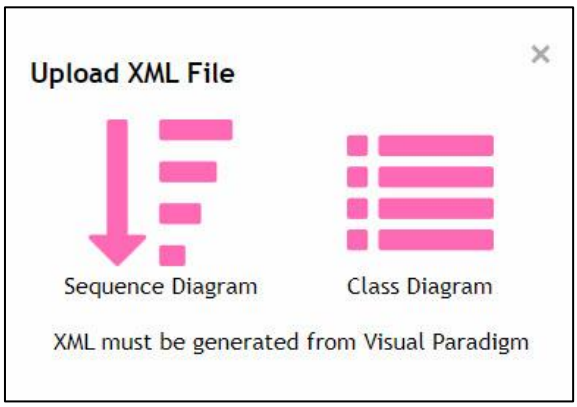
1. หน้าต่างหลัก (Main Page)

หน้าต่างหลักเป็นหน้าต่างแรกของตัวสร้างซึ่งทำหน้าที่เป็นเมนู (Menu) ของตัวสร้าง แสดงในรูปที่ 4-56 องค์ประกอบของหน้าต่างหลักประกอบด้วยปุ่ม 4 ปุ่มได้แก่

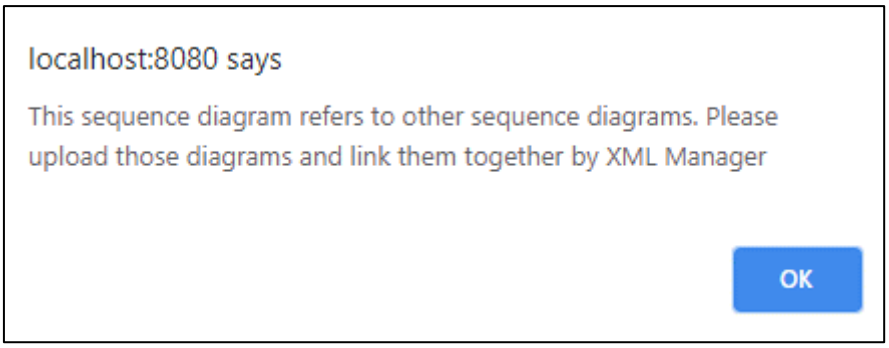


รูปที่ 4-56 หน้าต่างหลักของตัวสร้าง

1.1 ปุ่ม Upload Diagram ใช้ในการนำเข้าไฟล์แผนภาพ โดยเมื่อกดปุ่มนี้ ตัวสร้างจะแสดงหน้าต่างป๊อปอัพเพื่อให้เลือกประเภทของแผนภาพที่ต้องการนำเข้า แสดงในรูปที่ 4-57 จากนั้นตัวสร้างจะให้เลือกไฟล์ที่ต้องการนำเข้าและตัวสร้างจะประมวลผลไฟล์แผนภาพที่นำเข้า ในกรณีที่แผนภาพลำดับที่นำเข้ามีการอ้างอิงถึงแผนภาพลำดับอื่น ตัวสร้างจะแสดงข้อความเตือนให้ให้เชื่อมต่อกับแผนภาพลำดับที่ถูกอ้างอิงแสดงในรูปที่ 4-58

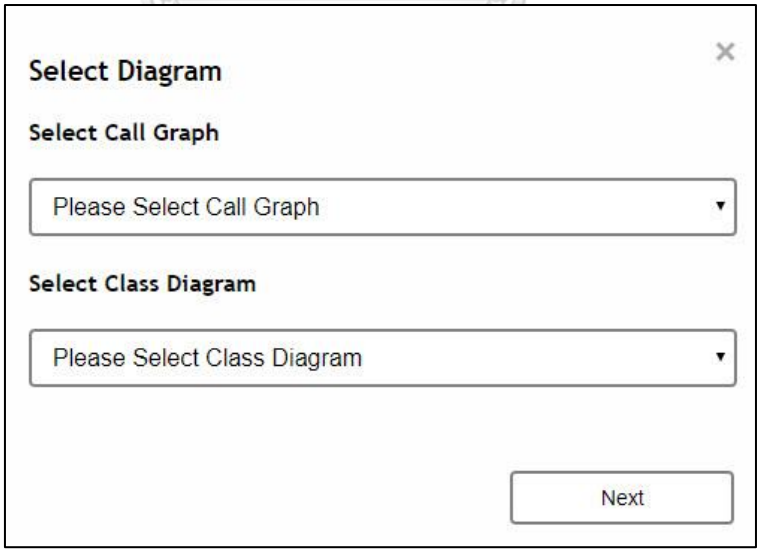


รูปที่ 4-57 หน้าต่างป๊อปอัพของการนำเข้าไฟล์

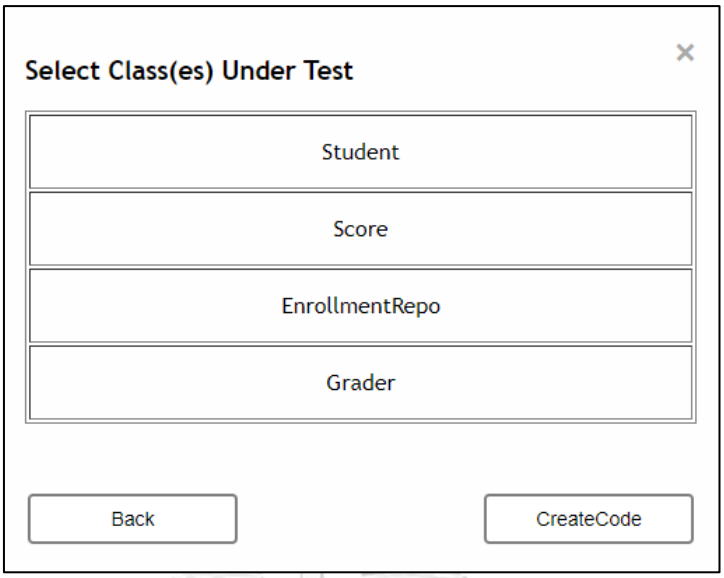


รูปที่ 4-58 ข้อความแจ้งเตือนเมื่อพบการอ้างอิงแผนภาพลำดับอื่น

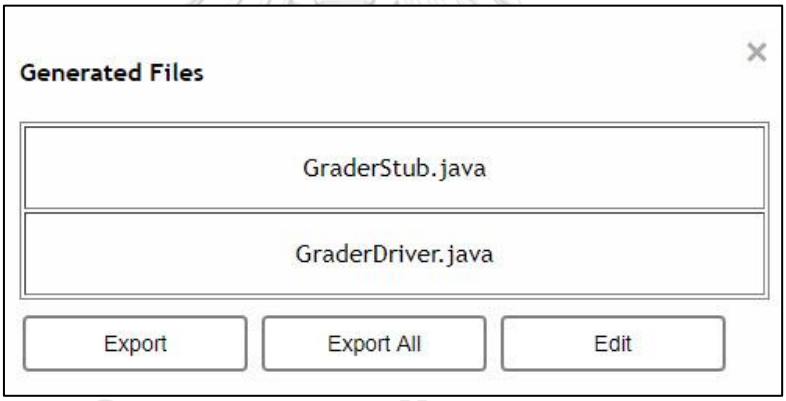
1.2 ปุ่ม Create Source Code ใช้ในการสร้างรหัสต้นฉบับของสตับและไดร์เวอร์ โดยเมื่อกดปุ่มตัวสร้างจะแสดงหน้าต่างป๊อปอัพเพื่อให้ผู้ทดสอบเลือกแผนภาพลำดับที่ต้องการทดสอบและแผนภาพคลาสที่สอดคล้องกับแผนภาพลำดับที่เลือกดังแสดงในรูปที่ 4-59 และเมื่อผู้ทดสอบกดปุ่ม Next ตัวสร้างจะแสดงรายการของอ็อบเจกต์ ในแผนภาพลำดับดังกล่าวให้ผู้ทดสอบเลือกคลาสภายใต้การทดสอบดังแสดงในรูปที่ 4-60 เมื่อผู้ทดสอบกด Next ตัวสร้างจะสร้างรหัสต้นฉบับของคลาสภายใต้การทดสอบที่เลือก และแสดงรายการของสตับและไดร์เวอร์ที่สร้างขึ้นมาดังแสดงในรูปที่ 4-61 ผู้ทดสอบสามารถกดปุ่ม Edit เพื่อแก้ไขรหัสต้นฉบับ หรือกดปุ่ม Export เพื่อส่งออกไฟล์ที่เลือก หรือกดปุ่ม Export All เพื่อส่งออกไฟล์รหัสต้นฉบับทั้งหมด โดยตัวสร้างจะบีบอัดไฟล์ในรูปแบบไฟล์ซิป (Zip) ก่อนส่งออกไฟล์



รูปที่ 4-59 หน้าต่างป๊อปอัพการเลือกแผนภาพ



รูปที่ 4-60 หน้าต่างป๊อปอัพการเลือกคลาสภายใต้การทดสอบ



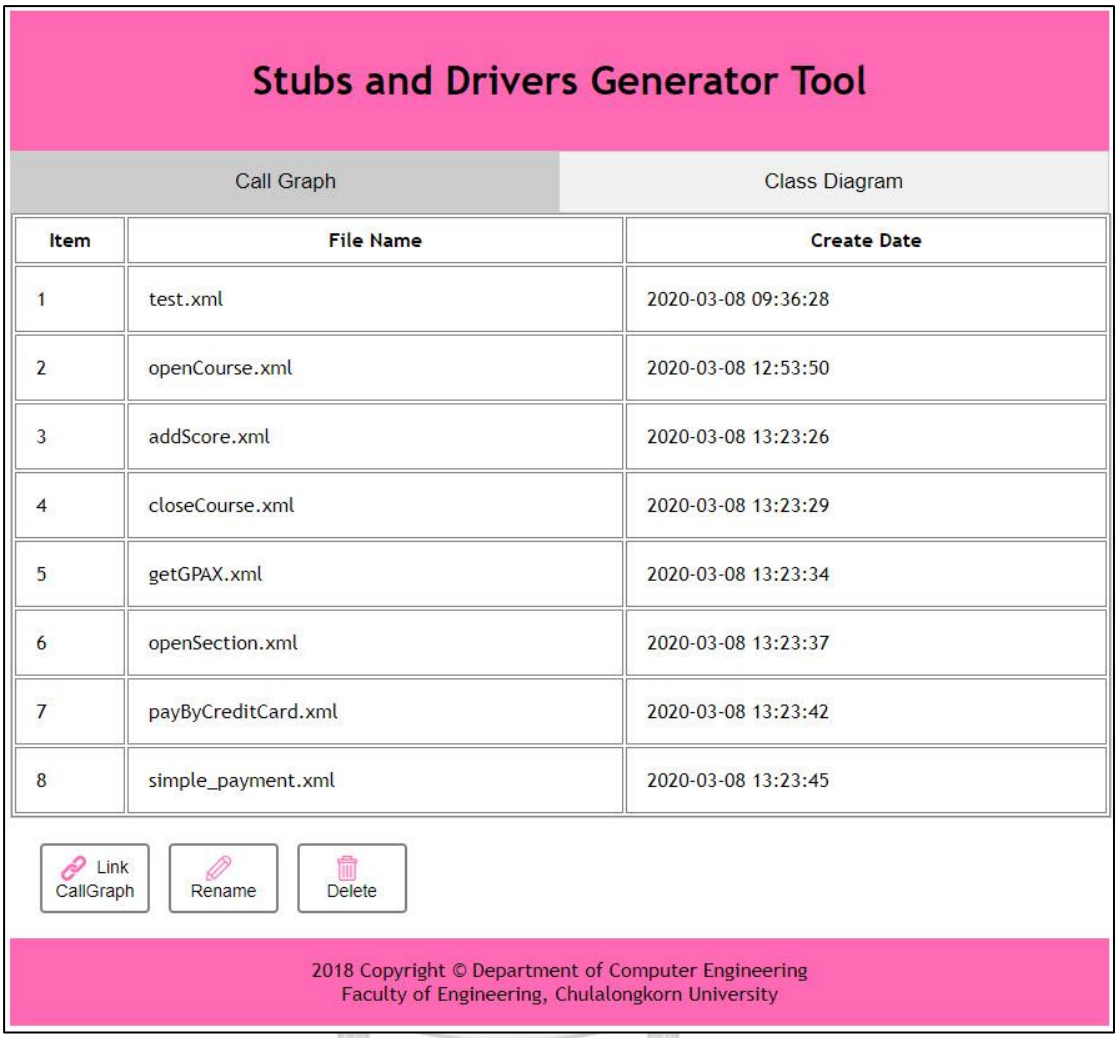
รูปที่ 4-61 หน้าต่างป๊อปอัพแสดงรายการของไฟล์รหัสต้นฉบับ

1.3 ปุ่ม XML Manager เป็นปุ่มที่นำทางไปหน้าต่างการจัดการไฟล์แผนภาพ

1.4 ปุ่ม Source Code Manager เป็นปุ่มที่นำทางไปหน้าต่างการจัดการไฟล์รหัสต้นฉบับ

2. หน้าต่างการจัดการแผนภาพ (Diagram Manager)

หน้าต่างการจัดการแผนภาพเป็นหน้าต่างที่ใช้แสดงแผนภาพลำดับและแผนภาพคลาสที่นำเข้ามาดังแสดงในรูปที่ 4-62 หน้าต่างนี้มีฟังก์ชันการใช้งาน 3 ฟังก์ชันได้แก่ การเชื่อมต่อแผนภาพลำดับ การลบแผนภาพ และการเปลี่ยนชื่อแผนภาพ โดยมีรายละเอียดดังต่อไปนี้

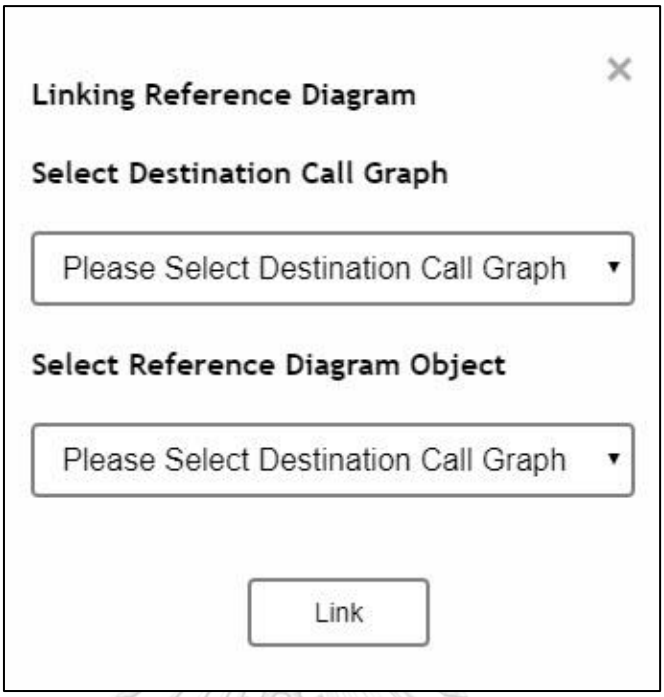


รูปที่ 4-62 หน้าต่างการจัดการแผนภาพ

2.1 การเชื่อมต่อแผนภาพลำดับ ผู้ทดสอบสามารถเชื่อมต่อแผนภาพลำดับที่มีการอ้างอิงถึงแผนภาพลำดับอื่นโดยเลือกแผนภาพลำดับในตารางและกดปุ่ม Link CallGraph ตัวสร้างจะแสดงหน้าต่างป๊อปอัพให้ผู้ทดสอบเลือกแผนภาพลำดับปลายทาง และอ็อบเจกต์ในแผนภาพลำดับต้นทางที่เป็นประเภทการอ้างอิงดังแสดงในรูปที่ 4-63

2.2 การลบแผนภาพ ผู้ทดสอบสามารถลบแผนภาพลำดับหรือแผนภาพคลาส โดยเลือกแผนภาพลำดับหรือแผนภาพคลาสในตารางและกดปุ่ม Delete

2.3 การเปลี่ยนชื่อแผนภาพ ผู้ทดสอบสามารถเปลี่ยนชื่อแผนภาพลำดับหรือแผนภาพ โดยเลือกแผนภาพลำดับหรือแผนภาพคลาสในตารางและกดปุ่ม Rename เมื่อกดปุ่ม Rename ตัวสร้างจะแสดงหน้าต่างป๊อปอัพให้ผู้ทดสอบใส่ชื่อไฟล์ใหม่ โดยหน้าต่างป๊อปอัพแสดงในรูปที่ 4-64

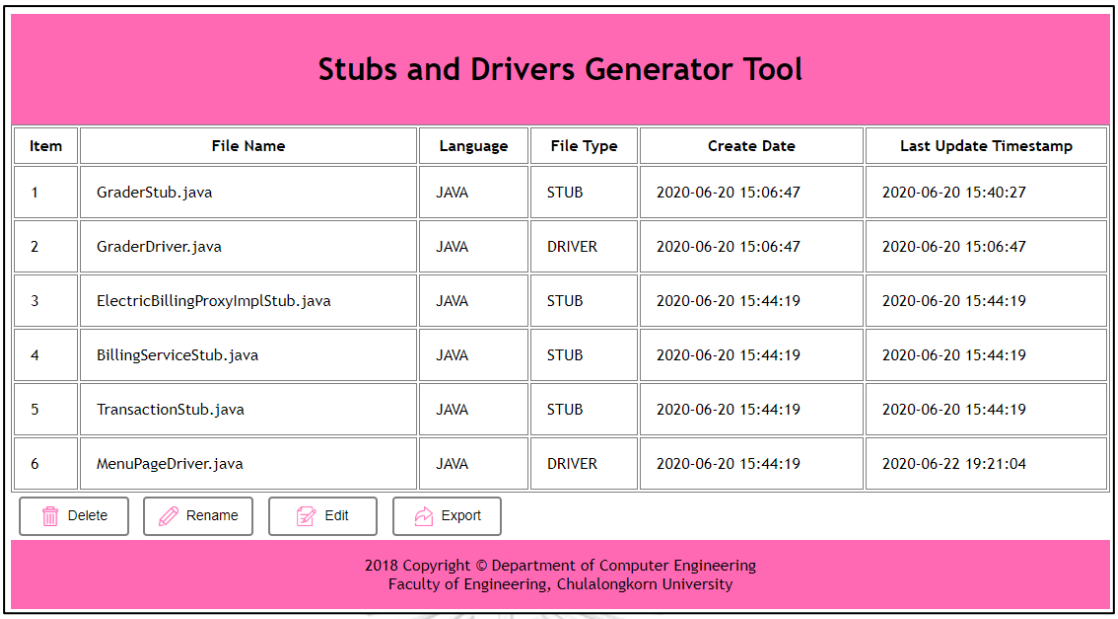


รูปที่ 4-63 หน้าต่างป๊อปอัพการเชื่อมต่อแผนภาพลำดับ

รูปที่ 4-64 หน้าต่างป๊อปอัพการเปลี่ยนชื่อแผนภาพ

3. หน้าต่างการจัดการไฟล์รหัสต้นฉบับ (Source Code Manager)

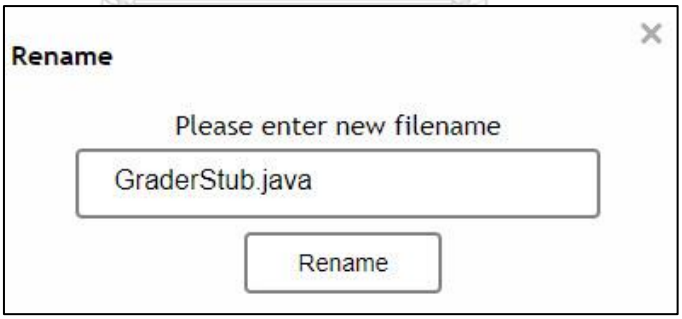
หน้าต่างการจัดการไฟล์รหัสต้นฉบับเป็นหน้าต่างที่ใช้แสดงรายการของไฟล์รหัสต้นฉบับที่เก็บไว้ในตัวสร้าง แสดงในรูปที่ 4-65 หน้าต่างนี้มีฟังก์ชันการใช้งาน 4 ฟังก์ชันได้แก่ การลบไฟล์ การเปลี่ยนชื่อไฟล์ การส่งออกไฟล์ และการแก้ไขไฟล์ โดยรายละเอียดของแต่ละฟังก์ชันมีดังต่อไปนี้



รูปที่ 4-65 หน้าต่างการจัดการไฟล์รหัสต้นฉบับ

3.1 การลบไฟล์ ผู้ทดสอบสามารถลบไฟล์รหัสต้นฉบับออกจากตัวสร้างโดยการเลือกไฟล์ในตารางและกดปุ่ม Delete

3.2 การเปลี่ยนชื่อไฟล์ ผู้ทดสอบสามารถเปลี่ยนชื่อไฟล์ได้โดยเลือกไฟล์ในตารางและกดปุ่ม Rename ตัวสร้างจะแสดงหน้าต่างป๊อปอัพเพื่อให้ผู้ทดสอบใส่ชื่อไฟล์ใหม่ดังรูปที่ 4-66



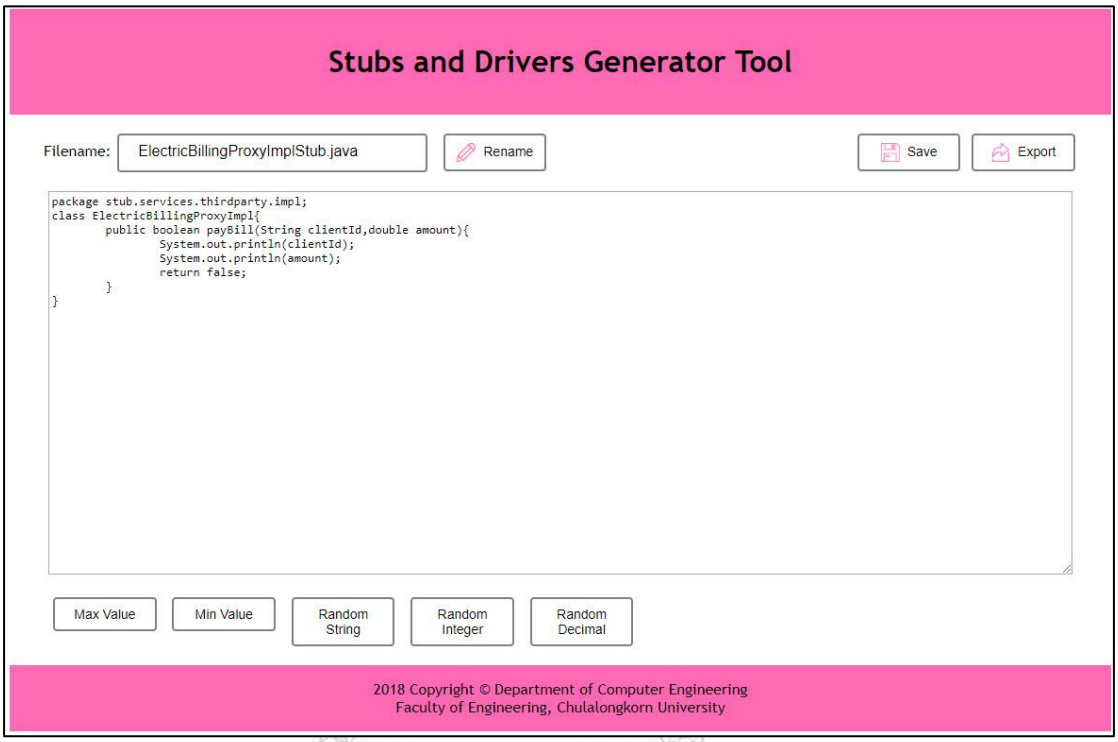
รูปที่ 4-66 หน้าต่างป๊อปอัพการเปลี่ยนชื่อไฟล์

3.3 การส่งออกไฟล์ ผู้ทดสอบสามารถส่งออกไฟล์ได้โดยการเลือกไฟล์ในตารางและกดปุ่ม Export

3.4 การแก้ไขไฟล์ ผู้ทดสอบสามารถแก้ไขรหัสต้นฉบับได้โดยการเลือกไฟล์ในตารางและกดปุ่ม Edit ตัวสร้างจะนำทางไปหน้าต่างการแก้ไขรหัสต้นฉบับ

4. หน้าต่างการแก้ไขรหัสต้นฉบับ (Code Editor)

หน้าต่างการแก้ไขรหัสต้นฉบับ เป็นหน้าต่างที่ใช้แก้ไขรหัสต้นฉบับโดยสามารถแทรกค่าคงที่หรือค่าสุ่มตามประเภทของข้อมูลได้ นอกจากนี้ยังสามารถเปลี่ยนชื่อไฟล์และส่งออกไฟล์ผ่านหน้าต่างนี้ได้ หน้าต่างการแก้ไขรหัสต้นฉบับแสดงในรูปที่ 4-67



รูปที่ 4-67 หน้าต่างการแก้ไขรหัสต้นฉบับ

บทที่ 5
การทดสอบตัวสร้าง

ในบทนี้จะกล่าวถึงการทดสอบเครื่องมือสร้างสตับและไดร์เวอร์ โดยใช้กรณีศึกษา 3 กรณี เพื่อทดสอบการสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาสในกรณีต่าง ๆ ที่แตกต่างกัน โดยจะกล่าวถึงขั้นตอนการทดสอบโดยภาพรวมและรายละเอียดของกรณีศึกษาทั้ง 3 กรณี

5.1 ขั้นตอนการทดสอบโดยภาพรวม

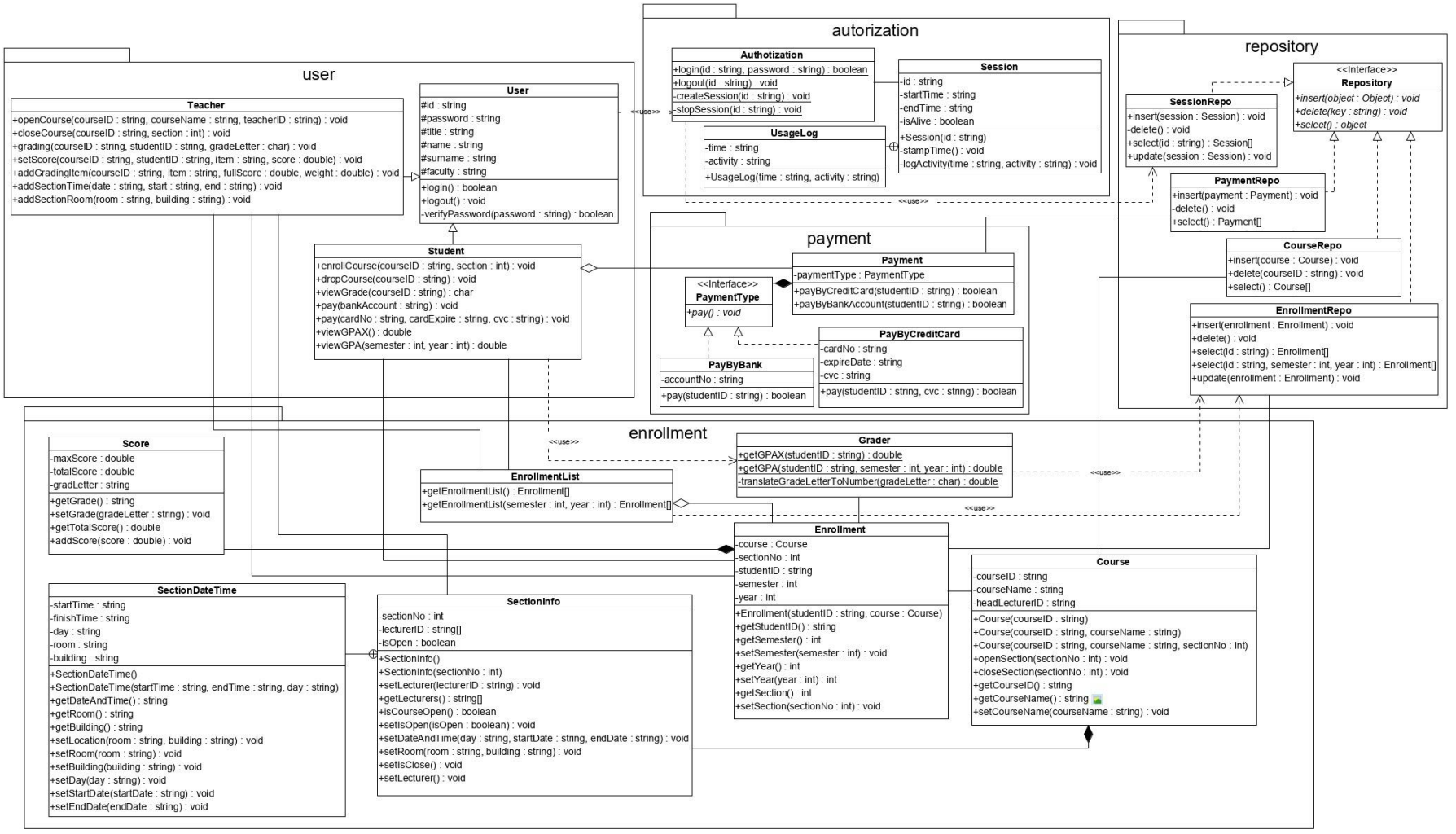
การทดสอบตัวสร้างในกรณีศึกษาต่าง ๆ มีขั้นตอนการทดสอบโดยรวมที่เหมือนกันโดยมีขั้นตอนหลัก 4 ขั้นตอนได้แก่

1. นำเข้าไฟล์แผนภาพลำดับและแผนภาพคลาส ในกรณีที่แผนภาพลำดับที่นำเข้ามีการอ้างอิงถึงแผนภาพลำดับอื่น ผู้ทดสอบต้องนำเข้าแผนภาพลำดับที่ถูกอ้างอิงและเชื่อมต่อเข้าด้วยกันก่อนดำเนินการขั้นตอนถัดไป
2. เลือกแผนภาพลำดับและแผนภาพภาพคลาสที่ต้องการทดสอบ ขั้นตอนนี้ผู้วิจัยได้ตั้งขอบเขตว่าแผนภาพคลาสที่เลือกต้องสอดคล้องกับแผนภาพลำดับที่เลือก
3. เลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบ
4. ตรวจสอบรหัสต้นฉบับที่ถูกสร้างขึ้น

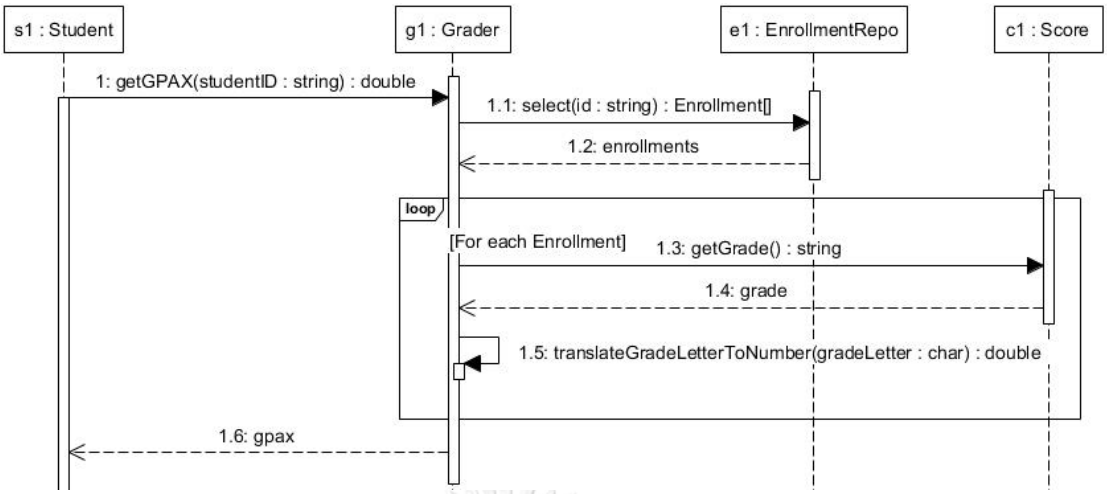
5.2 กรณีศึกษาที่ 1 ระบบการลงทะเบียนเรียนอย่างง่าย

กรณีศึกษาระบบลงทะเบียนเรียนอย่างง่ายเป็นระบบสมมติที่ใช้สำหรับลงทะเบียนเรียนของนักศึกษาโดยมีฟังก์ชันพื้นฐานได้แก่ การเพิ่มรายวิชา การลดรายวิชา และการดูเกรดเฉลี่ย สำหรับนักศึกษา และการเปิดรายวิชา การเปิดตอนเรียน การปิดวิชา และการกรอกคะแนน สำหรับอาจารย์ โดยแผนภาพคลาสของระบบลงทะเบียนเรียนแสดงในรูปที่ 5-1

สำหรับกรณีศึกษานี้ผู้ทดสอบนำเข้าไฟล์แผนภาพคลาสชื่อ register_system.xml ผ่านส่วนต่อประสานผู้ใช้ และไฟล์แผนภาพลำดับชื่อ getGPAX.xml ซึ่งเป็นแผนภาพลำดับแสดงการดูเกรดของนักศึกษาแสดงในรูปที่ 5-2 (การทวนสอบหลังจากการนำเข้าไฟล์แผนภาพลำดับและแผนภาพคลาสแสดงรายละเอียดในภาคผนวก ข.) กรณีศึกษานี้จะใช้แผนภาพคลาสและแผนภาพลำดับดังกล่าวในการทดสอบโดยมีรายละเอียดของแต่ละกรณีทดสอบดังต่อไปนี้



รูปที่ 5-1 แผนภาพคลาสชื่อ register_system.xml

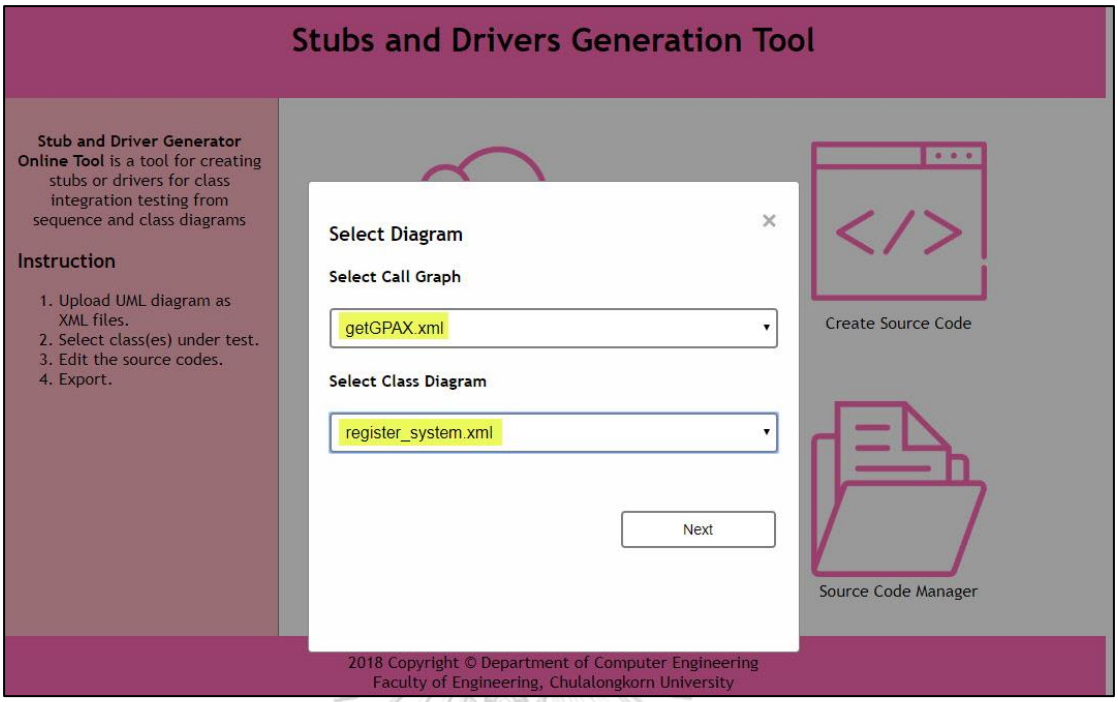


รูปที่ 5-2 แผนภาพลำดับชื่อ getGPAX.xml

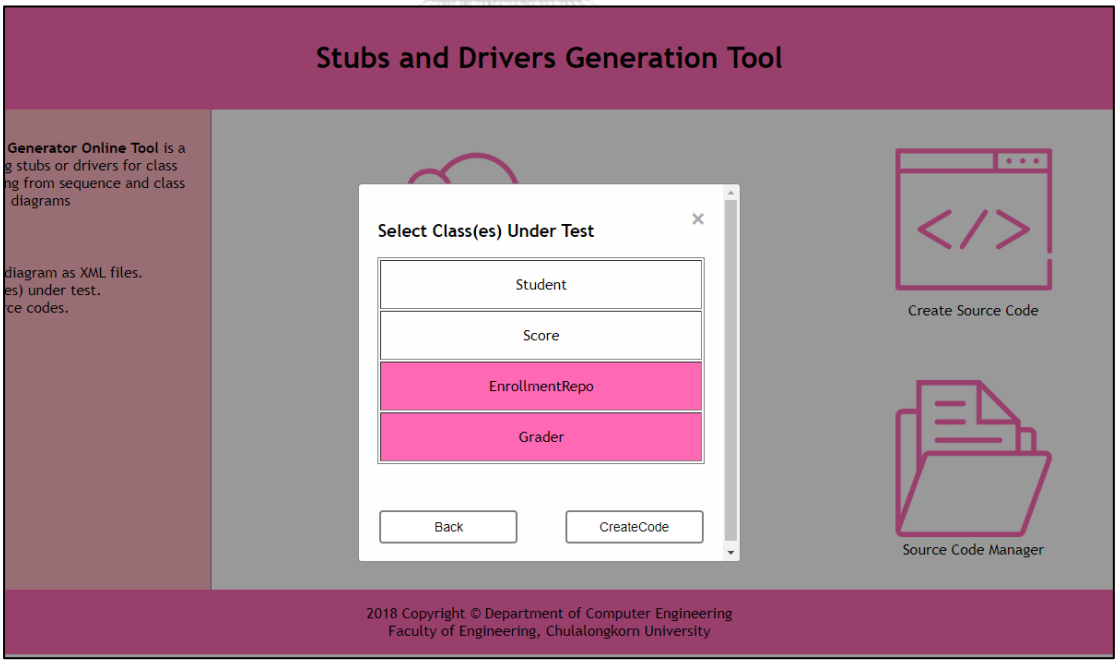
5.2.1 กรณีทดสอบ – เลือกกลุ่มของคลาสภายใต้การทดสอบที่อยู่ติดกัน

เริ่มต้นผู้ทดสอบกดปุ่ม Create Source Code ของตัวสร้าง ตัวสร้างจะให้เลือกแผนภาพลำดับที่ต้องการทดสอบและแผนภาพคลาสที่เกี่ยวข้องโดยผู้ทดสอบเลือกแผนภาพลำดับ getGPAX.xml และแผนภาพคลาส register_system.xml ดังแสดงในรูปที่ 5-3 จากนั้นผู้ทดสอบกดปุ่ม Next ตัวสร้างจะแสดงรายการของอ็อบเจกต์ ในแผนภาพลำดับ getGPAX.xml และผู้ทดสอบเลือกคลาส Grader และคลาส EnrollmentRepo เป็นกลุ่มของคลาสภายใต้การทดสอบดังแสดงในรูปที่ 5-4 เมื่อผู้ทดสอบกดปุ่ม Create Code ตัวสร้างจะแสดงรายการของไฟล์รหัสต้นฉบับของสตับและไดร์เวอร์ที่ต้องใช้ทดสอบคลาสดังกล่าวดังแสดงในรูปที่ 5-5

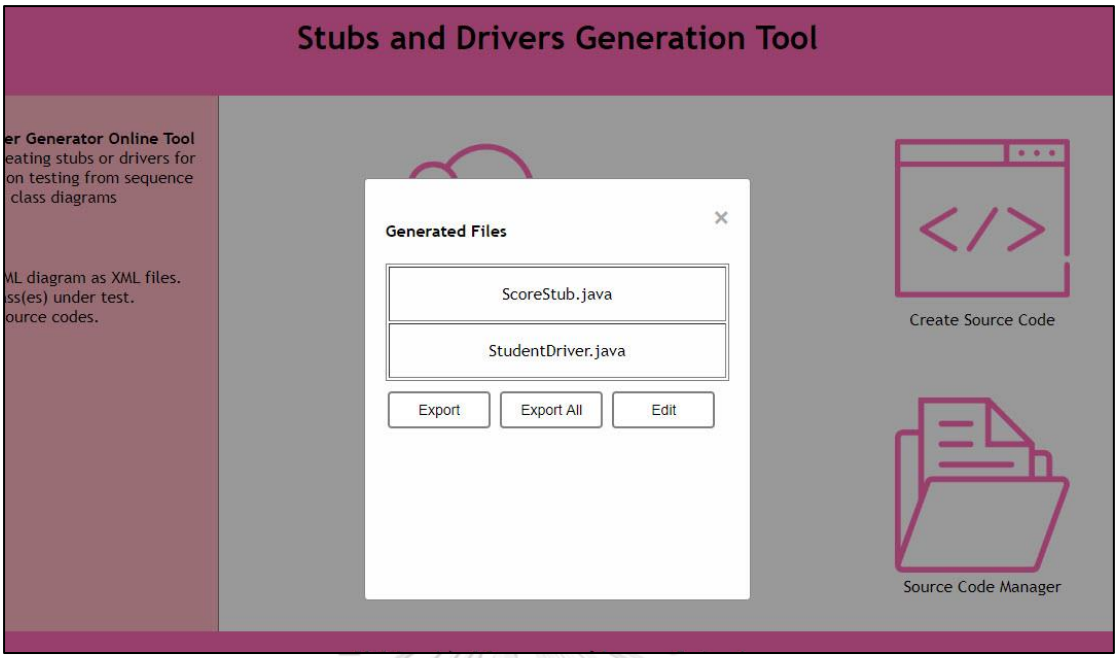
จากรูปที่ 5-2 หากเลือกคลาส EnrollmentRepo และคลาส Grader เป็นคลาสภายใต้การทดสอบจะต้องใช้สตับของคลาส Score และไดร์เวอร์ของคลาส Student ตัวสร้างจะสร้างสตับของคลาส Score โดยใช้ชื่อว่า ScoreStub เพื่อสื่อว่าเป็นสตับของคลาส Score และสร้างไดร์เวอร์ของคลาส Student โดยใช้ชื่อว่า StudentDriver เพื่อสื่อว่าเป็นไดร์เวอร์ที่จำลองการเรียกใช้งานจากคลาส Student จากรูปที่ 5-5 เมื่อส่งออกไฟล์ทั้งสอบด้วยปุ่ม Export All ตัวสร้างจะบีบอัดไฟล์ทั้งสองในรูปแบบไฟล์ซิป (Zip) รูปที่ 5-6 แสดงไฟล์ซิปและไฟล์รหัสต้นฉบับข้างต้นในไฟล์ ไฟล์รหัสต้นฉบับ ScoreStub.java จะประกอบด้วยเมท็อด getGrade เพียงเมท็อดเดียว รูปที่ 5-7 แสดงเนื้อหาของไฟล์ ScoreStub.java สำหรับไฟล์รหัสต้นฉบับ StudentDriver.java จะประกอบด้วยเมท็อดหนึ่งเมท็อดชื่อ testGetGPAXInGrader โดยเมท็อดดังกล่าวจะเรียกใช้งานเมท็อด getGPAX ในคลาส Grader รูปที่ 5-8 แสดงเนื้อหาของไฟล์ StudentDriver.java



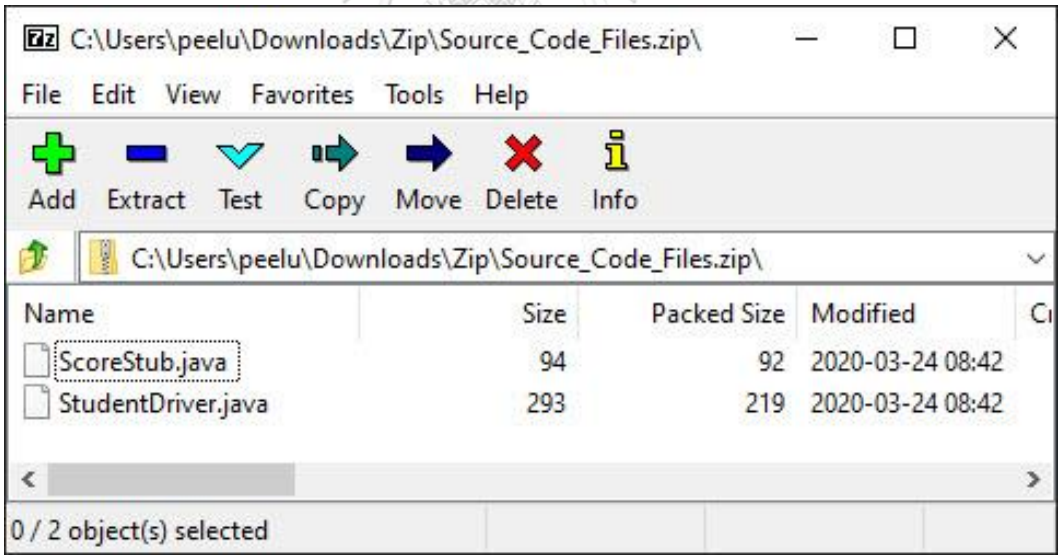
รูปที่ 5-3 การเลือกแผนภาพลำดับ getGPAX.xml และแผนภาพคลาส register_system.xml เพื่อทดสอบ



รูปที่ 5-4 การเลือกคลาส EnrollmentRepo และคลาส Grader เป็นคลาสภายใต้การทดสอบ



รูปที่ 5-5 รายการของรหัสต้นฉบับของสตับและไดร์เวอร์สำหรับทดสอบคลาส EnrollmentRepo และคลาส Grader



รูปที่ 5-6 ไฟล์ซิปของรหัสต้นฉบับ

```
package stub.enrollment;
public class ScoreStub{
    public string getGrade(){
        return "FZ3bNJZfu";
    }
}
```

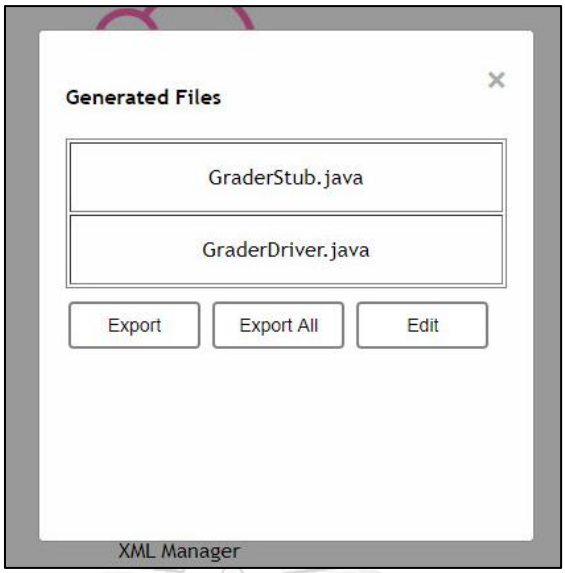
รูปที่ 5-7 รหัสต้นฉบับของสตับของคลาส Score

```
package driver.user;
/*--- AUTO IMPORT START HERE ---*/
import enrollment.Grader;
/*--- AUTO IMPORT END HERE ---*/
public class StudentDriver{
    @Test
    public void testGetGPAXInGrader(){
        double actualResult = Grader.getGPAX("akN");
        assertEquals(4.5123631461671,actualResult,0.285);
    }
}
```

รูปที่ 5-8 รหัสต้นฉบับของไดร์เวอร์ตัวแทนคลาส Student

5.2.2 กรณีทดสอบ – เลือกกลุ่มของคลาสภายใต้การทดสอบที่ต้องใช้สตับและไดร์เวอร์ที่แสดงถึงคลาสเดียวกัน

กรณีทดสอบนี้ผู้ทดสอบต้องการเลือกคลาสภายใต้การทดสอบที่ต้องใช้สตับและไดร์เวอร์ที่มีพื้นฐานมาจากคลาสเดียวกัน จากรูปที่ 5-2 หากผู้ทดสอบเลือกคลาส Student และ EnrollmentRepo เป็นคลาสภายใต้การทดสอบ ตัวสร้างจะสร้างสตับและไดร์เวอร์ของคลาส Grader ขึ้นมาดังแสดงในรูปที่ 5-9 โดยไฟล์รหัสต้นฉบับของสตับของคลาส Grader มีชื่อว่า GraderStub.java ซึ่งมีเมท็อดหนึ่งเมท็อดได้แก่ เมท็อด getGPAX ดังแสดงในรูปที่ 5-10 และไฟล์รหัสต้นฉบับของไดร์เวอร์ของคลาส Grader จะมีชื่อว่า GraderDriver.java ซึ่งประกอบด้วยมธอดสองเมท็อดเรียกใช้งานเมท็อด select ในคลาส EnrollmentRepo เนื่องจากคลาส EnrollmentRepo มีการโอเวอร์โหลดดิ้งของเมท็อด select ดังแสดงในแผนภาพคลาสรูปที่ 5-1 ทำให้ตัวสร้างเรียกใช้งานเมท็อดดังกล่าวทั้งสองรูป



รูปที่ 5-9 รายการของรหัสต้นฉบับสำหรับทดสอบคลาส Student และคลาส EnrollmentRepo

```java
package stub.enrollment;
public class GraderStub{
    public static double getGPAX(String studentID){
        System.out.println(studentID);
        return 1.0080509064458;
    }
}
```

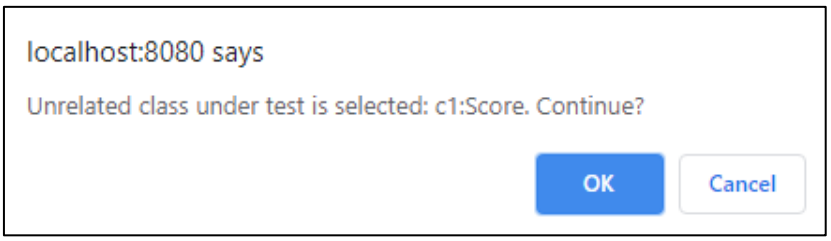
รูปที่ 5-10 รหัสต้นฉบับของสตับของคลาส Grader


```java
package driver.enrollment;
/*--- AUTO IMPORT START HERE ---*/
import repository.EnrollmentRepo;
/*--- AUTO IMPORT END HERE ---*/
class GraderDriver{
    @Test
    public void testSelectInEnrollmentRepo(){
        EnrollmentRepo enrollmentRepo = new EnrollmentRepo();
        Enrollment actualResult = enrollmentRepo.select("gQKTNb");
        assertEquals(null,actualResult);
    }
    @Test
    public void testSelectInEnrollmentRepo1(){
        EnrollmentRepo enrollmentRepo = new EnrollmentRepo();
        Enrollment actualResult = enrollmentRepo.select("9BGEmM41Y",244909650,186629185);
        assertEquals(null,actualResult);
    }
}
```

รูปที่ 5-11 ไฟล์รหัสต้นฉบับของไดร์เวอร์ตัวแทนคลาส Grader

5.2.3 กรณีทดสอบ – เลือกกลุ่มของคลาสภายใต้การทดสอบที่ไม่เกี่ยวข้องกัน

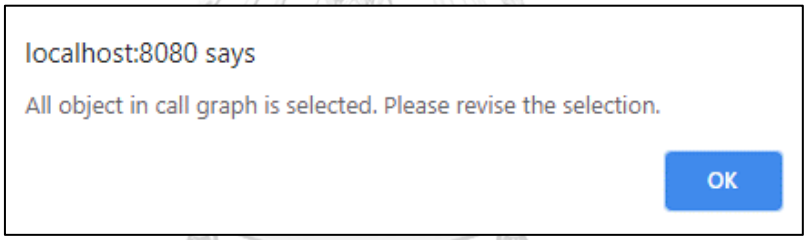
กรณีทดสอบนี้ผู้ทดสอบเลือกคลาสภายใต้การทดสอบที่ไม่มีการเรียกใช้งานถึงกัน จากแผนภาพลำดับรูปที่ 5-2 หากเลือกคลาส Score และคลาส EnrollmentRepo เป็นคลาสภายใต้การทดสอบ ตัวสร้างจะแสดงข้อความเตือนดังรูปที่ 5-12



รูปที่ 5-12 ข้อความแจ้งเตือนเมื่อเลือกคลาสภายใต้การทดสอบที่ไม่เกี่ยวข้องกัน

5.2.4 กรณีทดสอบ – เลือกกลุ่มของคลาสภายใต้การทดสอบทุกคลาสในแผนภาพ

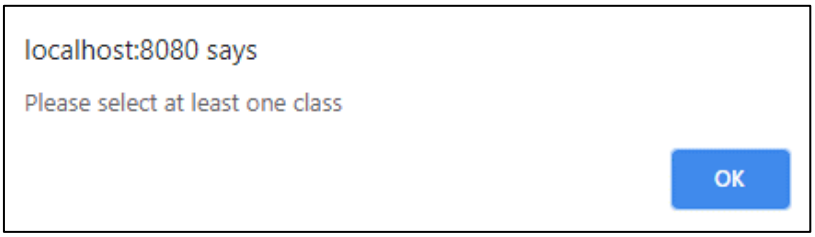
กรณีทดสอบนี้ผู้ทดสอบเลือกคลาสทุกคลาสในแผนภาพลำดับเป็นคลาสภายใต้การทดสอบ ตัวสร้างจะแสดงข้อความเตือนดังแสดงในรูปที่ 5-13



รูปที่ 5-13 ข้อความแจ้งเตือนเมื่อเลือกคลาสทุกคลาสในแผนภาพลำดับ

5.2.5 กรณีทดสอบ – ไม่เลือกคลาสใด ๆ เป็นคลาสภายใต้การทดสอบ

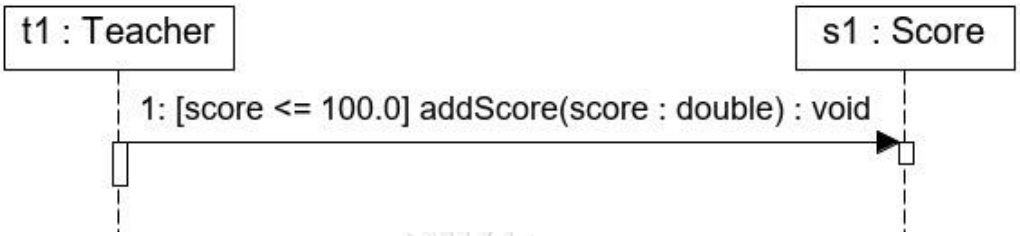
กรณีทดสอบนี้ ผู้ทดสอบกดปุ่ม Create Code เพื่อสร้างรหัสต้นฉบับโดยไม่เลือกคลาสใดเป็นคลาสภายใต้การทดสอบ ตัวสร้างจะแสดงข้อความแจ้งเตือนเพื่อให้ผู้ทดสอบเลือกคลาสอย่างน้อยหนึ่งคลาสเป็นคลาสภายใต้การทดสอบดังแสดงในรูปที่ 5-14



รูปที่ 5-14 ข้อความแจ้งเตือนเมื่อผู้ทดสอบไม่เลือกคลาสใด ๆ เป็นคลาสภายใต้การทดสอบ

5.2.6 กรณีทดสอบ – เมสเสจถูกกำกับด้วยเงื่อนไขปฏิสมพันธ์

กรณีทดสอบนี้ผู้ทดสอบต้องการทดสอบการสร้างไดร์เวอร์ที่มีการ์ดคอนดิชัน โดยผู้ทดสอบได้นำเข้าไฟล์แผนภาพลำดับชื่อ addScore.xml โดยแผนภาพลำดับแสดงในรูปที่ 5-15



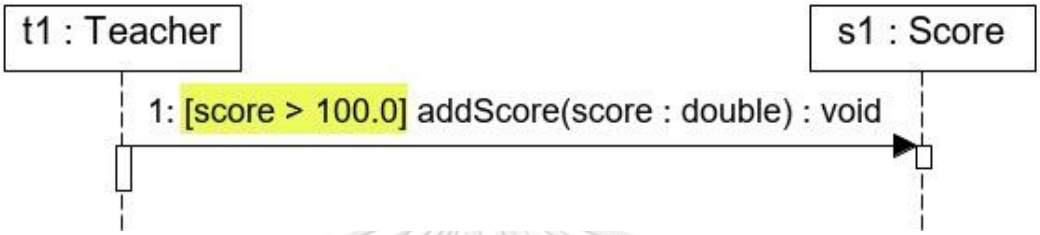
รูปที่ 5-15 แผนภาพลำดับ AddScore.xml

จากรูปที่ 5-15 เมสเสจ addScore ถูกกำกับด้วยการ์ดคอนดิชัน score <= 100.0 กล่าวคือ เมสเสจจะถูกส่งเมื่อ score ซึ่งเป็นพารามิเตอร์ของเมสเสจมีค่าน้อยกว่าหรือเท่ากับ 100 เพราะฉะนั้น หากเลือกคลาส Score เป็นคลาสภายใต้การทดสอบ ไดร์เวอร์ของคลาส Teacher ต้องส่งค่าพารามิเตอร์ที่น้อยกว่าหรือเท่ากับ 100 เท่านั้น รูปที่ 5-16 แสดงรหัสต้นฉบับของไดร์เวอร์ของคลาส Teacher ที่มีชื่อว่า TeacherDriver.java ที่มีการเรียกเมท็อด addScore โดยส่งค่าแบบสุ่มที่น้อยกว่าหรือเท่ากับ 100

```
package driver.user;
/*--- AUTO IMPORT START HERE ---*/
import enrollment.Score;
/*--- AUTO IMPORT END HERE ---*/
class TeacherDriver{
    @Test
    public void testAddScoreInScore(){
        Score score = new Score();
        score.addScore(57.9);
    }
}
```

รูปที่ 5-16 รหัสต้นฉบับของไดร์เวอร์ของคลาส Teacher

จากนั้นผู้ทดสอบได้แก้ไขแผนภาพลำดับในรูปที่ 5-15 โดยเปลี่ยนการ์ดคอนดิชันจาก score <= 100 เป็น score > 100 และนำเข้าตัวสร้างใหม่ รูปที่ 5-17 แสดงแผนภาพลำดับ AddScore.xml ที่ถูกแก้ไขแล้ว และผู้ทดสอบได้ลบไฟล์รหัสต้นฉบับ TeacherDriver.java ออก เพื่อให้ตัวสร้าง สร้างไฟล์รหัสต้นฉบับขึ้นมาใหม่ จากนั้นผู้ทดสอบได้สร้างไดร์เวอร์เพื่อทดสอบคลาส Score ขึ้นมาใหม่พบว่าพารามิเตอร์ที่ไดร์เวอร์ส่งมีค่ามากกว่า 100 รูปที่ 5-18 แสดงรหัสต้นฉบับของไดร์เวอร์ของคลาส Teacher ที่ถูกสร้างใหม่



รูปที่ 5-17 แผนภาพลำดับ addScore.xml ที่ถูกแก้ไขการ์ดคอนดิชัน

```
package driver.user;
/*--- AUTO IMPORT START HERE ---*/
import enrollment.Score;
/*--- AUTO IMPORT END HERE ---*/
class TeacherDriver{
    @Test
    public void testAddScoreInScore(){
        Score score = new Score();
        score.addScore(192.4);
    }
}
```

รูปที่ 5-18 รหัสต้นฉบับของไดร์เวอร์ของคลาส Teacher ที่ถูกสร้างขึ้นมาใหม่

5.2.7 การทดสอบแบบรวมระหว่างระบบจำลองกับสตับและไดร์เวอร์

หลังจากสร้างสตับและไดร์เวอร์จากตัวสร้างแล้วผู้ทดสอบได้ทดลองนำสตับและไดร์เวอร์ที่สร้างขึ้นทดสอบแบบรวมกับระบบจำลอง โดยทดสอบรวมคลาส Student และคลาส EnrollmentRepo กับ สตับและไดร์เวอร์ที่สร้างขึ้นจากตัวสร้าง รูปที่ 5-19 แสดงรหัสต้นฉบับของคลาส Student ที่พัฒนาเสร็จแล้วบางส่วน โดยคลาสดังกล่าวมีการเรียกใช้งานคลาส Grader ซึ่งยัง

พัฒนาไม่เสร็จและผู้ทดสอบได้เปลี่ยนให้คลาส Student เรียกใช้งานคลาส GraderStub แทน ดังแสดงในรูปที่ 5-20

```
public class Student extends User{
    public double viewGPAX(){
        double gpax = Grader.getGPAX(id);
        return gpax;
    }
}
```

รูปที่ 5-19 คลาส Student ที่พัฒนาเสร็จแล้วบางส่วน

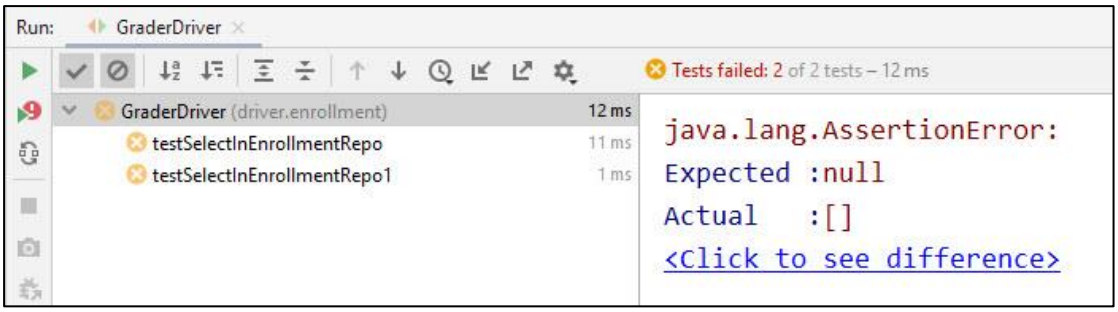
```
public class Student extends User{
    public double viewGPAX(){
        //double gpax = Grader.getGPAX(id);
        double gpax = GraderStub.getGPAX("123456789");
        return gpax;
    }
}
```

รูปที่ 5-20 คลาส Student ที่เรียกใช้สตับของคลาส Grader

ถัดมาผู้ทดสอบได้ทดลองเรียกใช้งานคลาส EnrollmentRepo ด้วยไดร์เวอร์ที่สร้างจากตัวสร้างแสดงในรูปที่ 5-11 โดยมีการแก้ไขเล็กน้อยโดยเปลี่ยนการประกาศตัวแปร actualResult เป็นรายการ (List) เนื่องจากเมท็อด select ส่งค่ากลับมาเป็นรายการของ Enrollment จึงจำเป็นต้องแก้ไขรหัสต้นฉบับเพื่อให้สามารถทำงานได้ รูปที่ 5-21 แสดงรหัสต้นฉบับของ GraderDriver ที่ถูกแก้ไข และรูปที่ 5-22 แสดงผลการเรียกใช้งานไดร์เวอร์ซึ่งสามารถเรียกใช้งานได้ แต่ผลการเปรียบเทียบค่าของผลลัพธ์ไม่ถูกต้องเนื่องจากค่าทั้งหมดเป็นค่าสุ่มจึงให้ผลลัพธ์ที่ไม่ถูกต้อง

```
public class GraderDriver{
    @Test
    public void testSelectInEnrollmentRepo(){
        EnrollmentRepo enrollmentRepo = new EnrollmentRepo();
        List<Enrollment> actualResult = enrollmentRepo.select("gQKTNb");
        assertEquals(null,actualResult);
    }
    @Test
    public void testSelectInEnrollmentRepo1(){
        EnrollmentRepo enrollmentRepo = new EnrollmentRepo();
        List<Enrollment> actualResult = enrollmentRepo.select("9BGEmM41Y",244909650,186629185);
        assertEquals(null,actualResult);
    }
}
```

รูปที่ 5-21 รหัสต้นฉบับ GraderDriver.java ที่มีการแก้ไข

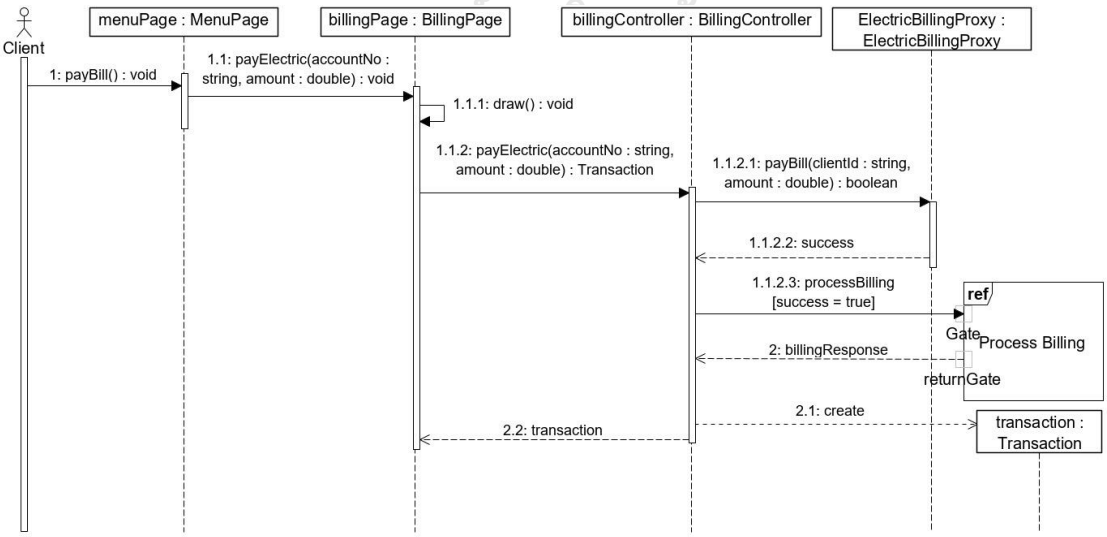


รูปที่ 5-22 ผลลัพธ์การเรียกใช้งาน GraderDriver

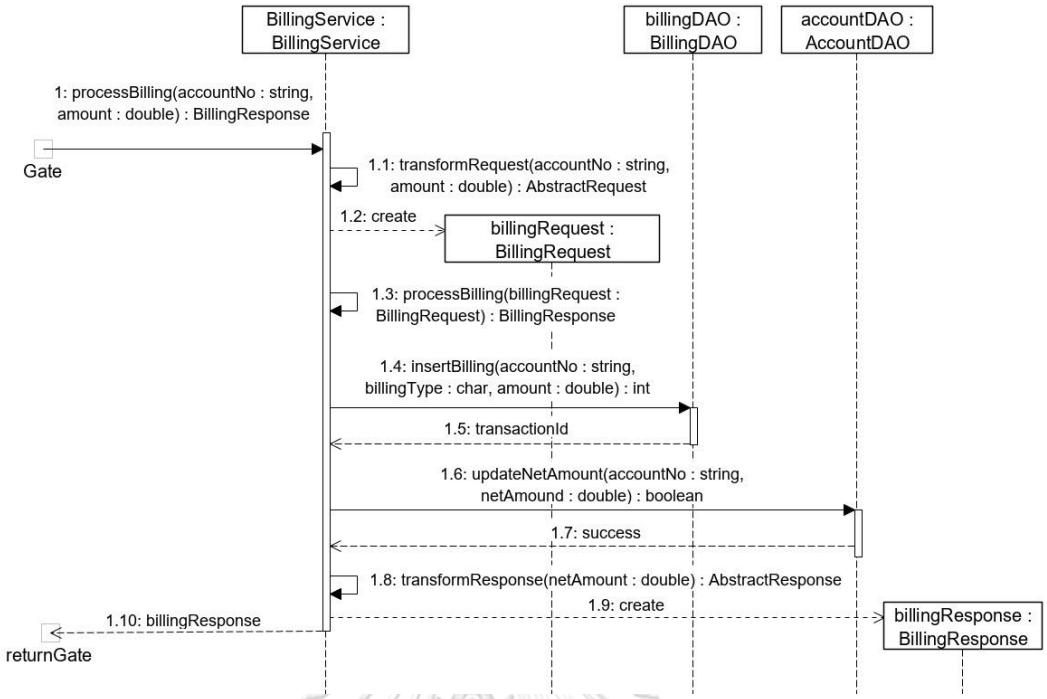
5.3 กรณีศึกษาที่ 2 ระบบธนาคาร

กรณีศึกษาระบบธนาคารเป็นระบบธนาคารจำลอง ที่ประกอบด้วยฟังก์ชันได้แก่ การฝากเงิน การถอนเงิน การโอนเงิน และการจ่ายบิล โดยการจ่ายบิลแบ่งเป็นการจ่ายค่าน้ำ ค่าไฟฟ้า และค่าโทรศัพท์ ซึ่งระบบอยู่ในรูปของเว็บแอปพลิเคชัน โดยรายละเอียดของระบบแผนภาพคลาสของระบบแสดงในภาคผนวก ค.

สำหรับกรณีศึกษานี้ ผู้ทดสอบได้นำเข้าไฟล์แผนภาพคลาส และแผนภาพลำดับของการจ่ายบิลค่าไฟฟ้า ซึ่งแผนภาพลำดับดังกล่าวมีการอ้างอิงถึงแผนภาพลำดับอื่นผ่านอ็อบเจกต์ ชื่อ Process Billing ดังแสดงในรูปที่ 5-23 และผู้ทดสอบได้นำเข้าไฟล์แผนภาพลำดับของการประมวลผลการทำธุรกรรมประเภทการจ่ายบิลดังแสดงในรูปที่ 5-24 และเป็นแผนภาพลำดับที่แผนภาพลำดับการจ่ายบิลค่าไฟฟ้าอ้างอิงถึง กรณีศึกษานี้จะใช้แผนภาพคลาสและแผนภาพลำดับดังกล่าวในการทดสอบโดยมีรายละเอียดของแต่ละกรณีทดสอบดังต่อไปนี้



รูปที่ 5-23 แผนภาพลำดับการจ่ายค่าไฟฟ้า



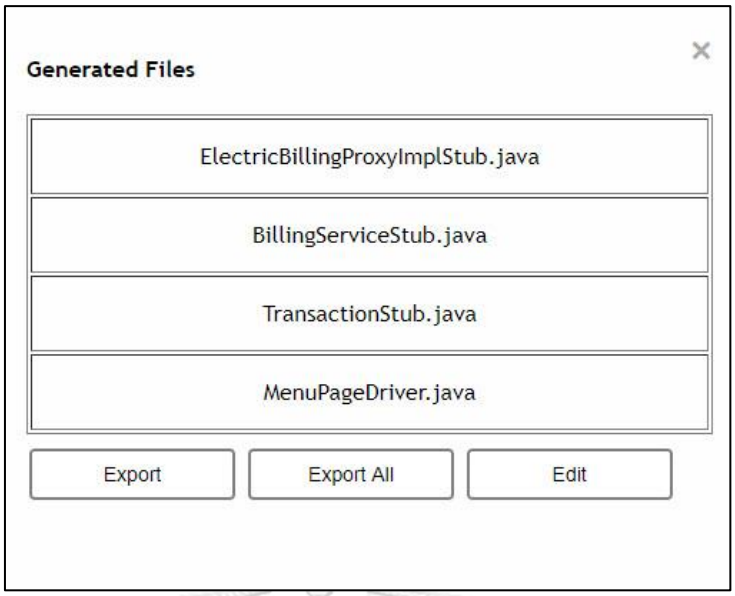
รูปที่ 5-24 แผนภาพลำดับชื่อ Process Billing

5.3.1 กรณีทดสอบ – คลาสภายใต้การทดสอบเรียกใช้คลาสในแผนภาพลำดับอื่น

กรณีทดสอบนี้ ผู้ทดสอบเลือกคลาส BillingPage และ BillingController ในแผนภาพลำดับรูปที่ 5-23 เป็นกลุ่มของคลาสภายใต้การทดสอบ ซึ่งจากรูปที่ 5-23 พบว่าจะต้องใช้สตับทั้งหมด 3 ตัว ได้แก่ สตับของคลาส ElectricBillingProxy แต่เนื่องจากคลาส ElectricBillingProxy เป็นคลาสนามธรรม ตัวสร้างจะสร้างสตับของคลาส ElectricBillingProxyImpl ขึ้นมาแทน สตับของคลาส Transaction และสตับที่เป็นตัวแทนของแผนภาพลำดับ Process Billing จากรูปที่ 5-24 คลาสแรกที่ปรากฏในแผนภาพลำดับ Process Billing ได้แก่คลาส BillingService เพราะฉะนั้นตัวสร้างจะสร้างสตับของคลาส BillingService นอกจากนี้ยังต้องใช้ไดร์เวอร์ที่เป็นตัวแทนของคลาส MenuPage ด้วย รูปที่ 5-25 แสดงผลลัพธ์จากการสร้างสตับและไดร์เวอร์สำหรับกลุ่มของคลาสภายใต้การทดสอบดังกล่าว

สำหรับรหัสต้นฉบับของสตับและไดร์เวอร์แต่ละไฟล์ในรูปที่ 5-25 มีรายละเอียดดังต่อไปนี้

1. ElectricBillingProxyImpleStub.java เป็นสตับที่สร้างขึ้นมาเป็นตัวแทนของคลาส ElectricBillingProxy แต่เนื่องจากคลาส ElectricBillingProxy เป็นคลาสนามธรรมจึงจำเป็นต้องสร้างสตับของคลาส ElectricBillingProxyImpl ซึ่งเป็นคลาสรูปธรรมที่สืบทอดมาจากคลาสดังกล่าวขึ้นมาแทน รูปที่ 5-26 แสดงรหัสต้นฉบับของสตับของคลาส Electric-BillingProxyImpl



รูปที่ 5-25 รายการของสตับและไดร์เวอร์ที่ใช้ทดสอบคลาส BillingPage และ BillingController

```
package stub.services.thirdparty.impl;
public class ElectricBillingProxyImplStub{
    public boolean payBill(String clientId,double amount){
        System.out.println(clientId);
        System.out.println(amount);
        return true;
    }
}
```

รูปที่ 5-26 รหัสต้นฉบับของสตับของคลาส ElectricBillingProxyImpl

2. TransactionStub.java เป็นสตับที่สร้างขึ้นมาเป็นตัวแทนของคลาส Transaction โดยจากรูปที่ 5-23 คลาส BillingController ส่งเมสเสจ create สร้างคลาส Transaction ขึ้นมาแต่เนื่องจากคลาส Transaction ไม่มีการระบุโครงสร้างของคอนสตรัคเตอร์ (Constructor) ในแผนภาพคลาส ตัวสร้างจึงใช้คอนสตรัคเตอร์เริ่มต้น (Default Constructor) [19] ของคลาสดังกล่าวแทน ตัวสร้างจึงไม่สร้างรหัสต้นฉบับของคอนสตรัคเตอร์เริ่มต้นขึ้นมาดังแสดงในรูปที่ 5-27

```
package stub.businessbean;
public class TransactionStub{
}
```

รูปที่ 5-27 รหัสต้นฉบับของสตับของคลาส Transaction

3. BillingServiceStub.java เป็นสตับที่สร้างขึ้นมาเป็นตัวแทนของคลาส BillingService ซึ่งคลาส BillingService ก็เป็นตัวแทนของแผนภาพลำดับ Process Billing โดยรหัสต้นฉบับของสตับของคลาส BilliingService แสดงในรูปที่ 5-28 เนื่องเมท็อด processBilling มีการฟ้องรูปของเมท็อด ตัวสร้างจึงสร้างรหัสต้นฉบับของเมท็อดทั้งกล่าวทุกรูปแบบ

```
package stub.services;
public class BillingServiceStub{
    public BillingResponse processBilling(BillingRequest billingRequest){
        System.out.println(billingRequest);
        return null;
    }
    public BillingResponse processBilling(String accountNo,double amount){
        System.out.println(accountNo);
        System.out.println(amount);
        return null;
    }
}
```

รูปที่ 5-28 รหัสต้นฉบับของสตับของคลาส BillingService

4. MenuPageDriver.java เป็นไดร์เวอร์ที่สร้างขึ้นมาเป็นตัวแทนของคลาส MenuPage ที่เรียกใช้งานคลาส BillingPage โดยรหัสต้นฉบับของไดร์เวอร์ของคลาส MenuPage แสดงในรูปที่ 5-29

```
package driver.view;
/*--- AUTO IMPORT START HERE ---*/
import view.billing.BillingPage;
/*--- AUTO IMPORT END HERE ---*/
public class MenuPageDriver{
    @Test
    public void testPayElectricInBillingPage(){
        BillingPage billingPage = new BillingPage();
        billingPage.payElectric("fI7RwjNJ",0.23895356499466);
    }
}
```

รูปที่ 5-29 รหัสต้นฉบับของไดร์เวอร์ตัวแทนคลาส MainPage

5.3.2 การทดสอบแบบรวมระหว่างระบบจำลองกับสตับและไดร์เวอร์

หลังจากสร้างสตับและไดร์เวอร์จากตัวสร้างแล้วผู้ทดสอบได้ทดลองนำสตับและไดร์เวอร์ที่สร้างขึ้นทดสอบแบบรวมกับระบบจำลอง โดยทดสอบรวมระหว่างคลาส BillingPage และคลาส Billing-Controller กับสตับและไดร์เวอร์ที่สร้างจากตัวสร้าง รูปที่ 5-30 แสดงรหัสต้นฉบับของคลาส BillingController พี่พัฒนาเสร็จแล้วบางส่วนโดยมีการเรียกใช้งานคลาส BillingService คลาส ElectricBillingProxyImpl ซึ่งเป็นคลาสรูปธรรมของคลาส ElectricBillingProxy และคลาส Transaction ซึ่งคลาสดังกล่าวทั้งหมดยังพัฒนาไม่เสร็จและผู้ทดสอบได้เปลี่ยนให้คลาส BillingController เรียกใช้งานคลาส BillingServiceStub คลาส ElectricBillingProxyImplStub และคลาส TransactionStub แทนดังแสดงในรูปที่ 5-31

อย่างไรก็ตามจากรูปที่ 5-31 การแทนที่คลาส Transaction ด้วยคลาส TransactionStub ส่งผลให้เกิดข้อผิดพลาดทางไวยกรณ์ (Syntax Error) เนื่องจากซิกเนเจอร์เมท็อด payElectric มีประเภทของข้อมูลส่งกลับเป็นอ็อบเจกต์ ที่สร้างจากคลาส Transaction ไม่ใช่ TransactionStub ซึ่งเป็นข้อจำกัดของตัวสร้างที่ไม่สามารถคาดการณ์ได้ว่ารหัสต้นฉบับของจริงจะมีรายละเอียดเป็นอย่างไร

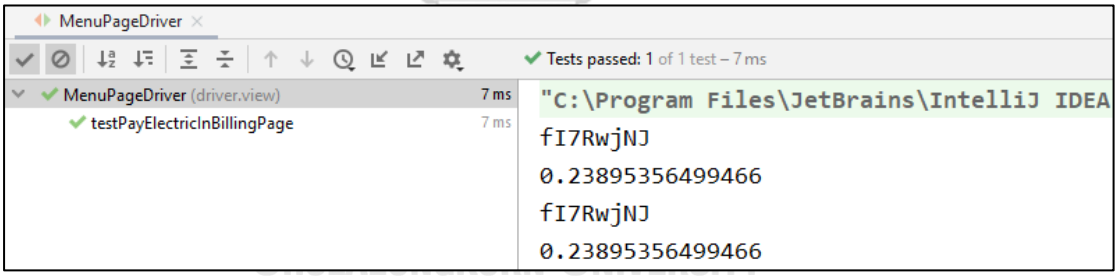
ถัดมาผู้ทดสอบได้ทดลองเรียกใช้งานคลาส BillingPage ด้วยไดร์เวอร์ของคลาส MenuPage รูปที่ 5-32 แสดงผลการเรียกใช้งานไดร์เวอร์ซึ่งสามารถเรียกใช้งานคลาส BillingPage ได้โดยค่า null และ 0.0 ในรูปเกิดจากการพิมพ์ค่าจากคลาส ElectricBillingProxyImplStub.java

```
public class BillingController extends AbstractController {
    @Override
    public void loadPage() {

    }

    @Override
    public void redirect() {

    }
    public Transaction payElectric(String accountNo, double amount){
        BillingService billingService = new BillingService();
        ElectricBillingProxyImpl electricBillingProxy = new ElectricBillingProxyImpl();
        if(electricBillingProxy.payBill(accountNo,amount)){
            BillingResponse billingResponse = billingService.processBilling(accountNo,amount);
        }
        Transaction transaction = new Transaction();
        return transaction;
    }

}
```

รูปที่ 5-30 รหัสต้นฉบับของคลาส BillingController ที่พัฒนาเสร็จแล้วบางส่วน

```
public class BillingController extends AbstractController {
    @Override
    public void loadPage() {

    }

    @Override
    public void redirect() {

    }
    public Transaction payElectric(String accountNo, double amount){
//        BillingService billingService = new BillingService();
        BillingServiceStub billingService = new BillingServiceStub();
//          ElectricBillingProxyImpl electricBillingProxy = new ElectricBillingProxyImpl();
        ElectricBillingProxyImplStub electricBillingProxy = new ElectricBillingProxyImplStub();
        if(electricBillingProxy.payBill(accountNo,amount)){
            BillingResponse billingResponse = billingService.processBilling(accountNo,amount);
        }
//          Transaction transaction = new Transaction();
        TransactionStub transaction = new TransactionStub();
        return transaction;
    }

}
```

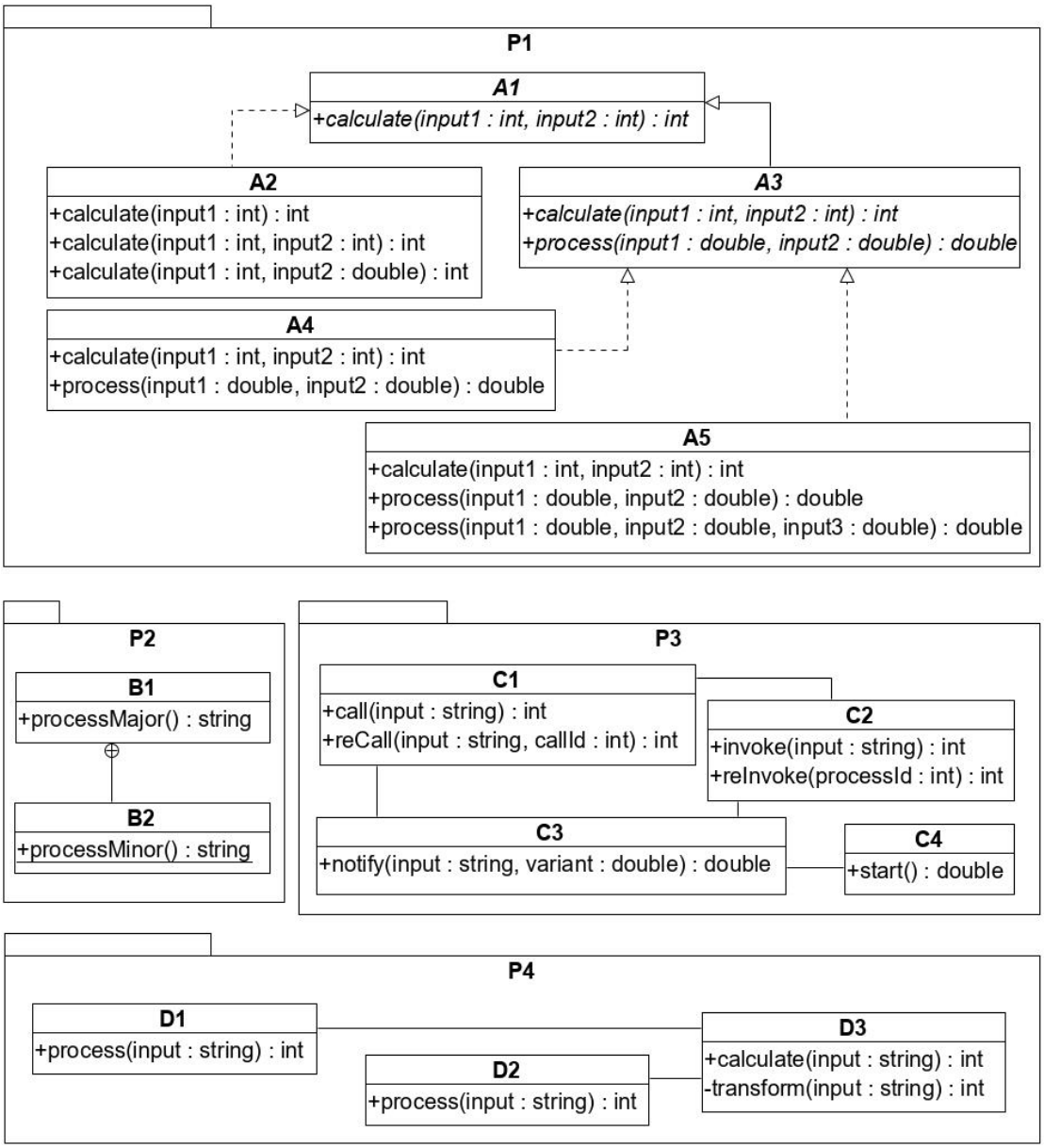
รูปที่ 5-31 รหัสต้นฉบับ BillingController ที่เรียกใช้สตับของคลาส BillingService สตับของคลาส ElectricBillingProxyImpl และสตับของคลาส Transaction



รูปที่ 5-32 ผลลัพธ์การเรียกใช้งาน MainPageDriver

5.4 กรณีศึกษาที่ 3 ระบบจำลอง

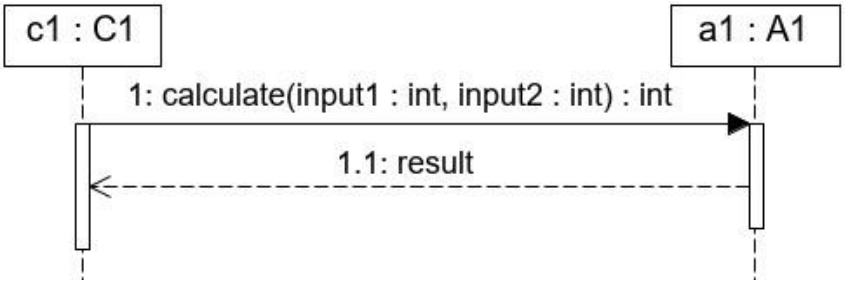
กรณีศึกษาระบบจำลอง เป็นระบบที่สร้างขึ้นมาโดยไม่ได้อ้างอิงถึงระบบในโลกความเป็นจริงใด ๆ โดยระบบดังกล่าวมีวัตถุประสงค์เพื่อจำลองเหตุการณ์บางเหตุการณ์ที่อธิบายได้ยากด้วยระบบจริง แผนภาพคลาสของระบบจำลองแสดงในรูปที่ 5-33



รูปที่ 5-33 แผนภาพคลาสของระบบจำลอง

5.4.1 กรณีทดสอบ - เลือกคลาสภายใต้การทดสอบที่เรียกใช้งานคลาสภายใน

กรณีทดสอบนี้จะทดสอบการสร้างสตับของคลาสภายในโดยใช้แผนภาพลำดับรูปที่ 5-34 จากแผนภาพลำดับ คลาส C1 เรียกใช้งานคลาส B2 ซึ่งเป็นคลาสภายในของคลาส B1 แสดงในรูปที่ 5-33 โดยผู้ทดสอบเลือกคลาส C1 เป็นคลาสภายใต้การทดสอบเพื่อให้ตัวสร้างสร้างสตับของคลาส B2 ขึ้นมา ซึ่งตัวสร้างได้สร้างรหัสต้นฉบับของคลาส B2 แสดงในรูปที่ 5-35 และผู้ทดสอบได้ทดลองนำสตับที่สร้างไปแทนที่คลาส B2 ในคลาส C1 รูปที่ 5-36 แสดงรหัสต้นฉบับบางส่วนของคลาส C1 ที่เรียกใช้งานสตับแทนคลาส B2



รูปที่ 5-34 แผนภาพลำดับที่มีการเรียกใช้งานคลาสนามธรรม

```
package stub.P2;
public class B2Stub{
    public static String processMinor(){
        return "UjHeti";
    }
}
```

รูปที่ 5-35 รหัสต้นฉบับของสตับของคลาส B2

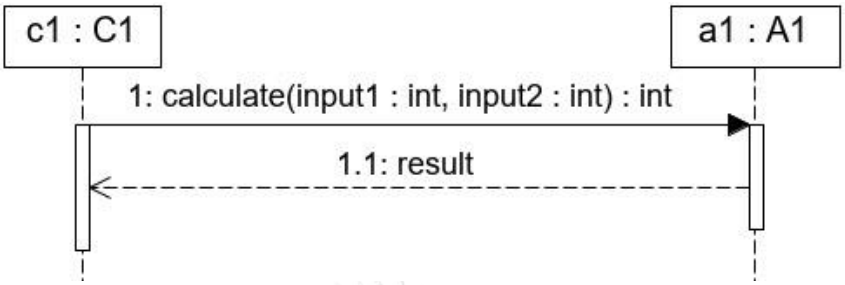
```
public String processMinor(){
    //B1.B2 b2 = new B1.B2();
    B2Stub b2 = new B2Stub();
    String processResult = b2.processMinor();
    return processMinor();
}
```

รูปที่ 5-36 รหัสต้นฉบับบางส่วนของคลาส C1 ที่เรียกใช้สตับของคลาส B2 แทนคลาส B2

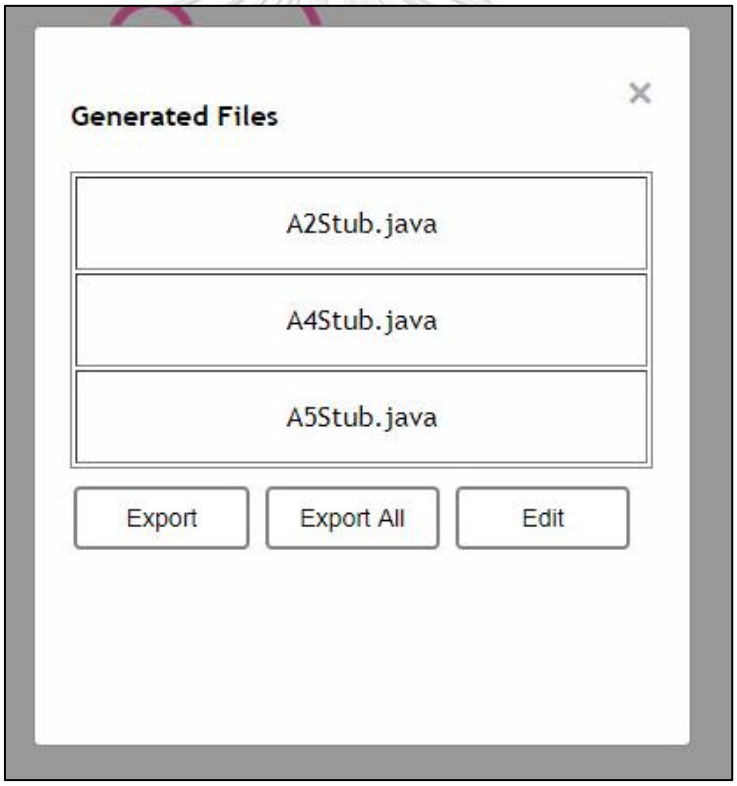
5.4.2 กรณีทดสอบ – คลาสลูกเป็นคลาสนามธรรม

กรณีทดสอบนี้เป็นการทดสอบคลาสที่เรียกใช้งานคลาสนามธรรม แต่คลาสลูกของคลาสนามธรรมนั้นเป็นคลาสนามธรรมเช่นกันโดยกรณีทดสอบนี้ใช้แผนภาพลำดับแสดงในรูปที่ 5-37 จากแผนภาพลำดับคลาส C1 เรียกใช้งานคลาส A1 ซึ่งเป็นคลาสนามธรรม จากแผนภาพคลาสรูปที่ 5-33 พบว่าจะต้องสร้างสตับของคลาส A2 และคลาส A3 เป็นตัวแทนของคลาส A1 แต่เนื่องจากคลาส A3

เป็นคลาสนามธรรมเช่นกัน ดังนั้นจึงจำเป็นต้องสร้างสตับของคลาส A4 และคลาส A5 ขึ้นมาเป็นตัวแทนของคลาส A3 รูปที่ 5-38 แสดงผลลัพธ์ของการสร้างสตับเพื่อทดสอบคลาส C1 ของตัวสร้าง



รูปที่ 5-37 แผนภาพลำดับที่มีการเรียกใช้งานคลาสนามธรรม

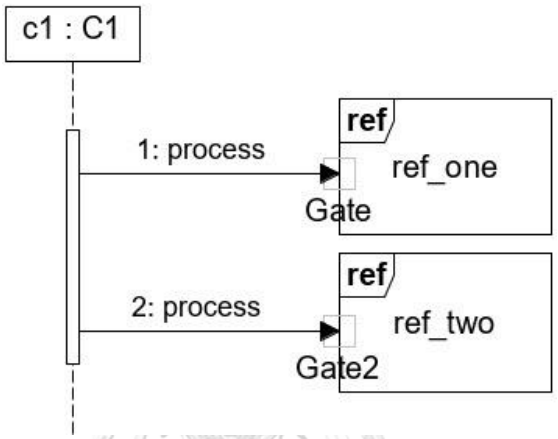


รูปที่ 5-38 รายการของสตับสำหรับทดสอบคลาส C1

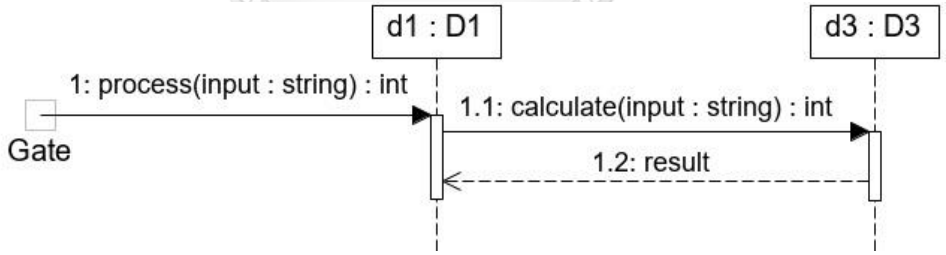
5.4.3 กรณีทดสอบ – แผนภาพลำดับที่อ้างอิงแผนภาพลำดับอื่นมากกว่าหนึ่งแผนภาพ

กรณีทดสอบนี้เป็นการทดสอบการสร้างสตับเมื่อคลาสภายใต้การทดสอบเรียกใช้งานแผนภาพลำดับอื่นมากกว่าหนึ่งแผนภาพ รูปที่ 5-39 แสดงแผนภาพลำดับที่มีการอ้างอิงถึงแผนภาพ

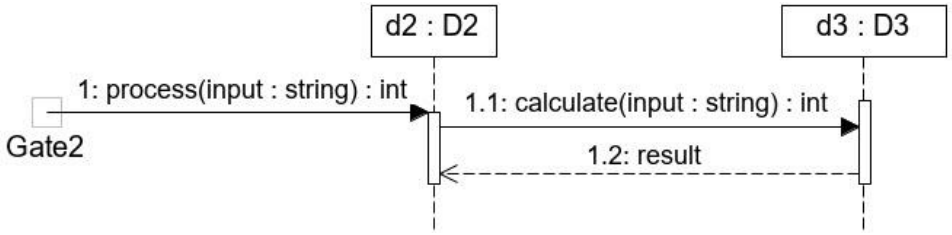
ลำดับอื่นสองแผนภาพได้แก่ แผนภาพลำดับ ref_one ซึ่งแสดงในรูปที่ 5-40 และแผนภาพลำดับ ref_two ดังแสดงในรูปที่ 5-41 ผู้ทดสอบได้นำเข้าแผนภาพลำดับทั้ง 3 และเชื่อมต่อแผนภาพลำดับดังกล่าวเข้าด้วยกัน จากนั้นผู้ทดสอบเลือกคลาส C1 เป็นคลาสภายใต้การทดสอบ จากรูปที่ 5-40 คลาสแรกที่ปรากฏในแผนภาพลำดับ ref_one ได้แก่คลาส D1 และจากรูปที่ 5-41 คลาสแรกที่ปรากฏในแผนภาพลำดับ ref_two ได้แก่คลาส D2 เพราะฉะนั้นตัวสร้างจะสร้างสตับของคลาส D1 และ คลาส D2 เป็นตัวแทนของแผนภาพลำดับ ref_one และแผนภาพลำดับ ref_two ตามลำดับ รูปที่ 5-42 แสดงรายการของสตับและไดร์เวอร์ที่ใช้ทดสอบคลาส C1



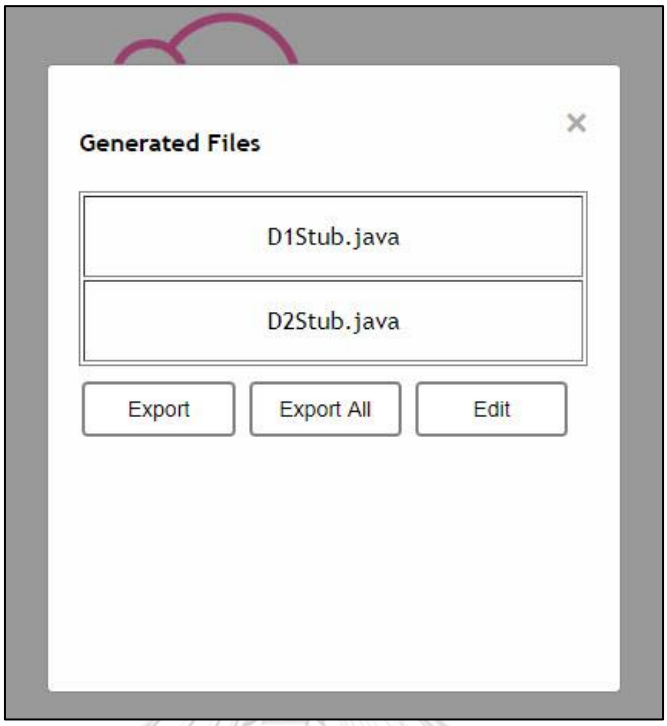
รูปที่ 5-39 แผนภาพลำดับที่อ้างอิงถึงแผนภาพลำดับอื่นมากกว่าหนึ่งแผนภาพ



รูปที่ 5-40 แผนภาพลำดับ ref_one



รูปที่ 5-41 แผนภาพลำดับ ref two



รูปที่ 5-42 รายการของสตับที่ต้องใช้ทดสอบคลาส C1

5.5 สรุปผลการทดสอบ

จากการทดสอบการสร้างสตับและไดร์เวอร์กับกรณีศึกษาทั้ง 3 กรณี พบว่าตัวสร้างสามารถสร้างรหัสต้นฉบับของสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาสได้อย่างถูกต้องทั้งกรณีปกติและกรณีที่ไม่สร้างรหัสต้นฉบับ รวมทั้งรหัสต้นฉบับของสตับและไดร์เวอร์ที่สร้างขึ้นสามารถนำไปแทนที่คลาสที่ยังพัฒนาไม่เสร็จได้โดยมีการแก้ไขรหัสต้นฉบับของคลาสภายใต้การทดสอบและไดร์-เวอร์เพียงเล็กน้อยได้แก่ การเปลี่ยนให้รหัสต้นฉบับของคลาสภายใต้การทดสอบเรียกใช้สตับแทนคลาสจริง และการเปลี่ยนการประกาศตัวแปรของไดร์เวอร์เมื่อเมท็อดของคลาสภายใต้การทดสอบคืนค่าออกมาเป็นรายการดังแสดงในหัวข้อที่ 5.2.7 นอกจากนี้รหัสต้นฉบับของไดร์เวอร์ที่สร้างขึ้นสำหรับเมสเสจที่กำกับด้วยการ์ดคอนดิชันสามารถสร้างค่าสุ่มที่ตรงเงื่อนไขได้ดังแสดงในหัวข้อที่ 5.2.6 อย่างไรก็ตามถึงแม้ว่าไดร์เวอร์จะสามารถทำให้เรียกใช้งานคลาสภายใต้การทดสอบได้แต่เนื่องจากค่าของข้อมูลนำเข้าและข้อมูลส่งออกของเมท็อดในสตับและไดร์เวอร์ที่สร้างขึ้นเป็นค่าสุ่มทำให้ผลลัพธ์ที่ได้ไม่ถูกต้องตามการทำงานจริงของระบบดังแสดงในหัวข้อที่ 5.2.7 และ 5.3.2 ผู้ทดสอบต้องแก้ไขข้อมูลดังกล่าวก่อนให้สอดคล้องกับการทำงานจริงของระบบเพื่อให้ทำงานได้อย่างถูกต้อง

บทที่ 6
สรุปผลของงานวิจัยและข้อเสนอแนะ

จากการวิเคราะห์ ศึกษา วิจัยและพัฒนาตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาส สามารถสรุปผลการวิจัย ข้อจำกัดของตัวสร้าง และแนวทางในการพัฒนาต่อในอนาคต โดยมีรายละเอียดดังต่อไปนี้

6.1 สรุปผลงานวิจัย

งานวิจัยนี้นำเสนอตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาส โดยเริ่มต้นผู้ทดสอบนำเข้าไฟล์แผนภาพลำดับและไฟล์แผนภาพคลาสในรูปแบบไฟล์เอกซ์เอ็มแอล จากนั้นตัวสร้างจะประมวลผลไฟล์ดังกล่าว สร้างกราฟกราฟการเรียกใช้งานจากแผนภาพลำดับ และเก็บข้อมูลของแผนภาพคลาส ผู้ทดสอบสามารถเลือกคลาสหรือกลุ่มของคลาสภายใต้การทดสอบเพื่อทดสอบรวมได้ผ่านส่วนต่อประสานผู้ใช้ของตัวสร้าง และตัวสร้างจะสร้างรหัสต้นฉบับของสตับและไดร์เวอร์สำหรับทดสอบคลาสดังกล่าว รวมทั้งสุ่มค่าของพารามิเตอร์และข้อมูลส่งออกตามชนิดของข้อมูล ก่อนส่งออกไฟล์ดังกล่าวเพื่อให้ผู้ทดสอบนำรหัสต้นฉบับดังกล่าวไปแทนที่รหัสต้นฉบับของคลาสที่ยังพัฒนาไม่เสร็จ ทั้งนี้ผู้วิจัยได้ทดสอบตัวสร้างในกรณีศึกษา 3 กรณี พบว่าตัวสร้างสามารถสร้างสตับและไดร์เวอร์ได้อย่างถูกต้องและสามารถนำรหัสต้นฉบับของสตับและไดร์เวอร์ไปแทนที่คลาสที่ยังพัฒนาไม่เสร็จได้โดยแก้ไขรหัสต้นฉบับเพียงเล็กน้อย

6.2 ข้อจำกัดของตัวสร้าง

ตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาสมีข้อจำกัดดังต่อไปนี้

1. หากแผนภาพลำดับที่นำเข้ามีการอ้างอิงถึงแผนภาพลำดับอื่นผู้ทดสอบต้องนำเข้าแผนภาพลำดับที่ถูกอ้างอิงและเชื่อมต่อแผนภาพลำดับเข้าด้วยกันก่อนสร้างรหัสต้นฉบับ

2. ตัวสร้างจะสร้างรหัสต้นฉบับโดยใช้ข้อมูลจากแผนภาพลำดับและแผนภาพคลาสเท่านั้นโดยไม่สนใจรหัสต้นฉบับ

3. ตัวสร้างไม่รองรับการสร้างตัวแปรที่เป็นโครงสร้างข้อมูลเช่น อาเรย์ ลิสต์

4. ตัวสร้างจะสุ่มค่าของพารามิเตอร์และข้อมูลส่งออกสำหรับข้อมูลประเภทจำนวนเต็ม (int, short, long) ตัวอักษร (string, char) ทศนิยม (double, float) และ Boolean เท่านั้น สำหรับข้อมูลชนิดอื่นตัวสร้างจะใช้ค่านัลแทน

5. ตัวสร้างจะไม่สร้างคอนสตรัคเตอร์แผนภาพคลาสไม่มีการระบุเมท็อดซิกเนเจอร์ของคอนสตรัคเตอร์

6. ตัวสร้างจะไม่สร้างไฟล์รหัสต้นฉบับของสตับและไดร์เวอร์ใหม่ ถ้ามีไฟล์เดิมอยู่ในฐานข้อมูลของตัวสร้าง

7. ตัวสร้างจะสร้างชุดข้อมูลทดสอบเพียงชุดเดียวสำหรับเรียกใช้งานเมท็อดของคลาสภายใต้การทดสอบ

6.3 ข้อเสนอแนะและแนวทางการดำเนินงานต่อ

ตัวสร้างสตับและไดร์เวอร์จากแผนภาพลำดับและแผนภาพคลาส สามารถนาไปต่อยอดพัฒนาเพิ่มเติมให้รองรับฟังก์ชันเพิ่มเติมและแก้ไขข้อจำกัดได้ดังต่อไปนี้

1. พัฒนาตัวสร้างให้รองรับการ์ดคอนดิชันเชิงประกอบได้เช่น n>5 && n<=100

2. พัฒนาตัวสร้างให้รองรับการสุ่มค่าสำหรับข้อมูลที่เป็นอ็อบเจกต์ และโครงสร้างข้อมูลได้

3. พัฒนาตัวสร้างให้รองรับแผนภาพลำดับและแผนภาพคลาสที่สร้างจากเครื่องมืออื่น ๆ ได้

4. พัฒนาตัวสร้างให้สามารถสร้างรหัสต้นฉบับภาษาอื่น ๆ ได้

5. พัฒนาตัวสร้างให้สามารถสุ่มค่าของพารามิเตอร์และข้อมูลส่งออกที่ไม่สอดคล้องกับการ์ดคอนดิชันได้

6. พัฒนาตัวสร้างให้รองรับการสร้างกรณีทดสอบในไดร์เวอร์ด้วยเทคนิคการทดสอบค่าขอบ (Boundary Value Testing) [1]

บรรณานุกรม

1. Jorgensen, P., *Software Testing A Craftsman's Approach*. 4 ed. 2014, NY: CRC Press.
2. Pressman, R. and B. Maxim, *Software Engineering A Practitioner's Approach*. 8 ed. 2015, NY: McGraw-Hill Education.
3. Myers, G., T. Badgett, and C. Sandler, *The Art of Software Testing*. 3 ed. 2012, NJ: Wiley.
4. Luengruengroj, P. and T. Suwannasart, *Stubs and Drivers Generator for Object-Oriented Program Testing Using Sequence and Class Diagrams*, in *2018 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII)*. 2018: Yonago. p. 32-36.
5. Zhang, Y., et al., *An approach of class integration test order determination based on test levels.* Software-Practice & Experience 2015. **45**(5): p. 657-687.
6. Sarma, M., D. Kundu, and R. Mall, *Automatic Test Case Generation from UML Sequence Diagram*, in *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*. 2007: Guwahati, Assam. p. 60-67.
7. Monpratarnchai, S., et al., *An Automated Testing Tool for Java Application Using Symbolic Execution Based Test Case Generation*, in *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*. 2013: Bangkok. p. 93-98.
8. Li, Y. and L. Jiang, *The research on test case generation technology of UML sequence diagram*, in *2014 9th International Conference on Computer Science & Education*. 2014: Vancouver. p. 1067-1069.
9. Khatun, A. and K. Sakib, *An automatic test suite regeneration technique ensuring state model coverage using UML diagrams and source syntax*, in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. 2016: Dhaka. p. 88-93.
10. Fraikin, F. and T. Leonhardt. *SeDiTeC-testing based on sequence diagrams*. in *17th IEEE International Conference on Automated Software Engineering*. 2002. Edinburgh.

11. Dhineshkumar, M. and Galeebathullah, *An Approach to Generate Test Cases from Sequence Diagram*, in *2014 International Conference on Intelligent Computing Applications*. 2014: Coimbatore. p. 345-349.
12. De Pontes Cafeo, B.B., et al. *A catalogue of stub and driver patterns to support integration testing of aspect-oriented programs.* in *Proceedings of the 8th Latin American Conference on Pattern Languages of Programs*. 2010. New York: ACM.
13. Clarke, P.J., et al., *Intra-Class Testing of Abstract Class Features*, in *The 18th IEEE International Symposium on Software Reliability (ISSRE '07)*. 2007: Trollhattan. p. 191-200.
14. World Wide Web Consortium (w3c). *Extensible Markup Language (XML).* [cited 2019 April, 14]; Available from: http://www.w3.org/XML/
15. Sanyal, A., B. Sathe, and U. Khedker, *Data Flow Analysis Theory and Practice*. 2009, FL: CRC Press.
16. Bechtold, S., et al. *JUnit 5 User Guide*. 2019 [cited 2019 June, 14]; Available from: https://junit.org/junit5/docs/current/user-guide/
17. Fowler, M., *Pattern of Enterprise Application Architecture*. 2002: Addison-Wesley Professional.
18. Dennis, A., B. Wixom, and D. Tegarden, *System Analysis and Design; An Object-Oriented Approach with UML*. 5 ed. 2015: Wiley.
19. Horstmann, C.S. and G. Cornell., *Core Java Volume I-Fundamentals*. 9 ed. 2012: Prentice Hall.
20. Ramakrishnan, R. and J. Gehrke, *Database Management Systems*. 3 ed. 2003: McGraw-Hill Education.

ภาคผนวก

ภาคผนวก ก

รายละเอียดยูสเคสของตัวสร้าง

ในภาคผนวก ก จะแสดงรายละเอียดยูสเคสของแต่ละยูสเคสดังต่อไปนี้

ตารางที่ ก-1 รายละเอียดยูสเคสอัปโหลดแผนภาพลำดับ

หมายเลขกรณีทดสอบ	UC01
ชื่อกรณีทดสอบ	อัปโหลดแผนภาพลำดับ
รายละเอียดกรณีทดสอบ	เพื่อนำเข้าไฟล์เอกซ์เอ็มแอลของแผนภาพลำดับสู่ตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	Include: ประมวลผลแผนภาพลำดับ
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบกดปุ่ม Upload XML 2. ตัวสร้างแสดงหน้าต่างให้เลือกประเภทของแผนภาพ 3. ผู้ทดสอบกดปุ่ม Sequence Diagram 4. ผู้ทดสอบเลือกไฟล์ที่ต้องการนำเข้า 5. ตัวสร้างนำเข้าไฟล์และจัดเก็บไว้ในฐานข้อมูลของตัวสร้าง
เงื่อนไขภายหลัง	ตัวสร้างปิดหน้าต่างการเลือกแผนภาพ

ตารางที่ ก-2 รายละเอียดยูสเคสประมวลผลแผนภาพลำดับ

หมายเลขกรณีทดสอบ	UC02
ชื่อกรณีทดสอบ	ประมวลผลแผนภาพลำดับ
รายละเอียดกรณีทดสอบ	เพื่อนำข้อมูลในแผนภาพลำดับไปสร้างเป็นกราฟการเรียกใช้งาน
ผู้กระทำ	-
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ตัวสร้างอ่านไฟล์เอกซ์เอ็มแอล 2. ตัวสร้างรวบรวมข้อมูลของอ็อบเจกต์ ภายในแผนภาพ 3. ตัวสร้างรวบรวมข้อมูลของเมสเสจที่ส่งระหว่างแต่ละอ็อบเจกต์

ตารางที่ ก-2 รายละเอียดยูสเคสประมวลผลแผนภาพลำดับ (ต่อ)

ขั้นตอน	4. ตัวสร้างรวบรวมข้อมูลของการ์ดคอนดิชัน 5. ตัวสร้างสร้างกราฟการเรียกใช้งาน 6. ตัวสร้างจัดเก็บกราฟการเรียกใช้งานลงฐานข้อมูล
เงื่อนไขภายหลัง	-

ตารางที่ ก-3 รายละเอียดยูสเคสอัปโหลดแผนภาพคลาส

หมายเลขกรณีทดสอบ	UC03
ชื่อกรณีทดสอบ	อัปโหลดแผนภาพคลาส
รายละเอียดกรณีทดสอบ	เพื่อนำเข้าไฟล์เอกซ์เอ็มแอลของแผนภาพคลาสสู่ตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	Include: ประมวลผลแผนภาพคลาส
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบกดปุ่ม Upload XML 2. ตัวสร้างแสดงหน้าต่างให้เลือกประเภทของแผนภาพ 3. ผู้ทดสอบกดปุ่ม Class Diagram 4. ผู้ทดสอบเลือกไฟล์ที่ต้องการนำเข้า 5. ตัวสร้างนำเข้าไฟล์และจัดเก็บไว้ในฐานข้อมูลของตัวสร้าง
เงื่อนไขภายหลัง	ตัวสร้างปิดหน้าต่างการเลือกแผนภาพ

ตารางที่ ก-4 รายละเอียดยูสเคสประมวลผลแผนภาพคลาส

หมายเลขกรณีทดสอบ	UC04
ชื่อกรณีทดสอบ	ประมวลผลแผนภาพคลาส
รายละเอียดกรณีทดสอบ	เพื่อรวบรวมข้อมูลของแผนภาพคลาส
ผู้กระทำ	-
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ตัวสร้างอ่านไฟล์เอกซ์เอ็มแอล 2. ตัวสร้างรวบรวมข้อมูลของแพ็คเกจในแผนภาพ

ตารางที่ ก-4 รายละเอียดยูสเคสประมวลผลแผนภาพคลาส (ต่อ)

ขั้นตอน	3. ตัวสร้างรวบรวมข้อมูลของคลาสในแต่ละแพ็คเกจ 4. ตัวสร้างรวบรวมข้อมูลของเมท็อดในแต่ละคลาส 5. ตัวสร้างรวบรวมข้อมูลของซิกเนเจอร์ของแต่ละเมท็อด 6. ตัวสร้างข้อมูลของแผนภาพคลาสลงฐานข้อมูล
เงื่อนไขภายหลัง	-

ตารางที่ ก-5 รายละเอียดยูสเคสเลือกคลาสภายใต้การทดสอบ

หมายเลขกรณีทดสอบ	UC05
ชื่อกรณีทดสอบ	เลือกคลาสภายใต้การทดสอบ
รายละเอียดกรณีทดสอบ	เพื่อสร้างสตับและไดร์เวอร์สำหรับคลาสภายใต้การทดสอบ
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	Include: ตรวจหาสตับที่ต้องใช้, ตรวจหาไดร์เวอร์ที่ต้องใช้, สร้างรหัสต้นฉบับ
เงื่อนไขก่อนหน้า	หากแผนภาพลำดับมีการอ้างอิงถึงแผนภาพลำดับอื่น จะต้องมีการเชื่อมต่อแผนภาพเข้าด้วยกันแล้ว
ขั้นตอน	1. ผู้ทดสอบเลือกกราฟการเรียกใช้งานที่ต้องการทดสอบ 2. ผู้ทดสอบเลือกแผนภาพคลาสที่สอดคล้องกับกราฟการเรียกใช้งาน 3. ผู้ทดสอบเลือกคลาสภายใต้การทดสอบ 4. ผู้ทดสอบกดปุ่ม Create Code 5. ตัวสร้างตรวจสอบรายการของคลาสภายใต้การทดสอบเพื่อเตรียมการสร้างรหัสต้นฉบับ (UC06-UC08)
ขั้นตอนการทางานทางเลือก/พิเศษ	5a ในขั้นตอนที่ 5 1. กรณีที่ผู้ทดสอบไม่ได้เลือกคลาสใด ๆ ตัวสร้างจะแสดงเมสเสจเตือนให้เลือกคลาสภายใต้การทดสอบอย่างน้อยหนึ่งคลาส 2. กรณีที่ผู้ทดสอบเลือกคลาสทุกคลาสในกราฟการเรียกใช้งานเป็นคลาสภายใต้การทดสอบ ตัวสร้างจะแสดงเมสเสจเตือนให้เลือกคลาสใหม่ 3. กรณีที่ผู้ทดสอบเลือกคลาสที่ไม่เกี่ยวข้องกันเป็นคลาสภายใต้การทดสอบ ตัวสร้างจะแสดงเมสเสจเตือนว่าคลาสที่เลือกไม่เกี่ยวข้องกัน

ตารางที่ ก-5 รายละเอียดยูสเคสเลือกคลาสภายใต้การทดสอบ (ต่อ)

เงื่อนไขภายหลัง	ตัวสร้างแสดงรายการของสตับและไดร์เวอร์ที่ต้องใช้

ตารางที่ ก-6 รายละเอียดยูสเคสตรวจหาสตับที่ต้องใช้

หมายเลขกรณีทดสอบ	UC06
ชื่อกรณีทดสอบ	ตรวจหาสตับที่ต้องใช้
รายละเอียดกรณีทดสอบ	เพื่อรวบรวมสตับที่ต้องใช้ในการทดสอบคลาสภายใต้การทดสอบที่ผู้ทดสอบเลือก
ผู้กระทำ	-
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	ผู้ทดสอบเลือกคลาสภายใต้การทดสอบแล้ว
ขั้นตอน	1. ตัวสร้างตรวจหาอ็อบเจกต์ ที่คลาสภายใต้การทดสอบเรียกใช้ 1.1 หากอ็อบเจกต์ เป็นการอ้างอิงแผนภาพลำดับอื่น ตัวสร้างจะตรวจสอบแผนภาพลำดับที่ถูกอ้างอิงว่ามีอ็อบเจกต์ ใดเป็นอ็อบเจกต์แรก 2. ตัวสร้างนำอ็อบเจกต์ ไปเทียบกับแผนภาพคลาส 2.1 หากพบว่าอ็อบเจกต์ เป็นคลาสนามธรรม ตัวสร้างจะค้นหาคลาสรูปธรรมที่สืบทอดคลาสดังกล่าว 3. ตัวสร้างรวบรวมรายการของสตับที่ต้องใช้
เงื่อนไขภายหลัง	-

ตารางที่ ก-7 รายละเอียดยูสเคสสร้างรหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC07
ชื่อกรณีทดสอบ	ตรวจหาไดร์เวอร์ที่ต้องใช้
รายละเอียดกรณีทดสอบ	เพื่อรวบรวมไดร์เวอร์ที่ต้องใช้ในการทดสอบคลาสภายใต้การทดสอบที่ผู้ทดสอบเลือก
ผู้กระทำ	-
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	ผู้ทดสอบเลือกคลาสภายใต้การทดสอบแล้ว

ตารางที่ ก-7 รายละเอียดยูสเคสสร้างรหัสต้นฉบับ (ต่อ)

ขั้นตอน	1. ตัวสร้างตรวจหาอ็อบเจกต์ ที่เรียกใช้คลาสภายใต้การทดสอบ 2. ตัวสร้างนำอ็อบเจกต์ ไปเทียบกับแผนภาพคลาส 3. ตัวสร้างรวบรวมรายการของไดร์เวอร์ที่ต้องใช้
เงื่อนไขภายหลัง	-

ตารางที่ ก-8 รายละเอียดยูสเคสสร้างรหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC08
ชื่อกรณีทดสอบ	สร้างรหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อสร้างรหัสต้นฉบับของสตับและไดร์เวอร์
ผู้กระทำ	-
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	ตัวสร้างได้รายการของสตับและไดร์เวอร์จากยูสเคส UC07 และ UC08
ขั้นตอน	1. ตัวสร้างสร้างชื่อไฟล์รหัสต้นฉบับ 2. ตัวสร้างค้นหาไฟล์รหัสต้นฉบับในฐานข้อมูลด้วยชื่อไฟล์ในขั้นตอนก่อนหน้า 2.1 หากพบ ตัวสร้างจะเปิดไฟล์ดังกล่าวและข้ามไปขั้นตอนที่ 4 2.2 หากไม่พบ ทำขั้นตอนที่ 3 3. ตัวสร้างสร้างโครงสร้างของไฟล์และจัดเก็บลงฐานข้อมูล 4. ตัวสร้างสร้างโครงสร้างของเมท็อดจากซิกเนอเจอร์ของเมท็อด 5. ตัวสร้างค้นหาเมท็อดที่มีซิกเนอเจอร์เหมือนกันในไฟล์รหัสต้นฉบับ 5.1 หากพบ ตัวสร้างจะปิดไฟล์และจบการทำงาน 5.2 หากไม่พบ ทำขั้นตอนที่ 6 6. ตัวสร้างแทรกรหัสต้นฉบับลงไฟล์ 7. ตัวสร้างปิดไฟลีและบันทึกไฟล์ลงฐานข้อมูล
เงื่อนไขภายหลัง	-

ตารางที่ ก-9 รายละเอียดยูสเคสเชื่อมต่อแผนภาพลำดับ

หมายเลขกรณีทดสอบ	UC09
ชื่อกรณีทดสอบ	เชื่อมต่อแผนภาพลำดับ
รายละเอียดกรณีทดสอบ	เพื่อเชื่อมต่อแผนภาพลำดับที่มีการอ้างอิงถึงแผนภาพลำดับอื่นเข้าด้วยกันกับแผนภาพลำดับที่ถูกอ้างอิง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	แผนภาพลำดับต้นทางและแผนภาพลำดับที่ถูกอ้างอิง ถูกนำเข้าสู่ตัวสร้างแล้วทั้งคู่
ขั้นตอน	1. ผู้ทดสอบเลือกไฟล์แผนภาพลำดับต้นทาง 2. ผู้ทดสอบกดปุ่ม Link Call Graph 3. ผู้ทดสอบเลือกแผนภาพลำดับที่ถูกอ้างอิง 4. ผู้ทดสอบเลือกอ็อบเจกต์ ประเภทอ้างอิงที่ต้องการเชื่อมต่อ 5. ผู้ทดสอบกดปุ่ม Link 6. ตัวสร้างบันทึกการเชื่อมต่อลงฐานข้อมูล
ขั้นตอนการทางานทางเลือก/พิเศษ	2a ในขั้นตอนที่ 2 1. หากแผนภาพลำดับต้นทางไม่มีการอ้างอิงถึงแผนภาพลำดับอื่น ตัวสร้างจะไม่ดำเนินการต่อ
เงื่อนไขภายหลัง	-

ตารางที่ ก-10 รายละเอียดยูสเคสลบไฟล์แผนภาพ

หมายเลขกรณีทดสอบ	UC10
ชื่อกรณีทดสอบ	ลบไฟล์แผนภาพ
รายละเอียดกรณีทดสอบ	เพื่อลบไฟล์แผนภาพและข้อมูลของแผนภาพออกจากตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกแผนภาพที่ต้องการลบ 2. ผู้ทดสอบกดปุ่ม Delete 3. ตัวสร้างลบไฟล์และข้อมูลของแผนภาพออกจากระบบ

ตารางที่ ก-10 รายละเอียดยูสเคสลบไฟล์แผนภาพ (ต่อ)

เงื่อนไขภายหลัง	-

ตารางที่ ก-11 รายละเอียดยูสเคสเปลี่ยนชื่อไฟล์แผนภาพ

หมายเลขกรณีทดสอบ	UC11
ชื่อกรณีทดสอบ	เปลี่ยนชื่อไฟล์แผนภาพ
รายละเอียดกรณีทดสอบ	เพื่อเปลี่ยนชื่อไฟล์แผนภาพในตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกแผนภาพที่ต้องการเปลี่ยนชื่อ 2. ผู้ทดสอบกดปุ่ม Rename 3. ผู้ทดสอบใส่ชื่อไฟล์ใหม่ 4. ผู้ทดสอบกดปุ่ม Rename 5. ตัวสร้างบันทึกความเปลี่ยนแปลงลงฐานข้อมูล
เงื่อนไขภายหลัง	-

ตารางที่ ก-12 รายละเอียดยูสเคสแก้ไขรหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC12
ชื่อกรณีทดสอบ	แก้ไขรหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อแก้ไขเนื้อหาของไฟล์รหัสต้นฉบับ
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกไฟล์ที่ต้องการแก้ไข 2. ผู้ทดสอบกดปุ่ม Edit 3. ตัวสร้างแสดงหน้าต่างการแก้ไขไฟล์
เงื่อนไขภายหลัง	-

ตารางที่ ก-13 รายละเอียดยูสเคสแทรกค่าสุ่มในรหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC13
ชื่อกรณีทดสอบ	แทรกค่าสุ่มในรหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อแทรกค่าสุ่มตามประเภทของข้อมูลลงในรหัสต้นฉบับ
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกตำแหน่งที่ต้องการแทรกค่าสุ่มในรหัสต้นฉบับ 2. ผู้ทดสอบเลือกประเภทของข้อมูลที่ต้องการสุ่ม 3. ผู้ทดสอบกดปุ่ม Random 4. ตัวสร้างแทรกค่าสุ่มลงในรหัสต้นฉบับตามตำแหน่งที่เลือก
เงื่อนไขภายหลัง	-

ตารางที่ ก-14 รายละเอียดยูสเคสส่งออกไฟล์รหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC14
ชื่อกรณีทดสอบ	ส่งออกไฟล์รหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อส่งออกไฟล์รหัสต้นฉบับออกจากตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกไฟล์ที่ต้องการส่งออก 2. ผู้ทดสอบกดปุ่ม Export 3. ตัวสร้างส่งออกไฟล์รหัสต้นฉบับ
ขั้นตอนการทางาน ทางเลือก/พิเศษ	3a ในขั้นตอนที่ 3 1. หากไฟล์ที่ผู้ทดสอบเลือกมีมากกว่าหนึ่งไฟล์ ตัวสร้างจะบีบอัดทั้งหมดเป็นไฟล์ซิปไฟล์เดียวก่อนส่งออก
เงื่อนไขภายหลัง	-

ตารางที่ ก-15 รายละเอียดยูสเคสเปลี่ยนชื่อไฟล์รหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC15
ชื่อกรณีทดสอบ	เปลี่ยนชื่อไฟล์รหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อเปลี่ยนชื่อไฟล์รหัสต้นฉบับในตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกไฟล์รหัสต้นฉบับที่ต้องการเปลี่ยนชื่อ 2. ผู้ทดสอบกดปุ่ม Rename 3. ผู้ทดสอบใส่ชื่อไฟล์ใหม่ 4. ผู้ทดสอบกดปุ่ม Rename 5. ตัวสร้างบันทึกความเปลี่ยนแปลงลงฐานข้อมูล
เงื่อนไขภายหลัง	-

ตารางที่ ก-16 รายละเอียดยูสเคสบันทึกไฟล์รหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC16
ชื่อกรณีทดสอบ	บันทึกไฟล์รหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อบันทึกความเปลี่ยนแปลงของรหัสต้นฉบับลงฐานข้อมูลเมื่อมีการแก้ไขรหัสต้นฉบับด้วยตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	ตัวสร้างอยู่ที่หน้าต่างการแก้ไขรหัสต้นฉบับ
ขั้นตอน	1. ผู้ทดสอบกดปุ่ม Save 2. ตัวสร้างบันทึกรหัสต้นฉบับลงฐานข้อมูล 3. ตัวสร้างบันทึกเวลาที่บันทึกรหัสต้นฉบับ
เงื่อนไขภายหลัง	-

ตารางที่ ก-17 รายละเอียดยูสเคสลบไฟล์รหัสต้นฉบับ

หมายเลขกรณีทดสอบ	UC17
ชื่อกรณีทดสอบ	ไฟล์รหัสต้นฉบับ
รายละเอียดกรณีทดสอบ	เพื่อลบไฟล์แผนภาพและข้อมูลของไฟล์รหัสต้นฉบับออกจากตัวสร้าง
ผู้กระทำ	ผู้ทดสอบ
ความสัมพันธ์	-
เงื่อนไขก่อนหน้า	-
ขั้นตอน	1. ผู้ทดสอบเลือกไฟล์รหัสต้นฉบับที่ต้องการลบ 2. ผู้ทดสอบกดปุ่ม Delete 3. ตัวสร้างลบไฟล์และข้อมูลของรหัสต้นฉบับออกจากระบบ
เงื่อนไขภายหลัง	-

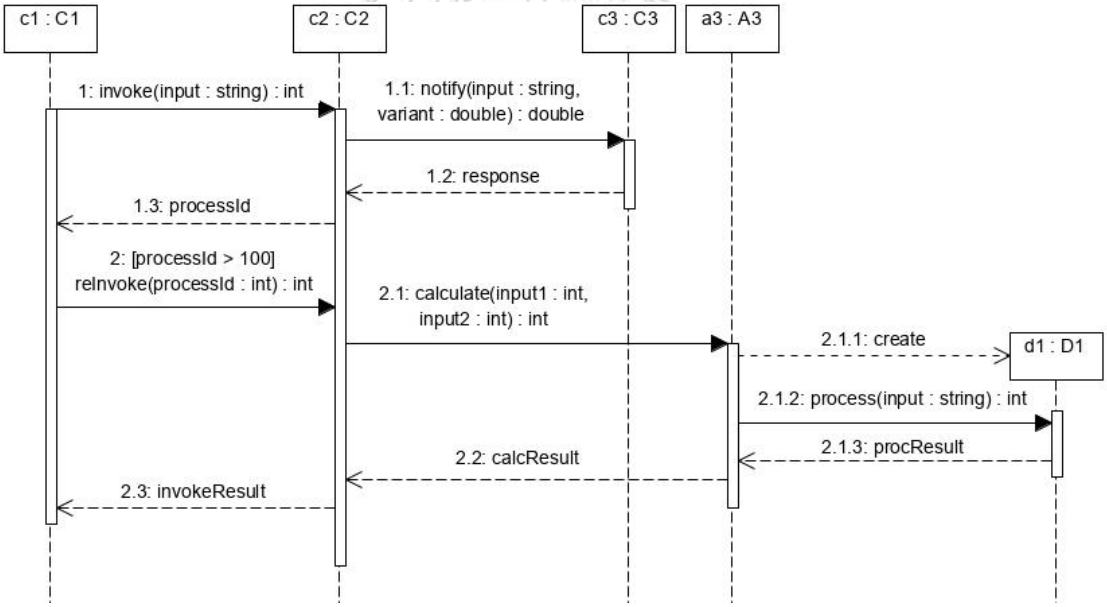
ภาคผนวก ข

การทวนสอบการนำเข้าแผนภาพลำดับและแผนภาพคลาส

บทนี้จะกล่าวถึงรายละเอียดและตัวอย่างการทวนสอบแผนภาพลำดับและแผนภาพคลาสหลังจากการนำเข้าไฟล์แผนภาพดังกล่าว

1. การทวนสอบการนำเข้าแผนภาพลำดับ

การทวนสอบการนำเข้าแผนภาพลำดับสามารถทำได้โดยการตรวจสอบฐานข้อมูลของตัวสร้างหลังจากการนำเข้าไฟล์แผนภาพลำดับ โดยจะตรวจสอบอ็อบเจกต์ เมสเสจ และการ์ดคอนดิชันที่ถูกเก็บในฐานข้อมูล การทวนสอบแผนภาพลำดับแสดงตัวอย่างด้วยแผนภาพลำดับชื่อ example1.xml แสดงในรูปที่ ข-1 โดยการทวนสอบมีรายละเอียดดังต่อไปนี้



รูปที่ ข-1 แผนภาพลำดับ example1.xml

1.1 ตาราง callgraph.graph

ตาราง callgraph.graph เป็นตารางที่ใช้เก็บข้อมูลของแผนภาพลำดับหรือกราฟการเรียกใช้งาน ตารางที่ ข-1 แสดงการเก็บข้อมูลของแผนภาพลำดับของตาราง callgraph.graph ในฐานข้อมูล

ตารางที่ ข-1 การเก็บข้อมูลของแผนภาพลำดับ example1.xml

callGraphId	callGraphName	filePath	createTimeStamp
1	example1.xml	../../SequenceDiagrams/ 1593074969_example1.xml	2020-06-25 15:49:29

1.2 ตาราง callgraph.objectnode

ตาราง callgraph.objectnode เป็นตารางที่ใช้เก็บข้อมูลของอ็อบเจกต์ในแผนภาพลำดับ โดยคอลัมน์ (Column) callGraphId จะอ้างอิงถึงคอลัมน์ callgraphId ในตาราง callgraph.graph ตารางที่ ข-2 แสดงการเก็บข้อมูลของอ็อบเจกต์ในแผนภาพลำดับ example1.xml ในฐานข้อมูล

ตารางที่ ข-2 การเก็บข้อมูลของอ็อบเจกต์ในแผนภาพลำดับ example1.xml

objectId	callGraphId	objectName	baseIdentifier
1	1	d1	D1
2	1	a3	A3
3	1	c3	C3
4	1	c2	C2
5	1	c1	C1

1.3 ตาราง callgraph.message

ตาราง callgraph.message เป็นตารางที่ใช้เก็บข้อมูลของเมสเสจในแผนภาพลำดับ โดยคอลัมน์ fromObjectId และ toObjectId จะอ้างอิงถึงคอลัมน์ objectId ในตาราง callgraph. objectnode ตารางที่ ข-3 แสดงการเก็บข้อมูลของเมสเสจในแผนภาพลำดับ example1.xml ในฐานข้อมูล

ตารางที่ ข-3 การเก็บข้อมูลของเมสเสจในแผนภาพลำดับ example1.xml

messageId	fromObjectId	toObjectId	messageName	messageType
1	5	4	invoke	CALLING
2	4	5	processId	RETURN
3	4	3	notify	CALLING
4	3	4	response	RETURN

ตารางที่ ข-3 การเก็บข้อมูลของเมสเสจในแผนภาพลำดับ example1.xml (ต่อ)

messageId	fromObjectId	toObjectId	messageName	messageType
5	5	4	reInvoke	CALLING
6	4	5	invokeResult	RETURN
7	4	2	calculate	CALLING
8	2	4	calcResult	RETURN
9	2	1	process	CALLING
10	1	2	procResult	RETURN
11	2	1	create	CREATE

1.4 ตาราง callgraph.guardcondition

ตาราง callgraph.guardcondition เป็นตารางที่ใช้เก็บข้อมูลของการ์ดคอนดิชันในแผนภาพลำดับ คอลัมน์ messageId จะอ้างอิงถึงคอลัมน์ messageId ในตาราง callgraph.message ตารางที่ ข-4 แสดงการเก็บข้อมูลของการ์ดคอนดิชันที่ปรากฏในแผนภาพลำดับ example1.xml ในฐานข้อมูล

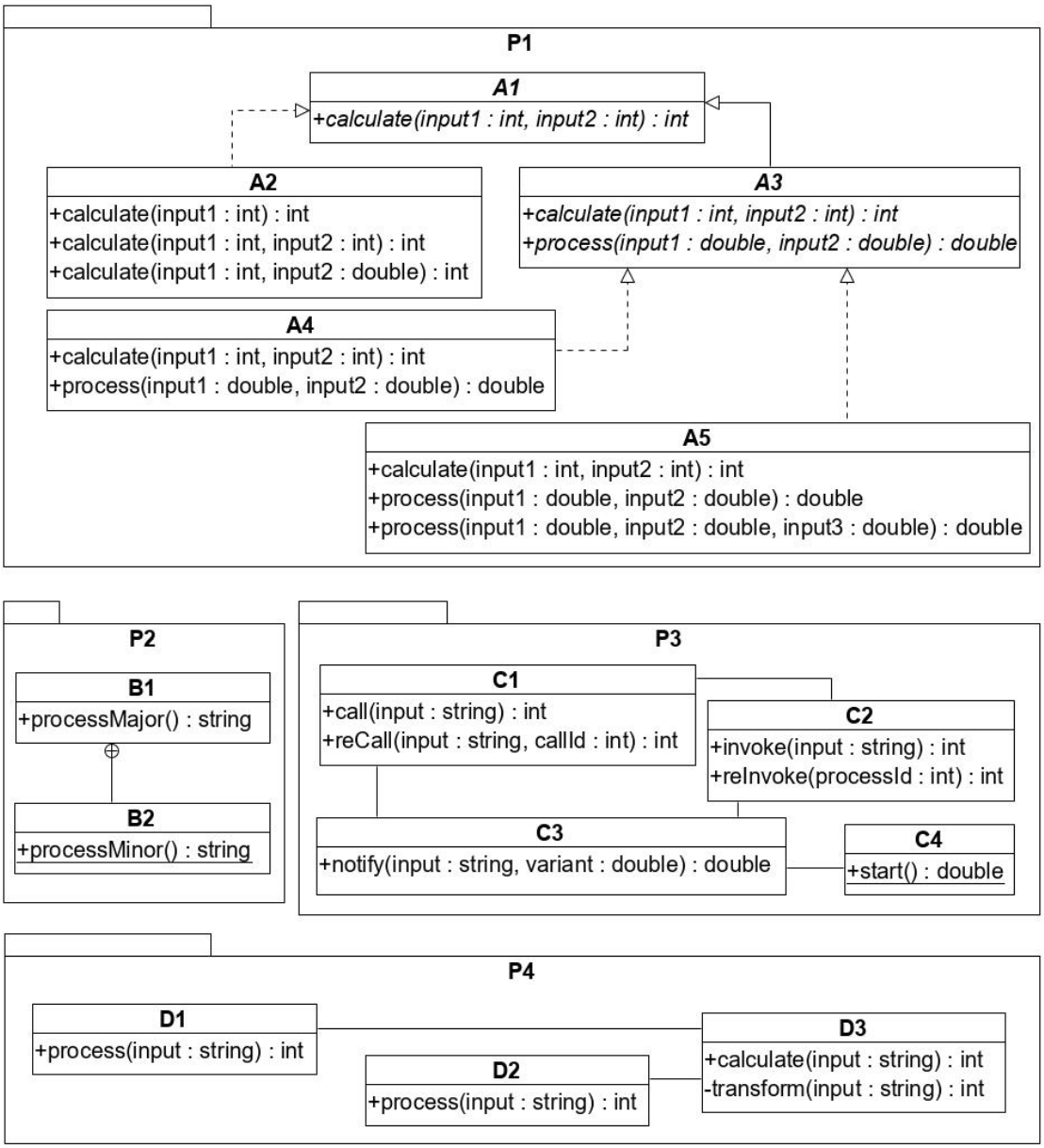
ตารางที่ ข-4 การเก็บข้อมูลของการ์ดคอนดิชันในแผนภาพลำดับ example1.xml

guardCondId	messageId	statement
1	5	processId > 100

จากตัวอย่างการทวนสอบการนำเข้าแผนภาพลำดับ example1.xml พบว่าตัวสร้างสามารถรวบรวมอ็อบเจกต์ เมสเสจ และการ์ดคอนดิชันได้อย่างถูกต้องและครบถ้วน

2. การทวนสอบการนำเข้าแผนภาพคลาส

การทวนสอบการนำเข้าแผนภาพคลาส สามารถทำได้โดยการตรวจสอบฐานข้อมูลของตัวสร้างหลังจากการนำเข้าไฟล์แผนภาพคลาส โดยพิจารณาการเก็บข้อมูลของแพ็คเกจ คลาส เมท็อด และเมท็อดซิกเนเจอร์ในฐานข้อมูล การทวนสอบแผนภาพคลาสแสดงตัวอย่างด้วยแผนภาพคลาสชื่อ ideal_system.xml แสดงในรูปที่ ข-2 โดยการทวนสอบมีรายละเอียดดังต่อไปนี้



รูปที่ ข-2 แผนภาพคลาส ideal_system.xml

2.1 ตาราง classdiagram.diagram

ตาราง classdiagram.diagram เป็นตารางที่ใช้เก็บข้อมูลของแผนภาพคลาสที่นำเข้า ตารางที่ ข-5 แสดงการเก็บข้อมูลของแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-5 การเก็บข้อมูลของแผนภาพคลาส ideal_system.xml

diagramId	diagramName	filePath	createTimeStamp
1	ideal_system.xml	../../ClassDiagrams/1593144040_ideal_system.xml	2020-06-26 11:00:40

2.2 ตาราง classdiagram.package

ตาราง classdiagram.package เป็นตารางที่ใช้เก็บข้อมูลของแพ็คเกจที่ปรากฏในแผนภาพคลาส โดยคอลัมน์ diagramId จะอ้างอิงถึงคอลัมน์ diagramId ในตาราง classdiagram.diagram ตารางที่ ข-6 แสดงการเก็บข้อมูลของแพ็คเกจในแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-6 การเก็บข้อมูลของแพ็คเกจในแผนภาพคลาส ideal_system.xml

packageId	diagramId	packageName	namespace
1	1	P4	P4
2	1	P2	P2
3	1	P3	P3
4	1	P1	P1

2.3 ตาราง classdiagram.class

ตาราง classdiagram.class เป็นตารางที่ใช้เก็บข้อมูลของคลาสในแต่ละแพ็คเกจ โดยคอลัมน์ packageId จะอ้างอิงถึงคอลัมน์ packageId ในตาราง classdiagram.package ตารางที่ ข-7 แสดงการเก็บข้อมูลของคลาสในแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-7 การเก็บข้อมูลของคลาสในแผนภาพคลาส ideal_system.xml

classId	packageId	className	instanceType
1	1	D1	CONCRETE
2	1	D2	CONCRETE
3	1	D3	CONCRETE
4	2	B1	CONCRETE
5	2	B2	CONCRETE

ตารางที่ ข-7 การเก็บข้อมูลของคลาสในแผนภาพคลาส ideal_system.xml (ต่อ)

classId	packageId	className	instanceType
6	3	C1	CONCRETE
7	3	C2	CONCRETE
8	3	C3	CONCRETE
9	3	C4	CONCRETE
10	4	A1	ABSTRACT
11	4	A2	CONCRETE
12	4	A3	ABSTRACT
13	4	A4	CONCRETE
14	4	A5	CONCRETE

2.4 ตาราง classdiagram.inheritance

ตาราง classdiagram.inheritance เป็นตารางที่ใช้เก็บการสืบทอดของคลาสภายในแผนภาพคลาสโดยคอลัมน์ superClassId และคอลัมน์ childClassId จะอ้างอิงถึงคอลัมน์ classId ในตาราง classdiagram.class ตารางที่ ข-8 แสดงการเก็บข้อมูลของการสืบทอดของคลาสภายในแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-8 การเก็บข้อมูลของการสืบทอดของคลาสในแผนภาพคลาส ideal_system.xml

inheritId	superClassId	childClassId
1	10	11
2	10	12
3	12	13
4	12	14

2.5 ตาราง classdiagram.method

ตาราง classdiagram.method เป็นตารางที่ใช้เก็บข้อมูลของเมท็อดซิกเนเจอร์ของแต่ละเมท็อดของแต่ละคลาสในแผนภาพคลาส คอลัมน์ classId จะอ้างอิงถึงคอลัมน์ classId ในตาราง

classdiagram.class ตารางที่ ข-9 แสดงการเก็บข้อมูลของเมท็อดซิกเนเจอร์ของแต่ละเมท็อดในแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-9 การเก็บข้อมูลของเมท็อดซิกเนเจอร์ของแต่ละเมท็อดในแผนภาพคลาส ideal_system.xml

methodId	classId	methodName	Visibility	returnType	instanceType
1	1	process	public	int	CONCRETE
2	2	process	public	int	CONCRETE
3	3	calculate	public	int	CONCRETE
4	3	transform	private	int	CONCRETE
5	5	processMinor	public	string	STATIC
6	4	processMajor	public	string	CONCRETE
7	6	call	public	int	CONCRETE
8	6	reCall	public	int	CONCRETE
9	7	invoke	public	int	CONCRETE
10	7	reInvoke	public	int	CONCRETE
11	8	notify	public	double	CONCRETE
12	9	start	public	double	CONCRETE
13	10	calculate	public	int	ABSTRACT
14	11	calculate	public	int	CONCRETE
15	11	calculate	public	int	CONCRETE
16	11	calculate	public	int	CONCRETE
17	12	calculate	public	int	ABSTRACT
18	12	process	public	double	ABSTRACT
19	13	calculate	public	int	CONCRETE
20	13	process	public	double	CONCRETE
21	14	calculate	public	int	CONCRETE
22	14	process	public	double	CONCRETE
23	14	process	public	double	CONCRETE

2.6 ตาราง classdiagram.param

ตาราง classdiagram.param เป็นตารางที่ใช้เก็บข้อมูลของพารามิเตอร์ของเมท็อดแต่ละเมท็อดในแผนภาพคลาส โดยคอลัมน์ methodId จะอ้างอิงถึงคอลัมน์ methodId ของตาราง classdiagram.method ตารางที่ ข-10 แสดงการเก็บข้อมูลของพารามิเตอร์ของแต่ละเมท็อดในแผนภาพคลาส ideal_system.xml ในฐานข้อมูล

ตารางที่ ข-10 การเก็บข้อมูลของพารามิเตอร์ของแต่ละเมท็อดในแผนภาพคลาส ideal_system.xml

paramId	methodId	paramName	dataType	seqIdx
1	1	input	string	1
2	2	input	string	1
3	3	input	string	1
4	4	input	string	1
5	7	input	string	1
6	8	input	string	1
7	8	callId	int	2
8	9	input	string	1
9	10	processId	int	1
10	11	input	string	1
11	11	variant	double	2
12	13	input1	int	1
13	13	input2	int	2
14	14	input1	int	1
15	15	input1	int	1
16	15	input2	int	2
17	16	input1	int	1
18	16	input2	double	2
19	17	input1	int	1
20	17	input2	int	2
21	18	input1	double	1
22	18	input2	double	2

ตารางที่ ข-10 การเก็บข้อมูลของพารามิเตอร์ของแต่ละเมท็อดในแผนภาพคลาส ideal_system.xml (ต่อ)

paramId	methodId	paramName	dataType	seqIdx
23	19	input1	int	1
24	19	input2	int	2
25	20	input1	double	1
26	20	input2	double	2
27	21	input1	int	1
28	21	input2	int	2
29	22	input1	double	1
30	22	input2	double	2
31	23	input1	double	1
32	23	input2	double	2
33	23	input3	double	3

จากตัวอย่างการทวนสอบการนำเข้าแผนภาพคลาส ideal_system.xml พบว่าตัวสร้างสามารถรวบรวมแพ็คเกจ คลาส เมท็อด และเมท็อดซิกเนเจอร์ได้อย่างถูกต้องและครบถ้วน

ภาคผนวก ค

รายละเอียดของกรณีศึกษาระบบธนาคาร

บทนี้จะแสดงรายละเอียดของกรณีศึกษาระบบธนาคารในหัวข้อที่ 5.3 โดยจะกล่าวถึงโครงสร้างของระบบเพื่อใช้เป็นข้อมูลในการทดสอบตัวสร้าง

1. โครงสร้างของระบบ

ระบบธนาคารเป็นระบบที่ออกแบบด้วยรูปแบบเอ็มวีซีแสดงด้วยแผนภาพคลาสในรูปที่ ค-1 ในแผนภาพคลาสประกอบด้วย 7 แพ็คเกจ โดยแต่ละแพ็คเกจมีรายละเอียดดังต่อไปนี้

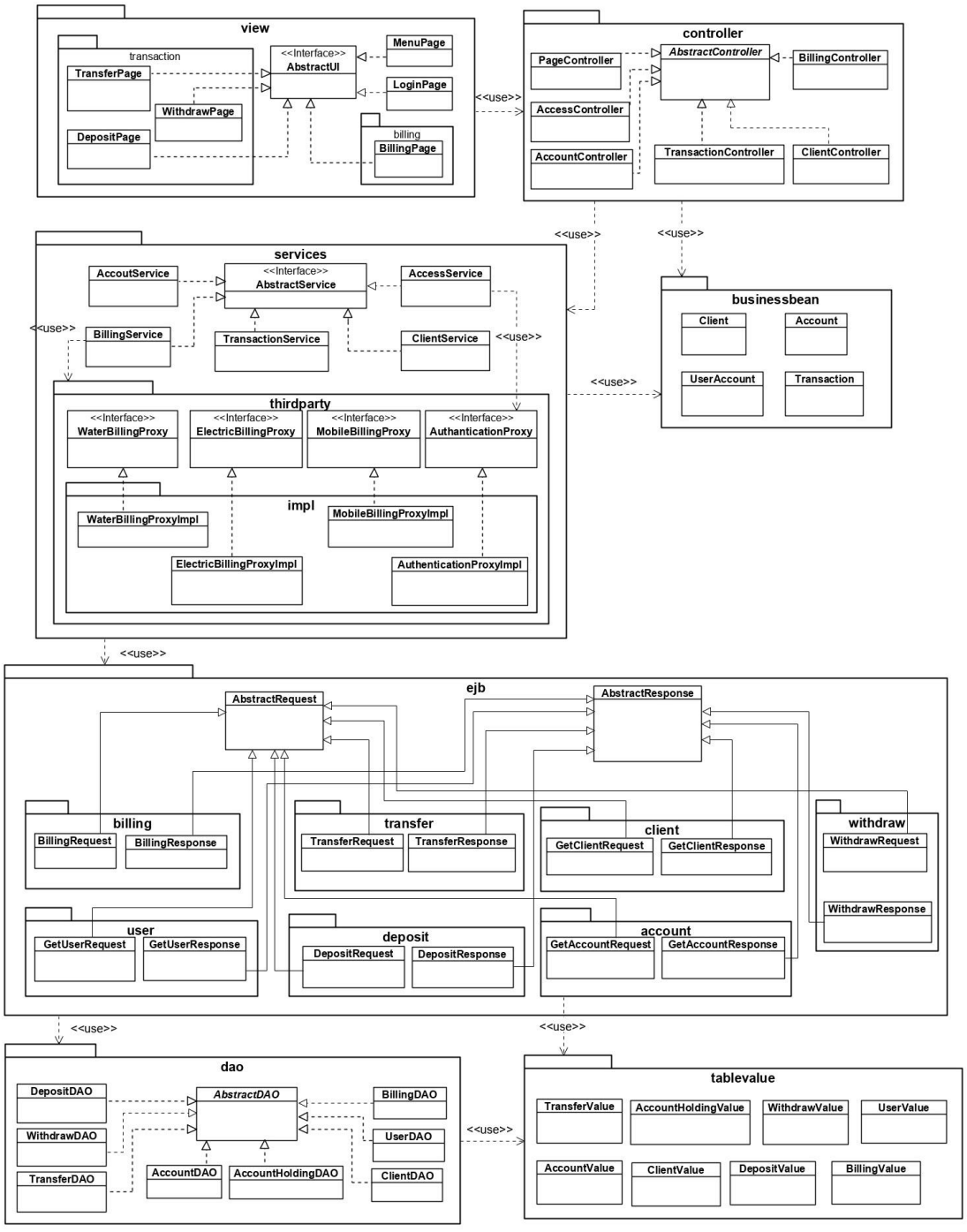
1. แพ็คเกจ view เป็นแพ็คเกจที่ประกอบด้วยคลาสในกลุ่มวิวตามรูปแบบเอ็มวีซี โดยภายในแพ็คเกจประกอบด้วยคลาส 7 ซึ่งเป็นคลาสที่แสดงถึงหน้าจอต่าง ๆ รูปที่ ค-2 แสดงรายละเอียดของคลาสแต่ละคลาสในแพ็คเกจ view

2. แพ็คเกจ controller เป็นแพ็คเกจที่ประกอบด้วยคลาสกลุ่มคอลโทลเลอร์ตามรูปแบบเอ็มวีซี โดยภายในแพ็คเกจประกอบด้วยคลาส 7 คลาสซึ่งเป็นคลาสที่ทำปฏิบัติฟังก์ชันที่ได้รับจากคลาสกลุ่ม view รูปที่ ค-3 แสดงรายละเอียดของคลาสแต่ละคลาสในแพ็คเกจ controller

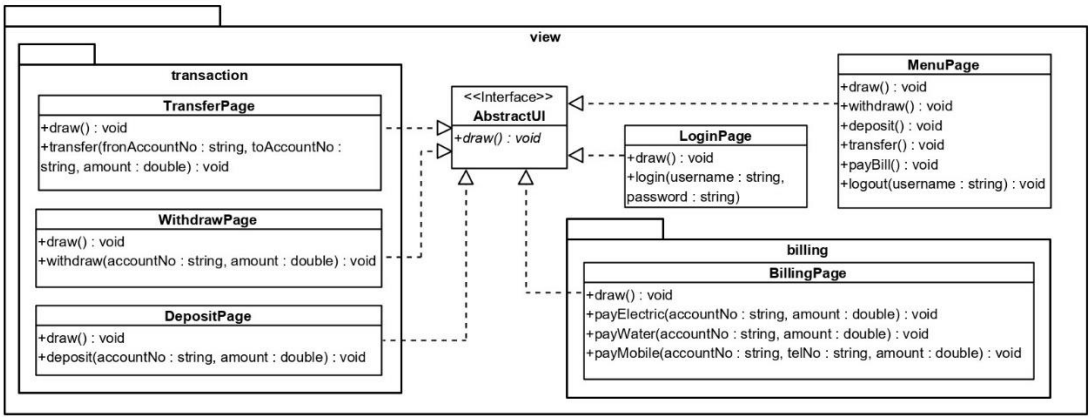
3. แพ็คเกจ businessbean เป็นแพ็คเกจที่ประกอบด้วยคลาสกลุ่มโมเดลตามรูปแบบเอ็มวีซี ซึ่งคลาสภายในแพ็คเกจนี้จะแสดงถึงข้อมูลของผู้ใช้ในเชิงธุรกิจ คลาส businessbean ประกอบด้วยคลาสทั้งหมด 4 คลาสซึ่งแสดงรายละเอียดของแต่ละคลาสในรูปที่ ค-4

4. แพ็คเกจ dao เป็นแพ็คเกจที่ประกอบด้วยคลาสที่ใช้เชื่อมต่อกับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database System) [20] ซึ่งเป็นฐานข้อมูลของระบบ โดยแพ็คเกจนี้ประกอบด้วยคลาสทั้งหมด 8 คลาสซึ่งแต่ละคลาสจะแสดงถึงตารางในฐานข้อมูลแต่ละตาราง รูปที่ ค-5 แสดงรายละเอียดของคลาสแต่ละคลาสในแพ็คเกจ dao

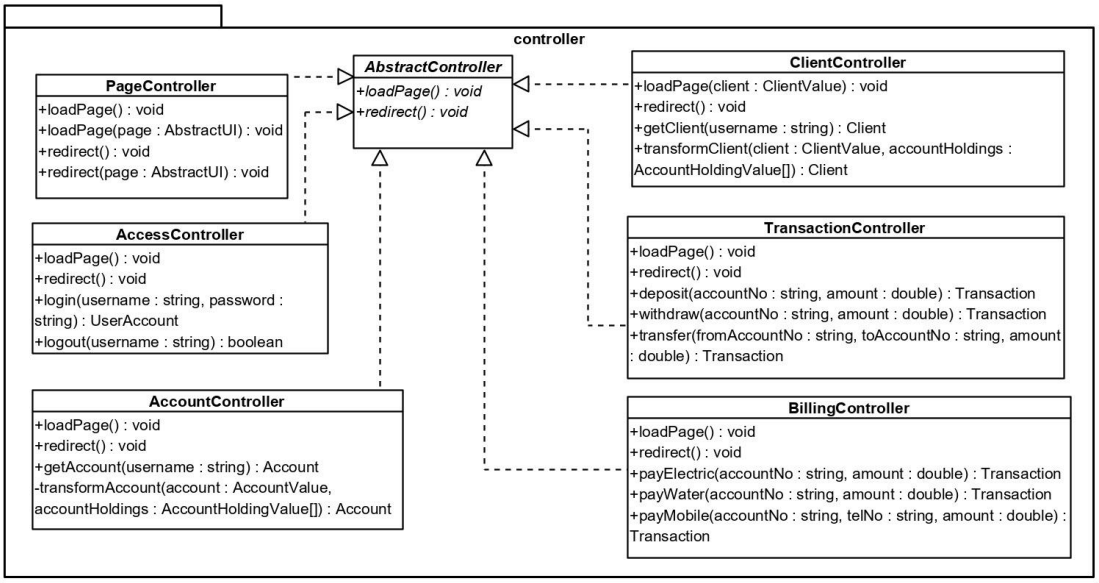
5. แพ็คเกจ services เป็นแพ็คเกจที่ประกอบด้วยคลาสที่ทำหน้าที่เป็นตัวกลางระหว่างคลาสฐานข้อมูลในแพ็คเกจและคลาสกลุ่มคอลโทลเลอร์ ซึ่งช่วยรับส่งและแปลงข้อมูลระหว่างคลาสทั้งสองกลุ่ม รายละเอียดของคลาสแต่ละคลาสในแพ็คเกจ services แสดงในรูปที่ ค-6 แพ็คเกจ services ยังประกอบด้วยแพ็คเกจย่อยหนึ่งแพ็คเกจได้แก่ แพ็คเกจ thirdparty ซึ่งประกอบด้วยคลาสที่ทำหน้าที่เป็นพร็อคซี่อินเตอร์เฟส (Proxy Interface) [17] ซึ่งจะเชื่อมต่อไประบบภายนอก และในแพ็คเกจ thirdparty ยังประกอบด้วยแพ็คเกจ impl ซึ่งเป็นแพ็คเกจที่ประกอบด้วยคลาสรูปธรรมที่สืบทอดมาจากพร็อคซี่อินเตอร์เฟส



รูปที่ ค-1 แผนภาพคลาสของระบบธนาคาร



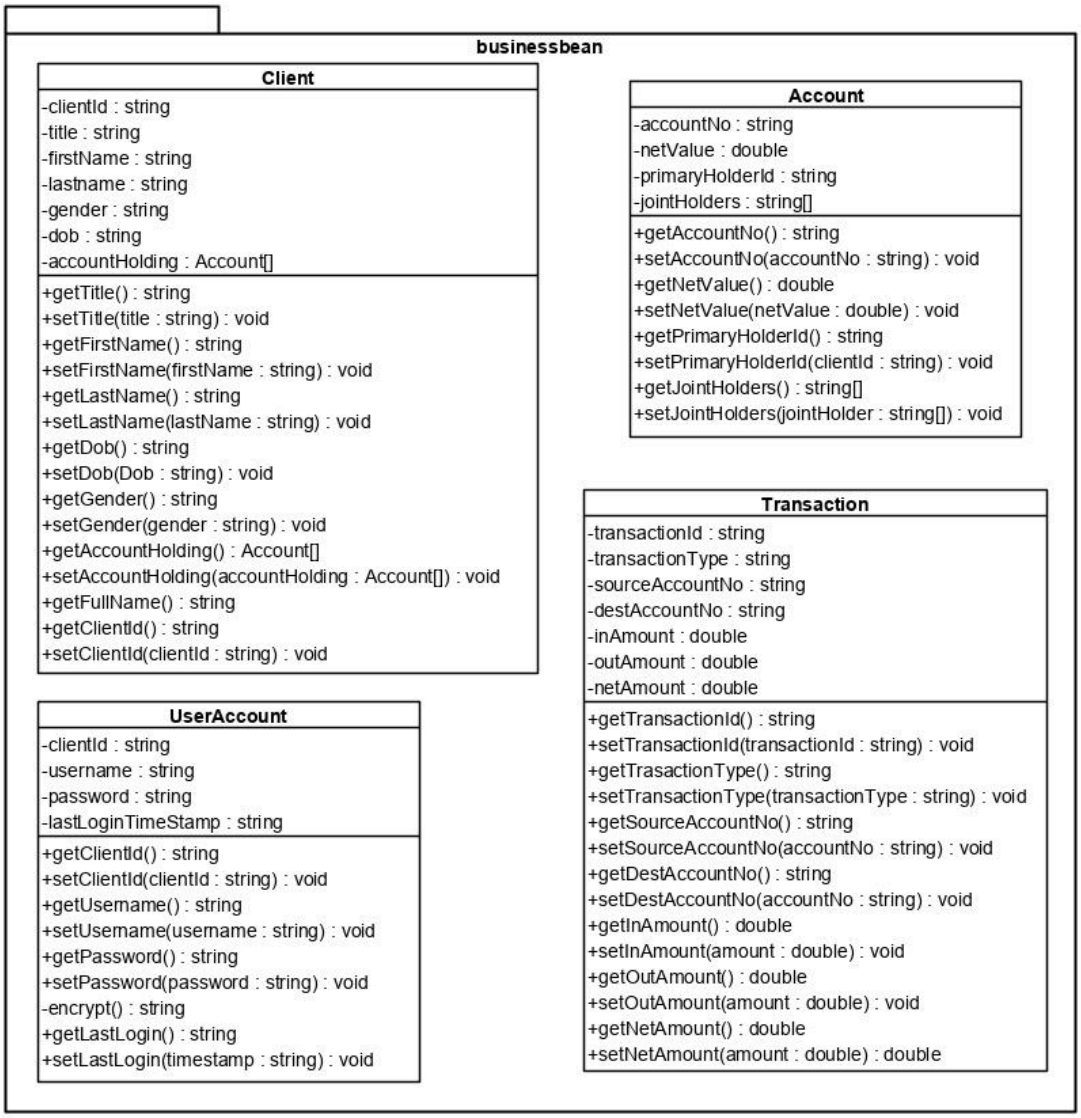
รูปที่ ค-2 แพ็คเกจ view



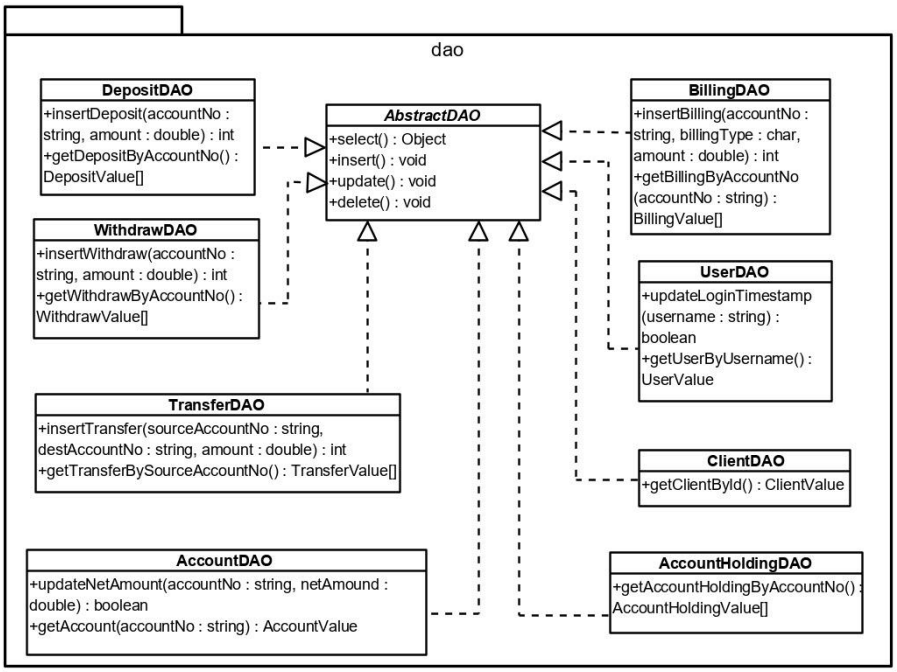
รูปที่ ค-3 แพ็คเกจ controller

6. แพ็คเกจ tablevalue เป็นแพ็คเกจที่ประกอบด้วยคลาสกลุ่มโมเดลตามรูปแบบเอ็มวีซีและแต่ละคลาสยังทำหน้าที่โครงสร้างข้อมูลของแต่ละตารางในฐานข้อมูล แพ็คเกจ tablevalue ประกอบด้วยคลาสทั้งหมด 8 คลาสโดยรายละเอียดของแต่ละคลาสในแพ็คเกจนี้แสดงในรูปที่ ค-7

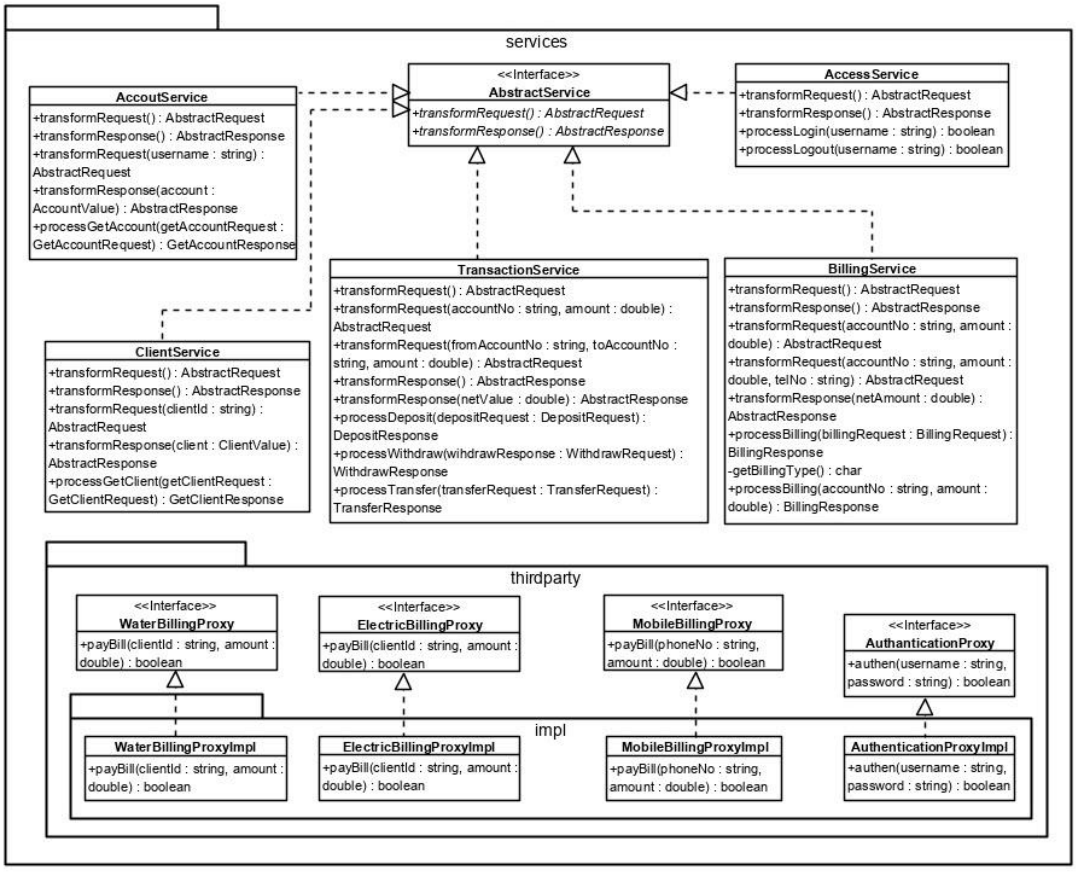
7. แพ็คเกจ ejb เป็นแพ็คเกจที่ประกอบด้วยคลาสที่ทำหน้าที่เป็นโครงสร้างข้อมูลที่ใช้รับส่งข้อมูลระหว่างคลาสในแพ็คเกจ services กับคลาสฐานข้อมูล โดยรายละเอียดของคลาสแต่ละคลาสในแพ็คเกจ ejb แสดงในรูปที่ ค-8



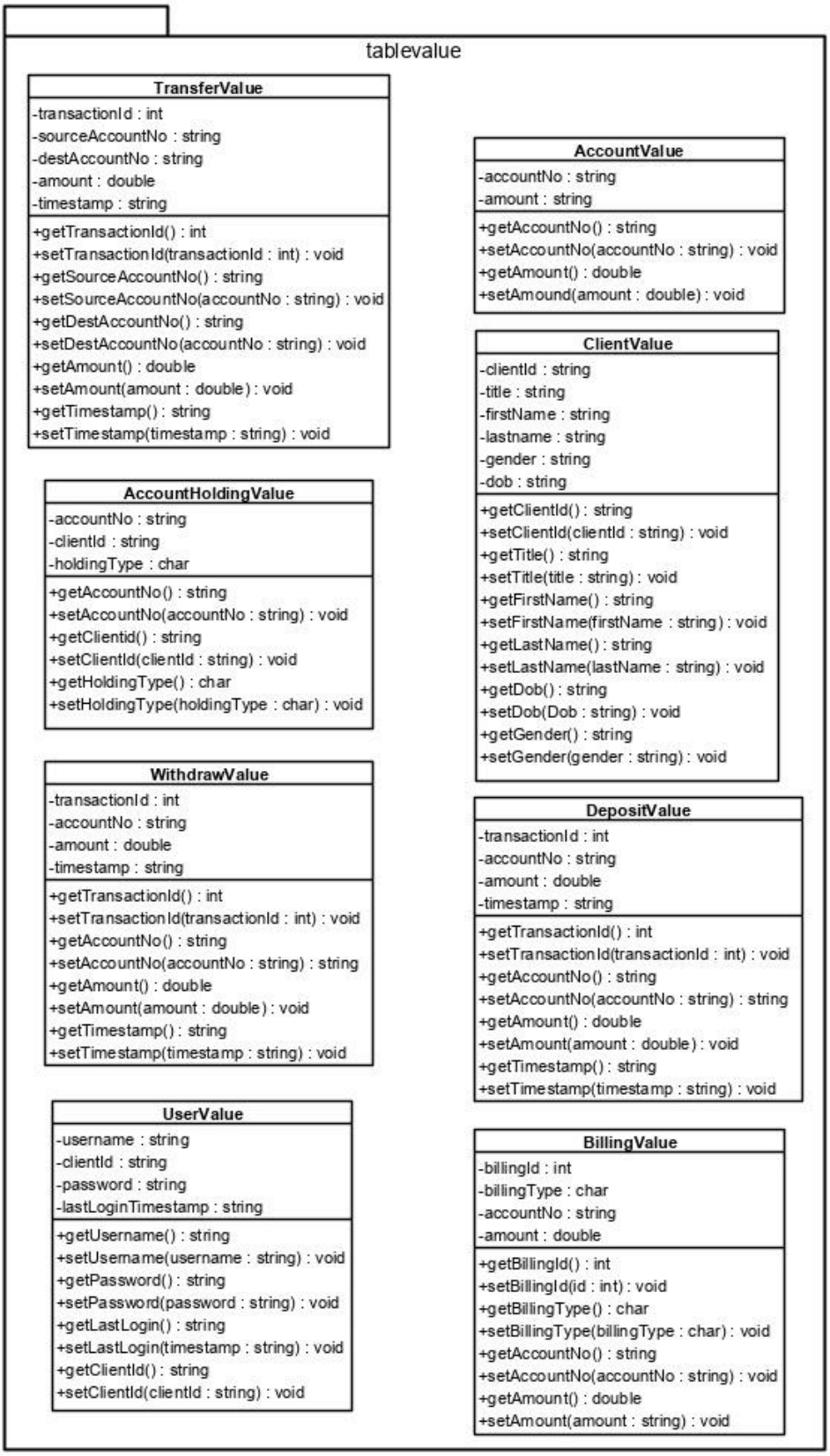
รูปที่ ค-4 แพ็คเกจ businessbean



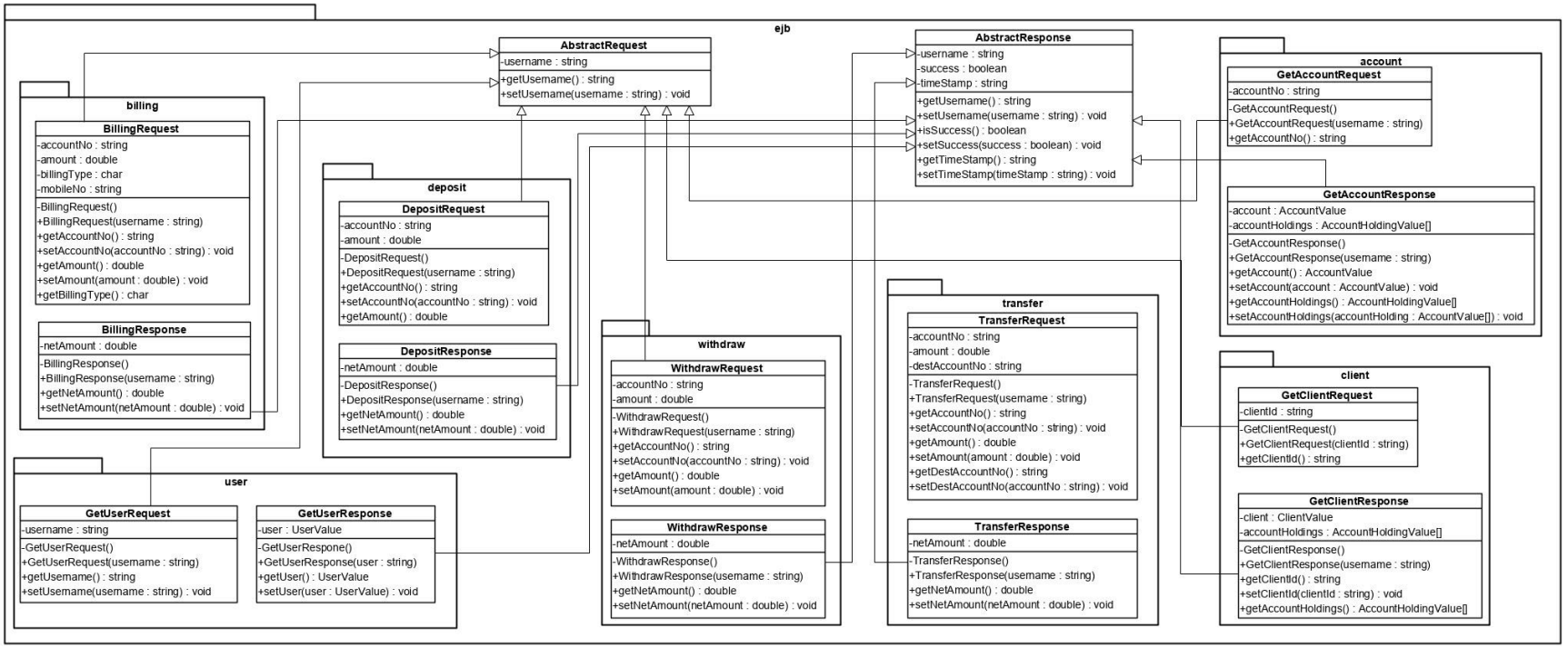
รูปที่ ค-5 แพ็คเกจ dao



รูปที่ ค-6 แพ็คเกจ services



รูปที่ ค-7 แพ็คเกจ tablevalue



รูปที่ ค-8 แพ็คเกจ ejb

ประวัติผู้เขียน

ชื่อ-สกุล	นาย พีรวุฒิ เหลืองเรืองโรจน์
วัน เดือน ปี เกิด	21 ธันวาคม 2538
สถานที่เกิด	กรุงเทพมหานคร
ที่อยู่ปัจจุบัน	88/191 หมู่ 4 ซอยกระทุ่มล้ม 27 ถนนพุทธมณฑลสายสี่ ตำบลกระทุ่มล้ม อำเภอสามพราน นครปฐม 73220
ผลงานตีพิมพ์	P. Luengruengroj and T. Suwannasart, "Stubs and Drivers Generator for Object-Oriented Program Testing Using Sequence and Class Diagrams," 2018 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII), Yonago, 2018, pp. 32-36.