บทที่ 3

คลาสของอุปกรณ์

ในงานวิจัยนี้เป็นการเขียนซอฟต์แวร์บนเครื่องไมโครคอมพิวเตอร์ เพื่อใช้เป็นเครื่องมือช่วยในการวาดรูปวงจรอิเล็กทรอนิกส์ทั่วไป เนื่องจากงานวิจัยนี้ได้นำเทคนิคการโปรแกรมเชิงวัตถุมาใช้ ดังนั้นงานส่วนใหญ่จะเป็นไปในรูปของการออกแบบคลาส เพื่อให้วัตถุเป็นวัตถุที่มีประสิทธิภาพสูง และคลาสนั้นสามารถนำกลับมาใช้ใหม่ได้สามารถเข้าใจได้ง่าย และสามารถบำรุงรักษาได้ง่าย ซึ่งในงานวิจัยนี้ได้จัดแบ่งคลาสเป็นไปตามกลุ่มของอุปกรณ์

<u>ชนิดของอุปกรณ์</u>

ในการสร้างโปรแกรมเมื่อมองดูข้อมูลเข้าของตัวโปรแกรมก็จะเห็นว่าเป็นตัวอุปกรณ์ต่างๆ ซึ่งผู้ใช้ต้องการเรียกขึ้นมา เพื่อต่อเป็นวงจรไฟฟ้า แล้วจึงนำวงจรนั้นไปวิเคราะห์ตามแบบที่ต้องการ เพื่อหาคำตอบของวงจรที่ต้องการ ดังนั้นจึงได้แบ่งข้อมูลเข้าตามชนิดของอุปกรณ์เป็น สายไฟ, กราวด์, แหล่งกำเนิดแรงดัน, แหล่งกำเนิดกระแส, ตัวต้านทาน, ตัวเหนี่ยวนำ, ตัวเก็บประจุ, ไดโอด, ทรานซิสเตอร์, หม้อแปลงไฟฟ้า, ออปแอมป์, สวิทช์, โมเดล และ Sweep ซึ่งเก็บแบบของการวิเคราะห์วงจรนั้น โดยแต่ละอุปกรณ์จะมีพิกัดบนจอภาพ และชื่อของอุปกรณ์นั้นเป็นข้อมูลร่วมกันของทุกอุปกรณ์ และมีข้อมูลเฉพาะ ซึ่งมีในแต่ละอุปกรณ์ไม่เหมือนกันก็คือรูปร่าง และข้อมูลเฉพาะอุปกรณ์นั้น ต่อไปนี้จะแสดงรูปร่างอุปกรณ์ และข้อมูลของอุปกรณ์นั้นๆ

- สายไฟ

ตำแหน่งปลายที่หนึ่งของสายไฟ
ตำแหน่งปลายที่สองของสายไฟ

ข้อมูลแสดงว่าสายไฟปรากฏหรือไม่

- กราวด์

ตำแหน่งของกราวด์
ข้อมูลแสดงว่ากราวด์ปรากฏหรือไม่

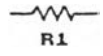
- แหล่งกำเนิดแรงดัน

ตำแหน่งของแหล่งกำเนิดแรงดัน
ข้อมูลแสดงว่าแหล่งกำเนิดแรงดันปรากฏหรือไม่
ชื่อของแหล่งกำเนิดแรงดัน
ค่าความต้านทานภายใน
แบบของแหล่งกำเนิดแรงดัน
ข้อมูลของแหล่งกำเนิดแรงดันแต่ละแบบ

- แหล่งกำเนิดกระแส

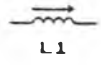
ตำแหน่งของแหล่งกำเนิดกระแส
ข้อมูลแสดงว่าแหล่งกำเนิดกระแสปรากฏหรือไม่
ชื่อของแหล่งกำเนิดกระแส
ค่าความต้านทานภายใน
แบบของแหล่งกำเนิดกระแส
ข้อมูลของแหล่งกำเนิดกระแสแต่ละแบบ

- ตัวต้านทาน



ตำแหน่งของตัวต้านทาน
ข้อมูลแสดงว่าตัวต้านทานปรากฏหรือไม่
ชื่อของตัวต้านทาน
ค่าความต้านทาน

- ตัวเหนี่ยวนำ

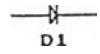


ตำแหน่งของตัวเหนี่ยวนำ
ข้อมูลแสดงว่าตัวเหนี่ยวนำปรากฏหรือไม่
ชื่อของตัวเหนี่ยวนำ
ค่าความเหนี่ยวนำ
ค่ากระแสเริ่มต้น

- ตัวเก็บประจุ

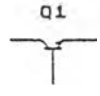
ตำแหน่งของตัวเก็บประจุ
ข้อมูลแสดงว่าตัวเก็บประจุปรากฏหรือไม่
ชื่อของตัวเก็บประจุ
ค่าความจุ
ค่าแรงดันเริ่มต้น

- ไดโอด



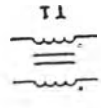
ตำแหน่งของไดโอด
ข้อมูลแสดงว่าไดโอดปรากฏหรือไม่
ชื่อของไดโอด
เบอร์ของไดโอด
ค่าแรงดันที่ทำให้ไดโอดนำกระแส

- ทรานซิสเตอร์



ตำแหน่งของทรานซิสเตอร์
ข้อมูลแสดงว่าทรานซิสเตอร์ปรากฏหรือไม่
ชื่อของทรานซิสเตอร์
เบอร์ของทรานซิสเตอร์
ค่าแรงดันระหว่างขา Base กับขา Collector
ค่าแรงดันระหว่างขา Base กับขา Emitter

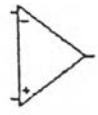
- หม้อแปลงไฟฟ้า



ตำแหน่งของหม้อแปลง
ข้อมูลแสดงว่าหม้อแปลงปรากฏหรือไม่
ชื่อของหม้อแปลง
จำนวนชุดขดลวดของหม้อแปลง
จำนวนรอบของขดลวดแต่ละชุด

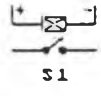
ค่าความต้านทานภายในขดลวดแต่ละชุด

- ออปแอมป์



ตำแหน่งของออปแอมป์
ข้อมูลแสดงว่าออปแอมป์ปรากฏหรือไม่
ชื่อของออปแอมป์
เบอร์ของออปแอมป์
แรงดันขาเข้าเริ่มต้น
แรงดันขาออกเริ่มต้น

- สวิทช์



ตำแหน่งของสวิทช์
ข้อมูลแสดงว่าสวิทช์ปรากฏหรือไม่
ชื่อของสวิทช์
แบบของสวิทช์
ค่าความต้านทานภายใน
ค่าแรงดันที่ทำให้สวิทช์ทำงาน
ค่าแรงดันที่คอยล์

- โมเดล

โมเดลเป็นที่เก็บคุณสมบัติของไดโอด, ทรานซิสเตอร์ และออปแอมป์ เบอร์ที่ผู้ใช้ป้อนเข้ามา และเก็บข้อมูลเกี่ยวกับเบอร์นั้น ดังนั้นข้อมูลของโมเดลก็คือ

ชื่อของโมเดล
ชนิดของอุปกรณ์

ข้อมูลของอุปกรณ์นั้น

- Sweep

Sweep เป็นที่เก็บแบบของการวิเคราะห์วงจรที่วาดข้อมูลที่เก็บใน Sweep จะเป็นแบบของการวิเคราะห์ และข้อมูลที่จำเป็นสำหรับแบบการวิเคราะห์นั้นๆ

ที่กล่าวมาข้างต้นเป็นข้อมูลที่เกี่ยวข้องกับอุปกรณ์ต่างๆ ที่ใช้ในการสร้างวงจรขึ้นมา เป็นข้อมูลจำเพาะของแต่ละอุปกรณ์ตามหลักการโปรแกรมเชิงวัตถุการติดต่อกับข้อมูลแต่ละตัวนั้นจะต้องผ่านส่วนเชื่อมต่อ ซึ่งวัตถุอนุญาตให้เท่านั้น จึงจะทำการติดต่อกับข้อมูลเหล่านั้นได้ ในหัวข้อถัดไปจะอธิบายถึงส่วนเชื่อมต่อของแต่ละอุปกรณ์ เพื่อใช้ในการติดต่อกับวัตถุ

<u>การติดต่อกับวัตถุ</u>

การติดต่อกับวัตถุให้ทำงานต่างๆ ตามที่เราต้องการนั้นจะมีฟังก์ชัน ซึ่งเราใช้ในการติดต่อเข้าถึงข้อมูลในวัตถุนั้น โดยมากเราจะเรียกฟังก์ชันเหล่านี้ว่า message เมื่อเรามีอุปกรณ์ต่างๆ ตามที่ได้จัดแบ่งไว้ และมีข้อมูลต่างๆในอุปกรณ์นั้นแล้ว อุปกรณ์เหล่านั้นต้องมีความสามารถในการที่จะแสดงข้อมูล ซึ่งผู้ใช้ต้องการออกมาได้ หรือนำข้อมูลใหม่ ซึ่งผู้ใช้กำหนดใส่เข้าไปได้ ตลอดจนอุปกรณ์เหล่านั้นต้องสามารถเคลื่อนที่ และหมุน เพื่อให้ผู้ใช้ต่ออุปกรณ์เหล่านั้นเป็นวงจรอยู่บนจอภาพคอมพิวเตอร์ได้ ดังนั้นอุปกรณ์ต่างๆ ควรมีพังก์ชันในการทำงานกับอุปกรณ์ ดังนี้

- สายไฟ

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่งแต่ละปลายของสายไฟ
ฟังก์ชันในการแสดง และไม่แสดงสายไฟ

- กราวด์

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง
ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม
ฟังก์ชันในการแสดง และไม่แสดงกราวด์
ฟังก์ชันในการย้ายตำแหน่งกราวด์

- แหล่งกำเนิดแรงดัน

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงแหล่งกำเนิดแรงดัน

ฟังก์ชันในการย้ายตำแหน่งแหล่งกำเนิดแรงดัน

ฟังก์ชันในการกำหนดค่า และอ่านค่าความต้านทานภายใน

ฟังก์ชันในการกำหนดแบบ และอ่านแบบของแหล่งกำเนิดแรงดัน

ฟังก์ชันในการกำหนดข้อมูล และอ่านข้อมูลของแหล่งกำเนิดแรงดันแต่ละแบบ

- แหล่งกำเนิดกระแส

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงแหล่งกำเนิดกระแส

ฟังก์ชันในการย้ายตำแหน่งแหล่งกำเนิดกระแส

ฟังก์ชันในการกำหนดค่า และอ่านค่าความต้านทานภายใน

ฟังก์ชันในการกำหนดแบบ และอ่านแบบของแหล่งกำเนิดกระแส

ฟังก์ชันในการกำหนดข้อมูล และอ่านข้อมูลของแหล่งกำเนิดกระแสแต่ละแบบ

- ตัวต้านทาน

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงตัวต้านทาน

ฟังก์ชันในการย้ายตำแหน่งตัวด้านทาน
ฟังก์ชันในการกำหนดค่า และอ่านค่าความด้านทาน

- ตัวเหนี่ยวนำ

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง
ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม
ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงตัวเหนี่ยวนำ
ฟังก์ชันในการย้ายตำแหน่งตัวเหนี่ยวนำ
ฟังก์ชันในการกำหนดค่า และอ่านค่าความเหนี่ยวนำไฟฟ้า
ฟังก์ชันในการกำหนดค่า และอ่านค่ากระแสเริ่มต้น

- ตัวเก็บประจุ

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านตำแหน่ง
ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม
ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงตัวเก็บประจุ
ฟังก์ชันในการย้ายตำแหน่งตัวเก็บประจุ
ฟังก์ชันในการกำหนดค่า และอ่านค่าความจุไฟฟ้า
ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันเริ่มต้น

- ไดโอด

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านค่าตำแหน่ง
ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม
ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงไดโอด
ฟังก์ชันในการย้ายตำแหน่งไดโอด

ฟังก์ชันในการกำหนด และอ่านเบอร์ของไดโอด

ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันที่ทำให้ไดโอดนำกระแส

- ทรานซิสเตอร์

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านค่าตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงทรานซิสเตอร์

ฟังก์ชันในการย้ายตำแหน่งทรานซิสเตอร์

ฟังก์ชันในการกำหนด และอ่านเบอร์ของทรานซิสเตอร์

ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันระหว่างขา Base กับขา Collector

ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันระหว่างขา Base กับขา Emitter

- หม้อแปลงไฟฟ้า

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านค่าตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์

ฟังก์ชันในการแสดง และไม่แสดงหม้อแปลงไฟฟ้า

ฟังก์ชันในการย้ายตำแหน่งหม้อแปลงไฟฟ้า

ฟังก์ชันในการกำหนดจำนวนชุด และอ่านจำนวนชุดขดลวดของหม้อแปลงไฟฟ้า

ฟังก์ชันในการกำหนดจำนวนรอบ และอ่านจำนวนรอบของขดลวดแต่ละชุด

ฟังก์ชันในการกำหนดค่า และอ่านค่าความต้านทานภายในขดลวดแต่ละชุด

- ออปแอมป์

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านค่าตำแหน่ง

ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม

ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงออปแอมป์
ฟังก์ชันในการย้ายตำแหน่งออปแอมป์
ฟังก์ชันในการกำหนด และอ่านเบอร์ของออปแอมป์
ฟังก์ชันในการกำหนด และอ่านแรงดันขาเข้าเริ่มต้น
ฟังก์ชันในการกำหนด และอ่านแรงดันขาออกเริ่มต้น

- สวิทช์

ฟังก์ชันในการกำหนดตำแหน่ง และอ่านค่าตำแหน่ง
ฟังก์ชันในการกำหนดค่า และอ่านค่ามุม
ฟังก์ชันในการกำหนดชื่อ และอ่านชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงชื่ออุปกรณ์
ฟังก์ชันในการแสดง และไม่แสดงสวิทช์
ฟังก์ชันในการย้ายตำแหน่งสวิทช์
ฟังก์ชันในการกำหนด และอ่านแบบของสวิทช์
ฟังก์ชันในการกำหนดค่า และอ่านค่าความต้านทานภายใน
ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันที่ทำให้สวิทช์ทำงาน
ฟังก์ชันในการกำหนดค่า และอ่านค่าแรงดันที่คอยล์

- โมเดล

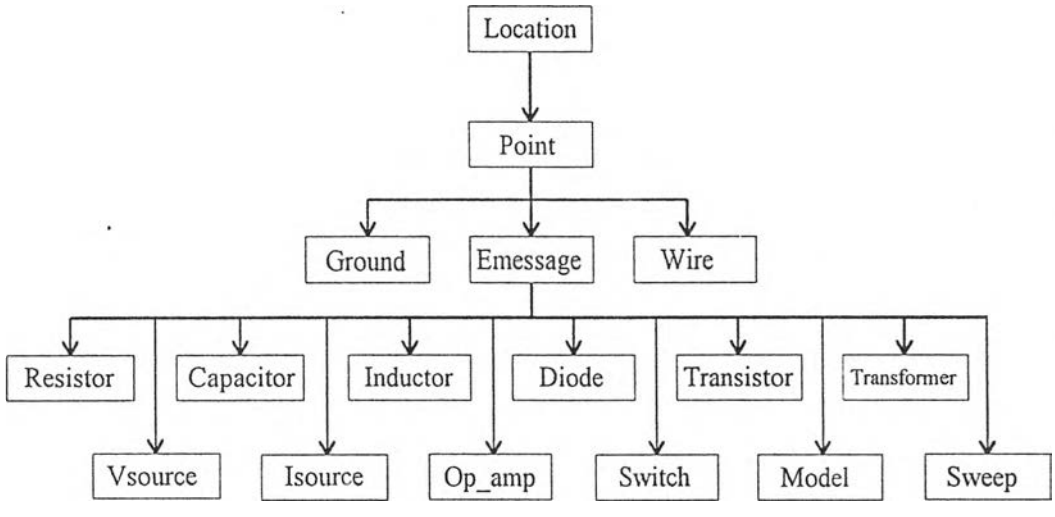
ฟังก์ชันในการกำหนดชื่อ และอ่านชื่อของโมเดล
ฟังก์ชันในการกำหนดชนิด และอ่านชนิดของอุปกรณ์
ฟังก์ชันในการกำหนดข้อมูล และอ่านข้อมูลของแต่ละอุปกรณ์

- Sweep

ฟังก์ชันในการกำหนด และอ่านค่าแบบของการวิเคราะห์วงจร
ฟังก์ชันในการกำหนดข้อมูล และอ่านข้อมูลของการวิเคราะห์วงจรแต่ละแบบ

สรุปคลาสของอุปกรณ์

จากสองหัวข้อข้างต้นจะเห็นว่าอุปกรณ์ควรมีข้อมูล และฟังก์ชันบางส่วน ซึ่งซ้ำกันอยู่ร่วมกัน ดังนั้นในการโปรแกรม จึงได้จัดให้มีคลาส Location ซึ่งเก็บข้อมูลเกี่ยวกับตำแหน่งและมุมของอุปกรณ์ โดยมีฟังก์ชันในการกำหนดค่าตำแหน่ง, อ่านค่าตำแหน่ง กำหนดค่ามุม และอ่านค่ามุมของอุปกรณ์ และมีคลาส Point เก็บข้อมูล ซึ่งแสดงว่าอุปกรณ์ปรากฏหรือไม่ โดยมีฟังก์ชันชนิดที่เป็น Virtual Function ใช้ในการแสดงอุปกรณ์, ไม่แสดงอุปกรณ์ และย้ายตำแหน่งอุปกรณ์ นอกจากนั้นมีคลาส Emessage ซึ่งเก็บข้อมูลที่เป็นชื่อของอุปกรณ์ โดยมีฟังก์ชันในการกำหนดชื่ออุปกรณ์, อ่านชื่ออุปกรณ์, แสดงชื่ออุปกรณ์ และไม่แสดงชื่ออุปกรณ์ ดังนั้นความสัมพันธ์ของการ Inheritance ของคลาสต่างๆ จึงเป็นดังแผนภาพ



รูปที่ 3.1 แสดงโครงสร้างการ Inheritance ของคลาสหลักในงานวิจัย

โดยมีรายละเอียดของการนิยามคลาสในแต่ละคลาสดังนี้

```
class Location {
protected:
  int X;
  int Y;
  angle Angle;
```

```
 void init(int InitX, int InitY, angle InitAngle = right)
 {
   X = InitX;
   Y = InitY;
   Angle = InitAngle;
 }
public:
 Location(int InitX, int InitY, angle InitAngle = right)
   {X = InitX; Y = InitY; Angle = InitAngle;}
 Location() { init(50,50); }
 int GetX() {return X;}
 int GetY() {return Y;}
 void GiveX(int Gx) {X=Gx;}
 void GiveY(int Gy) {Y=Gy;}
 void GiveAngle(angle Gangle) {Angle=Gangle;}
 angle GetAngle() {return Angle;}
};

class Point : public Location {
protected:
  Boolean Visible;
  void init(Boolean InitVisible)
  {
    Visible = InitVisible;
  }
public:
  Point(int InitX, int InitY);
  Point() { init(false); }
  virtual void Show();       // Show and Hide are virtual
  virtual void Hide();
```

```
 virtual void ShowCursor(); // Cursor Show and Hide are virtual
 virtual void HideCursor();
 virtual void Drag(int DragBy); // new virtual drag function
 virtual void DragCursor(int DragBy);
 Boolean IsVisible() {return Visible;}
 void GiveVisible(Boolean visible) {Visible = visible;}
 void MoveTo(int NewX, int NewY, angle NewAngle);
 void MoveToCursor(int NewX, int NewY);
};

class EMessage : public Point {
  char far *msg;              // message to be displayed
  int Font;                   // BGI font to use
  int Field;                  // size of field for text scaling
  void init(int MsgFont, int FieldSize, char *text)
  {
  if (!(msg = new char[9])) {
    exit (1);
  }
    strcpy(msg,text);
    Font = MsgFont;
    Field = FieldSize;
  }
public:
  EMessage(int InitMsgX, int InitMsgY, int MsgFont, int FieldSize,
     char *text);
  EMessage() { init(0,12,""); }
  ~EMessage() { delete(msg); }      // new add for del msg in init()
  virtual void Givemsg(char *);
  virtual char far *Getmsg() { return msg; }
```

```
 void Show(angle Angle);          // show message
 void Hide(angle Angle);          // hide message
};

class Wire : public Point {    .     // Derived from class Point
protected:
  int x2;
  int y2;
  void init(int Initx2, int Inity2)
  {
    x2 = Initx2;
    y2 = Inity2;
  }
public:
  Wire(int InitX, int InitY, int Initx2, int Inity2);
  Wire() { init(50,50); }
  void Show();
  void Hide();
  int GetX2() {return x2;}
  int GetY2() {return y2;}
  void GiveX2(int Gx) {x2=Gx;}
  void GiveY2(int Gy) {y2=Gy;}
};

class Ground : public Point {     // Derived from class Point
public:
  Ground(int InitX, int InitY);
  Ground() {}
  void Show();
  void Hide();
```

```
};

class Vsource : public EMessage {        // Derived from class EMessage
protected:
  char far *Rs;
  char far *Type;
  char far *Vdc_ac_m_BrkPts_CtlFr;
  char far *f_CtlTo;
  char far *Ph_Gain;
  char far *T1;
  char far *V1;
  char far *T2;
  char far *V2;
  char far *T3;
  char far *V3;
  char far *T4;
  char far *V4;
  char far *T5;
  char far *V5;
  char far *T6;
  char far *V6;
  void init(char *InitRs, char *InitType, char *InitVdc_ac_m_BrkPts_CtlFr,
        char *Initf_CtlTo, char *InitPh_Gain,
        char *InitT1, char *InitV1, char *InitT2, char *InitV2,
        char *InitT3, char *InitV3, char *InitT4, char *InitV4,
        char *InitT5, char *InitV5, char *InitT6, char *InitV6)
  {
    Rs = InitRs;
    Type = InitType;
    Vdc_ac_m_BrkPts_CtlFr = InitVdc_ac_m_BrkPts_CtlFr;
```

```
  f_CtlTo = Initf_CtlTo;
  Ph_Gain = InitPh_Gain;
  T1 = InitT1; V1 = InitV1;    T2 = InitT2;     V2 = InitV2;
  T3 = InitT3; V3 = InitV3;    T4 = InitT4;     V4 = InitV4;
  T5 = InitT5; V5 = InitV5;    T6 = InitT6;     V6 = InitV6;
 }
 void GiveVdc_ac_m_BrkPts_CtlFr(char *InitVdc_ac_m_BrkPts_CtlFr);
 char *GetVdc_ac_m_BrkPts_CtlFr(void);
 void Givef_CtlTo(char *Initf_CtlTo);
 char *Getf_CtlTo(void);
 void GivePh_Gain(char *InitPh_Gain);
 char *GetPh_Gain(void);
public:
  Vsource(char *Initmsg, int Font, char *InitRs,
       int InitX, int InitY, char *InitType,
       char *InitVdc_ac_m_BrkPts_CtlFr,
       char *Initf_CtlTo, char *InitPh_Gain,
       char *T1, char *V1, char *T2, char *V2, char *T3, char *V3,
       char *T4, char *V4, char *T5, char *V5, char *T6, char *V6);
  Vsource() { init("","","","","","","","","","","","","","","","",""); }
  void Show();
  void Hide();
  void GiveValue(char *);
  char *GetValue(void);
  void GiveRs(char *);
  char *GetRs(void);
  void GiveType(char *);
  char *GetType(void);
  void GiveVdc(char *);
  char *GetVdc(void);
```

```
void GiveVac(char *);
char *GetVac(void);
void GiveVm(char *);
char *GetVm(void);
void GiveBrkPts(char *);
char *GetBrkPts(void);
void GiveCtlFr(char *);
char *GetCtlFr(void);
void Givef(char *);
char *Getf(void);
void GiveCtlTo(char *);
char *GetCtlTo(void);
void GivePh(char *);
char *GetPh(void);
void GiveGain(char *);
char *GetGain(void);
void GiveT1(char *);
char *GetT1(void);
void GiveT2(char *);
char *GetT2(void);
void GiveT3(char *);
char *GetT3(void);
void GiveT4(char *);
char *GetT4(void);
void GiveT5(char *);
char *GetT5(void);
void GiveT6(char *);
char *GetT6(void);
void GiveV1(char *);
char *GetV1(void);
```

```
  void GiveV2(char *);
  char *GetV2(void);
  void GiveV3(char *);
  char *GetV3(void);
  void GiveV4(char *);
  char *GetV4(void);
  void GiveV5(char *);
  char *GetV5(void);
  void GiveV6(char *);
  char *GetV6(void);
};

class Isource : public EMessage {          // Derived from class EMessage
protected:
  char far *Gs;
  char far *Type;
  char far *Idc_ac_m_BrkPts_CtlFr;
  char far *f_CtlTo;
  char far *Ph_Gm;
  char far *T1;
  char far *I1;
  char far *T2;
  char far *I2;
  char far *T3;
  char far *I3;
  char far *T4;
  char far *I4;
  char far *T5;
  char far *I5;
  char far *T6;
```

```
char far *I6;
void init(char *InitGs, char *InitType, char *InitIdc_ac_m_BrkPts_CtlFr,
      char *Initf_CtlTo, char *InitPh_Gm,
      char *InitT1, char *InitI1, char *InitT2, char *InitI2,
      char *InitT3, char *InitI3, char *InitT4, char *InitI4,
      char *InitT5, char *InitI5, char *InitT6, char *InitI6)
{
  Gs = InitGs;
  Type = InitType;
  Idc_ac_m_BrkPts_CtlFr = InitIdc_ac_m_BrkPts_CtlFr;
  f_CtlTo = Initf_CtlTo;
  Ph_Gm = InitPh_Gm;
  T1 = InitT1; I1 = InitI1;     T2 = InitT2;    I2 = InitI2;
  T3 = InitT3; I3 = InitI3;     T4 = InitT4;    I4 = InitI4;
  T5 = InitT5; I5 = InitI5;     T6 = InitT6;    I6 = InitI6;
}
void GiveIdc_ac_m_BrkPts_CtlFr(char *InitIdc_ac_m_BrkPts_CtlFr);
char *GetIdc_ac_m_BrkPts_CtlFr(void);
void Givef_CtlTo(char *Initf_CtlTo);
char *Getf_CtlTo(void);
void GivePh_Gm(char *InitPh_Gm);
char *GetPh_Gm(void);
public:
 Isource(char *Initmsg, int Font, char *InitGs,
       int InitX, int InitY, char *InitType,
       char *InitIdc_ac_m_BrkPts_CtlFr,
       char *Initf_CtlTo, char *InitPh_Gm,
       char *T1, char *I1, char *T2, char *I2, char *T3, char *I3,
       char *T4, char *I4, char *T5, char *I5, char *T6, char *I6);
 Isource() { init("","","","","","","","","","","","","","","","",""); }
```

```
void Show();
void Hide();
void GiveValue(char *);
char *GetValue(void);
void GiveGs(char *);
char *GetGs(void);
void GiveType(char *);
char *GetType(void);
void GiveIdc(char *);
char *GetIdc(void);
void GiveIac(char *);
char *GetIac(void);
void GiveIm(char *);
char *GetIm(void);
void GiveBrkPts(char *);
char *GetBrkPts(void);
void GiveCtlFr(char *);
char *GetCtlFr(void);
void Givef(char *);
char *Getf(void);
void GiveCtlTo(char *);
char *GetCtlTo(void);
void GivePh(char *);
char *GetPh(void);
void GiveGm(char *);
char *GetGm(void);
void GiveT1(char *);
char *GetT1(void);
void GiveT2(char *);
char *GetT2(void);
```

```
void GiveT3(char *);
char *GetT3(void);
void GiveT4(char *);
char *GetT4(void);
void GiveT5(char *);
char *GetT5(void);
void GiveT6(char *);
char *GetT6(void);
void GiveI1(char *);
char *GetI1(void);
void GiveI2(char *);
char *GetI2(void);
void GiveI3(char *);
char *GetI3(void);
void GiveI4(char *);
char *GetI4(void);
void GiveI5(char *);
char *GetI5(void);
void GiveI6(char *);
char *GetI6(void);
};

class Resistor : public EMessage {        // Derived from class EMessage
protected:
  char far *Value;
  void init(char *InitValue)
  {
  if (!(Value = new char[9])) {
    exit (1);
  }
```

```
    Value = InitValue;
  }
public:
  Resistor(char *Initmsg, int Font, char *InitValue,
        int InitX, int InitY);
  Resistor() { init("1"); }
  void Show();
  void Hide();
  void GiveValue(char *);
  char *GetValue(void);
};


class Inductor : public EMessage {        // Derived from class EMessage
protected:
  char far *Value;
  char far *Il0;
  void init(char *InitValue, char *InitIl0)
  {
    Value = InitValue;
    Il0 = InitIl0;
  }
public:
  Inductor(char *Initmsg, int Font, char *InitValue, char *InitIl0,
        int InitX, int InitY);
  Inductor() { init("1","0"); }
  void Show();
  void Hide();
  void GiveValue(char *);
  char *GetValue(void);
```

```
  void GiveIl0(char *);
  char *GetIl0(void);
};

class Capacitor : public EMessage {       // Derived from class EMessage
protected:
  char far *Value;
  char far *Vc0;
  void init(char *InitValue, char *InitVc0)
  {
    Value = InitValue;
    Vc0 = InitVc0;
  }
public:
  Capacitor(char *Initmsg, int Font, char *InitValue, char *InitVc0,
        int InitX, int InitY);
  Capacitor() { init("1","0"); }
  void Show();
  void Hide();
  void GiveValue(char *);
  char *GetValue(void);
  void GiveVc0(char *);
  char *GetVc0(void);
};

class Diode : public EMessage {           // Derived from class EMessage
protected:
  char far *Part;
  char far *Vd0;
  void init(char *InitPart, char *InitVd0)
```

```
{
  Part = InitPart;
  Vd0 = InitVd0;
}
public:
 Diode(char *Initmsg, int Font, char *InitPart, char *InitVd0,
       int InitX, int InitY);
 Diode() { init("1N916","0"); }
 void Show();
 void Hide();
 void GiveModel(char *);
 char *GetModel(void);
 void GiveVd(char *);
 char *GetVd(void);
};

class Transistor : public EMessage {      // Derived from class EMessage
protected:
 char far *Part;
 char far *Vbc;
 char far *Vbe;
 void init(char *InitPart, char *InitVbc, char *InitVbe)
 {
   Part = InitPart;
   Vbc = InitVbc;
   Vbe = InitVbe;
 }
public:
 Transistor(char *Initmsg, int Font, char *InitPart, char *InitVbc,
       char *InitVbe, int InitX, int InitY);
```

```
 Transistor() { init("BC549","0","0"); }
 void Show();
 void Hide();
 void GiveModel(char *);
 char *GetModel(void);
 void GiveVbc(char *);
 char *GetVbc(void);
 void GiveVbe(char *);
 char *GetVbe(void);
};

class Transformer : public EMessage {   // Derived from class EMessage
protected:
  char far *Coils;
  char far *N1;
  char far *Rs1;
  char far *N2;
  char far *Rs2;
  char far *N3;
  char far *Rs3;
  char far *N4;
  char far *Rs4;
  void init(char *InitCoils, char *InitN1, char *InitRs1,
        char *InitN2, char *InitRs2, char *InitN3, char *InitRs3,
        char *InitN4, char *InitRs4)
  {
    Coils = InitCoils;
    N1 = InitN1;
    Rs1 = InitRs1;
    N2 = InitN2;
```

```
    Rs2 = InitRs2;
    N3 = InitN3;
    Rs3 = InitRs3;
    N4 = InitN4;
    Rs4 = InitRs4;
  }
public:
  Transformer(char *Initmsg, int Font, char far *InitCoils,
        char far *InitN1, char far *InitRs1,
        char far *InitN2, char far *InitRs2,
        char far *InitN3, char far *InitRs3,
        char far *InitN4, char far *InitRs4,
        int InitX, int InitY);
  Transformer() { init("","","","","","","","","","","",""); }
  void Show();
  void Hide();
  void GiveCoils(char *);
  char *GetCoils(void);
  void GiveN1(char *);
  char *GetN1(void);
  void GiveRs1(char *);
  char *GetRs1(void);
  void GiveN2(char *);
  char *GetN2(void);
  void GiveRs2(char *);
  char *GetRs2(void);
  void GiveN3(char *);
  char *GetN3(void);
  void GiveRs3(char *);
  char *GetRs3(void);
```

```
  void GiveN4(char *);
  char *GetN4(void);
  void GiveRs4(char *);
  char *Getks4(void);
};

class Op_amp : public EMessage {        // Derived from class EMessage
protected:
  char far *Part;
  char far *Vi0;
  char far *Vo0;
  void init(char *InitPart, char *InitVi0, char *InitVo0)
  {
    Part = InitPart;
    Vi0 = InitVi0;
    Vo0 = InitVo0;
  }
public:
  Op_amp(char *Initmsg, int Font, char *InitPart, char *InitVi0,
        char *InitVo0, int InitX, int InitY);
  Op_amp() { init("","0","0"); }
  void Show();
  void Hide();
  void GiveModel(char *);
  char *GetModel(void);
  void GiveVi0(char *);
  char *GetVi0(void);
  void GiveVo0(char *);
  char *GetVo0(void);
};
```

```
class Switch : public EMessage {          // Derived from class EMessage
protected:
  char far *Type;
  char far *Rs;
  char far *Vth;
  char far *Vcoil;
  void init(char *InitType, char *InitRs, char *InitVth, char *InitVcoil)
  {
    Type = InitType;
    Rs = InitRs;
    Vth = InitVth;
    Vcoil = InitVcoil;
  }
public:
  Switch(char *Initmsg, int Font, char *InitType, char *InitRs,
       char *InitVth, char *InitVcoil, int InitX, int InitY);
  Switch() { init("","","","",""); }
  void Show();
  void Hide();
  void GiveType(char *);
  char *GetType(void);
  void GiveRs(char *);
  char *GetRs(void);
  void GiveVth(char *);
  char *GetVth(void);
  void GiveVcoil(char *);
  char *GetVcoil(void);
};
```

```
class Model : public EMessage {          // Derived from class EMessage
protected:
  char far *Device;
  char far *QType_Gain;
  char far *EType;
  char far *BetaF_Ri_Ron;
  char far *BetaR_Ro_Roff;
  char far *Cbc_Vsupply_Vz;
  char far *Cbe_Slew_Rz;
  char far *Is_Pole_Rpi;
  char far *Va_Vcutin;
  char far *Rce;
  char far *Pole1;
  char far *Pole2;
  char far *Pole3;
  void init(char *InitDevice, char *InitQType_Gain, char *InitEType,
            char far *InitBetaF_Ri_Ron, char far *InitBetaR_Ro_Roff,
            char far *InitCbc_Vsupply_Vz, char far *InitCbe_Slew_Rz,
            char far *InitIs_Pole_Rpi, char far *InitVa_Vcutin,
            char far *InitRce, char far *InitPole1, char far *InitPole2,
            char far *InitPole3)
  {
    Device = InitDevice;
    QType_Gain = InitQType_Gain;
    EType = InitEType;
    BetaF_Ri_Ron = InitBetaF_Ri_Ron;
    BetaR_Ro_Roff = InitBetaR_Ro_Roff;
    Cbc_Vsupply_Vz = InitCbc_Vsupply_Vz;
    Cbe_Slew_Rz = InitCbe_Slew_Rz;
    Is_Pole_Rpi = InitIs_Pole_Rpi;
```

```
    Va_Vcutin = InitVa_Vcutin;
    Rce = InitRce;
    Pole1 = InitPole1;
    Pole2 = InitPole2;
    Pole3 = InitPole3;
  }
public:
  Model() { init("","","","","","","","","","","","",""); }
  void GiveDevice(char *);
  char *GetDevice(void);
  void GiveQType(char *);
  char *GetQType(void);
  void GiveGain(char *);
  char *GetGain(void);
  void GiveVcutin(char *);
  char *GetVcutin(void);
  void GiveEType(char *);
  char *GetEType(void);
  void GiveBetaF(char *);
  char *GetBetaF(void);
  void GiveRi(char *);
  char *GetRi(void);
  void GiveRpi(char *);
  char *GetRpi(void);
  void GiveRon(char *);
  char *GetRon(void);
  void GiveBetaR(char *);
  char *GetBetaR(void);
  void GiveRo(char *);
  char *GetRo(void);
```

```
void GiveRce(char *);
char *GetRce(void);
void GiveRoff(char *);
char *GetRoff(void);
void GiveCbc(char *);
char *GetCbc(void);
void GiveVsupply(char *);
char *GetVsupply(void);
void GiveVz(char *);
char *GetVz(void);
void GiveCbe(char *);
char *GetCbe(void);
void GiveSlew(char *);
char *GetSlew(void);
void GiveRz(char *);
char *GetRz(void);
void GiveIs(char *);
char *GetIs(void);
void GivePole(char *);
char *GetPole(void);
void GiveVa(char *);
char *GetVa(void);
void GivePole1(char *);
char *GetPole1(void);
void GivePole2(char *);
char *GetPole2(void);
void GivePole3(char *);
char *GetPole3(void);
};
```

```
class Sweep : public EMessage {          // Derived from class EMessage
protected:
  char far *Var;
  char far *Fstart_Tstop;
  char far *Fstop_Tstep;
  char far *Fpts_StepCtrl;
  char far *Fscale_MaxTRiter;
  char far *LTEv;
  char far *LTEi;
  void init(char far *InitVar, char far *InitFstart_Tstop, char far *InitFstop_Tstep,
       char far *InitFpts_StepCtrl, char far *InitFscale_MaxTRiter, char far *InitLTEv,
       char far *InitLTEi)
  {
  Var = InitVar;
  Fstart_Tstop = InitFstart_Tstop;
  Fstop_Tstep = InitFstop_Tstep;
  Fpts_StepCtrl = InitFpts_StepCtrl;
  Fscale_MaxTRiter = InitFscale_MaxTRiter;
  LTEv = InitLTEv;
  LTEi = InitLTEi;
  }
public:
  Sweep() { init("","","","","","","","",""); }
  void GiveVar(char *);
  char *GetVar(void);
  void GiveFstart(char *);
  char *GetFstart(void);
  void GiveTstop(char *);
  char *GetTstop(void);
  void GiveFstop(char *);
```

```
char *GetFstop(void);
void GiveTstep(char *);
char *GetTstep(void);
void GiveFpts(char *);
char *GetFpts(void);
void GiveStepCtrl(char *);
char *GetStepCtrl(void);
void GiveFscale(char *);
char *GetFscale(void);
void GiveMaxTRiter(char *);
char *GetMaxTRiter(void);
void GiveLTEv(char *);
char *GetLTEv(void);
void GiveLTEi(char *);
char *GetLTEi(void);
};
```

จากหลักการโปรแกรมเชิงวัตถุ จะเห็นว่าคลาสติดต่อกับภายนอก โดยผ่านส่วน Interface ซึ่งไม่สนใจว่าในคลาสนั้นมีการจัดเก็บข้อมูลอย่างไร ตัวอย่างเช่น ในการนิยามคลาสของคลาส Sweep ที่ผ่านมาจะเห็นว่าข้อมูล Fstart และ Tstop จริงแล้ว เก็บไว้ในที่เดียวกันแต่เวลาผู้ใช้สั่ง GetFstart ก็จะได้ค่า Fstart ผู้ใช้สั่ง GiveFstart ก็จะเป็นการให้ค่าข้อมูลกับ Fstart ผู้ใช้สั่ง GetTstop หรือ GiveTstop ก็จะเป็นการอ่านค่า หรือเก็บค่า Tstop ตามลำดับ โดยผู้ใช้ไม่จำเป็นต้องรู้ว่าภายในคลาสเก็บค่าเหล่านี้อย่างไร