การพัฒนาแอพพลิเคชันบนเว็บโดยใช้ Java และ ActiveX

WEB APPLICATION DEVELOPMENT (JAVA & ACTIVEX)

โดย

นางสาวจารุวรรณ ณ ระนอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

การพัฒนาแอพพลิเคชันบนเว็บโดยใช้ Java และ ActiveX

WEB Application Development (Java & ActiveX)

โดย

นางสาวจารุวรรณ ณ ระนอง 37014053

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาแอพพลิเคชันบนเว็บโดยใช้ Java และ ActiveX

ผู้จัดทำ

นางสาวจารุวรรณ ณ ระนอง

..... อาจารย์ที่ปรึกษา

(.....)

การพัฒนาแอพพลิเคชันบนเว็บโดยใช้ Java และ ActiveX

นางสาวจารุวรรณ ณ ระนอง

ดร.วรวัฒน์ ลิ้มโภคา

ปีการศึกษา 2540

บทคัดย่อ

โครงงานนี้เป็นการนำเสนอวิธีการสร้างแอพพลิเคชันให้ผู้ใช้สามารถเรียกใช้งานได้โดยผ่านเว็บบราวเซอร์ของบริการเวิลด์ไวด์เว็บ ซึ่งเป็นบริการของระบบเครือข่ายอินเตอร์เน็ตที่เป็นที่นิยมใช้งานกันอย่างกว้างขวางมากที่สุดในปัจจุบัน แอพพลิเคชันถูกสร้างขึ้นโดยใช้ภาษาจาวา ซึ่งสามารถถูกประมวลผลได้ที่ฝั่งเว็บเซิร์ฟเวอร์และที่บราวเซอร์บนฝั่งไคล์เอ็นท์หรือประมวลผลที่บราวเซอร์บนฝั่งไคล์เอ็นท์เพียงที่เดียว

แอพพลิเคชันที่สร้างขึ้นสามารถอนุญาติให้ผู้ใช้งานแต่ละคนทำงานร่วมกับผู้ใช้งานคนอื่นบนเว็บบราวเซอร์ โดยที่แอพพลิเคชันจะทำการเชื่อมต่อแต่ละผู้ใช้งานผ่านเว็บเซิร์ฟเวอร์และแสดงผลข้อมูลของแต่ละผู้ใช้งานที่เว็บบราวเซอร์ของผู้ใช้งานเอง ผู้ใช้สามารถเรียกใช้งานแอพพลิเคชัน และเข้าทำงานร่วมกันกับผู้ใช้งานคนอื่นได้โดยไม่จำกัดชนิดของเครื่องคอมพิวเตอร์ที่ผู้ใช้งานใช้งานอยู่ ไม่จำกัดชนิดของระบบปฏิบัติการ เพียงแต่เครื่องคอมพิวเตอร์ที่ผู้ใช้งานใช้อยู่ได้เชื่อมต่อกับระบบเครือข่ายอินเตอร์เน็ตก็เพียงพอแล้ว

Web Application Development (Java & ActiveX)

Jaruwan Naranong

Dr.Voravat Limpoka

1997

Abstract

This project presents how to create an application that allow to call and work through the Web browser of the WWW service in the INTERNET. The application can be created with JAVA that will be executed either at Web browser in the Web Server or at Web browser in the client site or only at Web browser in the client site.

The application can allow user to work with other user on Internet through Web browser. Application can connect each user through Web Server and display their data on Web browser at their site. User can call the application for running and working with other user without the limited of the computer platform, or the operating system. The computer should be connect to the INTERNET

สารบัญ

หน้า

สารบัญตาราง

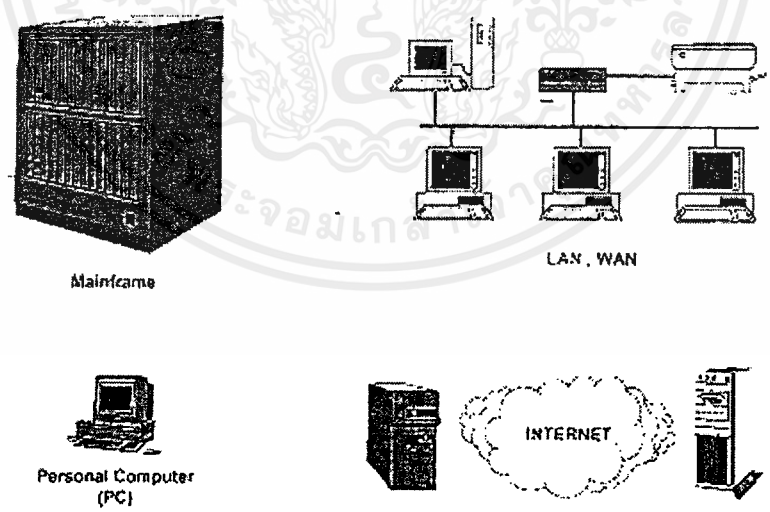
สารบัญภาพ

รูปภาพที่ หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เมื่อมีการนำคอมพิวเตอร์หลายๆเครื่องเข้ามาเชื่อมต่อกันเพื่อเป็นการแบ่งปันทรัพยากรที่มีอยู่ให้สามารถใช้งานร่วมกัน เกิดความคุ้มค่ามากขึ้น กลายเป็นระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น (LAN : Local Area Network) และระบบเครือข่ายคอมพิวเตอร์ขนาดใหญ่ (WAN : Wide Area Network) จนกระทั่งในปัจจุบันนี้ ระบบเครือข่ายคอมพิวเตอร์ได้ถูกพัฒนาขึ้นให้สามารถติดต่อสื่อสารกันได้ทั่วโลก กลายเป็นระบบเครือข่ายอินเตอร์เน็ต (INTERNET) ซึ่ง ได้เป็นที่นิยมใช้กันอย่างกว้างขวาง มีการเชื่อมต่อเครื่องคอมพิวเตอร์หลากหลายยี่ห้อที่อยู่ในสถานที่ต่าง ๆ ซึ่งห่างไกลกันได้ทั่วโลก ทำให้สามารถใช้งานและแบ่งข้อมูลต่างๆ เช่น ฐานข้อมูลซึ่งช่วยให้เกิดการพัฒนาในด้านต่างๆอย่างรวดเร็ว เช่นการค้นคว้าข้อมูลในการทำวิจัยของนักวิชาการ นักวิทยาศาสตร์ การค้นคว้าข้อมูลสำหรับการทำธุรกิจ การค้นคว้าข้อมูลของนักศึกษาในการเรียนการศึกษา หรือแม้กระทั่งการค้นหาข้อมูลต่างๆ เพื่อความบันเทิงของผู้ใช้งานทั่วไป ดังนั้นแนวโน้มของการใช้งานระบบเครือข่ายอินเตอร์เน็ต จึงมีมากมายและเพิ่มขึ้นอย่างต่อเนื่อง



รูปที่ 1 – 1 ระบบคอมพิวเตอร์ในยุคต่างๆ

ประโยชน์ของที่ได้รับในการสื่อสารโดยใช้ระบบเครือข่ายอินเตอร์เน็ต

- เป็น แหล่งที่ทำให้เกิดการพัฒนาเทคโนโลยีใหม่ ๆ
- เกิดการติดต่อสื่อสารระหว่างกัน และให้บริการข่าวสาร

- ทำให้เกิดการแลกเปลี่ยนความรู้ แนวความคิด และ เกิดการศึกษา
- เป็นการติดต่อสื่อสารข้อมูลที่เร็วมาก
- ใช้สื่อสารทางด้านธุรกิจผ่านเครือข่ายคอมพิวเตอร์

การใช้งานระบบเครือข่ายอินเตอร์เน็ตที่เป็นที่นิยมมากในปัจจุบันนี้ได้แก่การใช้งานบริการ World Wide Web (WWW) ในช่วงแรกของการเริ่มมีการบริการเวิลด์ไวด์เว็บนั้น โปรแกรมเว็บบราวเซอร์สามารถอ่านข้อมูลจากเว็บเซิร์ฟเวอร์และสามารถแสดงเฉพาะตัวอักษรได้เพียงอย่างเดียว ซึ่งในยุคนั้นบริการนี้ก็ไม่ได้รับความนิยมมากเท่าไรนัก แต่ต่อมามีการพัฒนาให้เว็บบราวเซอร์สามารถแสดงข้อมูลในโหมดกราฟฟิกได้ ซึ่งหมายความว่าในหน้าเว็บเพจ สามารถที่จะใส่ทั้งข้อความ รูปภาพ และไฟล์แบบต่างๆ เช่น เสียงหรือวีดีโอลงไปได้ด้วย จุดนี้ทำให้บริการเวิลด์ไวด์เว็บเป็นที่นิยมอย่างท่วมท้น และได้แซงบริการตัวอื่นๆ ในอินเตอร์เน็ตไปอย่างรวดเร็ว จนบางครั้งหลายคนเข้าใจว่า " เวิลด์ไวด์เว็บคืออินเตอร์เน็ต" ไปเลยทีเดียว

การบริการเวิลด์ไวด์เว็บสามารถช่วยให้ผู้ใช้งานสามารถค้นหาข้อมูลต่างๆได้ง่าย และรวดเร็ว ซึ่งข้อมูลที่ได้จากบริการ WWW นี้มีทั้งข้อมูลที่เป็นข้อมูลตัวอักษร รูปภาพ ภาพเคลื่อนไหว หรือแม้กระทั่งข้อมูลเสียงก็มี นอกจากนี้รูปแบบการใช้งานมีการติดต่อกับผู้ใช้แบบกราฟฟิก (GUI : Graphic User Interface) ซึ่งทำให้ใช้งานได้ง่ายขึ้น และดูน่าสนใจยิ่งขึ้น

การใช้บริการเวิลด์ไวด์เว็บนี้นอกจากจะเรียกข้อมูลที่ต้องการขึ้นมาดูได้แล้ว ยังสามารถแพร่ข้อมูล และสามารถทำงานกับฐานข้อมูลที่มีอยู่ที่ฝั่งเซิร์ฟเวอร์ (Server) ได้ด้วย โดยสามารถเพิ่มเติม แก้ไขข้อมูลในฐานข้อมูลก็ได้ หรือจะสั่งให้แอพพลิเคชั่น (Application) ที่อยู่ที่ฝั่งเซิร์ฟเวอร์ทำการประมวลผลหรือทำงานอื่นใดก็ได้ตามที่ต้องการ เช่น ระบบการเล่นเกมส์แบบ Multiplayer ผ่านทางอินเตอร์เน็ต ซึ่งต้องมีเซิร์ฟเวอร์ที่คอยประมวลผลและส่งข้อมูลถึงกันระหว่างผู้เล่น เป็นต้น

ซึ่งจากข้างต้นจะเห็นว่ารูปแบบของการนำเสนอข้อมูลแบบเสียง ภาพเคลื่อนไหวบนเว็บ การเชื่อมต่อกับระบบเครือข่าย จะสามารถดึงดูดให้ผู้ใช้งานบริการเวิล์ดไวด์เว็บสนใจในเว็บเพจนั้นๆ ซึ่งถ้าหากจะกล่าวถึงภาษาโปรแกรมมิ่งที่จะสามารถทำงานได้บนเว็บบราวเซอร์ และกำลังเป็นที่สนใจของผู้เขียนโปรแกรมนั้น ก็คือภาษาจาวา (JAVA) ซึ่งจาวานี้มีความสามารถมากกว่าภาษาที่ใช้งานบนเว็บได้เหมือนกัน เช่น HTML ซึ่งแสดงข้อมูลได้เพียงแต่ข้อความและรูปภาพเท่านั้น ต่างจากภาษาจาวาที่มีความสามารถในตอบสนองความต้องการด้านมีเดียต่างๆ ไม่ว่าจะเป็นภาพเคลื่อนไหว วีดีโอ เสียง และการติดต่อ สื่อสารโต้ตอบกันระหว่างเครือข่ายได้ดีกว่า

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1ศึกษาทฤษฎีของภาษาจาวา ซึ่งเป็นภาษาที่สามารถทำงานบนหรือเรียกใช้บริการทางเวิลด์ไวด์เว็บได้

1.2.2ศึกษาการสร้างแอพพลิเคชันที่สามารถเรียกใช้งานผ่านทางบริการเวิลด์ไวด์เว็บได้ และมีรูปแบบของการนำเสนอที่น่าสนใจ

1.2.3ศึกษาการสร้างแอพพลิเคชันที่สามารถปฏิบัติงานร่วมกันได้หลายผู้ใช้งานผ่านทางบริการเวิลด์ไวด์เว็บได้

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้เป็นสร้างแอพพลิเคชั่นที่สามารถเรียกใช้งานผ่านทางบริการเวิลด์ไวด์เว็บได้ โดยการนำเอาภาษาที่สามารถทำงานได้บนบราวเซอร์ (Browser) คือ ภาษา JAVA โดยมีการสร้างระบบการติดต่อกับผู้ใช้ (User Interface) เพื่อให้ผู้ใช้งานสามารถใช้งานระบบงานนี้ได้โดยง่าย และผู้ใช้งานสามารถทำงานติดต่อกันได้บนระบบเครือข่าย และแอพพลิเคชันที่สร้างขึ้นมานี้จะสามารถทำงานได้โดยอิสระ ไม่ขึ้นกับระบบปฏิบัติการที่ผู้ใช้งานใช้อยู่ เช่น Windows , Unix , Macintosh , OS/2 เป็นต้น

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎี พื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย คือทฤษฎีของภาษา Java ทั้งคำสั่งพื้นฐาน ด้านกราฟฟิก และการติดต่อกับเครือข่าย ซึ่งมีรายละเอียดดังในบทที่ 2 , 3 , 4 จากนั้นก็จะนำเอาความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบมาออกแบบแอพพลิเคชัน ในด้านการติดต่อกับผู้ใช้งาน การติดต่อกับระบบเครือข่าย และพัฒนาแอพพลิเคชัน ซึ่งมีรายละเอียดในบทที่ 5 และ บทที่ 6

สำหรับบทที่ 7 นั้นก็จะเป็นการทดสอบแอพพลิเคชันทั้งหมด และสำหรับบทที่ 8 ซึ่งเป็นบทสุดท้ายก็จะเป็นการสรุปการทำงาน ผลที่ได้รับจากงานวิจัยชิ้นนี้ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
ทฤษฎีเบื้องต้นของภาษาจาวา

2.1 บทนำ

ภาษา Java หรือ Java Programming Language คือ ภาษาคอมพิวเตอร์แบบ Object-Oriented ที่มีความคล้ายกับภาษา C/C++ มากแต่ก็ไม่เหมือนกันทีเดียว โดยตัดข้อเสียบางอย่างของ C/C++ ออกไปและเพิ่มข้อดีหลายๆอย่างเข้าไป ทั้งนี้เป็นเพราะ Java ถูกพัฒนาขึ้นเพื่อใช้งานกับระบบเครือข่าย Internet ซึ่งต้องปรับปรุงและพัฒนาความสามารถหลายๆอย่างให้เหมาะสมกับการใช้งานบนเครือข่ายคอมพิวเตอร์ แต่ยังคงความง่ายในการเขียนโปรแกรมให้เหมือนกับภาษา C/C++ Java ถูกคิดและพัฒนาขึ้นโดยนาย James Gosling และทีมงาน Green group ของบริษัท Sun Microsystems ในปี 1991 มีชื่อเดิมของภาษาว่า Oak ถูกประกาศตัวครั้งแรกในปี 1995 ในงาน SunWorld'95 ในชื่อว่า **Java** และบริษัทซันยังได้เปิดตัว Web-browser ตัวใหม่ในชื่อว่า **Hotjava** อีกด้วย โปรแกรมที่เขียนด้วยภาษา Java นี้จะมีสีสันขึ้นจากการใช้งานเว็บแบบเดิม คือจะมีคุณสมบัติของเสียงและภาพเคลื่อนไหวประกอบเพิ่มขึ้น สามารถเล่นเกม พูดคุย รับข้อมูลที่ทันสมัยเสมอได้

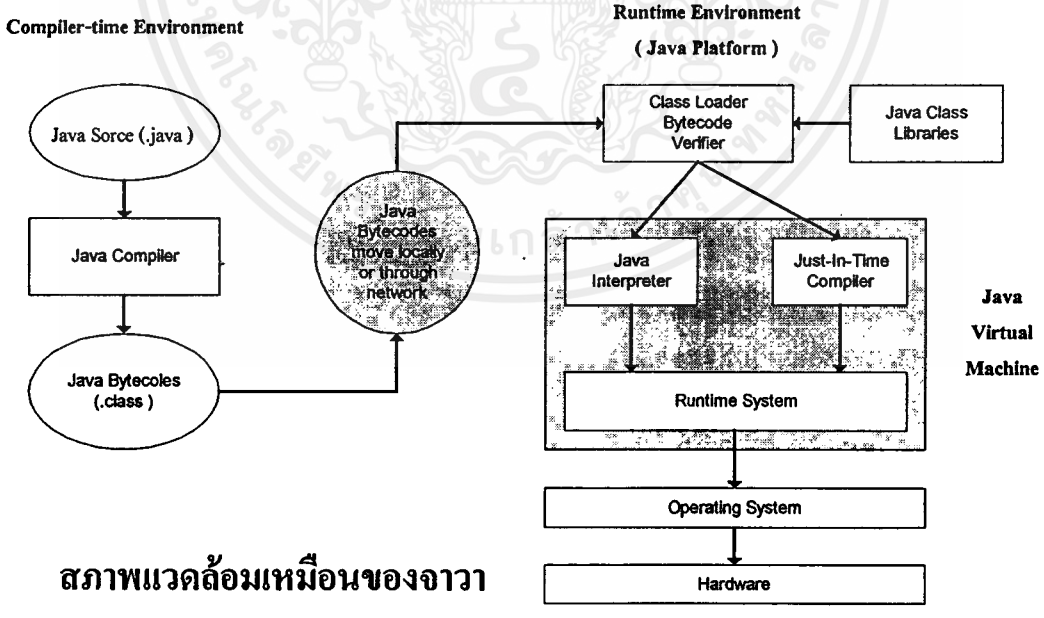
2.2 ความสัมพันธ์ระหว่าง Java กับ Web

Java ในตอนแรกถูกพัฒนาขึ้นมาเพื่อใช้กับระบบปฏิบัติการ set-top box ซึ่งเป็นโครงการของระบบตลาดอิเล็กทรอนิกส์ มีการเชื่อมต่อและแลกเปลี่ยนข้อมูลข่าวสารซึ่งกันและกันทางธุรกิจผ่าน เครือข่ายคอมพิวเตอร์ แต่โครงการนี้ต้องถูกล้มเลิกไปเพราะบริษัทแพ้การประมูล ในขณะเดียวกัน WWW (World Wide Web หรือเรียกสั้นๆว่า Web) ซึ่งกำลังได้รับความนิยมอย่างสูงและมีการทำงานแบบ Client/Server มีการแลกเปลี่ยนข้อมูลกันบนเครือข่ายคอมพิวเตอร์เหมือนกัน ดังนั้นจึงได้เกิดแนวความคิดในการนำเอา Java มาเป็นส่วนหนึ่งของ Web เพื่อเพิ่มประสิทธิภาพในการทำงาน เมื่อ Client หรือผู้ขอใช้บริการซึ่งอยู่ในที่ต่างๆของระบบเครือข่าย Internet สามารถทำงานโต้ตอบ(Interactive) กับ Homepage นั้นๆได้ สามารถ Download โปรแกรมซึ่งเป็นโปรแกรม Java จาก Server หรือผู้ให้บริการมาปฏิบัติการหรือรัน (Run) บนเครื่อง Client ได้ ซึ่ง Java ได้ถูกออกแบบให้มีความปลอดภัยในการใช้งานกับระบบเครือข่าย Internet โดยเฉพาะ

2.3 Java Virtual Machine

Java Virtual Machine เป็นหลักการสร้างคอมพิวเตอร์จำลองของจาวา โดยการสมมติให้มีคอมพิวเตอร์อีกเครื่องหนึ่งขึ้นมาโดยเครื่องนี้จะใช้ในการคอมไพล์โปรแกรมภาษาจาวาทุกโปรแกรม เมื่อต้องการให้โปรแกรมภาษาจาวาไปทำงานบนคอมพิวเตอร์จริงๆ เครื่องใด เราก็เพียงแต่สร้างตัวอินเตอร์พรีเตอร์ (Interpreter) ของคอมพิวเตอร์จำลองตัวนี้บนเครื่องนั้น ๆ ภาษาจาวาทุกโปรแกรมก็จะสามารถทำงานบนระบบคอมพิวเตอร์นั้น ๆ ได้ตามต้องการ ดังแสดงในรูปที่ 2 - 1 โดย Java Virtual Machine นี้จะคอยให้บริการอยู่ระหว่างไบท์โค้ดกับระบบปฏิบัติการของระบบคอมพิวเตอร์ สภาพแวดล้อมของจาวาเช่นนี้จะเห็นได้บนระบบอินเตอร์เน็ตโดยการรัน จาวาแอพเพล็ต (Java Applet) ผ่านเว็บบราวเซอร์ซึ่งทำหน้าที่เป็นสภาพแวดล้อมของจาวา จะเห็นได้ว่าคอมพิวเตอร์และระบบปฏิบัติการเครื่องใดสามารถรันเว็บบราวเซอร์อย่างเช่น Netscape Navigator หรือ Microsoft Internet Explorer ฉะนั้นสภาพแวดล้อมสำหรับจาวาจึงมีเพียงหนึ่งเดียวเท่านั้นและเป็นมาตรฐานกลางสำหรับการเขียนโปรแกรมคอมพิวเตอร์

Java Virtual Machine จะอนุญาตให้เฉพาะจาวาแอพเพล็ทหรือจาวาแอพพลิเคชั่นเท่านั้นที่สามารถรันได้โดยที่ไบท์โค้ดจะปราศจากไวรัสหรือส่วนที่จะทำอันตรายต่อระบบ อันเนื่องจาก Java Virtual Machine จะจำกัดสิทธิ์การเข้าใช้ทรัพยากรของไบท์โค้ด ซึ่งตรงนี้ที่เป็นที่มาของคุณสมบัติของจาวาที่ไม่ขึ้นกับแพลทฟอร์มและฮาร์ดแวร์ใด ๆ หรือพอร์ตเทเบิล (Portable) หรือ Java มีความปลอดภัยสำหรับเหตุผลที่ต้องมี Java VM นี้ จริง ๆ แล้วเป็นการกำหนดคำขึ้นมาเพื่อเป็นคำจัดกัดความเฉพาะ



รูปที่ 2 - 1 สภาพแวดล้อมเหมือนของจาวา

ในความคิดเท่านั้น เพื่อให้นักพัฒนาจะได้ไม่ถูกบังคับให้ต้องสร้างตัวอินเตอร์พรีเตอร์ตามแนวทางแบบใดแบบหนึ่งโดยเฉพาะ แต่ตัวอินเตอร์พรีเตอร์ที่สร้างขึ้นตามข้อกำหนดดังกล่าว ไม่ว่าจะอยู่บนแพลทฟอร์มใด จะสามารถรันโปรแกรมที่เขียนขึ้นในภาษาจาวาได้ โดยให้ผลลัพธ์ออกมาเหมือนกัน

จากที่กล่าวมาข้างต้นถึงคุณสมบัติของจาวาที่ไม่ยึดติดอยู่กับแพลทฟอร์มใด ๆ ทำให้การทำงานกับระบบคอมพิวเตอร์ที่ทำงานแบบกระจาย (Distributed computing) ได้รับการตอบสนองอย่างเหมาะสม นั่นคือทำให้เกิดความพอร์ตเทเบิลนั่นเอง

อย่างไรก็ตาม Java virtual Machine ก็ยังมีข้อจำกัดอยู่ ข้อจำกัดของ Java virtual Machine นั้นก็อยู่ที่ข้อจำกัดของการออกแบบตัวอินเตอร์พรีเตอร์แทน เช่นการจำกัด ค่าของโอเปอร์แรนด์ และขนาดของสแต็ก เป็นต้น ข้อกำหนดเหล่านี้หมายถึงว่า Java Virtual Machine สามารถจะอ้างถึงหน่วยความจำได้เฉพาะในห้องแอดเดรสเท่าที่มีอยู่เท่านั้น

ข้อจำกัดภายในของตัว Java Virtual Machine เองนั้นมีห้วงแอดเดรสให้ใช้อยู่ถึง 4 GB เพราะขนาดของความกว้างของการอ้างแอดเดรสเป็น 32 บิต method ต่าง ๆ ของจาวามีขนาดได้เพียง 32 KB เพราะมีข้อจำกัดของการใช้คำสั่งกระโดด เป็นแบบ 16 บิต (โดยมีบิตแรกเป็นตัวบอกว่ากระโดดไปข้างหน้า หรือข้างหลัง และบิตต่อๆไป บอกระยะทางการกระโดดจากจุดที่ทำงานอยู่) จำนวนตัวแปรแต่ละชุดสแต็กจะถูกจำกัดอยู่ที่ 256 ตัว เพราะดัชนีที่ใช้ชี้ตัวแปรเหล่านี้มีขนาดเพียง 8 บิต นอกจากนั้น จำนวนของค่าคงที่ที่อยู่ในส่วนกลาง ยังถูกจำกัดด้วยขนาดของดัชนี 16 บิต ทำให้มีจำนวนได้เพียง 32,000 ค่าในแต่ละ method

การที่เราถือว่า การมีค่าเหล่านี้ เป็นข้อจำกัด ก็อาจจะเป็นการมองการณ์ไกลไปสักหน่อย เนื่องจากปัจจุบันเครื่องคอมพิวเตอร์ส่วนใหญ่มีหน่วยความจำเพียง 16 หรือ 32 MB กันเท่านั้น หน่วยความจำขนาด 4 GB จึงเป็นเรื่องที่ไม่ต้องคิดหนักในตอนนี้ ในขณะที่ขนาดของ method ที่จำกัดที่ 32 KB ก็เป็นเพียง method เดียวเท่านั้น

2.4 หลักการทำงานของโปรแกรมภาษาจาวา

การทำงานของภาษาจาวาเริ่มตั้งแต่การคอมไพล์ตัวโปรแกรมจนกระทั่งนำไปเรียกใช้งานในเว็บเพจ เป็นดังนี้

2.4.1 การคอมไพล์

คอมไพเลอร์ของภาษาจาวาก็เช่นเดียวกับคอมไพเลอร์ในภาษาอื่น ๆ นั่นคือ มันจะสร้างรหัสภาษาเครื่อง (Machine code หรือ assembler code) จากภาษาในระดับที่สูงกว่าเพื่อให้ซีพียูสามารถนำไปใช้งานได้ แต่ข้อแตกต่างที่สำคัญระหว่างคอมไพเลอร์ของภาษาจาวากับภาษาอื่น ๆ คือ ซีพียูหรือโพรเซสเซอร์ที่จะทำหน้าที่ในการปฏิบัติตามคำสั่งที่ได้จากการคอมไพล์ภาษาจาวานั้นไม่มีอยู่จริง เป็นเพียงสิ่งที่สมมติขึ้นมาที่เรียกว่า Java Virtual Machine นอกจากนี้การอ้างถึงส่วนต่าง ๆ ของโปรแกรมที่คอมไพล์ด้วยคอมไพเลอร์ของภาษาจาวาก็จะมีวิธีการที่แตกต่างออกไป โดยสิ่งที่ได้จากการคอมไพล์ในภาษาจาวาเราเรียกว่า ไบท์โค้ด (ByteCode)

คอมไพเลอร์ของภาษาจาวาจะไม่เปลี่ยนแปลงการอ้างถึงส่วนของโปรแกรมจากการใช้ชื่อแบบในภาษาสูง ไปเป็นตัวเลขเหมือนคอมไพเลอร์ภาษาอื่น ๆ ทำกัน และคอมไพเลอร์ภาษาจาวาจะไม่มีการสร้างแผนที่ของการจัดวางโปรแกรมบนหน่วยความจำขึ้นมาในระหว่างการคอมไพล์ด้วยเหตุผลที่สำคัญคือ เพื่อเป็นการสร้างความพอร์ตเทเบิลให้กับตัวโปรแกรม เพราะการจัดวางตำแหน่งของโปรแกรมจะต้องขึ้นอยู่กับลักษณะการทำงานของโปรเซสเซอร์ตัวใดตัวหนึ่ง การยังไม่จัดวางตำแหน่งช่วยทำให้โปรแกรมที่ได้จากการคอมไพล์มีความเป็นกลาง สามารถนำไปใช้บนคอมพิวเตอร์แพลทฟอร์มอื่น ๆ ได้ นอกจากนั้นยังทำให้เกิดความปลอดภัยอีกด้วย

2.4.2 การวางตำแหน่งในหน่วยความจำ

ในภาษาจาวาจะไม่มีการลดรูปแบบการอ้างถึงส่วนต่าง ๆ ของโปรแกรมจากการเรียกเป็นชื่อให้เหลือเพียงตัวเลขหรือแอดเดรสที่กำหนดขึ้นจากการจัดวางตำแหน่งของโปรแกรมลงในหน่วยความจำ คอมไพลเลอร์ภาษาจาวาจะทิ้งชื่อของแต่ละส่วนของโปรแกรม (โดยเฉพาะ method) เอาไว้ในตัวโปรแกรมที่สร้างขึ้น เมื่อโปรแกรมทำงานจะเป็นหน้าที่ของตัวอินเตอร์พรีเตอร์ที่จะคอยเปิดตารางค้นหาที่อยู่ของ method ที่ต้องการเรียกใช้งาน โดยก่อนที่จะเริ่มทำงานจริง อินเตอร์พรีเตอร์จะต้องสร้างแผนที่ในการจัดวางสิ่งต่าง ๆ ลงในหน่วยความจำขึ้นมาก่อนแล้วจึงสร้างตารางขึ้นมา เพื่อช่วยหาตำแหน่งของ method เมื่อมีการเรียกใช้งานโดยใช้ชื่อของ method

2.4.3 การรันโปรแกรม

การรันโค้ดที่คอมไพล์เอาไว้สำหรับ Java Virtual Machine เป็นหน้าที่ของตัวอินเตอร์พรีเตอร์ การรันโปรแกรมจะแบ่งได้เป็น 3 ขั้นตอนหลัก ๆ คือ การอ่าน การตรวจสอบความถูกต้อง และการทำงานตามโค้ด หน้าที่ในการอ่านโค้ดเข้าสู่ระบบจะเป็นของ Class Loader หน้าที่การทำงานในส่วนนี้จะไม่ได้อ่านเข้ามาเฉพาะไฟล์จาวาที่กำลังจะเรียกใช้เท่านั้น แต่จะอ่านคลาสที่มีการอ้างถึงและคลาสที่มีการ inherited มาโดยคลาสที่อ้างถึง เมื่อผ่านขั้นตอนนี้แล้ว โค้ดทั้งหมดก็จะถูกส่งผ่านตัวตรวจสอบไบท์โค้ดที่ส่งมามีความถูกต้องตามมาตรฐานของจาวา และจะไม่รบกวนเสถียรภาพของระบบ เมื่อผ่านการตรวจสอบ

แล้ว โค้ดก็จะถูกส่งต่อไปยังระบบรันไทม์ (Run-time System) ซึ่งจะสั่งงานไปยังฮาร์ดแวร์อีกต่อหนึ่ง ซึ่งหลักการทำงานในแต่ละขั้นตอนที่กล่างมาข้างต้นเป็นดังนี้

2.4.3.1 Class Loader

ตัวคลาสโหลดเดอร์ ทำหน้าที่ดึงโค้ดทั้งหมดที่จำเป็นในการทำงานของแอพพลิเคชัน ไม่ว่าจะเป็นคลาสที่ถูก inherited มา หรือคลาสอื่นๆ ที่มีการใช้เรียกใช้ เมื่อโหลดเดอร์ดึงคลาสใดเข้ามาแล้วก็จะจัดคลาสนั้น ๆ ใส่เข้าไว้ใน namespace ของมันเอง โดยการเก็บจะใช้ชื่อคลาสเป็นสำคัญ ไม่ได้ใช้การอ้างอิงเป็นตัวเลข หลักการนี้ก็จะเหมือนกันกับการทำงานของ Virtual Machine ที่โอเอส (OS : Operating System) สร้างขึ้นให้แอพพลิเคชันแต่ละตัวทำงาน ถ้าไม่ได้มีการเจาะจงเรียกใช้คลาสที่อยู่นอก namespace นี้ การเรียกใช้ชื่อต่างๆ ในคลาส ก็จะไม่มีการรบกวนกันระหว่างคลาสเลย

คลาสทั้งหมดที่อยู่บนเครื่องโลคอลเอง จะได้รับห้วงแอดเดรส (Address Space) เป็นของตัวเอง ส่วนคลาสต่างๆ ที่ดึงมาจากภายนอกจะได้รับ namespace เป็นของตัวเอง การทำงานลักษณะนี้จะช่วยให้คลาสที่อยู่บนโลคอลทำงานได้ประสิทธิภาพดีขึ้น เพราะใช้ namespace ร่วมกันได้ แต่ก็ยังมีการป้องกันความผิดพลาดที่อาจจะเกิดจากคลาสที่ดึงเข้ามาจากภายนอก และในทางกลับกัน คลาสที่อิมพอร์ตเข้ามาก็ปลอดภัยจากความผิดพลาดที่อาจจะเกิดขึ้นจากคลาสโลคอลด้วย

เมื่อคลาสทั้งหมดที่เกี่ยวข้องกับการทำงานถูกอิมพอร์ตเข้ามาเรียบร้อยแล้ว การจัดวางหน่วยความจำสำหรับเริ่มต้นการทำงานก็จะเกิดขึ้นได้ การเรียกชื่อต่างๆก็จะสามารถจับคู่กับแอดเดรสจริงๆ ของหน่วยความจำได้ แล้วตัวโหลดเดอร์จะสร้างตารางสำหรับค้นหาที่อยู่ของสิ่งต่างๆขึ้น (Look-up Table) การสร้างตารางขึ้น เป็นขั้นตอนสุดท้ายนี้ ทำให้ลดความเสี่ยงจากการทำงานผิดปกติของซุปเปอร์คลาส (Super Class) และการอ้างแอดเดรสที่ไม่ถูกต้องได้

2.4.3.2 การตรวจสอบไบท์โค้ด

เมื่อโค้ดเดินทางมาจนถึงขั้นตอนการสร้างตารางจับคู่ชื่อกับแอดเดรสแล้ว ก็ยังไม่สามารถแน่ใจได้ว่าโค้ดที่อ่านเข้ามาจะมีความปลอดภัย ดังนั้นจึงต้องมีตัว Verifier หรือตัวตรวจสอบไบท์โค้ด ทำหน้าที่ตรวจสอบความถูกต้องทีละบรรทัดว่าเป็นไปตามข้อกำหนดของจาวา และสอดคล้องกับการทำงานของตัวโปรแกรมเองหรือไม่ การตรวจสอบไบท์โค้ดในเชิงทฤษฎีจะสร้างค้นหาปัญหาต่างๆ ได้หลายอย่าง เช่น จะไม่มีการสร้างพอยต์เตอร์ที่เกินกว่าหน่วยความจำจริง ไม่มีคำสั่งใดสามารถละเมิดสิทธิ์การทำงานของตัวโปรแกรมได้ ไม่มีการจับคู่ออบเจ็คท์ผิด จะไม่มีการให้โอเปอร์แรนด์มากหรือน้อยเกินไป การกำหนดค่าต่างๆสำหรับไบท์โค้ดจะต้องถูกต้องครบถ้วน และจะไม่มีการแปลงข้อมูลผิดรูปแบบ

การใช้ตัวตรวจสอบตอบสนองจุดประสงค์ 2 ประการคือ สิ่งต่างๆดังที่กล่าวมาแล้วจะถูกตรวจสอบก่อนทำให้ตัวอินเตอร์พรีเตอร์มั่นใจได้ว่า ไบท์โค้ดที่ส่งเข้าไปทำงานจะไม่มีขั้นตอนการทำงานที่สร้างปัญหาให้กับตัวระบบ และจุดประสงค์ที่สองก็คือ ตัวอินเตอร์พรีเตอร์จะทำงานตามไบท์โค้ดได้รวดเร็วกว่า เพราะไม่ต้องคอยระวังว่าจะมีปัญหาเกิดขึ้น และไม่ต้องหยุดเป็นช่วงๆ เมื่อพบปัญหาและต้องแก้ไข

ในการทำงาน ไบท์โค้ดจะถูกตรวจสอบเพียงครั้งเดียวเท่านั้น และจะทำงานไปได้ตลอดไม่ต้องมีการตรวจสอบซ้ำอีกเมื่อมีการเรียกกลับมาทำงานที่ส่วนเดิมของโปรแกรม

2.4.3.3 การทำงานตามโค้ด

เมื่อตัวโหลดเดอร์ได้รวบรวมโค้ดเข้ามาสู่ระบบทำการจัดวางในหน่วยความจำ และตัวตรวจสอบได้ทำการตรวจสอบความถูกต้องแล้ว โค้ดก็จะถูกส่งต่อไปยังตัวอินเตอร์พรีเตอร์เพื่อทำงานตามคำสั่ง การทำงานตามคำสั่งของโค้ดก็คือการเปลี่ยนโค้ดให้กลายเป็นคำสั่ง การทำงานจริงที่ตัวระบบไคลเอ็นท์ที่รันโค้ดนี้สามารถทำงานได้ ซึ่งวิธีการที่ทำได้ก็มีอยู่ 2 วิธีด้วยกันคือ ตัวอินเตอร์พรีเตอร์ทำการคอมไพล์โค้ดเหล่านี้ให้กลายเป็นเนทีฟโค้ดที่ตัวเครื่องไคลเอ็นท์เข้าใจ แล้วค่อยทำงาน เพื่อให้ได้ความเร็วสูงสุดในการทำงาน กับอีกวิธีหนึ่งก็คือ ตัวอินเตอร์พรีเตอร์อ่านโค้ดเข้ามาแล้วตีความทำงานไปทีละคำสั่ง และทำการตีความไปเรื่อยๆ ตลอดเวลาที่มีการทำงาน

โดยปกติแล้ว ผู้สร้างตัวอินเตอร์พรีเตอร์มักจะเลือกใช้วิธีการหลัง รูปแบบของไบท์โค้ดในภาษาจาวามีความยืดหยุ่นเพียงพอที่จะสามารถเปลี่ยนไปทำงานบนเครื่องไคลเอ็นท์แบบต่างๆ ได้โดยไม่มีการก่อให้เกิดโอเวอร์เฮดมากมายเกินความจำเป็น อย่างไรก็ตามไคลเอ็นท์ของจาวาบางระบบจะมีความสามารถในการทำงานได้ทั้งสองวิธี คือ โปรแกรมเมอร์สามารถจะเลือกใช้วิธีการคอมไพล์กับงานที่เน้นการคำนวณมากๆ เพื่อเป็นการเพิ่มสมรรถนะในการทำงานให้ได้เต็มที่ ซึ่งไคลเอ็นท์แบบนี้จะให้ได้ทั้งความพอร์ทเทเบิล และสมรรถนะที่ดี

การสร้างระบบรันไทม์ที่ดีจะต้องถ่วงดุลย์ความสำคัญ 3 ประการให้ได้พอเหมาะ นั่นคือ ความพอร์ทเทเบิล ความปลอดภัย และสมรรถนะ เรื่องของความพอร์ทเทเบิลทำได้โดยการใช้รูปแบบของไบท์โค้ดที่มีความเป็นกลางเพียงพอ สามารถนำไปใช้รันบนเครื่องคอมพิวเตอร์แบบต่างๆได้โดยง่าย นอกจากนั้น การที่ตัวอินเตอร์พรีเตอร์ทำการกำหนดการจัดวางตำแหน่งในหน่วยความจำในช่วงรันไทม์ (แทนที่จะเป็นระหว่างการคอมไพล์เหมือนภาษาอื่นๆ ก็เป็นการเพิ่มความแน่นอนว่า คลาสต่างๆ ที่อิมพอร์ตเข้ามาจะยังคงใช้ได้อยู่ตลอดเวลา เรื่องความปลอดภัยนั้น เป็นสิ่งที่มีการคำนึงถึงอยู่ตลอดกระบวนการทำงานของระบบรันไทม์ โดยเฉพาะอย่างยิ่งในส่วนของตัวตรวจสอบไบท์โค้ด ที่ทำให้แน่ใจได้ว่าโปรแกรมจะทำงานได้ถูกต้องตามข้อกำหนดของจาวา ส่วนเรื่องของสมรรถนะนั้นก็สามารถจัดการได้ในสองระยะ คือ

พยายามเอาโอเวอร์เฮดทั้งหลายไปใส่ไว้ที่ตอนเริ่มต้น โหลดโปรแกรมเข้ามาสู่ระบบ หรือไม่ก็กำหนดให้ทำงานเป็นแบบแบ็กกราวนด์ (Back-Ground Thread)

ด้วยสิ่งต่างๆเหล่านี้ ทำให้จาวาสามารถปล่อยสมรรถนะระดับที่น่าพอใจออกมาได้ โดยที่ยังคงไว้ซึ่งความพอร์ทเทเบิล และสภาพแวดล้อมที่ปลอดภัย นอกจากนั้น ยังสามารถจะดึงสมรรถนะระดับสูงสุดออกมาใช้ได้ทันทีเมื่อต้องการ

2.4.4 ออบโค้ดและโอเปอแรนด์ในจาวา

ส่วนต่าง ๆ ของโปรแกรมเมื่อเราไปอ้างอิงในคอมพิวเตอร์แบบต่าง ๆ จะประกอบด้วยส่วนประกอบ 2 ส่วนด้วยกัน คือ

1. ออบโค้ด คือ คำสั่งที่จะให้คอมพิวเตอร์ทำงานอย่างใดอย่างหนึ่ง
2. โอเปอแรนด์ คือ ข้อมูลที่จำเป็นต้องใช้ในการทำงานตามออบโค้ด

ทั้งออบโค้ดและโอเปอแรนด์จะถูกจัดวางเรียงอยู่เป็นแถวเพื่อให้เครื่องคอมพิวเตอร์เรียกเข้าไปทำงาน
เรียงไปตามลำดับ ตามคำสั่งอาจจะมีการเรียกข้อมูลตัวเลขออกมาจากหน่วยความจำแล้วเก็บใส่ไว้ในสแต็ก คำสั่งอาจจะให้ดึงตัวเลขออกมาอีกตัวหนึ่งแล้วก็ใส่เข้าไปในสแต็กอีก จากนั้นบวกข้อมูลตัวเลขทั้งสองเข้าด้วยกันแล้วเก็บผลลัพธ์กลับเข้าสู่หน่วยความจำ คำสั่งตามรูปแบบชุดคำสั่งของ Java Virtual Machine ที่ทำงานตามนี้ ก็คือ

lload address

lload address

ladd

lstore address

ซึ่งจะสามารถคอมไพล์ออกมาเป็นภาษาเครื่องได้ดังนี้

22 xxxx

22 xxxx

97

55 xxxx

จะเห็นได้ว่าออบโค้ดจะเป็นตัวเลขขนาด 8 บิตธรรมดา เพื่อบอกให้เครื่องรู้ว่าจะให้ทำงานอะไร ส่วนตัวแปรแอดเดรสเป็นตัวเลขที่ใช้บอกเครื่องว่าให้ไปดึงค่าของตัวแปรมาจากที่ไหนในหน่วยความจำ ส่วนนี้ก็คือโอเปอร์แรนด์ ซึ่งขนาดของตัวแปรแอดเดรสแต่ละตัวจะเป็นเลข 32 บิต ดังนั้น จากออบโค้ด

และโอเปอแรนด์ข้างต้นรวมกันและจะใช้พื้นที่ในหน่วยความจำ 16 ไบท์หรือ 128 บิต ซึ่งถ้าโปรแกรมส่วนเล็ก ๆ นี้เป็นสมาชิกประเภท method ของคลาส มันก็จะถูกผนึกรวมเข้าไปในทุก ๆ method ที่อยู่ในคลาสเดียวกัน เมื่อตัวคอมไพเลอร์ทำการสร้างโค้ด

ส่วนวิธีการที่คอมไพเลอร์จะทราบตำแหน่งที่อยู่และกระโดดไปทำงานที่ส่วนของโปรแกรมที่ถูกต้องเมื่อมีส่วนอื่นๆ ของโปรแกรมเรียกใช้นั้น ก็คือคอมไพเลอร์จะเก็บขนาดความยาวของทุกๆ ส่วนของโค้ดเอาไว้แล้วจัดวางลงบนหน่วยความจำเรียงตามลำดับ หลังจากนั้นก็เพียงแต่บอกให้เครื่องกระโดดไปทำงานที่ตำแหน่งของแอดเดรสที่เป็นจุดเริ่มต้นของ method ที่ต้องการจะเรียกใช้งานเท่านั้น ในการเรียกใช้ method จะต้องใช้คำสั่ง jsr address ซึ่งแปลออกมาเป็นภาษาเครื่องได้ 168 xx โดยแอดเดรสในที่นี้เป็นเลข 16 บิต

2.5 Java Programming Advantages

ภาษาจาวาเป็นภาษาโปรแกรมแบบ OOP (Object Oriented Programming) แต่มีจุดเด่นอยู่ที่สามารถสร้างโปรแกรมขนาดเล็ก เพื่อใช้งานผ่านเน็ตเวิร์กร่วมกับบราวเซอร์ที่สามารถอินเตอร์พรีเตอร์ไบท์โค้ดที่ถูกสร้างขึ้นด้วยจาวาคอมไพเลอร์ได้ เทคนิคการออกแบบของจาวานั้น ไม่ขึ้นอยู่กับสถาปัตยกรรมของฮาร์ดแวร์ดังที่กล่าวมาแล้ว เราเรียกว่า Architecture Neutral ฉะนั้นโปรแกรมเมอร์ที่เขียนด้วยภาษาจาวาไม่จำเป็นจะต้องสนใจว่าโปรแกรมของตนจะถูกนำไปใช้บนเครื่องใด บราวเซอร์จะทำหน้าที่แปลไบท์โค้ดให้กับฮาร์ดแวร์เอง ซึ่งถือเป็นกุญแจสำคัญของการออกแบบด้านเทคนิคของจาวา และเป็นเหตุผลสำคัญที่ทำให้จาวาเหมาะกับเว็บ และระบบออนไลน์มากกว่า

เป้าหมายของการฝังภาษาจาวากับเว็บ ก็คือ เพื่อให้เนื้อหาของเว็บสามารถเอ็กซิคิวท์ได้ ซึ่งทำให้ผู้ใช้สามารถโต้ตอบ หรือ Interactive ได้ สิ่งที่จำเป็นก็เพียงแต่มีบราวเซอร์ที่สนับสนุนภาษาจาวาติดตั้งอยู่บนเครื่องเท่านั้นและเชื่อมต่อกับอินเตอร์เน็ตก็เพียงพอแล้ว

2.6 คุณสมบัติของภาษาจาวาที่โดดเด่น

Java เป็นภาษาคอมพิวเตอร์ชั้นสูงที่มีความสามารถหลายประการดังนี้

1. ความง่าย (Simple)
ด้วยโครงสร้างของภาษาที่คล้ายคลึงกับภาษาที่เป็นที่นิยม และคุ้นตาอย่างภาษา C/C++

2. คุณสมบัติแบบ OOP (Object-Oriented Programming)

เป็นภาษาแบบ OOP ทำให้สามารถนำเอา code เก่ามาใช้ได้อีก และมีการจัดการกับข้อมูลอย่างมีประสิทธิภาพ และปลอดภัย Java ใช้ Methods จัดการกับข้อมูลแทน Functions ในภาษาองค์ประกอบของข้อมูล และโอเปอเรชัน (Operation) ที่เรียกว่า methods ที่จะเรียกใช้ข้อมูลเหล่านั้น methods เหล่านี้จะผนึกรวมมากับข้อมูล encatsulation นอกจากจะป้องกันการเข้าถึงข้อมูลของออบเจ็คท์โดยตรง เพราะ method เป็นวิธีเดียวที่จะเปลี่ยนสถานะของข้อมูล หลักการทั่วไปของ Object-Oriented คือ inheritance ออบเจ็คท์ที่ใช้คุณลักษณะจาก ออบเจ็คท์อื่นโดยไม่จำเป็นต้องสร้างฟังก์ชั่นเหล่านั้นใหม่ ดังนั้น inheritance ช่วยให้ซอฟท์แวร์นำกลับมาใช้ใหม่ได้ ข้อได้เปรียบอีกประการหนึ่งของการจัดการซอฟท์แวร์แบบ inheritance จะถูกจัดการไปตามคลาส หมายถึงแต่ละออบเจ็คท์ในคลาสจะมีคุณลักษณะออบเจ็คท์แม่ (parent objects) ซึ่งจะทำให้สามารถสร้างเอกสารของงาน ทำความเข้าใจ และผลประโยชน์ของซอฟท์แวร์ที่สร้างไว้ก่อนหน้า เพราะฟังก์ชั่นของซอฟท์แวร์มักจะเป็นการเพิ่มเติมจากออบเจ็คท์ที่สร้างขึ้นก่อนหน้านั้น ออบเจ็คท์ที่อยู่ปลายของ inheritance มักจะมีคุณลักษณะพิเศษและทำงานด้วยความเร็วสูง

3. การไม่ขึ้นกับแพลทฟอร์มและฮาร์ดแวร์ใด ๆ (Platform Independent)

เป็นข้อเด่นที่ทำให้ Java แพร่หลายไปได้อย่างรวดเร็ว ด้วยคุณสมบัติของภาษาจาวาที่เมื่อเขียนขึ้นแล้วจะถูกคอมไพล์ให้เป็นรูปแบบไบท์โค้ด (Byte Code Format) และเมื่อนำไปใช้งานในแพลทฟอร์มใด ๆ ก็จะถูกอ่านและรันโดยอินเตอร์พรีเตอร์ในแต่ละแพลทฟอร์มนั้น ๆ ด้วยคุณสมบัตินี้ ทำให้การใช้งานจาวาไม่ขึ้นกับแพลทฟอร์ม หรือฮาร์ดแวร์ใด ๆ

4. ความปลอดภัย (Safety)

ปลอดภัยจากไวรัสและการบุกรุกเข้าระบบคอมพิวเตอร์โดยไม่ได้รับอนุญาติ มีระบบรักษาความปลอดภัยของตัวภาษา ไม่ถูกลบหรือเปลี่ยนแปลงไฟล์ข้อมูล

5. คุณสมบัติ Distributed

ผู้ใช้สามารถดึงเอา Class Library จากที่ต่าง ๆ ผ่าน HTTP หรือ FTP ซึ่งทำงานอยู่บน TCP/IP โดย
การอ้างตำแหน่งผ่าน URL ของ Library เหล่านั้น

6. Interpreted

เนื่องจากจาวาคอมไพเลอร์แปลงไฟล์คลาสที่เป็น Source ของจาวา ให้อยู่ในรูปไบท์โค้ดแล้วไฟล์คลาสในรูปไบท์โค้ดนี้จะสามารถรันอยู่บนเครื่องใดก็ได้ ทำให้จาวาไม่ขึ้นอยู่กับฮาร์ดแวร์แพลทฟอร์มใดและแยกคอมไพเลอร์ ออกจากการรันโปรแกรมที่ไคลเอ็นท์ เนื่องจากไบทโค้ดไม่ได้จำเพาะเจาะจงกับเครื่อง ขอให้มี Virtual Machine ของจาวาก็ใช้ได้แล้ว และในการใช้ อินเตอร์เพรทเตอร์ (Interpreter) แปล Byte Code ในขณะที่ปฏิบัติงาน (Run-time) จะมีการตรวจสอบการทำงานของโปรแกรมไปพร้อม ๆ กับการแปลด้วย

7. Robust

โปรแกรมผ่านการตรวจสอบอย่างดีขณะเวลาแปลหรือคอมพาย (Compile) และขณะปฏิบัติงาน เพื่อให้แน่ใจว่าไม่สร้างปัญหาขึ้นภายหลังเมื่อเวลาใช้งานจริง ที่สำคัญ Java ไม่มีข้อมูลชนิด Pointer เพื่อป้องกันการเข้าใช้ข้อมูลกับหน่วยความจำ (Memory) โดยตรง อาจก่อให้เกิดผลเสียกับระบบได้

8. Architecture-neutral

ใช้ Byte Code รูปแบบเดียวกันหมด สามารถถูกปฏิบัติงานได้ทุกที่ไม่ขึ้นกับระบบบนเครือข่าย เพราะจาวาคอมไพเลอร์สร้างไบท์โค้ดที่จะถูกส่งไปยังบราวเซอร์ที่เรียกใช้ และอินเตอร์พรีตบนเครื่องปลายทาง ซึ่งมีบราวเซอร์ หรืออินเตอร์พรีเตอร์ของจาวาติดตั้งอยู่

9. High performance

ประสิทธิภาพในด้านความเร็วมีสูงเพราะ Byte Code ที่ถูกแปลก่อนการปฏิบัติงานโดย คอมพายเลอร์ (Compiler) มีรหัส (Code) ที่ใกล้เคียงกับภาษาเครื่อง (Machine Code) ของแต่ละเครื่อง

10. Multithreaded

ภาษาจาวาสามารถสร้างแอพพลิเคชั่นที่ทำงานได้หลายอย่างพร้อมกัน (Concurrency) เช่น ขณะที่ภาพกำลังเคลื่อนไหว เสียงดนตรีก็ไม่ขาดหาย เนื่องจากรูทีนของระบบที่อนุญาติให้ทำงานแบบ multiple threads ด้วยจาวา โปรแกรมเมอร์สามารถเขียนโปรแกรมแบบเรียลไทม์ และอินเตอร์แอคทีฟกับผู้ใช้ได้

11. Dynamic

Class Library จะถูกเชื่อมต่อ (link) รวมกันทั้งเวลาแปลและเวลาปฏิบัติงานทำให้ไม่ต้องแปลโปรแกรมใหม่ทั้งหมดเมื่อมีการแก้ไขเพียงบางส่วนของ Library ซึ่งจากโค้ดที่เขียนจาก C++ ที่ต้องคอมไพล์ซ้ำทุกครั้งที่มีการเปลี่ยน parent class

12. Portable

ชนิดและขนาดของข้อมูลเป็นมาตรฐานเดียวกันไม่ขึ้นกับ Hardware

13. Graphics

มีความสามารถในการเล่นไฟล์เสียง (Sound) และโดยเฉพาะ Animation Graphics

14. Int

แทนอินทีเจอร์ที่เป็น 32 บิต two's compliment

15. Float

แทนจำนวนโฟลทติงพอยต์แบบ 32 บิตเสมอ ตามมาตรฐาน IEEE 754

บทที่ 3
การเขียนโปรแกรมด้วยภาษาจาวา

สำหรับในที่นี้จะนำเสนอใน 5 รูปแบบด้วยกันคือ

1. หลักการเขียนโปรแกรมแบบ Object-Oriented Programming
2. โครงสร้างภาษาจาวา (Writing the Java Programs)
3. ออบเจ็คท์ (Objects)
4. คลาส (Classes)
5. อินเตอร์เฟซ (Interface)
6. การเขียนแอพเพล็ท (Writing Applets)

3.1 หลักการของการเขียนโปรแกรมแบบ Object-Oriented Programming

เนื่องด้วยภาษาจาวานี้ อาศัยหลักการเขียนโปรแกรมแบบ Object-Oriented Programming หรือ OOP ซึ่งมีรายละเอียดของคุณสมบัติของ OOP และ Java มีดังนี้

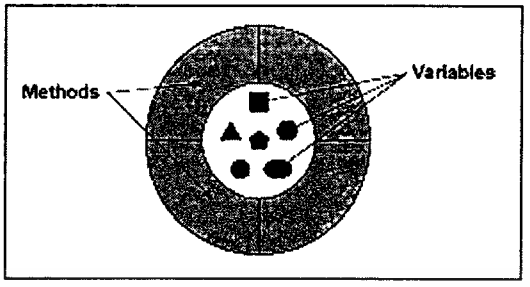
3.1.1 ความหมายของ Object-Oriented Programming

OOP คือ วิธีการเขียนโปรแกรมคอมพิวเตอร์โดยใช้แนวความคิดเสมือนการมองภาพวัตถุ (Object) ในโลกแห่งความเป็นจริง เป็นการจัดโครงสร้างของโปรแกรมให้เป็นระเบียบมากยิ่งขึ้น และเอื้ออำนวยต่อการพัฒนาโปรแกรมในรุ่นต่อๆไป OOP เป็นวิธีการจัดแบ่งประเภทของวัตถุในทางนามธรรม (Abstract) ออกเป็นกลุ่มๆ (Classes) ซึ่งในแต่ละกลุ่มก็มีสถานะ (States) และพฤติกรรม (Behaviors) เฉพาะของตัวเอง เพื่อเป็นต้นแบบให้แก่วัตถุที่จะถูกสร้างขึ้นใหม่ในอนาคต ข้อมูลหรือคุณสมบัติ (Characteristic) เฉพาะจะถูกเก็บซ่อน (Excapsulation) ไว้ภายในกลุ่มไม่ให้ไปปะปนกับกลุ่มอื่น วัตถุสามารถติดต่อสื่อสารซึ่งกันและกันโดยใช้ข่าวสาร (Message) และที่สำคัญวัตถุสามารถสืบทอด (Inheritance) คุณสมบัติจากบรรพบุรุษไปสู่ลูกหลานได้

3.1.1.2 อ็อปเจ็คท์(Object)

อ็อปเจ็คท์ คือ ซอฟท์แวร์ซึ่งประกอบด้วยข้อมูลและวิธีการจัดการกับข้อมูลนั้น อ็อปเจ็คท์เป็นกุญแจสำคัญในการทำความเข้าใจการเขียนโปรแกรมแบบ OOP โดยถ้าเราจะยกตัวอย่างอ็อปเจ็คท์ในโลกแห่งความเป็นจริง เช่น สุนัขเป็นสิ่งมีชีวิตที่มีสถานะ เช่น พันธุ์ สี ขนาด เป็นต้นและมีพฤติกรรมที่ตอบสนองต่อสถานะ เช่น สามารถเห่าได้ สามารถดมกลิ่นได้ เป็นต้น และอีกตัวอย่างหนึ่งคือ หลอดไฟเป็นสิ่ง

ของที่มีสถานะ เช่น รูปทรง กำลังวัตต์ สถานะของสวิทช์ไปเปิดปิด เป็นต้น และมีพฤติกรรม เช่น การส่องสว่าง การเกิดความร้อน เป็นต้น สำหรับการเขียนโปรแกรมแบบ OOP เราจะแทนสถานะด้วยตัวแปร (Variables) และแทนพฤติกรรมด้วยเม็ทธอด (Methods) ดังรูปที่ 3 - 1



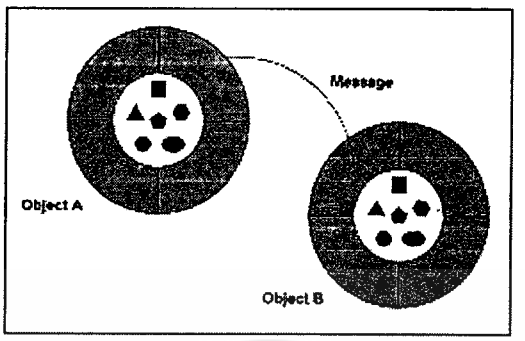
รูป 3 – 1 แสดงการจำลองส่วนประกอบของอ็อปเจ็คท์

จากรูป จะเห็นได้ว่าตัวแปรของอ็อปเจ็คท์ถูกจัดให้อยู่ตรงกลางของวงกลมอ็อปเจ็คท์และมีเม็ทธอดล้อมรอบเพื่อทำหน้าที่เก็บซ่อนส่วนกลางของอ็อปเจ็คท์ไว้ ประโยชน์ของการเก็บซ่อน (Encapsulation) นี้คือ ถ้าหากเราต้องเปลี่ยนแปลงข้อมูลภายในอ็อปเจ็คท์ การเปลี่ยนแปลงจะไม่ส่งผลต่ออ็อปเจ็คท์อื่นๆที่อยู่ภายนอกเพราะข้อมูลไม่ขึ้นต่อกัน (Independent) เราจึงสามารถเปลี่ยนแปลงหรือปรับปรุงส่วนของโปรแกรมที่เราต้องการเปลี่ยนแปลงได้ ในขณะที่คนอื่นก็สามารถเปลี่ยนแปลงส่วนของตนเองไปอย่างไม่เกิดการแซกแซงซึ่งกันและกัน และประโยชน์อีกข้อหนึ่งก็คือ เป็นการจัดการกับข้อมูลภายในอ็อปเจ็คท์ว่าข้อมูลส่วนนี้ต้องการที่จะเปิดเผยหรือไม่ ซึ่งสามารถกำหนดได้โดยการประกาศคุณสมบัติให้กับอ็อปเจ็คท์ เช่น ถ้ากำหนดเป็นข้อมูลแบบส่วนตัว(Private) อ็อปเจ็คท์อื่นๆจะไม่สามารถเข้ามาใช้ข้อมูลส่วนตัวนี้ได้ แต่ถ้าหากเรากำหนดให้เป็นแบบสาธารณะ(Public) อ็อปเจ็คท์อื่นๆจะสามารถเข้ามาใช้ข้อมูลนี้ได้ ซึ่งก็เป็นประโยชน์เมื่อต้องใช้ข้อมูลเหล่านั้นร่วมกัน ข้อมูลในที่นี้หมายถึงทั้งตัวแปรและเม็ทธอดภายในอ็อปเจ็คท์นั้นๆ

นอกจากนี้ในภาษาจาวามีการใช้ออบเจ็คท์ในการแทนสิ่งที่ไม่มีตัวตนได้เช่น มีการแทนออบเจ็คท์ในเหตุการณ์ (event) ต่างๆ เช่น การกดเมาส์ การปล่อยเมาส์ การกดคีย์บอร์ด เป็นต้น สำหรับคุณสมบัติการ Encapsulation ของออบเจ็คท์นี้ ทำให้การใช้คุณสมบัติออบเจ็คท์ของจาวานี้ได้เปรียบการเขียนโปรแกรมแบบอื่นๆคือ ออบเจ็คท์แต่ละออบเจ็คท์จะเป็นอิสระจากตัวซอร์สโค้ด นั่นคือ เราสามารถนำออบเจ็คท์ไปใช้ที่อื่นก็ได้ หรือนำไปใช้ที่ไหนก็ได้ และออบเจ็คท์ยังทำให้สามารถซ่อนข้อมูลบางอย่างที่ไม่จำเป็นต้องแสดงออกมา ทำให้เราสามารถเปลี่ยนแปลงข้อมูลในส่วนนี้ได้ตลอดเวลา โดยไม่มีผลกระทบต่อการทำงานของ method ปกติ

3.1.1.3 ข่าวสาร(Messages)

การติดต่อสื่อสารและแลกเปลี่ยนข้อมูลระหว่างอ็อปเจ็คท์ทำได้โดยการรับส่งข่าวสาร ยกตัวอย่าง เช่น เมื่ออ็อปเจ็คท์ A ต้องการทำงานกับเม็ทธอดของอ็อปเจ็คท์ B อ็อปเจ็คท์ A ก็จะส่งข่าวสารไปบอกแก่อ็อปเจ็คท์ B ดังรูป 3 – 2



รูป 3 –2 แสดงการส่งข่าวสารระหว่างอ็อปเจ็คท์

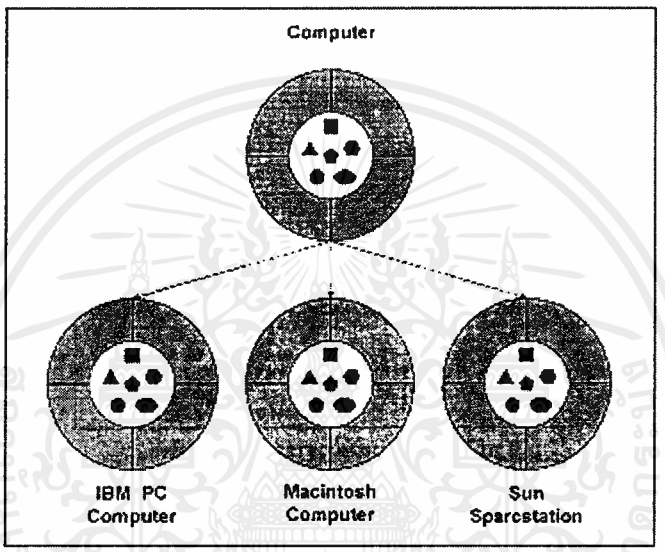
3.1.1.4 คลาส (Class)

คลาส คือ ต้นแบบของการกำหนดตัวแปรและเม็ทธอดของอ็อปเจ็คท์ ฉนั้นคลาสหนึ่งคลาสสามารถเป็นต้นแบบให้แก่อ็อปเจ็คท์หลายๆอ็อปเจ็คท์ได้ ยกตัวอย่างเช่น รถจักรยานหลายๆคันมีคุณสมบัติการเป็นรถจักรยานเหมือนกัน มีล้อสองล้อ มีตัวถัง มีอานสำหรับนั่งเหมือนกัน เป็นต้น ที่โรงงานผลิตรถจักรยานก็จะมีต้นแบบพื้นฐานของรถจักรยานที่จะผลิตเช่น แบบพิมพ์เขียน (คลาส) ของรถจักรยานเป็นต้น แต่เมื่อผลิตรถจักรยานจริงๆ (อ็อปเจ็คท์) ออกมาเสร็จแล้ว รายละเอียดปีกย่อยอาจจะแตกต่างกันไปแล้วแต่รุ่น เช่น สี ขนาด น้ำหนัก รูปทรง เป็นต้น การที่เราสร้างอ็อปเจ็คท์ขึ้นมาหนึ่งอ็อปเจ็คท์หรือหลายๆอ็อปเจ็คท์เราเรียกว่า อินสแตนซ์ (Instance) เราต้องสร้างอินสแตนซ์ขึ้นมาจากคลาสก่อนที่เราจะมีการใช้ตัวแปรหรือเม็ทธอดของคลาสนั้น และจะเรียกว่าอินสแตนซ์ ว่าเป็นอ็อปเจ็คท์เมื่อมีการเปลี่ยนแปลงข้อมูลหรือเรียกใช้เม็ทธอดของอินสแตนซ์นั้นเกิดขึ้น

3.1.1.5 การสืบทอด (Inheritance)

การสืบทอด คือ การที่คลาสเก่าที่มีอยู่แล้วเป็นต้นแบบให้กับคลาสใหม่ที่จะถูกสร้างขึ้น เกิดการถ่ายทอดคุณสมบัติจากคลาสแม่ (Superclass) ไปสู่คลาสลูก (Subclass) หรือพูดในทางกลับกันก็คือ คลาสลูกสืบทอดคุณสมบัติมาจากคลาสแม่ และที่สำคัญนอกจากคลาสลูกซึ่งสืบทอดคุณสมบัติทุกอย่างมาจากคลาสแม่แล้วในขณะเดียวกันคลาสลูกก็สามารถที่จะเพิ่มคุณสมบัติให้กับตัวเองได้อีกด้วย เช่น การเพิ่มตัวแปร หรือเม็ทธอดให้กับตัวเอง ดังรูป 3 - 3 เป็นตัวอย่างของการสืบทอดคุณสมบัติของเครื่องคอมพิวเตอร์ ซึ่งคลาสลูกจะเป็นคอมพิวเตอร์เหมือนกันแต่ได้เพิ่มคุณสมบัติเฉพาะตัวเองเข้าไปอีกจนเกิดเป็นคอมพิวเตอร์ชนิดใหม่ขึ้นมา

คลาสลูกสามารถที่จะใช้ชื่อของเม็ทธอดเหมือนกันกับชื่อเม็ทธอดของคลาสแม่ได้ซึ่งเป็นประโยชน์เพื่อเพิ่มความสามารถพิเศษให้แก่เม็ทธอดเดิม (Overriding) การสืบทอดแบบนี้ถือว่าเป็นคุณสมบัติเด่นของการเขียนโปรแกรมแบบ OOP เพราะทำให้เราสามารถใช้รหัสหรือโค้ด (Code) เดิมได้โดยไม่ต้องมีการเขียนโปรแกรมที่เคยเขียนไว้แล้วขึ้นมาใหม่อีก เพราะคอมพายเลอร์ (Compiler) จะค้นหาโค้ดให้เราเองเมื่อเวลาคอมพาย (Compile) โปรแกรม โปรแกรมเมอร์สามารถกำหนดต้นแบบของคลาส (Abstract Classes) เพื่อให้โปรแกรมเมอร์คนอื่นๆนำไปพัฒนาต่อโดยการเพิ่มรายละเอียดเฉพาะของตัวเองเข้าไป



รูปที่ 3 – 3 แสดงการสืบทอดคุณสมบัติของเครื่องคอมพิวเตอร์

3.2 โครงสร้างของภาษา Java

สำหรับในหัวข้อนี้ จะกล่าวถึงการเขียนโปรแกรมด้วยภาษาจาวา ว่าประกอบด้วย องค์ประกอบพื้นฐานอะไรบ้าง

3.2.1 โครงสร้างพื้นฐาน

ซอร์สโค้ด (Source Code) หรือ ไฟล์ตัวอักษรธรรมดาที่เราเขียนขึ้นโดยภาษา Java มีนามสกุลของไฟล์ (File Extension) เป็น .java และชนิดของโปรแกรม Java ซึ่งถูกคอมพายแล้วเป็น .class มีอยู่ 2 แบบคือ Application และ Applet แต่ไม่ว่าโปรแกรม Java จะเป็นแบบใดก็จะใช้โครงสร้างการเขียนแบบเดียวกันคือ ในแต่ละไฟล์ .java จะมีหน่วยที่ต้องถูกคอมพาย (Compilation Unit) อยู่อย่างน้อย 1 หน่วย และในแต่ละหน่วยนี้จะประกอบด้วยส่วนย่อยอีก 4 ส่วนคือ Package statements , Import statements ,

Class declaration , และ Interface declaration ซึ่งที่สำคัญที่สุดคือคลาส (Class) เพราะอย่างน้อย โปรแกรม Java หนึ่งโปรแกรมจะต้องมีคลาสอย่างน้อย 1 คลาส และคลาสนี้จะประกอบไปด้วยส่วนสำคัญอยู่ 2 ส่วนคือ ตัวแปร (Variable) และเม็ทธอด (Method) ซึ่งเม็ทธอดก็คือฟังก์ชั่น (Function) ซึ่งทำหน้าที่จัดการกับข้อมูลเหมือนภาษา C/C++ โปแกรม Java ที่เป็น Application ทุกตัวที่จะถูกรันเหมือนโปรแกรมใช้งานทั่วไปต้องมีเม็ทธอด main() บรรจุอยู่ด้วยทุกโปรแกรม (เหมือนกับภาษา C/C++) สำหรับเป็นจุดเริ่มต้นของโปรแกรมเมื่อถูกรันจากอินเตอร์พรีเตอร์ (Interpreter) และโครงสร้างพื้นฐานของโปรแกรม Java Application จะเป็นดังนี้

```
Class ClassName {
    Public static void main ( String args[] ) {
            // add code here
    }
}
```

จากบรรทัดที่สองจะอธิบายได้ว่า เม็ทธอด main ( ) จะรับค่าตัวแปรจาก Command-line ผ่านทางตัวแปรอาร์เรย์ (Array) ชนิดสตริง (String) ที่ชื่อ arg [ ] เม็ทธอดนี้ไม่มีการส่งค่ากลับคืน (void) ทุกๆอินสแตนซ์ที่ถูกสร้างขึ้นจะใช้เม็ทธอดนี้ร่วมกัน (Static) และอ็อปเจ็คท์อื่นมีสิทธิ์ที่จะขอเรียกใช้เม็ทธอดนี้ (Public) โดยทั่วไปแล้วเรามักจะมีการใช้เม็ทธอดอื่นที่จำเป็นจากสิ่งแวดล้อมของ Java (Java Environment) โดยใช้คำหลัก **Import** เพื่อดึงเอาเม็ทธอดหรือโปรแกรมย่อย (Subroutine) จากคลาสต่างๆ ของ Java มาใช้ และโครงสร้างการเขียนจะเป็นดังนี้

```
Import ClassName

Class ClassName  {
        public static void main ( String args [ ] ) {
                // add code here
    }
}
```

3.2.2 วงเล็บปีกกาและบล็อค (Braces and Blocks)

การกำหนดขอบเขตของกลุ่มคำสั่งของ Java จะใช้วงเล็บปีกกาเปิด "{" เป็นจุดเริ่มต้น และ จะใช้วงเล็บปีกกาปิด "}" เป็นจุดสิ้นสุดของขอบเขตหรือบล็อค (Block) นั้น คำสั่งหรือคำหลักต่างๆที่อยู่ภายใต้บล็อคนี้ก็จะเป็นอิสระจากบล็อคอื่น และที่สำคัญต้องมีวงเล็บปีกกาเปิดและปิดอยู่คู่กันเสมอไม่เช่นนั้น

และจะคอมพายไม่ผ่าน ภายในหนึ่งบล็อคสามารถบรรจุบล็อคย่อยอีกหลายๆบล็อคและจะสืบทอดกันไปเรื่อยๆ ยกตัวอย่างเช่นบล็อคของเม็ทธอด main ( ) { } สามารถมีบล็อคของการวนซ้ำหรือลูป (Loop) อยู่หลายๆลูป เป็นต้น

3.2.3 การตั้งชื่อ (Indentifiers)

การตั้งชื่อสำหรับ ตัวแปร คลาส และเม็ทธอด ต้องขึ้นต้นด้วยตัวอักษร 'a - 'z','A' - 'Z' หรือเครื่องหมายขีดล่าง (Underscore) '_' หรือ เครื่องหมายดอลล่าร์ (Dollar Sign) '$' เท่านั้น ตัวอักษรต่อๆไปสามารถเป็นตัวเลข '0' - '9' หรือตัวอักษรพิเศษอื่นๆก็ได้เช่น π เป็นต้น ตัวอักษรตัวเล็กและตัวใหญ่ เช่น 'a' กับ 'A' ภาษา Java จะมองเป็นคนละตัวกัน เช่นคำว่า "World" กับ "world" จะเป็นคนละคำกัน การตั้งชื่อควรจะสื่อความหมายและหน้าที่ของ ตัวแปร หรือ คลาส หรือ เม็ทธอด และที่สำคัญการตั้งชื่อคลาสตัวอักษรตัวแรกต้องเป็นตัวใหญ่เท่านั้น เช่น class MoutainBike , class School , class FirstClass เป็นต้น ส่วนตัวแปรและเม็ทธอดควรจะขึ้นต้นด้วยตัวอักษรตัวเล็ก เช่น int counter, float grade2 , String full_name เป็นต้น สำหรับการตั้งชื่อที่ใช้เป็นตัวอักษรตัวใหญ่หมดเลย เช่น PI , MAX , UFO หรือคำอื่นๆที่ใช้กับคำหลัก #define ของภาษา C/C++ เพื่อกำหนดค่าคงที่นั้น ไม่แนะนำให้ใช้ในภาษา Java เพราะภาษา Java สามารถเชื่อมต่อ (Linking) กับภาษา C/C++ ได้ จะกล่าวต่อไปในภายหลัง

3.2.4 คำหลัก (Keywords)

คำหลักถูกกำหนดขึ้นโดยภาษา Java เอง ฉะนั้นจึงไม่สามารถที่จะตั้งชื่อของตัวแปร คลาส และเม็ทธอดให้ซ้ำได้ เพื่อเป็นการดีให้สังเกตุว่าคำหลักทั้งหมดจะเป็นคำเดี่ยวๆที่ใช้ตัวอักษรตัวเล็ก และก่อนการตั้งชื่อเราควรจะตรวจดูคำที่เราจะกำหนดกับคำสงวน (Reserved Keywords) ของภาษา Java เสียก่อน ดังตารางที่ 3-1

คำหลักสงวนของ Java			
Abstract	else	int	static
Boolean	extends	interface	super
Break	false	long	switch
Byte	final	native	synchronized
Byvalue*	finally	new	this
Case	float	null	threadsafe
Cast*	for	operator*	throw
Catch	future*	outer*	thansient
Char	generic*	package	true

Class	goto*	private	try
Const*	if	protected	var*
Continue	inner*	public	void
Default	implements	rest*	while
Do	import	return	
Double	instanceof	short	
* เป็นคำหลักสงวน แต่ java เลิกใช้แล้ว			

ตารางที่ 3 – 1 แสดงคำหลักสงวนของ Java

3.2.5 การประกาศตัวแปร

จากข้างต้นที่ได้อธิบายถึงชนิดของตัวแปรต่างๆที่ใช้ในภาษา Java รวมไปถึงการประกาศชนิดของตัวแปร และการกำหนดค่าแรกให้แก่ตัวแปรด้วย ดังตัวอย่างจะเห็นได้ว่ามีการประกาศที่เหมือนกันกับการประกาศอ็อปเจ็คท์ คือ

```
Class ClassName {
    Type name1;
}
```

แต่การกำหนดค่าแรกนั้นจะแตกต่างกันคือตัวแปรสามารถกำหนดได้เลยโดยตรงโดยใช้เครื่องหมายเท่ากับแล้วตามด้วยค่าที่ต้องการกำหนด เช่น int counter = 10; float average = 0.45; เป็นต้น แต่การสร้างและกำหนดค่าแรกให้แก่อ็อปเจ็คท์ โดยมากแล้วมักจะทำผ่านตัวดำเนินการ new เพื่อสื่อความหมายว่าเป็น การสร้างอ็อปเจ็คท์

การประกาศตัวแปร จะมีองค์ประกอบอย่างเต็มยศดังนี้

```
[accessSpecifier][static][final][transient][volatile] type variableName
```

ตัวที่อยู่ใน [] หมายถึง จะใส่หรือไม่ก็ได้ สำหรับองค์ประกอบต่างๆเหล่านี้ สามารถอธิบายได้ดังนี้

1. accessSpecifier เป็นการกำหนดควบคุมการเข้าใช้หรือควบคุมการเข้าถึงตัวแปร หรือ method ว่าได้หรือไม่

2. static จะแสดงให้เห็นว่าตัวแปร หรือ method นั้นๆเป็น class member variable ซึ่งตรงกันข้ามกับ instance member variable และ class method ตามลำดับ
3. final บอกให้รู้ว่าตัวแปรนี้เป็นค่าคงที่
4. volatile แสดงให้เห็นว่าตัวแปร สามารถเปลี่ยนแปลงได้แบบ Asynchronous

3.2.6 เม็ทธอด (Method)

เม็ทธอดเปรียบได้กับฟังก์ชั่น (Function) ในภาษา C/C++ ซึ่งทำหน้าที่ได้หลายอย่าง เช่น เป็นโปรแกรมย่อย (Subroutine) ไว้ทำงานซ้ำๆให้เม็ทธอดอื่นเรียกใช้งานได้บ่อยๆ หรือถูกเรียกใช้งานจากเม็ทธอดตัวมันเอง (Recursive Methods) เป็นต้น มีรูปแบบการเขียนดังนี้

```
ReturnType name (parameters) {
    // function body
    [ return returnValue;]
}
```

returnType คือ การกำหนดชนิดของข้อมูลที่จะส่งกลับ
name คือ ชื่อของเม็ทธอด
parameters คือ ข้อมูลที่จะรับเข้ามาสู่เม็ทธอด ประกาศเหมือนกับการประกาศตัวแปร
function body คือ คำสั่งต่างๆ ภายในเม็ทธอด
returnValue คือ ข้อมูลที่ต้องการจะส่งกลับ โดยใช้คำหลัก return นำหน้า และจะต้องมีทุกครั้ง เมื่อประกาศให้เม็ทธอดต้องส่งค่ากลับคืน

ถ้าหากต้องการกำหนดให้ไม่มีการรับข้อมูลเข้าหรือส่งผลลัพธ์กลับคืนเราสามารถใช้คำหลัก void ได้เหมือนภาษา C/C++ และไม่ต้องใช้คำหลัก return เมื่อไม่มีการส่งข้อมูลกลับคืน

3.2.7 เม็ทธอดเรียกตัวเอง (Recursive Method)

ภาษา Java สามารถเขียนโปรแกรมให้เม็ทธอดสามารถเรียกตัวเองได้ เช่น ภายในเม็ทธอดชื่อ A สามารถมีการเรียกใช้งานเม็ทธอดชื่อ A ได้เป็นต้น แต่การเขียนเม็ทธอดในรูปแบบของการเรียกตัวเองเราต้องระมัดระวังให้ดีเพื่อไม่ให้เกิดการเรียกตัวเองจนกระทั่งไม่มีทางออกหรือไม่มีที่สิ้นสุด

3.2.8 ตัวแปรและชนิดของข้อมูล

ตัวแปร

ในภาษาจาวาการประกาศตัวแปรจะ มีรูปแบบการประกาศ ทีประกอบด้วย 2 ส่วนทุกครั้งที่ประกาศ คือประกอบด้วย ชนิดของตัวแปร และชื่อของตัวแปร ดังตัวอย่างการกำหนดการประกาศและการสร้างตัวแปรในการให้ค่าเริ่มต้น

```
Int count = 0;          //ตัวแปรและชนิดของข้อมูล
```

สำหรับชนิดของตัวแปร หรือ ข้อมูลในภาษาจาวามี 2 แบบด้วยกัน คือ แบบข้อมูลมาตรฐาน และข้อมูลซับซ้อน

ชนิดข้อมูลมาตรฐาน

ตัวแปรทั้งหมดของภาษา Java ต้องมีการกำหนดชนิดข้อมูล ภาษา Java มีชนิดของข้อมูลที่ให้เลือกใช้ตามมาตรฐานของภาษาคอมพิวเตอร์ทั่วไปคือ ชนิดจำนวนเต็ม ชนิดจำนวนจริง ชนิดตัวอักษร และชนิดตรรก รายละเอียดของขนาดหน่วยความจำที่ใช้สำหรับเก็บตัวแปรหนึ่งๆ ในแต่ละชนิดข้อมูลมีดังตาราง

ชนิดของข้อมูล	ขนาดและรูปแบบ
Bye	8-bit two's complement
Short	16-bit two's complement
Int	32-bit two's complement
Long	64-bit two's complement
Float	32-bit IEEE 754 floating point
Double	64-bit IEEE 754 floating point
char	16-bit Unicode character

ตารางที่ 3 – 2 แสดงรายละเอียดของชนิดข้อมูล

ฉะนั้นจากตัวอย่างตัวแปร count คือชนิดข้อมูลจำนวนเต็ม จากการประกาศโดยใช้คำหลัก int นำหน้าชื่อตัวแปร และข้อความ " = 0 " คือการสร้างตัวแปรโดยการให้ค่า จากตัวอย่างถ้าปราศจากข้อความส่วนนี้เมื่อเวลาคอมพายแอพพลิเคชั่นหรือแอพเพล็ทจะคอมพายไม่ผ่านโดยคอมพายเลอร์จะแสดงคำเตือนข้อผิดพลาดที่เกิดขึ้นว่ามีตัวแปรที่ไม่ได้ถูกให้ค่าเริ่มต้น ซึ่งนั่นหมายถึงเราต้องให้ค่าแก่ตัวแปรที่เราประกาศไว้เสียก่อนที่จะมีการกระทำใดๆต่อไปกับตัวแปรนั้นเช่นการคำนวณเป็นต้น

ชนิดข้อมูลซับซ้อน

นอกจากชนิดข้อมูลมาตรฐานแล้วภาษา Java ยังมีข้อมูลชนิดซับซ้อน คือ อาเรย์ สตริง และอ็อปเจ็คท์ เวลาที่เราประกาศตัวแปรให้เป็นชนิดข้อมูลเหล่านี้เราสามารถกำหนดในรูปแบบของการประกาศแบบคลาสและแบบอินเตอร์เฟซ (Class and Interface definitions)
ตัวดำเนินการ

3.2.9 ตัวดำเนินการ (Operator)

ในภาษา Java ได้จัดแบ่งตัวดำเนินการของภาษาออกเป็น 4 ชนิดคือ ชนิดการคำนวณ ชนิดความสัมพันธ์ ชนิดตรรก และชนิดสตริง ความหมายของแต่ละชนิดมีดังต่อไปนี้

ตัวดำเนินการคำนวณ

ภาษา Java ได้สนับสนุนตัวดำเนินการคำนวณในหลายๆรูปแบบไม่ว่าจะเป็นการบวก ลบ คูณ หาร สำหรับข้อมูลตัวเลข และตัวดำเนินการคำนวณยังแบ่งออกได้อีก 2 แบบคือ แบบใช้กับตัวแปรเดี่ยวซึ่งอาจใช้เติมไว้ข้างหน้าหรือข้างหลังก็ได้ และแบบใช้กับตัวแปรสองตัว โดยมีตัวดำเนินการนั้นคั่นอยู่ตรงกลางระหว่างตัวแปรทั้งสอง รายละเอียดของตัวดำเนินการคำนวณทั้งสองแบบดังแสดงในตารางที่ และตารางที่ และยังมีตัวดำเนินการที่น่าสนใจอีก 5 ตัวคือ += -= *= /= และ %= ตัวดำเนินการเหล่านี้ทำงานโดยการให้ค่าแก่ตัวแปรผลลัพธ์ซ้ายมือโดยการบวก ลบ คูณ หาร หรือ หารเอาเฉพาะเศษ อย่างใดอย่างหนึ่งตามลำดับโดยตัวแปรหรือจำนวนใดๆกับผลลัพธ์นั้นอย่างเช่น x+=2; มีค่าเท่ากับ x = x+2; เป็นต้น

ตัวดำเนินการ	คำอธิบาย
-	เปลี่ยนจำนวนบวกเป็นลบและลบเป็นบวก
~	เปลี่ยน 0 เป็น 1 และ 1 เป็น 0
++	เพิ่มทีละหนึ่ง
--	ลดทีละหนึ่ง

ตารางที่ 3 – 3 แสดงตัวดำเนินการคำนวณแบบใช้กับตัวแปรเดี่ยว

ตัวดำเนินการ	คำอธิบาย
+	บวก
-	ลบ

*	คูณ
/	หาร
%	หารเอาเฉพาะเศษ (MOD)
&	และ (AND)
|	หรือ (OR)
^	ต่างกันได้จริงและเหมือนกันได้เท็จ (XOR)
<<	เลื่อนบิทข้อมูลไปทางซ้าย
>>	เลื่อนบิทข้อมูลไปทางขวา
>>>	เลื่อนบิทข้อมูลไปทางขวาแล้วเติมศูนย์จากทางซ้าย

ตารางที่ 3 - 4 แสดงตัวดำเนินการคำนวณแบบใช้กับตัวแปรสองตัว

ตัวดำเนินการสัมพันธ์

ทำหน้าที่เปรียบเทียบตัวแปรสองตัวและให้ผลลัพธ์เป็นข้อมูลชนิดตรรกคือจริงหรือเท็จ โดยรายละเอียดของตัวดำเนินการสัมพันธ์ทั้งหมดแสดงดังตาราง

ตัวดำเนินการ	การอธิบาย
<	น้อยกว่า
>	มากกว่า
<=	น้อยกว่าและเท่ากับ
>=	มากกว่าและเท่ากับ
==	เท่ากับ
!=	ไม่เท่ากับ

ตารางที่ 3 – 5 แสดงตัวดำเนินการสัมพันธ์

ตัวดำเนินการตรรก

ทำหน้าที่นำตัวแปรสองตัวมากระทำกันแบบตรรกและให้ผลลัพธ์เป็นแบบตรรกด้วย เช่น AND OR NOT XOR เป็นต้น ส่วนมากแล้วโปรแกรมเมอร์มักจะใช้ตัวดำเนินการตรรกกับการทดสอบเงื่อนไข รายละเอียดของตัวดำเนินการตรรกทั้งแสดงดังตาราง

ตัวดำเนินการ	คำอธิบาย
!	เปลี่ยนจากเท็จเป็นจริงและจริงเป็นเท็จ (NOT)
&	และ (AND)
|	หรือ (OR)
^	ต่างกันได้จริงและเหมือนกันได้เท็จ (XOR)
&=	ให้ค่าแก่ผลลัพธ์โดย AND ข้อมูลใดๆกับผลลัพธ์
|=	ให้ค่าแก่ผลลัพธ์โดย OR ข้อมูลใดๆกับผลลัพธ์
^=	ให้ค่าแก่ผลลัพธ์โดย XOR ข้อมูลใดๆกับผลลัพธ์
?=	เลือกการให้ค่าโดยการตรวจสอบเงื่อนไข

ตารางที่ 3 – 6 แสดงตัวดำเนินการตรรก

ตัวดำเนินการสตริง

เป็นตัวดำเนินการสุดท้ายที่จะกล่าวถึง ตัวอย่างเช่น

```
System.out.println("Input has" + count + "chars.");    // การรวมกันของข้อมูลสตริง
```

จากตัวอย่างแสดงการรวมสตริง "Input has" กับตัวแปรจำนวนเต็ม count และสตริง "chars." เข้าด้วยกันโดยตัวดำเนินการ '+' ถึงแม้ว่าตัวแปร count จะเป็นชนิดจำนวนเต็ม int แต่เมื่อเวลาคอมพายแอพพลิเคชั่นหรือแอพเพล็ท คอมพายเลอร์เมื่อเจอข้อมูลที่อยู่ภายในเครื่องหมายคำพูดแล้วถูกรวมโดยใช้ตัวดำเนินการบวกกับตัวแปรตัวเลขใดๆ คอมพายเลอร์จะแปลงข้อมูลตัวเลขนั้นให้เป็นสตริงทันที แล้วนำข้อมูลสตริงนี้ไปรวมเข้ากับข้อมูลจริงๆซึ่งอยู่ก่อนหน้าหรือต่อหลังอีกทีเพื่อแสดงผลข้อมูลสตริงออกหน้าจอภาพรวมกัน

ลำดับความสำคัญของตัวดำเนินการ

การให้ค่าแก่ตัวแปรหรืออาจจะเป็นการทดสอบเงื่อนไขจำเป็นต้องใช้สมการซึ่งมีความซับซ้อนของลำดับก่อนหลังการทำการคำนวณระหว่างคู่คำนวณหนึ่งๆในสมการอันซับซ้อน ในภาษา Java ลำดับดังกล่าวดังแสดงในตาราง โดยลำดับความสำคัญสูงสุดลงสู่ต่ำสุดจะเรียงตามแนวตั้งและจะเท่ากันในแนวขวาง

	[]	()		
++	--	!	~	instanceof
*	/	%		
+	-			
<<	>>	>>>		
<	>	<=	>=	
==	!=			
&				
^				
&&				
||				
?:				
=				
,	op=			

ตารางที่ 3 – 7 แสดงลำดับความสำคัญจากสูงสุดลงไปต่ำสุดของตัวดำเนินการ

3.2.10คำสั่งควบคุมทิศทาง (Control Flow Statements)

เป็นการกำหนดเงื่อนไขการเดินหน้าของโปรแกรมโดยการทำในแต่ละรูปแบบดังนี้

รูปแบบ	คีย์เวิร์ด
การตัดสินใจ	If-else , switch-case
ลูป	For , while , do-while
การยกเว้น	Try-catch-finally , throw
อื่นๆ	Break , continue , label: , return

ตารางที่ 3 – 8 แสดงคำสั่งควบคุมทิศทางอื่นๆ

การใช้ if-else

เราสามารถเขียนรูปแบบการใช้ if-else ได้ง่าย ๆ ดังนี้

if *(expression)*

statement1

else *statements*

โดยโปรแกรมจะทำตาม statement1 ก็ต่อเมื่อ expression เป็นจริง (True) แต่ถ้าไม่จริง ก็จะทำตาม statement2 ของ else แต่ตามปกติก็ไม่จำเป็นต้องมี else ทุกครั้งก็ได้ นอกจากนี้เรายังสามารถใช้ประโยคที่มีการกำหนดเงื่อนไข if-else ที่ย่อยเข้าไปมากๆได้ (Nested) เช่น

if *(exp1)*
stat1
else if (exp2)
stat2
else if *(exp3)*
stat3
else...

การใช้ switch

เป็นการใช้เงื่อนไขที่จะกระทำตามเงื่อนไขใดๆ โดยขึ้นกับเงื่อนไขที่กำหนด ที่มีหลายเงื่อนไข โดยมีรูปแบบการเขียนดังนี้

```
switch (test) {
case valueOne:
        resultOne;
        break;
case valueTwo:
        resultTwo;
        break;
...
default: defaultresult;
}
```

การใช้ Loop Statements

การใช้ for Loop มีรูปแบบการเขียนง่ายๆดังนี้

for (initialization;termination;increment)

statements

นั่นคือ โปรแกรมจะทำตาม statements ไปเรื่อยๆ โดยเริ่มต้นกำหนดค่าเริ่มต้นที่ initialization จนกระทั่งถึงจุดปลาย (termination) โดยขณะที่ทำตาม statements จนจบก็จะกลับมาตรวจดูค่าเงื่อนไขที่กำหนดแต่แรกเสมอ จากนั้นจะ increment ตามที่กำหนด

การใช้ do-while Loop

มีรูปแบบการใช้ดังนี้

```
do {
        statements
} while (booleanExpression);
```

นั่นคือ โปรแกรมจะยังคงทำตาม statements ตราบใดที่ค่าใน (Boolean Expression) ยังคงเป็นจริงตามเงื่อนไขที่กำหนด

ตัวอย่างแสดงการกำหนดคำสั่งควบคุมทิศทาง

```
while (System.in.read() != '\n')        // คำสั่งควบคุมทิศทางและการรับข้อมูลมาตรฐาน
```

จากตัวอย่างการใช้คำสั่งควบคุมทิศทาง while เพื่อควบคุมทิศทางของการทำงานของแอพพลิเคชั่นหรือแอพเพล็ท ให้วนรอบหลายๆรอบ โดยจำนวนรอบสูงสุดนี้ถูกควบคุมโดยเงื่อนไขที่เป็นจริง เมื่อใดก็ตามที่เงื่อนไขนั้นไม่เป็นจริงหรือเป็นเท็จวงรอบการทำงานนั้นก็จะสิ้นสุดและการทำงานจะหลุดออกจากวงรอบนั้นไปทำงานส่วนถัดไปทันที สำหรับรูปแบบการเขียนของคำสั่งควบคุมทิศทาง while แสดงได้ดังนี้

```
While (expression)
    Statement       // หรืออาจมีบล็อคของวงเล็บปีกกา "{...}" ล้อมรอบ statements ก็ได้
```

ดังนั้นจึงอธิบายได้ว่า ตราบใดที่ expression เป็นจริง ส่วนของ statement ก็ยังคงทำงานตลอด

3.2.11 การละเว้นความผิดพลาด (Exception Handling Statement)

```
Int x = 0;
Int y = 2;
System.out.println("Answer = "+y/x);        # ความผิดพลาดที่เกิดขึ้นขณะรัน
```

จะเห็นได้ว่าจากตัวอย่างโปรแกรม y/x รูปแบบของการเขียนโปรแกรมถูกต้องตามหลักไวยากรณ์ของภาษาคอมพิวเตอร์ทุกประการ เมื่อเวลาแอพพลิเคชั่นหรือแอพเพล็ทถูกคอมไพล์หรือแปล คอมพายเลอร์ก็จะไม่แสดงความผิดพลาดใดๆของโปรแกรมออกมาให้เราทราบเลย แต่ความผิดพลาดนี้จะไปแสดงผลในขณะที่รันโปรแกรมซึ่งเครื่องคอมพิวเตอร์และระบบปฏิบัติการโดยทั่วไปจะไม่ยอมรับการคำนวณแบบนี้ และอาจทำให้โปรแกรมหยุดทำงานลงทันที

อย่างไรก็ตามเหตุการณ์ของการเกิดความผิดพลาดระหว่างการรันโปรแกรมนี้ ระบบคอมพิวเตอร์จะมีวิธีการที่จะจัดการกับเหตุการณ์เช่นนี้ได้โดยใช้วิธีการที่เรียกว่า "การละเว้น" (Exception) ในภาษา Java เราสามารถที่จะใช้ประโยชน์ของการละเว้นความผิดพลาดซึ่งระบบ Java จะคอยตรวจสอบข้อผิดพลาดเหล่านั้น เราสามารถเลือดที่จะจับเอาความผิดพลาดแต่ละแบบที่เกิดขึ้นจากการตรวจสอบของระบบ Java และจัดการกับความผิดพลาดนั้น (Exception handler) เช่น ความผิดพลาดบางกรณีที่ไม่ร้ายแรงมาก นักเราก็อาจจะไม่ทำอะไรเลยก็ได้ แต่ในบางกรณีที่เกิดความผิดพลาดร้ายแรงเราก็ต้องหาวิธีจัดการที่เหมาะสมมากกว่า เช่น แจ้งให้ผู้ใช้ทราบ

Exception มี 2 ส่วนคือ สัญลักษณ์ของ Exception และการรองรับความผิดพลาด สัญลักษณ์ของ Exception ใช้ Try ในการกำหนด ส่วนการรองรับความผิดพลาดใช้คำสั่ง Catch หากต้องการบอกกับระบบว่ามีความผิดพลาดเกิดขึ้นให้ใช้คำสั่ง Throw

ตามหลักของภาษา Java แล้วทุกเม็ทธอดต้องมีการกำหนดการละเว้นความผิดพลาดหลังจากการประกาศชื่อเม็ทธอดในกรณีที่เนื้อหาภายในเม็ทธอดนั้นอาจก่อให้เกิดความผิดพลาดแบบนั้นๆเกิดขึ้น

ชื่อ Exception	คำอธิบาย
ArithmeticException	ตัวเลขถูกหารด้วยศูนย์
ArrayIndexOutOfBoundException	เลขอินเด็กซ์ของอาร์เรย์ไม่ถูกต้อง
ArrayStoreException	ชนิดของวัตถุที่ไม่ถูกต้องถูกเก็บลงในอาร์เรย์
ClassCastException	การเปลี่ยนแปลงชนิดผิดพลาด
ClassNotFoundException	คลาสที่ใช้ไม่สามารถหาพบ
CloneNotSupportedException	วัตถุไม่สามารถสำเนาได้
Exception	Exception มาตรฐานทั่วไป
FileNotFound	ไฟล์ไม่สามารถใช้ได้
IllegalAccessException	วิธีการเฉพาะที่ระบุไม่สามารถหาพบ
IllegalArgumentException	อาร์กิวเมนต์ที่ส่งค่าไม่ถูกต้อง

IllegalMonitorStateException	การติดต่อกับมอนิเตอร์มีการทำงานผิดพลาด
IndexOutOfBoundsException	ค่าของอินเด็กซ์เกินขอบเขต
InstantiationException	ไม่สามารถเริ่มต้นการทำงานของคลาสได้
NegativeArraySizeException	ขนาดของอาร์เรย์ติดลบ
NoSuchMethodException	ไม่มีวิธีการที่ระบุ
NullPointerException	พอยน์เตอร์ของวัตถุระหว่างอาร์เรย์ชี้ไปยังตำแหน่ง Null
NumberFormatException	รูปแบบตัวเลขไม่ถูกต้อง
RuntimeException	รันไทม์ผิดพลาด
SecurityException	ความปลอดภัยผิดพลาด
StringIndexOutOfboundsException	อินเด็กซ์ของอักขระเกินขอบเขต

ตารางที่ 3 – 9 แสดงชนิดของ Exception ในจาวา

เมื่อมีความผิดพลาดเกิดขึ้น อาจมีความจำเป็นที่จะเรียกใช้โค้ดชุดหนึ่งทุกครั้งเพื่อยกเลิกการใช้งานทรัพยากรหรือปิดไฟล์ ซึ่ง Java มีการรองรับการทำงานแบบนี้ได้โดยการใช้คีย์เวิร์ด finally โค้ดในบล็อค finally จะถูกใช้งานเมื่อมีความผิดพลาดเกิดขึ้นไม่ว่าจะเกิดกรณีใดก็ตาม ดังตัวอย่างการใช้คีย์เวิร์ด try...catch...finally ร่วมกัน

```
try
{
        // โค้ดที่ใช้ในคีย์เวิร์ด Try
}
catch (Exception e)
{
        // เกิดความผิดพลาดในกรณีของ Catch เงื่อนไขผิดพลาด การรองรับผิดพลาด
}
finally
{
        // โค้ดที่จะถูกเรียกใช้ไม่ว่าจะเกิดอะไรขึ้น
}
```

Branching statement

จะเป็นการใช้ break โดยเมื่อพบเครื่องหมาย break ในโปรแกรมเมื่อใด โปรแกรมก็จะหยุดแล้วกลับไปยัง statement ถัดจาก statement ปัจจุบันทันที นอกจากนี้เรายังจะสามารถใช้ break เพื่อใช้กระโดด (jump) ไปยังจุด หรือ statement ที่เราใส่ชื่อไว้ (lable) โดยหลักการใส่ชื่อทำได้ดังตัวอย่าง สมมติต้องการตั้งชื่อ statements ว่า breakToHere ก็สามารถทำได้ดังนี้

breakToHere *: statemensts...*

เมื่อเราต้องการกระโดดไปยัง statements เหล่านั้นก็สามารถทำได้ดังนี้

break ***breakToHere;***

การใช้ return จะเป็นการใช้เพื่อออกจาก method ปัจจุบัน และกระโดดกลับไปสู่ statement ถัดจาก method ที่เรียกใช้นั้น สำหรับการใช้ return นี้ เราสามารถใช้เพื่อคืนค่าหรือไม่คืนค่าก็ได้ สำหรับกรณีที่ต้องการคืนค่ากลับไป ให้ใส่ค่านั้นที่ท้ายของ return ดังตัวอย่าง

return Count;

สำหรับกรณีที่ไม่ต้องการคืนค่าใด ๆ เพียงแต่ต้องการกลับไปสู่การเรียกใช้ method ก็เพียงแต่ใช้ return เฉย ๆ ดังนี้

return;

3.2.12 การใช้อาร์เรย์และสตริง

การใช้อาร์เรย์

การใช้อาร์เรย์ก็เช่นเดียวกันกับการใช้ตัวแปรทั่วๆไป นั่นคือ ก่อนใช้จะต้องมีการประกาศ อาร์เรย์นั้นก่อน โดยรูปแบบการประกาศจะใช้เช่นเดียวกันกับตัวแปรทั่วๆไป โดยบอกให้รู้ว่าเป็นอาร์เรย์ชนิดใด แต่จุดที่แตกต่างจากตัวแปรทั่วๆไป คือ จะต้องบอกให้รู้ว่าเป็นอาร์เรย์ โดยใช้เครื่องหมาย [] เช่น

int[] arrayOfInts;

เป็นการประกาศสร้างตัวแปร arrayOfInts ว่าเป็นอาร์เรย์ของ int หลังจากการประกาศแล้ว ก็จะต้องใช้โอเปอร์เตอร์ new เพื่อที่จะให้มีการ allocate เนื้อที่ในหน่วยความจำให้กับอาร์เรย์นั้นๆ (ตอนแรกที่ประกาศอาร์เรย์ จะยังไม่มีการ allocate เนื้อที่หน่วยความจำแต่อย่างใด) ดังนี้

int[] arrayOfInts = new int[10]

สำหรับรูปแบบการประกาศดังข้างต้นจะเป็นดังนี้

elementType[] arrayName = new elementType[arraySize]

ในการใช้ new จะมีการกำหนดขนาดของอาร์เรย์นั้นให้ด้วย และเมื่อเราใช้โอเปอร์เรเตอร์ new เรียบร้อยแล้ว เราก็สามารถให้ค่า (assign) กับอาร์เรย์เหล่านั้นได้แล้ว ดังเช่นตัวอย่างการให้ค่าอาร์เรย์ข้างล่างนี้

```
for(int j = 0;j<arrayOfInts.length;j++)
{
        arrayOfInt[j] = j;
}
```

นอกจากนี้ อาร์เรย์ยังสามารถเก็บออบเจ็คท์หรืออาร์เรย์ด้วยกันเองได้ด้วย

การใช้สตริง

การใช้สตริงสามารถใช้ได้ 2 รูปแบบได้แก่

String[] args เป็นการใช้โดยการประกาศให้รู้ว่าเป็นสตริง และมีการกำหนดชื่อของสตริงนั้น ซึ่งจากตัวอย่างก็คือ args นั่นเอง

สำหรับอีกวิธีหนึ่งก็คือ การใช้เครื่องหมายอัญประกาศ เช่น "Input has" ในกรณีนี้จะเป็นการกำหนดขนาดของสตริงแน่นอน

3.2.13 การรับข้อมูลมาตรฐาน

ตัวอย่างการรับข้อมูลเข้าแบบมาตรฐาน

```
While (System.in.read() != '\n')          //การรับข้อมูลมาตรฐาน
```

คลาส System เป็นสมาชิกตัวหนึ่งของแพคเกจ java.lang ซึ่งทุกคลาสของภาษา Java ต่างก็เป็นสมาชิกของแพคเกจนี้ไม่โดยตรงก็ทางอ้อมเพราะ java.lang อยู่เหนือสุดของคลาสทุกคลาสและเนื่องจากคลาสนี้ประกอบด้วยคลาสอื่นๆจำนวนมากจึงถูกเรียกเป็นแพคเกจ คลาส System ทำหน้าที่จัดการกับการรับข้อมูลมาตรฐานทั้งเข้าและออก จัดการกับการทำงานของตัวแปรอาร์เรย์ จัดการเรื่องวันเวลาของระบบ และจัดการตัวแปรสิ่งแวดล้อมของระบบเป็นต้น

3.2.14 กระแสข้อมูลเข้ามาตรฐาน

ตัวแปรแบบคลาส System.in ทำหน้าที่เป็นกระแสข้อมูลเข้ามาตรฐาน โดยแนวความคิดของกระแสข้อมูลเข้ามาตรฐานนี้ได้มาจากไลบรารีภาษา C ซึ่งภาษา Java ได้นำมาใช้ด้วย อาศัยหลักการง่ายๆที่เป็นมาตรฐานของระบบคอมพิวเตอร์เมื่อข้อมูลที่ทยอยกันเข้ามาภายในระบบซึ่งก็คือข้อมูลตัวอักษรเปรียบเสมือนกระแสน้ำภายในท่อประปาที่ไหลเข้ามาสู่ถังน้ำในที่นี้ก็คือ บัฟเฟอร์ (Buffer) หรือหน่วยความจำสำหรับเก็บตัวอักษรนั่นเอง และแหล่งที่มาของน้ำประปาอาจจะเป็นก็อกน้ำในห้องน้ำส่วนแหล่งที่มาของข้อมูลของระบบคอมพิวเตอร์ เช่น ข้อมูลจากคีย์บอร์ด ข้อมูลจากไฟล์ ข้อมูลจากอุปกรณ์ระบบเป็นต้น หลักการพื้นฐานนี้เป็นหลักการที่ใช้นานตั้งแต่อดีตซึ่งเป็นระบบตัวอักษร (Text base)

3.2.15 การอ่านข้อมูลจากกระแสข้อมูลเข้ามาตรฐาน

การใช้เม็ทธอด read () จากตัวแปรแบบคลาส System.in อ่านตัวอักษรทีละตัวและคืนค่ารหัสตัวอักษรให้แก่แอพพลิเคชั่น และจะคืนค่า -1 เมื่อไม่มีตัวอักษรจากกระแสข้อมูลเข้าอีกต่อไป

3.2.16 การแสดงข้อมูลมาตรฐาน

การแสดงข้อมูลมาตรฐานเป็นการใช้งานเม็ทธอดแสดงข้อมูลของคลาส System โดยผ่านตัวแปรแบบคลาส System.out ที่จะคอยทยอยเอาข้อมูลอักษรที่เก็บในบัฟเฟอร์เหมือนกระแสน้ำที่ไหลออกจากที่เก็บนั้นส่งไปตามท่อประปาแล้วถูกฉีดออกสู่สนามหญ้า และแน่นอนความหมายของสนามหญ้าในที่นี้ก็คือจอภาพที่แสดงตัวอักษรนั่นเอง

3.3 อ็อปเจ็คท์

อ็อบเจ็คท์ของจาวา จะมีตัวแปรและ method ที่แสดงถึงลักษณะของออบเจ็คท์ต่างๆ ตามหลักการของ OOP โดยการใช้งานออบเจ็คท์จะมีขั้นตอน หรือที่เรียกว่าวงจรชีวิตของออบเจ็คท์ 5 ขั้นตอนด้วยกัน ดังนี้

1. การประกาศอ็อปเจ็คท์
2. การสร้างอ็อปเจ็คท์
3. การกำหนดค่าให้อ็อปเจ็คท์
4. การใช้อ็อปเจ็คท์
5. การกำจัดอ็อปเจ็คท์

3.3.1 การประกาศอ็อปเจ็คท์

การประกาศอ็อปเจ็คท์ คือ การบอกให้คอมพายเลอร์ทราบว่าเราต้องการจะสร้างอ็อปเจ็คท์ขึ้นมาใหม่ตามแบบของคลาสนั้นๆ มีรูปแบบดังนี้

Type name;

type คือ ชื่อของคลาส
name คือ ชื่อของอ็อปเจ็คท์
ตัวอย่างเช่น Date today;

3.3.2 การสร้างอ็อปเจ็คท์

การสร้างอ็อปเจ็คท์ คือ การจองหน่วยความจำไว้เก็บข้อมูลของอ็อปเจ็คท์ ทำได้โดยการใช้ตัวดำเนินการ (Operator) **new** นำหน้าชื่อของคลาสที่ต้องการจะสร้าง และชื่อของอ็อปเจ็คท์ที่ต้องการสร้างจะอยู่ข้างหน้าเครื่องหมายเท่ากับแล้วตามด้วยคำหลัก new รูปแบบของคำหลัก new แสดงได้ดังนี้

```
Type name = new type;
Type name;
Name = new type;
```

type คือ ชื่อของคลาส

name คือ ชื่อของอ็อปเจ็คท์

ตัวอย่างเช่น Date today = new Date();

3.3.3 การกำหนดค่าแรกให้แก่อ็อปเจ็คท์

ในการสร้างคลาสขึ้นมาใหม่ โดยมากแล้วภายในคลาสหนึ่งๆ จะมีเม็ทธอดอยู่อย่างน้อยหนึ่งเม็ทธอด ซึ่งมีชื่อเดียวกันกับคลาสนั้น ทำหน้าที่ส่งค่ากลับคืนสำหรับการสร้างอ็อปเจ็คท์ (Constructor Method) เช่น คลาส Date มีเม็ทธอดการสร้างชื่อ Date() เหมือนกัน และเม็ทธอดตัวนี้นอกจากทำหน้าที่สร้างแล้วยังทำหน้าที่เป็นตัวกำหนดค่าแรกให้แก่อ็อปเจ็คท์ที่จะสร้างขึ้นใหม่อีกด้วย และนอกจากนี้คลาสๆหนึ่งสามารถมีเม็ทธอดการสร้างได้หลายๆเม็ทธอดอีกด้วย เช่น คลาส Date มีเม็ทธอดการสร้างซึ่งเราสามารถกำหนดค่าวันเดือนปีให้แก่อ็อปเจ็คท์ในขณะการสร้างได้โดยการป้อนค่าให้แก่เม็ทธอดการสร้างดังรูปแบบดังต่อไปนี้

```
Date (int year,int month,int day);
```

ในบางครั้งอาจจะเกิดความสงสัยว่า แล้วการที่คลาส Date มีเม็ทธอดซึ่งมีชื่อเหมือนกันหลายๆเม็ทธอด แล้วเวลาทำการกำหนดค่าแรกให้แก่อ็อปเจ็คท์ใหม่ คลาสจะเลือกเม็ทธอดการสร้างตัวใดจึงจะเหมาะสมที่สุดเพื่อมาทำการกำหนดค่าแรกให้แก่อ็อปเจ็คท์ คำตอบก็คือ แต่ละเม็ทธอดการสร้างจะมีชนิดและจำนวนของข้อมูลที่รับเข้ามาหรืออาร์กิวเมนท์ (Argument) แตกต่างกัน ฉนั้นในกรณีเช่นนั้น คลาสจึงสามารถเลือกเม็ทธอดตัวสร้างได้อย่างถูกต้อง ตัวอย่างเช่นการกำหนดอ็อปเจ็คท์ Date today = new Date (); ไม่มีการป้อนข้อมูลให้แก่คลาส Date ฉะนั้นคลาส Date จึงเลือกเม็ทธอดตัวสร้างซึ่งไม่มีการรับข้อมูลเข้ามา (Void) และทำหน้าที่กำหนดค่าวันเวลาในขณะนั้นเป็น Output แก่อ็อปเจ็คท์ใหม่นั่นเอง

3.3.4 การใช้ออบเจ็คท์

การใช้ออบเจ็คท์ในที่นี้ก็คือ การกระทำกับตัวแปรต่างๆนั่นเอง ในที่นี้ เราจะมาดูตัวอย่างการกระทำกับตัวแปรในรูปแบบต่างๆ

3.3.4.1 การอ้างอิงตัวแปรออบเจ็คท์

การอ้างอิงตัวแปรแต่ละตัวของออบเจ็คท์จะใช้รูปแบบดังนี้

ObjectReference.variable

นั่นคือ เราจะใช้เครื่องหมาย . คั่นระหว่างชื่อของออบเจ็คท์นั้นกับตัวแปร สมมติเรามี rect เป็นชื่อของสี่เหลี่ยม ซึ่งได้มีการประกาศและสร้างเรียบร้อยแล้ว และมีตัวแปร x และ y การอ้างถึงแต่ละตัวแปร ก็แทนได้ดังนี้ rect.x และ rect.y เมื่อเราต้องการเปลี่ยนแปลงค่าต่างๆ ของตัวแปร ก็ทำได้เหมือนกับเป็นตัวแปรทั่วๆไป อย่างในกรณีนี้เราต้องการจะเปลี่ยนจุดของสี่เหลี่ยมใหม่ ก็ทำได้ดังนี้

```
rect.x = 15;
rect.y = 50;
```

นอกจากนี้ การให้ค่า (assignment) การใช้เครื่องหมายต่างๆ (operator) ที่กล่าวมาข้างต้นสามารถมาใช้กับออบเจ็คท์นี้ทั้งหมด อย่างเช่นในกรณีนี้ สี่เหลี่ยนมีส่วนกว้างและยาว ให้แทนด้วย rect.width และ rect.height ฉะนั้นเราก็สามารถหาค่าของพื้นที่ของสี่เหลี่ยมนี้ได้ดังนี้

```
area = rect.height*rect.width;
```

3.3.4.2 การเรียกใช้ method ของออบเจ็คท์

ก็เช่นเดียวกันกับการอ้างอิงตัวแปร นั่นคือจะใช้รูปแบบดังนี้

```
ObjectReference.methodName(argumentList); หรือ
ObjectReference.methodName();
```

อย่างเช่นในกรณีที่เราต้องการย้ายสี่เหลี่ยมไปยังจุดใหม่ ก็สามารถใช้ method ชื่อ move() ได้

```
rect.move(200,300);
```

3.3.5 การกำจัดออบเจ็คท์ที่ไม่ต้องการ

เราจะใช้ finalize() ซึ่งเป็น method ที่ใช้เพื่อทำการจบ หรือกำจัดออบเจ็คท์ที่ไม่จำเป็นต้องใช้อีกแล้ว เราเรียกว่า finalization ซึ่งจะทำให้เรามีเนื้อที่ว่างมากขึ้น

3.4 คลาสและอินสแตนซ์

จากที่เราได้เคยกล่าวมาข้างต้นแล้ว่าคลาสทำหน้าที่เป็นต้นแบบ หรือ โปโตไทป์ของออบเจ็คท์ ในการใช้คลาส จะประกอบด้วย 2 องค์ประกอบด้วยกัน คือ

1. ส่วนประกาศคลาส (Class Declaratio)
2. ตัวคลาส (Class Body)

นอกจากนี้ในการใช้คลาส ยังมีองค์ประกอบอื่นๆที่ต้องพิจารณาอีกมาก ซึ่งจะกล่าวต่อไป ดังนี้

3.4.1 การประกาศคลาส (Class Declaration)

สำหรับการประกาศคลาสพื้นฐานจะประกอบด้วยคลาส ซึ่งเป็นคีย์เวิร์ดบอกให้รู้ว่าเป็นคลาส และชื่อของคลาสนั้น เพราะฉะนั้นรูปแบบพื้นฐานของการประกาศออกมาได้ดังนี้

```
class NameOfClass {
        ...
}
```

สำหรับหลักการประกาศคลาส ก็มีดังนี้

1. ชื่อของคลาส ต้องขึ้นต้นด้วยตัวใหญ่
2. บอกให้รู้ว่าซุปเปอร์คลาสนั้นคืออะไร

ตามปกติทุกๆคลาส จะต้องมีซุปเปอร์คลาสของมันเอง ซึ่งถ้าเราไม่ประกาศ โปรแกรมจะคาดว่ามัน
คือออบเจ็คท์คลาส (Object class)

สำหรับการบอกถึงซุปเปอร์คลาส ทำได้โดยการใส่คีย์เวิร์ด extend บวกกับชื่อของซุปเปอร์คลาส โดยเขียนอยู่ระหว่างชื่อคลาสนั้น ๆและเครื่องหมายวงเล็บปีกกาเปิด ({) ซึ่งเราสามารถเขียนรูปแบบของมันได้ดังนี้

```
class NameOfClass extends SuperClassName {
        ...
}
```

การใส่อินเตอร์เฟซที่คลาสนั้นใช้อินเตอร์เฟซเป็นการประกาศเซ็ทของ method และค่าคงที่ โดยที่เราไม่จำเป็นต้องบอกถึงรายละเอียด เนื่องจากมีการกล่าวรายละเอียดอยู่ในอินเตอร์เฟซนั้นแล้ว เมื่อใดที่เราอ้างถึง method ใดๆ ก็เป็นการอ้างอิงไปถึงอินเตอร์เฟซนั้นเอง

สำหรับรูปแบบของการใช้คีย์เวิร์ด implements บวกกับชื่อของอินเตอร์เฟซนั้นๆ เช่น

class ThreadAnimation extends Applet implements Runnable

เป็นการบอกว่า ThreadAnimation เป็นคลาสซึ่งมีซุปเปอร์คลาสชื่อ Applet และใช้อินเตอร์เฟซชื่อ Runnable ซึ่งในอินเตอร์เฟซนี้จะมี method ชื่อ start() และ stop() จึงสามารถสรุปการประกาศคลาสจะมีรูปแบบคร่าวๆดังนี้

[modifier] class ClassName [extends SuperClassName][implements InterfaceNames] {

...

}

สิ่งที่อยู่ภายในเครื่องหมาย [] หมายถึงว่าจะมีหรือไม่มีก็ได้

1. modifier จะประกาศให้รู้ว่าคลาสนี้เป็น public abstract หรือ final
2. ClassName จะเป็นชื่อคลาส
3. SuperClassName เป็นชื่อของซุปเปอร์คลาสของ ClassName นั้นๆ
4. InterfaceName เป็นชื่อของอินเตอร์เฟซที่ใช้ในกรณีที่มีมากกว่า 1 ให้ใส่เครื่องหมาย

ในกรณีที่เราไม่ได้ใส่ส่วนที่อยู่ใน [] คอมไพเลอร์ของจาวาจะกำหนดให้เป็น non-final , non-public, non-abstrace , สับคลาสของออบเจ็คท์ และไม่มีอินเตอร์เฟซใดๆ

3.4.2 ตัวคลาส (Class Body)

ตามรูปแบบของโปรแกรม ตัวคลาสจะอยู่ในส่วนในของโปรแกรม ดังข้างล่างนี้

ClassDeclaration {

ClassBody

}

สำหรับองค์ประกอบของตัวคลาสนี้มี 2 ส่วนใหญ่ๆก็คือ ตัวแปร และ method ซึ่งตามปกติเรามักจะนิยมประกาศตัวแปรก่อน แล้วตามด้วยการประกาศ method ดังรูปแบบข้างล่างนี้

```
ClassDeclaration {
        MemberVariableDeclarations
        MethodDeclarations
}
```

ซึ่งข้างล่างนี้เราจะแสดงถึงตัวอย่างการประกาศตัวแปร และ method ดังนี้

```
class Hello {
        int count;
        String message;
        Void Display(int Value,String Text_message) {
                Count = Value;
                Message = Text_message;
        }
}
```

3.4.3 คอนสตรักเตอร์ (Constructor)

คลาสของจาวาจะมี method พิเศษที่เรียกว่า คอนสตรักเตอร์ ซึ่งใช้ในการ initialize ออบเจ็คท์ใหม่ของคลาสนั้นๆ คอนสตรักเตอร์จะมีชื่อเดียวกับคลาส และคลาสแต่ละคลาส จะมีคอนสตรักเตอร์กี่ตัวก็ได้ และทุกตัวต้องมีชื่อเดียวกันดังตัวอย่างของ Rectangleclass อาจจะมีคอนสตรักเตอร์ดังนี้

```
public Rectangle()
public Rectangle(int width,int height)
public Rectangle(int x,int y,int width,int height)
public Rectangle(Dimention size)
public Rectangle(Point location)
public Rectangle(Point location , Dimension size)
```

ซึ่งแต่ละตัวจะมีค่าอาร์กิวเมนต์ที่แตกต่างกัน และในการประกาศคอนสตรักเตอร์ก็จะมีการกำหนดระดับการเข้าใช้ตัวแปร หรือ method 4 แบบเหมือนดังที่กล่าวมาก่อนหน้านี้ คือ private protected public และ package

3.4.4 ซับคลาส (Subclass) , ซุปเปอร์คลาส (Superclass) และ Inheritance

จากที่ได้กล่าวมาแล้วในส่วนของ OOP ที่ได้บอกเกี่ยวกับสับคลาสที่จะเกิดจากซุปเปอร์คลาส โดยอาศัยคุณสมบัติของ Inheritance ที่จะรับคุณสมบัติทุกอย่างของซุปเปอร์คลาสของมันมานั่นคือ ตัวแปร และ method

3.4.5 การสร้างสับคลาส

เราจะอาศัยการสร้างผ่านการประกาศคลาสที่ได้กล่าวมาแล้วข้างต้น ดังนี้

```
class Subclass extends SuperClass {
        ...
}
```

แต่อย่างไรก็ตามคลาสนี้ก็สามารถอ้างอิงได้เพียงซุปเปอร์คลาสขั้นเดียวเท่านั้น

3.5 อินเตอร์เฟซ (Interface)

อินเตอร์เฟซคือการรวบรวมความหมายของ method และค่าคงที่ต่างๆ เอาไว้อินเตอร์เฟซมีประโยชน์ดังนี้

1. เป็นการเก็บรวบรวมความเหมือนกันของคลาสต่างๆ ที่ไม่เกี่ยวข้องกัน
2. เป็นการประกาศ method ที่ต้องอาศัยคลาสหนึ่งหรือมากกว่าในการทำงาน

ในการสร้างอินเตอร์เฟซขึ้นมาเราต้องกระทำสองอย่างด้วยกัน คือ การประกาศอินเตอร์เฟซ และ Interface Body ดังรูปแบบข้างล่าง

```
InterfaceDeclaration {
        InterfaceBody
}
```

สำหรับการประกาศอินเตอร์เฟซ (Interface Declaration) จะต้องมีองค์ประกอบอย่างน้อยดังนี้

```
interface Countable {
        ...
}
```

สำหรับการตั้งชื่ออินเตอร์เฟซเราจะใช้ชื่อขึ้นต้นเป็นอักษรใหญ่ เช่นเดียวกับชื่อของคลาส นอกจากนี้อินเตอร์เฟซยังสามารถทำได้อย่างคลาสทั่วๆไป ในการใช้คุณสมบัติ Inheritance รับคุณสมบัติจากตัวอินเตอร์เฟซต้น โดยอาศัยคีย์เวิร์ด extends ดังรูปแบบข้างล่างนี้

```
[public] interface InterfaceName [extends listOfSuperInterfaces] {
        ...
}
```

3.6 การเขียนแอพเพล็ท(writing Applets)

สำหรับการเขียนแอพเพล็ทก็เช่นเดียวกับการเขียนจาวา ซึ่งเราต้องอาศัยหลักการพื้นฐานจากการเขียนภาษาจาวา และในแอพเพล็ทก็จะมีองค์ประกอบบางอย่างพื้นฐานที่แตกต่างจากจาวา ซึ่งจะขอกล่าวเป็นหัวข้อย่อย ๆ ดังนี้

1. หลักการพื้นฐานของแอพเพล็ท (Overview of Applets)
2. การสร้าง Applet User Interface
3. การใส่แอพเพล็ทลงในหน้าจอ HTML
4. การติดต่อสื่อสารกับโปรแกรมอื่น ๆ

3.6.1 หลักการพื้นฐานของแอพเพล็ท(Overview of Applets)

3.6.1.1 หลักการพื้นฐานของแอพเพล็ท

ก่อนที่จะเริ่มหลักพื้นฐานใด ๆ ของแอพเพล็ท ก็จะขอยกตัวอย่างแอพเพล็ทที่แสดงถึงสิ่งที่แอพเพล็ทสามารถทำได้ ดังนี้

```
import java.applet.Applet;
import java.awt.Graphics;
public class Simple extends Applet{
  StringBuffer buffer;
        public void init(){
                buffer = new StringBuffer();
                addItem("initializing");
        }
        public void start(){
```

```
                addItem("Starting... ");
        }
        public void stop(){
                addItem("Stopping");
        }
        public void destroy(){
                addItem("Preparing for Unloading...");
        }
        void addItem(String newWord){
                System.out.println(newWord);
                buffer.append(newWord);
                repaint();
        }
        public void paint(Graphics g){
                g.drawRect(0,0size().width-1,size().height-1);
                g.drawString(buffer.toString(),5,15);
        }
}
```

สำหรับการเขียน และหลักพื้นฐานทั่ว ๆไปของการใช้แอพเพล็ท จะกล่าวถึงในรายละเอียดดังต่อไปนี้

3.6.1.2 วงจรชีวิตพื้นฐานของแอพเพล็ท(Life Cycle of an Applet)

วงจรชีวิตพื้นฐานในที่นี้ หมายถึง วงจรที่แอพเพล็ทต้องทำงานด้วย method ต่าง ๆ ที่มันอ้างอิงถึง ซึ่งตามปกติจะมีการทำงานวนเวียนดังนี้

1. initialize ตัวมันเอง
2. เริ่มรันแอพเพล็ท
3. หยุดรันแอพเพล็ท
4. ทำลายตัวเอง เพื่อเตรียมพร้อมที่หยุดการทำงาน จนกว่าจะมีการเรียกใช้ใหม่

นั่นคือ มันจะเริ่มต้นที่การ Initialize ตัวเอง และจบลงด้วยการทำลายตัวเอง แต่ในขณะที่ทำงานบางครั้งเราทิ้งหน้าจอไปสักครู่ แล้วค่อยกลับมาดูอีกครั้งหนึ่ง เราจะพบว่า แอพเพล็ท จะไม่ทำลายตัวเอง

เพียงแต่ละหยุดการรันแอพเพล็ทเท่านั้น และเมื่อเรากลับมาดูอีกครั้งก็จะเริ่มมันใหม่ โดยไม่จำเป็นต้อง Initialize ตังเองอีกครั้ง ซึ่งจะทำให้การรันแอพเพล็ทเร็วขึ้นแต่อย่างไรก็ตาม มันก็ต้องใช้ทรัพยากรสิ้นเปลืองเช่นกัน

สำหรับวงจรชีวิตที่กล่าวถึงข้างต้น จะต้องอาศัย method พื้นฐานนี้

1. **init**

เม็ทธอด init() จะถูกเรียกเมื่อแอพเพล็ทเริ่มทำงานครั้งแรกเท่านั้น ส่วนมากแล้วเราจะใช้เม็ทธอดนี้เพื่อจัดเตรียมองค์ประกอบเริ่มต้นต่างๆที่จำเป็นให้พร้อมต่อการทำงาน เช่น การกำหนดค่าเริ่มต้นให้แก่ตัวแปร การโหลดไฟล์ภาพไฟล์เสียง ซึ่งกิจกรรมเหล่านี้บางครั้งถ้าไปทำในขณะที่ทำงานอยู่อาจจะทำให้การทำงานชักช้า ฉะนั้นกิจกรรมใดที่เราคิดว่าต้องทำเพียงครั้งเดียวหรือควรจัดการก่อนการเข้าสู่การทำงานจริงก็ควรจะจัดการในเม็ทธอด init() นี้

```
public void init()
```

2. **start**

เม็ทธอด start() จะถูกเรียกหลังจากเม็ทธอด init() (ซึ่งถูกเรียกเพียงครั้งเดียวตอนเริ่มต้นเท่านั้น) และจะทำงานทุกครั้งเมื่อแอพเพล็ทถูกเรียกกลับมาให้ทำงาน เช่น เมื่อเราสั่งให้ Netscape Navigator กลับไปโหลดโฮมเพจหน้าก่อนหน้านี้ (Back) แล้วเราสั่งให้กลับมาที่หน้าเดิมซึ่งมีแอพเพล็ทของเราทำงานอยู่ (forward) แอพเพล็ทก็จะได้รับเหตุการณ์ start อีกครั้งหนึ่ง

```
public void start()
```

3. **stop**

เม็ทธอด stop() จะทำงานตรงกันข้ามกับเม็ทธอด start() คือจะถูกเรียกเมื่อออกจากโฮมเพจหน้าซึ่งมีแอพเพล็ทนั้นๆทำงานอยู่เพื่อไปทำงานที่โฮมเพจหน้าอื่น และเมื่อเรากลับมาทำงานที่หน้าโฮมเพจหน้าเดิม เม็ทธอด start() ก็จะถูกเรียกใช้งานอีกครั้งหนึ่ง และสุดท้ายเม็ทธอด stop() จะถูกเรียกใช้งานก่อนเม็ทธอด destroy() ทุกครั้ง

```
public void stop()
```

4. **destroy**

เม็ทธอด destroy() จะถูกเรียกหลังจากเม็ทธอด stop() ทุกครั้ง เพื่อทำการสะสางทรัพยากรต่างๆที่ใช้งานก่อนหน้านี้ เช่น ยกเลิกการติดต่อกับระบบเน็ตเวิร์คใดๆ ที่เรากำลังติดต่ออยู่ หากเรามีกิจกรรมใดๆที่จะต้องทำก่อนจบการทำงานของแอพเพล็ทก็ควรจะจัดการในเม็ทธอดนี้

```
public void destroy()
```

3.6.1.3 method สำหรับใช้ใน Drawing และ Event Handling

method พื้นฐานที่เราใช้ในการแสดงผลออกสู่หน้ามี 2 อย่าง คือ

1.paint

เม็ทธอด paint() จะทำงานเมื่อมีความต้องการที่จะวาดภาพภายในพื้นที่แสดงผลของแอพเพล็ทใหม่ และเม็ทธอด paint() นี้คือเม็ทธอดเดียวที่เราสามารถใช้เม็ทธอดทางกราฟฟิกที่ทำหน้าที่วาดภาพหรือวาดตัวอักษรบนพื้นที่แสดงผลของแอพเพล็ท

```
public void paint(Graphics g)
```

2.update

เม็ทธอด update() ทำหน้าที่คล้ายเม็ทธอด paint() คือจัดการกับการแสดงผลในแต่ในแนวทางของการปรับปรุงภาพ และจะแตกต่างกันเมื่อเวลาที่มีการวาดภาพใหม่เมื่อเรียกเม็ทธอด update() จะไม่ทำการลบพื้นที่แสดงผลใหม่ โดยเม็ทธอด update() เมื่อถูกเรียกใช้งานจะไปเรียกให้เม็ทธอด paint() ให้ทำงานอีกต่อหนึ่ง ฉะนั้นเมื่อเราต้องการที่จะปรับปรุงส่วนใดๆบนพื้นที่การแสดงผล เราก็จะจัดการอยู่ภายในเม็ทธอดนี้

```
public void update()
```

3.6.1.4 method สำหรับการตอบสนองเหตุการณ์

method พื้นฐานสำหรับการตอบสนองเหตุการณ์ต่างๆ ประกอบด้วย

1.mouseDown

เม็ทธอด mouseDown() จะถูกเรียกเมื่อมีการกดปุ่มเม้าส์ภายในพื้นที่แสดงผลของแอพเพล็ทและภายในเม็ทธอด mouseDown() ยังรับข้อมูลของตำแหน่งของเม้าส์อีกด้วยคือ (x,y)

```
Public boolean mouseDown(Event evt,int x,int y)
```

2.mouseUp

เม็ทธอด mouseUp() ทำงานตรงข้ามกับเม็ทธอด mouseDown() โดยจะถูกเรียกเมื่อปุ่มของเม้าส์ถูกปล่อยหลังจากการกด และก็จะได้รับตำแหน่งของเม้าส (x,y) เช่นกัน

```
Public boolean mouseUp(Event evt,int x,int y)
```

3.mouseMove

เม็ทธอด mouseMove() จะถูกเรียกเมื่อมีการเลื่อนตำแหน่งของเม้าส์ภายในพื้นที่แสดงผลของแอพเพล็ท โดยไม่มีการกดปุ่มเม้าส์

```
Public boolean mouseMove(Event evt,int x,int y)
```

4.mouseDrag

เม็ทธอด mouseDrag() ทำงานคล้ายกับเม็ทธอด mouseMove() จะแตกต่างกันตรงที่เหตุการณ์นี้จะเกิดขึ้นเมื่อมีการกดปุ่มของเม้าสในขณะที่เคลื่อนเม้าสไปด้วย

```
Public boolean mouseDrag(Event evt,int x,int y)
```

5.mouseEnter

เม็ทธอด mouseEnter() จะทำงานเมื่อเราเลื่อนเม้าสจากภายนอกพื้นที่แสดงผลของแอพเพล็ทเข้ามาสู่ภายในบริเวณของพื้นที่แสดงผลของแอพเพล็ท

```
Public boolean mouseEnter(Event evt,int x,int y)
```

6.mouseExit

เม็ทธอด mouseExit() จะทำงานเมื่อตำแหน่งของเม้าสหลุดออกนอกพื้นที่ของแอพเพล็ทซึ่งตรงข้ามกัลเม็ทธอด mouseEnter(0 และค่าอาร์กิวเมนท์อินพุทของเม็ทธอด mouseExit() นี้ก็คือค่าตำแหน่งสุดท้ายของเม้าส์นั่นเอง

```
Public boolean mouseExit(Event evt,int x,int y)
```

7. keyDown

เม็ทธอด keyDown() จะทำงานเมื่อมีการกดปุ่มใดๆของคีย์บอร์ดในขณะที่แอพเพล็ทนั้นทำงานอยู่และจะได้รับอินพุทเป็นค่ารหัสคีย์บอร์ดที่กดด้วยในรูปของข้อมูลชนิดจำนวนเต็มฐานสิบ

```
Public boolean keyDown(Event evt,int key)
```

8.keyUp

ในทำนองเดียวกันกับเม้าสเมื่อมีการกดปุ่มก็ต้องมีการปล่อยปุ่ม สำหรับคีย์บอร์ดก็เช่นเดียวกัน เม็ทธอด keyUp() ทำหน้าที่ตรงกันข้ามกับเม็ทธอด keyDown() คือจะทำงานเมื่อมีการปล่อยปุ่มคีย์บอร์ดใดๆหลังจากที่กดปุ่มนั้นแล้ว ในขณะที่แอพเพล็ทนั้นๆทำงานอยู่

```
Public boolean keyUp(Event evt,int key)
```

3.6.2 การสร้างระบบติดต่อผู้ใช้ หรือ User interface

ระบบติดต่อผู้ใช้ (The User interface system) คือส่วนประกอบหนึ่งของโปรแกรมคอมพิวเตอร์ ซึ่งทำหน้าที่เป็นสื่อกลางสำหรับโต้ตอบกับผู้ใช้ ในความหมายของการโต้ตอบคือการรับคำสั่งและการแสดงผลลัพธ์ ระบบติดต่อผู้ใช้มีอยู่ด้วยกันหลายรูปแบบตั้งแต่อดีตจนถึงปัจจุบัน แต่ที่เห็นเด่นได้ชัดนั้นมีอยู่ 2 แบบคือ แบบรับคำสั่งทางคีย์บอร์ดแล้วแสดงผลเป็นตัวอักษร (Command-line interface) และอีกแบบหนึ่งคือแบบ"ชี้แล้วกด " (Point-and-Click) โดยการใช้เมาส์สั่งงานร่วมกับการแสดงผลแบบกราฟฟิก (Graphic User Interface หรือ GUI) ซึ่งระบบติดต่อกับผู้ใช้แบบนี้เป็นที่นิยมใช้กันในโปรแกรมประยุกต์สมัยใหม่ เพราะค่อนข้างเป็นมาตรฐานใช้งานได้ง่ายกว่าแบบแรกและไม่เสียเวลาในการเรียนรู้มากมายนัก

การทำงานในระดับล่างของระบบติดต่อกับผู้ใช้ ระบบปฏิบัติการ (Operating System หรือ OS) จะส่งข้อมูลอินพุทจากคีย์บอร์ดและเม้าส์ไปให้แก่โปรแกรมเพื่อประมวลผลและแสดงผลข้อมูลที่เอ้าท์พุทในรูปของจุดสีหรือจุดภาพ (Pixel) ซึ่งจะประกอบกันขึ้นเป็นตัวอักษรหรือรูปภาพบนจอภาพ AWT ได้ถูกออกแบบให้โปรแกรมเมอร์หรือผู้พัฒนาแอพพลิเคชั่นหรือแอพเพล็ท ไม่ต้องมาเป็นกังวลกับงานระดับล่าง เช่นการจัดการกับ Interrupt ของเม้าส์หรือคีย์บอร์ดหรือแม้กระทั่งกับรายละเอียดในการเขียนจุดสีบนจอภาพ

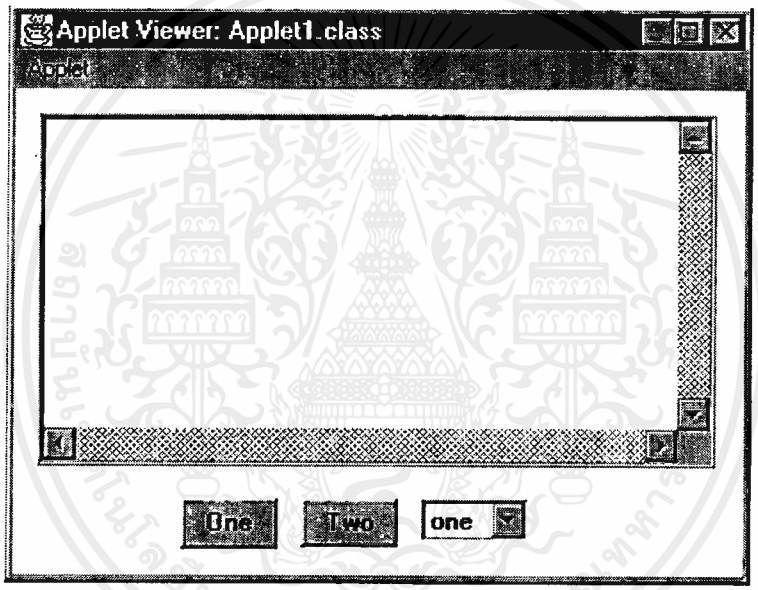
ด้วยเหตุผลที่จาวาเป็นภาษาคอมพิวเตอร์ที่ไม่ขึ้นกับระบบใด ฉะนั้นการปฏิบัติงานของ AWT ซึ่งเป็นส่วนหนึ่งของจาวาก็ต้องเป็นอิสระต่อระบบใดๆ เช่นกัน AWT ได้ถูกออกแบบให้มีเครื่องไม้เครื่องมือที่ใช้สำหรับติดต่อกับผู้ใช้ซึ่งสามารถปฏิบัติงานกับระบบปฏิบัติการหลายๆระบบได้ (เช่น Windows , MacOS , Solaris , Xwindows , OS/2 , ฯลฯ) โดยการใช้เครื่องมือดั้งเดิมของระบบปฏิบัติการนั้นๆ เพื่อรักษารูปร่างหน้าตาของเครื่องมือจะแตกต่างกันบ้างเมื่อทำงานอยู่บน OS ที่ต่างระบบกัน แต่สิ่งนี้ไม่ถือเป็นอุปสรรคที่ใหญ่โตเพราะการทำงานพื้นฐานเหมือนกันทุกประการ

คอมโพเนนท์และคอนเทนเนอร์

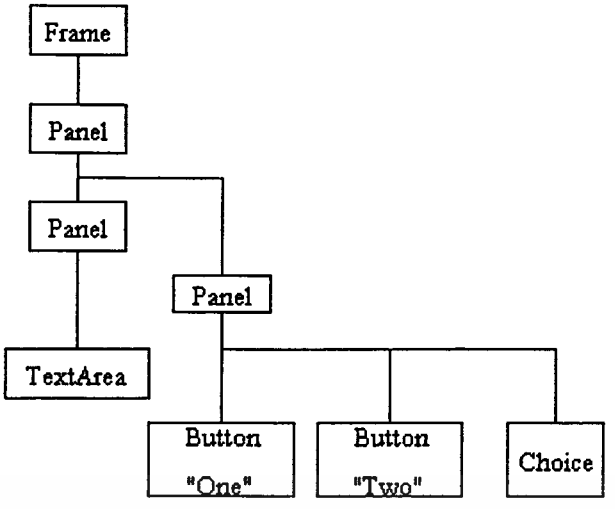
ระบบติดต่อผู้ใช้ถูกสร้างขึ้นจากส่วนประกอบย่อยๆ ที่เรียกกันว่า "คอมโพเนนท์" (Component) ตัวอย่างของคอมโพเนนท์ที่พบเห็นโดยทั่วไปเช่น ปุ่มกด แถบเลื่อน ช่องเติมตัวอักษร กล่องรายการ ฯลฯ คอมโพเนนท์คือส่วนที่ทำให้ผู้ใช้ทำการโต้ตอบกับโปรแกรม (การสั่งงาน) และคือส่วนที่ทำให้ผู้ใช้ทราบสถานะการปฏิบัติงานของโปรแกรม (การแสดงผล) ภายใน AWT คอมโพเนนท์ทุกๆคอมโพเนนท์ก็คือ "อินสแตนซ์" (Instance) หรือตัวตนหนึ่งๆของคลาส Component หรือซับไทป์ (Subtype) ของคลาส Component

คอมโพเนนท์ไม่ใช่เครื่องมือที่ทำงานอยู่อย่างโดดเดี่ยวอย่างอิสระแต่จะอยู่ภายใต้และถูกควบคุมโดย "คอนเทนเนอร์" (Container) ซึ่งคือที่สำหรับบรรจะคอมโพเนนท์หรือแม้กระทั่งบรรจุคอนเทนเนอร์ด้วยกัน อันเนื่องมาจากตัวคอนเทนเนอร์เองบางครั้งก็ทำหน้าที่เป็นคอมโพเนนท์เช่นกัน และคอนเทนเนอร์ทุกๆคอนเทนเนอร์ก็คืออินสแตนซ์ของคลาส Container หรือซับไทป์ของคลาส Container

คอมโพเนนท์จำเป็นต้องอยู่ภายในคอนเทนเนอร์ ซึ่งกลุ่มของคอนเทนเนอร์ (หรือคอนเทนเนอร์ด้วยกัน) ที่บรรจุอยู่ภายในคอนเทนเนอร์เหล่านี้ เปรียบเสมือนกับการสร้างโครงสร้างแบบรากต้นไม้ของคอมโพเนนท์ โดยที่คอนเทนเนอร์บนสุดเปรียบเสมือนกับรากแก้วซึ่งจะแตกแขนงรากย่อยๆ ออกไปในที่นี้ก็คือคอมโพเนนท์ ดังรูปที่ 3 – 4 จะแสดงระบบติดต่อกับผู้ใช้บน Windows95 และดังรูปที่ 3 – 5 จะแสดงโครงสร้างแบบรากต้นไม้ของระบบติดต่อผู้ใช้ของรูปที่ 3 – 4



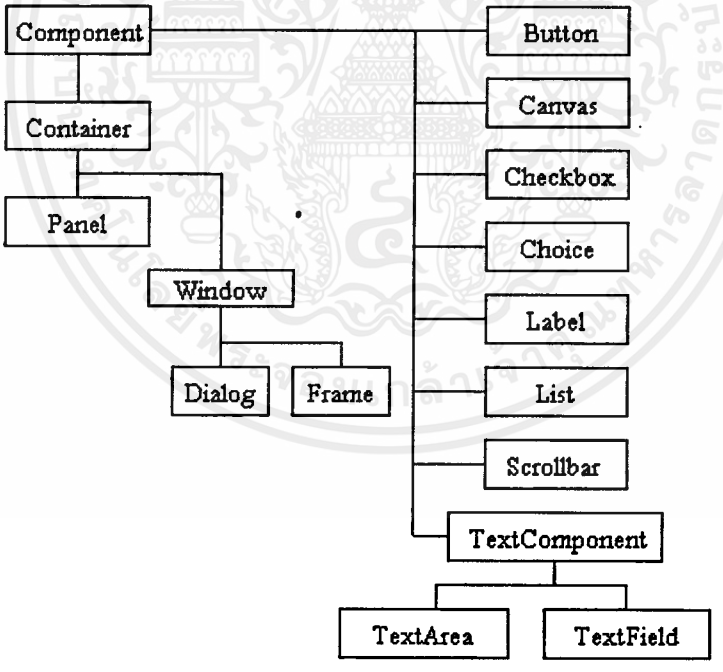
รูป 3 – 4 แสดงระบบติดต่อผู้ใช้ของ AWT บน Windows95



รูปที่ 3 – 5 แสดงโครงสร้างแบบรากต้นไม้ของระบบเชื่อมต่อจากรูปที่ 3 - 4

ชนิดของคอมโพเนนท์

ความสัมพันธ์แบบถ่ายทอดระหว่างคลาสต่างๆของ AWT จะแสดงได้ดังรูปที่ 3 - 6



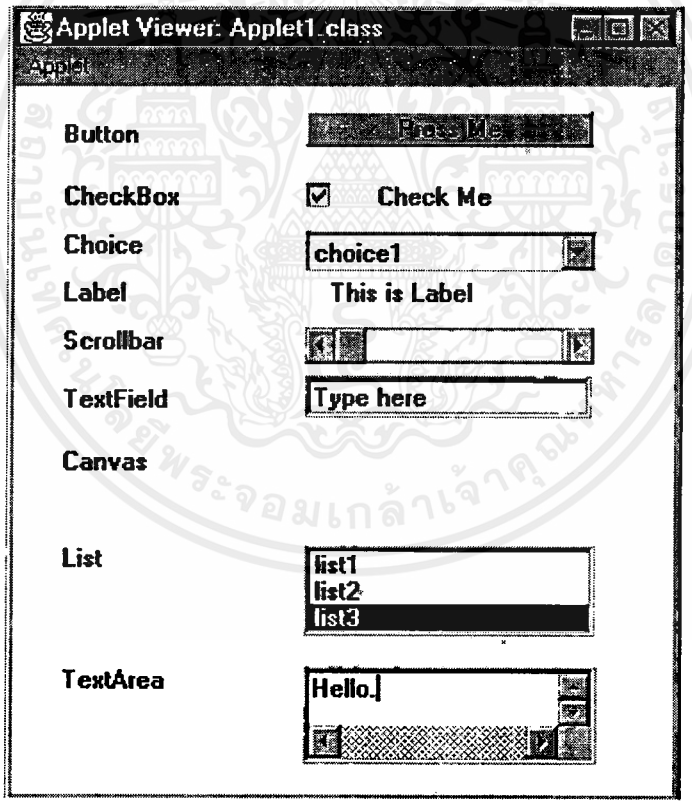
รูปที่ 3 – 6 แสดงความสัมพันธ์แบบถ่ายทอดของ AWT

คอมโพเนนท์พื้นฐานซึ่งไม่สามารถเป็นคอนเทนเนอร์ได้มีอยู่ 10 คลาสคือ

- Buttons(java.awt.Button) ปุ่มกด

- Checkboxes(java.awt.Checkbox) หรือช่องเลือก
- Single-line text fields(java.awt.TextField) หรือช่องเติมตัวอักษรบรรทัดเดียว
- Larger text display and editing areas(java.awt.TextArea) หรือช่องเติมตัวอักษรหลายบรรทัด
- Labels(java.awt.Label) หรือแถบแสดงตัวอักษร
- Lists(java.awt.List) หรือ กล่องรายการหลายบรรทัด
- Pop-up lists of choices(java.awt.Choice) หรือกล่องรายการบรรทัดเดียว
- Sliders and scrollbars(java.awt.Scrollbar) หรือแถบเลื่อน
- Drawing areas(java.awt.Canvas) หรือพื้นที่วาดภาพ
- Menus(java.awt.Menu,java.awt.MenuItem,Java.awt.checkboxMenuItem) หรือเมนู

นอกจากนั้นเรายังสามารถสร้างคอมโพเนน ท์ใหม่ขึ้นได้โดยการสืบทอดคุณสมบัติมาจากคอมโพเนนท์พื้นฐานต่างๆเหล่านี้ ดังรูปที่ 3 – 7 แสดงอินสแตนซ์ของแต่ละคลาส



รูปที่ 3 – 7 แสดงรูปร่างหน้าตาของคอมโพเนนท์พื้นฐาน

ชนิดของคอนเทนเนอร์

คอนเทนเนอร์มีอยู่ 4 ชนิดคือคลาส Window ซึ่งประกอบด้วยซับไทป์อีก 2 คลาสคือคลาส Frame และคลาส Dialog สำหรับคลาส Panel เป็นคลาสที่อยู่ระดับเดียวกันกับคลาส Window และนอกจากนั้น คลาส Applet ถือเป็นคอนเทนเนอร์ชนิดหนึ่งซึ่งได้สืบทอดมาจากคลาส Panel คลาส Applet จึงสามารถบรรจุคอมโพเนนท์ต่างๆ ได้ดังรูปที่ 3 – 4 และในตารางที่ จะแสดงคุณสมบัติของคลาสคอนเทนเนอร์ทั้ง 4

คลาส	คุณสมบัติ
Window	เป็นพื้นที่การแสดงผลระดับสูงสุด (a window) อินสแตนซ์ของคลาส Window จะไม่สามารถถูกเพิ่มเติมหรือฝังตัวภายในคอนเทนเนอร์ใดๆ ได้ รูปร่างหน้าตาของคลาส Window จะไม่แสดงกรอบและหัวชื่อเรื่องของวินโดว
Frame	เป็นพื้นที่การแสดงผลระดับสูงสุด (a Window) มีการแสดงกรอบและแถบชื่อของวินโดว อินสแตนซ์ของคลาส Frame อาจจะประกอบไปด้วยแถบเมนู การทำงานโดยทั่วไปจะเหมือนกับอินสแตนซ์ของคลาส Window
Dialog	เป็นพื้นที่การแสดงผลระดับสูงสุด (a Window) มีการแสดงกรอบและแถบชื่อของวินโดวอินสแตนซ์ของคลาส Dialog จะเกิดขึ้นไม่ได้ถ้าปราศจากการเชื่อมต่อกับอินแสตนซ์ของคลาส Frame
Panel	คอนเทนเนอร์ทั่วไปสำหรับบรรจุคอมโพเนนท์ โดยการใช้เม็ทธอด add () เพื่อเพิ่มเติมคอมโพเนนท์ให้แก่อินสแตนซ์ของคลาส Panel

ตารางที่ 3 – 10 แสดงคุณสมบัติของคลาสคอนเทนเนอร์

การสร้างคอนเทนเนอร์

การที่จะเพิ่มเติมคอมโพเนนท์ใดๆ เพื่อสร้างระบบติดต่อกับผู้ใช้นั้น เราในฐานะของโปรแกรมเมอรจำเป็นต้องสร้างคอนเทนเนอร์เพื่อเป็นที่เก็บคอมโพเนนท์เสียก่อน เมื่อทำการสร้างโปรแกรม “จาวาแอพพลิเคชั่น” สิ่งแรกที่เราต้องทำคือการสร้างอินสแตนซ์ของคลาส Window หรือคลาส Frame แต่สำหรับ “จาวาแอพเพล็ท” นั้นคลาส Frame ได้มีอยู่แล้วซึ่งก็คือวินโดวของเว็บบราวเซอร์นั่นเอง เนื่องมาจากคลาส Applet เป็นซับไทป์ของคลาส Panel ฉะนั้นเราจึงสามารถที่จะเพิ่มเติมคอมโพเนนท์ต่างๆให้แก่คลาส Applet ได้เลย

การเพิ่มเติมคอมโพเนนท์ให้แก่คอนเทนเนอร์

คอนโพเนนท์สามารถถูกเพิ่มแก่คอนเทนเนอร์ได้โดยการใช้เม็ทธอด add() การใช้งานของเม็ทธอด add() สามารถทำได้ 3 วิธีซึ่งจะขึ้นอยู่กับโครงร่างการวางตำแหน่งของคอมโพเนนท์ (Component Layout)

การวางตำแหน่งคอมโพเนนท์

โครงร่างหรือการวางตำแหน่งของคอมโพเนนท์ภายในคอมโพเนนท์นั้นจะไม่ได้ถูกควบคุมโดยตัวคอนเทนเนอร์เอง แต่จะถูกควบคุมจาก "เลย์เอาท์แมนเนเจอร์" (Layout Manager) ซึ่งทำหน้าที่เกี่ยวข้องโดยตรงกับตัวคอนเทนเนอร์ตัวนั้น เลย์เอาท์แมนเนเจอร์คือตัวที่จะตัดสินใจว่าจะให้คอมโพเนนท์นั้นๆ อยู่ในตำแหน่งใดๆ ในคอนเทนเนอร์

การจัดการโครงร่างของเลย์เอาท์แมนเนเจอร์มีทั้งหมดอยู่ 5 รูปแบบ จากรูปแบบง่ายๆธรรมดาๆ ไปจนถึงแบบที่ซับซ้อนมากๆ คือ FlowLayout , BorderLayout

การทำงานของคลาส FlowLayout จะวางตำแหน่งของคอมโพเนนท์จากซ้ายไปขวาในแนวคอลัมน์ (แนวนอน) ก่อนแล้วจะเริ่มแถวใหม่ลงมาเมื่อมีคอมโพเนนท์เกินออกมาในแถวนั้น การใช้เม็ทธอด add() สำหรับการเพิ่มหรือสร้างคอมโพเนนท์ใหม่จะใช้ "อาร์กิวเมนท์" (Argument) หรืออินพุทของเม็ทธอดเพียงหนึ่งตัวเท่านั้นซึ่งก็คือชื่อของคอมโพเนนท์

สำหรับคลาส BorderLayout ได้จัดแบ่งสัดส่วนพื้นที่จะมีชื่อเฉพาะคือ North (ด้านบน) , South (ด้านล่าง) , West (ด้านซ้าย) , East (ด้านขวา) และ Center (กึ่งกลาง) แต่ละคอมโพเนนท์ก็จะถูกจัดวางตำแหน่งภายในพื้นที่เหล่านี้ จุดเด่นของโครงร่างแบบนี้ก็คือ แต่ละพื้นที่จะรักษาสัดส่วนของตัวเองไว้เสมอ แม้ว่ามีการเปลี่ยนแปลงขนาดของคอนเทนเนอร์ไปในรูปแบบใดก็ตาม พื้นที่ที่จะมีขนาดกว้างใหญ่ที่สุดคือบริเวณ Center อาร์กิวเมนท์ที่จะให้กับเม็ทธอด add() จะมีอยู่ 2 ตัวคือ ชื่อของอ็อปเจ็คท์หรือคอมโพเนนท์นั้น และชื่อของพื้นที่ที่จะมีการวางตำแหน่งคอมโพเนนท์

คลาสที่เป็นคอนเทนเนอร์แต่ละแบบจะมีเลย์เอาท์แมนเนเจอร์โดยปริยาย (default) เช่นคลาส Frame และคลาส Dialog จะใช้เลย์เอาท์แมนเนเจอร์แบบ BorderLayout ส่วนเลย์เอาท์แมนเนเจอร์โดยปริยายของคลาส Panel และคลาส Applet คือแบบ FlowLayout

สำหรับคลาส Grid เป็นส่วนแสดงผลของแอพเพล็ทภายในกริดของคอลัมน์และแถว แต่ละส่วนควบคุมของยูสเซอร์อินเตอร์เฟซจะถูกแทนที่ในเซลล์ภายในกริดจากซ้ายไปขวา และจากบนลงล่าง เลย์เอาท์แมนเนเจอร์จะจัดส่วนควบคุมที่ถูกต้องไว้ โดยที่ส่วนประกอบของยูสเซอร์อินเตอร์เฟซจะปรากฏภายในแอพเพล็ท

สำหรับคลาส Card ใช้ทำที่ว่างบนจอได้ดีกว่า ใช้ตัวเลย์เอาท์แมนเนเจอร์เพื่อเลือกลำดับของส่วนควบคุมที่จะแสดงในเวลาต่างๆกัน เมื่อใส่ส่วนควบคุมเพิ่มไปยังเลย์เอาท์แบบ Card มันจะไม่แสดงส่วนควบคุมทั้งหมดพร้อมกัน แต่ละเลือกแสดงส่วนควบคุมได้โดยใช้ตัวเลย์เอาท์แมนเนเจอร์ ซึ่งจะใช้ในกรณีที่ต้องการยูสเซอร์อินเตอร์เฟซที่ซับซ้อน

สำหรับคลาส GridBagLayout คือเลย์เอาท์แมนเนเจอร์พิเศษ และอนุญาตให้แอพเพล็ตใส่ลำดับความสาคัญให้แต่ละส่วนควบคมภายในแอพเพลัตได ถ้าแอพเพลัตถกเปลี่ยนขนาดอีกครั้ง ส่วนควบคมที่มลาดบความสาคญมากทสดจะครอบครองพนททหนาจอมากขน

- ดงนน method พนฐานทใชในการเรยกใชระบบตดตอผใช ไดแก
- add() เปนการเพม UI ตาง ๆ
- remove() เปนการเอา UI ออกไป

- setLayout() เป็นการใช้งานกับ layout manager ซึ่งเป็นการจัดหน้าจอ

3.6.3 การใส่แอพเพล็ทลงในหน้าจอของ HTML

เราสามารถทำโดยการใส่ tag <APPLET> ลงในโค้ดของ HTML ในที่ขอยกรูปแบบมาตราฐานของการเขียนโค้ด ดังนี้

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
</APPLET>
```

จากตัวอย่างข้างต้น เราจะเห็นองค์ประกอบพื้นฐานต่อไปนี้

1. เริ่มต้นด้วยการประกาศเป็น APPLET-tag
2. ตามด้วยการบอกถึงไฟล์ที่จะใช้ในการรัน ซึ่งจะต้องเป็นไฟล์ที่คอมไพล์เรียบร้อยแล้ว นั่นคือเป็นไฟล์ .class จากนั้นก็บอกถึงขนาดความกว้างอละความยาวของหน้าจอ หรือพื้นที่ที่ต้องการให้แสดงแอพเพล็ทนั้น ๆ (WIDTH และ HEIGHT)
3. จากนั้นก็ปิดท้ายด้วย </APPLET>
4. นอกจากนี้ เรายังมีฟังก์ชันที่บอกถึงที่อยู่ในไฟล์ไบต์โค้ดนั้นด้วย ซึ่งมีรูปแบบดังนี้

```
<APPLET CODE=AppletSubclass.class CODEBASE=aURL
        WIDTH=anInt HEIGHT=anInt>
</APPLET>
```

ซึ่งในส่วนของ CODEBASE นี้ นอกจากจะใช้เป็น URL แล้ว ยังสามารถแทนด้วยที่อยู่ไดเร็กทอรี่ได้ด้วย และในกรณีที่มีการรับค่าเข้าสู่แอพเพล็ท ก็สามารถทำได้ผ่าน <PARAM>ตามรูปแบบ

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
<PARAM NAME=parameter1Name VALUE=aValue>
<PARAM NAME=parameter2Name VALUE=anotherValue>
</APPLET>
```

สำหรับการเขียนจริง ๆ ไม่จำเป็นที่จะต้องเป็นตัวใหญ่ทั้งหมด ดังตัวอย่างข้างล่าง

```
<applet code=AppletButton.class codebase=example width=350 height=60>
<param name=windowTypevalue=BorderWindow>
<param name=windowtextvalue="BordreLayout">
```

```
<param name=buttonTextvalue="Click here to see a BorderLayout in action">
<blockquote>
<hr>
<em>
Your browser can't run 1.0 Java appets,
so here's a picture of the window the program brins up:</em>
<p>
<img src=images/BorderEx1.gif width=302 height=138>
<hr>
</blockquote>
</applet>
```

จากตัวอย่างเราจะเห็น <blockquote>-tag ซึ่งใช้เป็นตัวบอกในกรณีที่บราวเซอร์นั้นไม่สามารถรันแอพเพล็ทนั้นไ ก็จะข้ามบล็อกนี้ไป

3.6.4 การติดต่อสื่อสารกับโปรแกรมอื่นๆ

แอพเพล็ทต่างๆสามารถติดต่อกับโปรแกรมอื่นๆได้ใน 3 ทางดังนี้

1. โดยการเรียกใช้ method ของแอพเพล็ทอื่นบนเพจเดียวกัน
2. โดยการใช้ API ซึ่งจะมี method ที่ช่วยให้แอพเพล็ทสามารถติอต่อกับบราวเซอร์ หรือ Applet Viewer ที่เก็บแอพเพล็ทนี้
3. โดยการใช้ API ช่วยให้สามารถติดต่อกับโปรแกรมอื่นๆ ผ่านเน็ตเวิร์ก โดยจุดสำคัญก็คือจะต้องอยู่ในเน็ตเวิร์กที่ใช้โฮสต์เดียวกัน

สำหรับการติดต่อสื่อสารจะมีหัวข้อย่อยๆ ที่จะต้องพิจารณาดังที่กล่าวมาข้างต้น ดังนี้

การส่ง message ไปยังแอพเพล็ทอื่นๆบนเพจเดียวกัน

ปกติแอพเพล็ทจะพยายามค้นหาแอพเพล็ทอื่นๆ และส่ง message ไปยังแอพเพล็ทนั้น โดยตั้งอยู่บนพื้นฐานทางด้านการรักษาความปลอดภัยดังนี้

1. แอพเพล็ทจะต้องรันอยู่ในเพจเดียวกัน และบนบราวเซอร์ตัวเดียวกัน
2. มี Applet Viewer หลายตัวที่ต้องการให้แอพเพล็ทเหล่านั้นเกิดจากเซิร์ฟเวอร์เดียวกันด้วย

การค้นหาแอพเพล็ทอื่น สามารถทำได้โดยใช้ method ชื่อ getApplet() โดยการหาอาจหาโดยหาตามรายชื่อแอพเพล็ท หรือการค้นหาทั้งหมดเลยก็ได้ ซึ่งตามปกติแอพเพล็ทจะไม่มีชื่อ การที่แอพเพล็ทจะมีชื่อได้จะต้องมีการกำหนดให้มันไว้ในไฟล์ HTML โดยการกำหนดสามารถทำได้ 2 แบบ ดังนี้

โดยการใส่ชื่อไว้ใน APPLET-tag

```
<applet codebase=example/ code=Sender.class width=450 height=200
 name = "buddy">
 ...
</applet>
```

โดยการใส่ใน PARAM-tag

```
<applet codebase=example/ code=Receiver.class width=450 height=35>
<param name="name" value="old pal">
...
</applet>
```

หลักการใช้ getApplet() เพื่อหาแอพเพล็ทบนเพจนั้น เมื่อมีการเรียกใช้ method นี้จะให้ค่าของแอพเพล็ททั้งหมดที่ปรากฏบนเพจนั้น

บทที่ 4
แนวการเขียนเกมส์บนเว็บ (Games on the Web)

จากการที่อินเตอร์เน็ตและ World Wide Web มีอัตราเติบโตอย่างรวดเร็ว มีเครือข่ายการเชื่อมต่อระหว่างผู้ใช้มากมาย ฉะนั้นแอพพลิเคชันที่น่าจะกล่าวถึงกันอย่างมากคือเกมส์ โดยที่เกมส์เหล่านี้จะถูกคิดค้นให้สามารถเล่นเป็นเครือข่ายได้ สร้างความบันเทิงแก่ผู้เล่น การเล่นเกมส์บนเว็บได้เปลี่ยนแปลงรูปแบบการสร้างความบันเทิง เพราะความไม่มีขอบเขตจำกัดในการเล่นเหมือนกับอินเตอร์เน็ต เราสามารถเล่นได้กับทุกคนที่กำลังใช้งานบนอินเตอร์เน็ตอยู่

ในหัวข้อนี้เราจะกล่าวถึงหัวข้อย่อยดังนี้

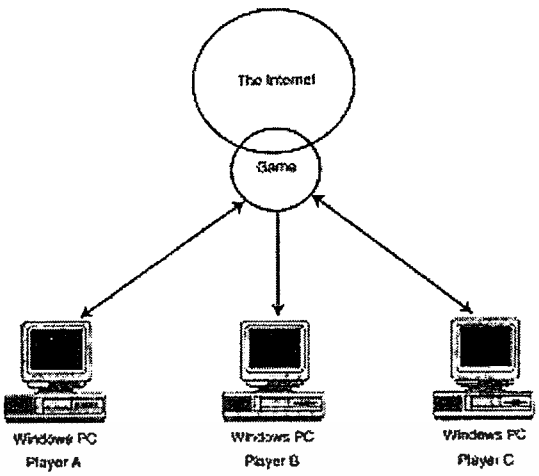
1. รูปแบบของเกมส์ต่างๆบนระบบอินเตอร์เน็ต
2. สิ่งที่ควรพิจารณาในการเขียนเกมส์ด้วยภาษาจาวา
3. การออกแบบเกมส์
4. ทฤษฎีของภาษาจาวาในทางกราฟฟิกที่ใช้ในการเขียนเกมส์
5. ระบบเครือข่ายกับภาษาจาวา
6. แนวความคิดพื้นฐานของการเล่นเกมส์แบบหลายคน
7. ปัญหาที่เกิดขึ้นในการเล่นเกมส์แบบเครือข่าย
8. การเขียนเกมส์โดยใช้ Multithreading

4.1 รูปแบบของเกมส์ต่างๆ

รูปแบบของเกมส์บนอินเตอร์เน็ตที่สามารถจำแนกได้เป็นดังนี้

4.1.1 Web Game

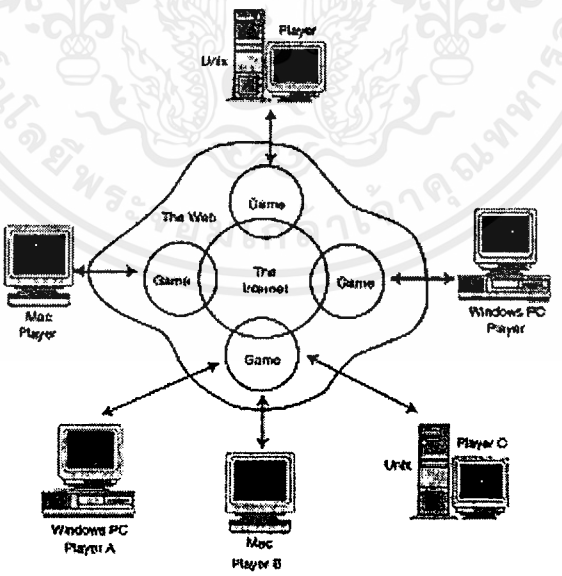
เป็นเกมส์ที่สามารถทำงานบนอินเตอร์เน็ตและการทำงานต้องอยู่บน Web Page โดยที่ไม่ขึ้นกับแพลทฟอร์มใดๆ ดังรูปที่ 4 – 1



รูปที่ 4 – 1 แสดง Web Game ที่สามารถรันได้ทั้งแบบเล่นคนเดียวและเล่นบนเครือข่ายอินเตอร์เน็ต

4.1.2 Non-Java Based Games

เป็นเกมส์ที่จะทำงานบนระบบเครือข่ายแต่ไม่มีการเชื่อมต่อกับ Web Page นอกจากนี้ non-Java game ยังต้องเล่นได้เพียงแพลทฟอร์มเดียวหรือสามารถเล่นได้เพียงบางแพลทฟอร์มเท่านั้น ดังแสดงได้รูปที่ 4 – 2



รูปที่ 4 – 2 แสดง Non-Web Based Games

ตัวอย่างของเกมส์ที่เป็นแบบ Non-Web Based เช่น Doom และ CivNet ซึ่งเกมส์เหล่านี้ต่างเป็นเกมส์ที่สามารถเล่นเป็นเครือข่ายได้แต่ไม่ได้เกี่ยวข้องกับ Web

4.2 สิ่งที่ควรพิจารณาในการเขียนเกมส์ด้วยภาษาจาวา

หลักการในการเขียนเกมส์ด้วยภาษาจาวาจำเป็นต้องพิจารณาสิ่งเหล่านี้คือ

1. กราฟฟิกและภาพเคลื่อนไหว (Graphics and Animation)
2. การป้อนอินพุท
3. เสียง
4. ระบบเครือข่าย
5. การจัดการด้านมีเดียต่างๆ

4.2.1 กราฟฟิกและภาพเคลื่อนไหว (Graphics and Animation)

เนื่องจากกราฟฟิกและภาพเคลื่อนไหวเป็นสิ่งสำคัญที่จะให้เกมส์ดูน่าเล่นหรือน่าเบื่อ ฉะนั้นจึงเป็นสิ่งสำคัญที่ควรจะพิจารณาเป็นอันดับแรกในการเขียนเกมส์ ภาษาจาวามี API ที่สนับสนุนการทำงานเกี่ยวกับกราฟฟิกต่างๆ เช่น รูปภาพ (image) , กราฟฟิก 2D เป็นต้น ส่วนภาพเคลื่อนไหวนั้น ถึงแม้ว่าจาวาจะไม่มี API ที่สามารถจัดการได้โดยตรง แต่ก็มีแนวทางในการเขียนที่สามารถทำให้ง่ายขึ้น

4.2.2 การป้อนอินพุท (User Input)

การป้อนอินพุทของผู้เล่นก็นับว่าเป็นสิ่งที่สำคัญในการเล่นเกมส์เพราะมันเป็นสิ่งที่จะบอกได้ว่าเกมส์นี้สร้างความรู้สึกอย่างไรกับผู้เล่น ภาษาจาวาได้สนับสนุนการทำงานของดีไวซ์ที่ป้อนอินพุท 2 ชนิด คือ คีย์เวิร์ดและเมาส์ โดยที่ในขณะที่เขียนโปรแกรมอยู่ ผู้เขียนสามารถทำการตรวจสอบการทำงานของดีไวซ์เหล่านี้โดยการเช็คเหตุการณ์ที่ถูกสร้างจากดีไวซ์ และถึงแม้ว่าการเล่นเกมส์จะได้อรรถรสมากถ้าหากสามารถทำการป้อนอินพุทจากดีไวซ์จำพวก joystick หรือ fight yokes แต่ในความเป็นจริงบางแพลทฟอร์มไม่สามารถทำงานร่วมกับดีไวซ์เหล่านี้ได้

4.2.3 เสียง (Sound)

เสียงก็นับว่าเป็นสิ่งที่สำคัญในเกมส์เหมือนกัน ภาษาจาวาสนับสนุนการทำงานกับเสียงได้แย่มากที่สุดในเกมส์ เนื่องจากทำงานได้กับรูปแบบของเสียงได้ไม่กี่แบบ เช่น .au , .midi

4.2.4 ระบบเครือข่าย (Networking)

ระบบเครือข่ายเป็นส่วนที่สำคัญที่ทำให้จาวาได้รับความนิยมเป็นอย่างมาก ระบบเครือข่ายของจาวาเป็นรูปแบบของการไม่ขึ้นกับแพลทฟอร์มใดๆ นั่นก็คือเราสามารถนั่งอยู่หน้าเครื่องวินโดว95 และเล่นเกมส์กับผู้อื่นที่กำลังใช้เครื่องชนิดอื่นเช่น Sun หรือ Macintoshes

4.2.5 การจัดการด้านมีเดีย (Media Management)

การจัดการด้านมีเดีย คือ การคอยตรวจสอบ (track) มีเดียต่าง ๆ เช่น กราฟฟิก เสียง ฯลฯ ว่าถ่ายโอนครบทุกอย่างหรือยัง เพื่อป้องกันรูปภาพแสดงไม่เต็มรูปขณะเล่นเกมส์อยู่

4.3 การออกแบบ (Game Design)

เป็นการพิจารณาว่าจะรูปแบบของเกมส์จะออกมาแนวไหน ซึ่งแบ่งหัวข้อพิจารณาเป็นหัวข้อย่อยดังนี้

1. แนวความคิดพื้นฐาน (Basic Idea)
2. แนวการดำเนินเกมส์ (Storyline)
3. โหมดการเล่นเกมส์ (Play Mode)

4.3.1 แนวความคิดพื้นฐาน (Basic Idea)

เป็นแนวความคิดเกี่ยวกับรูปแบบของเกมส์ว่าจะเป็นแบบเกมส์การผจญภัย , เกมส์แอกชัน หรือรวมกันหลายอย่าง

4.3.2 แนวการดำเนินเกมส์ (Storyline)

Storyline จะเป็นสิ่งที่ช่วยให้การคิดเกมส์แคบขึ้น ทำให้มองภาพพจน์ของเกมส์ทั้งหมดได้ชัดขึ้น และ Storyline จะช่วยให้การเขียนโปรแกรมหรือการพัฒนาโปรแกรมทำได้ง่ายขึ้น

4.3.3 โหมดการเล่นเกมส์ (Play Mode)

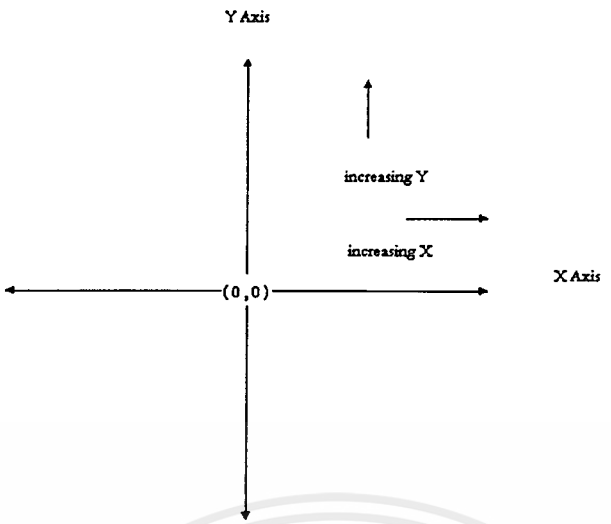
เป็นการระบุว่าจำนวนผู้เล่นมีเท่าไรเช่น เล่นคนเดียวต่อหน้าจอ หรือ สองผู้เล่น เป็นต้น

4.4 ทฤษฎีของภาษาจาวาในทางกราฟฟิกที่ใช้ในการเขียนเกมส์

ซึ่งแบ่งเป็นหัวข้อย่อยดังนี้

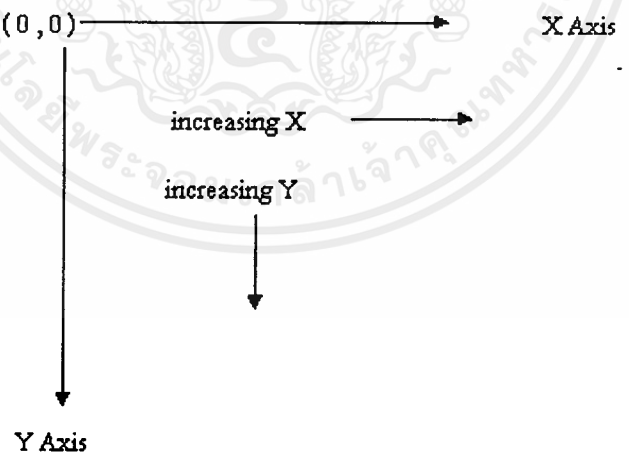
4.4.1 ระบบกราฟฟิกโคออดิเนต (The Graphics Coordinate System)

เป็นส่วนที่ระบุจุดต่างๆบนระบบ โดยที่ระบบโคออดิเนตมีจุดเริ่มต้นที่ (0,0) ของระบบกราฟฟิก โดยที่มีแนวการเพิ่มขึ้นหรือลดลงของค่าตามแนวลูกศร ตามรูปที่ 4 – 3



รูปที่ 4 – 3 แสดงระบบโคออดิเนต

ในระบบกราฟฟิกของภาษาจาวาใช้ระบบโคออดิเนตในการระบุว่า การวาดภาพจะเกิดที่ตำแหน่งไหนและอย่างไร เนื่องจากการวาดภาพของจาวาจะเกิดขึ้นในเฉพาะพื้นที่ของแอพเพล็ท ดังนั้นระบบโคออดิเนตในจาวาจึงมีจุดเริ่มต้นจากมุมซ้ายบนสุดของแอพเพล็ท และค่าทางแนวแกน X และ Y จะเพิ่มขึ้นเมื่อในทิศทางลงของแอพเพล็ทตามรูปที่ 4 – 4



รูปที่ 4 – 4 แสดงระบบโคออดิเนตของภาษาจาวา

4.4.2สีพื้นฐาน

ในภาษาจาวามีความคล้ายคลึงกับระบบทางกายภาพ (Graphical System) ที่ถูกใช้โดยจอสีทั่วๆ

ไป คือสีหนึ่งๆ ประกอบด้วยสีแม่สี คือ แดง เขียว น้ำเงิน ในอัตราต่างกันๆ หรือเรียกว่า RGB (Red Green Blue) ซึ่งเป็นมาตรฐานทั่วไปของระบบทางกายภาพของคอมพิวเตอร์ และนอกจากนี้ในจาวายังมีระบบสีอีกชนิดหนึ่งที่เรียกว่า HSB (Hue Saturation Brightness) โดยที่ระบบสีในนี้ถูกำหนดตามความแตกต่างของค่า hue , saturation และความสว่าง

Color	Red	Green	Blue
ขาว	255	255	255
ดำ	0	0	0
เทาสว่าง	192	192	192
เทามืด	128	128	128
แดง	255	0	0
เขียว	0	255	0
ฟ้า	0	0	255
เหลือง	255	255	0
ม่วง	255	0	255

ตารางที่ 4 – 1 แสดงค่าของ RGB ในสีพื้นฐาน

4.4.3การวาดภาพทางกราฟฟิกพื้นฐาน (Drawing Graphics Primitives)

Graphics primitives ประกอบด้วยเส้นตรง สี่เหลี่ยมผืนผ้า รูปหลายเหลี่ยม วงกลม หรือ วงรีเป็นต้น ซึ่งคลาส Graphics จะมี method ที่ใช้ในการวาดภาพเหล่านี้

เส้นตรง (Lines)

ในการวาดเส้นตรงมีรูปแบบของการเขียนดังนี้

```
void drawLine(int x1, int y1, int x2 ,int y2)
```

โดยที่ค่า x1 และ y1 คือจุดเริ่มต้นของการวาดเส้นตรง และค่า x2 , y2 คือจุดสิ้นสุดของการวาด สำหรับการวาดเส้นตรงในแอพเพล็ทสามารถทำได้โดยการเรียกใช้ method drawLine ใน method paint ของแอพเพล็ทตามตัวอย่างข้างล่าง

```
public void paint(Graphics g) {
        g.drawLine(5 , 10 , 15 , 55)
```

```
}
```

สี่เหลี่ยมผืนผ้า

การวาดสี่เหลี่ยมผืนผ้าสามารถทำได้โดยเรียกใช้ method drawRect โดยมีการระบุมุมซ้ายบนและ ความกว้าง-ยาวของสี่เหลี่ยมผืนผ้า method drawRect ถูกเก็บในคลาส Graphcs ตามรูปแบบข้างล่าง

```
void drawRect(int x , int y , int width , int height )
```

โดยที่ ค่า x และ y คือตำแหน่งของมุมซ้ายบนของสี่เหลี่ยม และ width , height คือความกว้างและความยาว ตัวอย่างของการใช้ method ใน แอพเพล็ท

```
public void paint (Graphics g) {
        g.drawRect(5 , 10 , 15 , 55 )
}
```

วงกลม (Circle)

รูปหลายเลี่ยม (Polygons)

ในการเขียนรูปหลายเหลี่ยมจำเป็นต้องมีค่า x และ y จำนวนหนึ่ง โดยที่การวาดจะเริ่มจากค่าที่หนึ่ง แล้วลากเส้นไปยังจุดที่สอง จากนั้นลากเส้นไปยังจุดที่สาม ต่อไปเรื่อยๆ ดังนั้นในการใช้ method drawPolygon() และ fillPolygon() จะต้องมีอาร์กิวเมนต์ต่อไปนี้

1. อาร์เรย์ของค่าตัวเลขที่เป็นค่าจุด x
2. อาร์เรย์ของค่าตัวเลขที่เป็นค่าจุด y
3. ค่าตัวเลขที่เก็บจำนวนตัวเลขทั้งหมด

โดยตัวอย่างของการใช้ method แสดงอยู่ข้างล่าง

```
public void paint(Graphics g) {
        int exes[] = { 39 , 94 , 97 , 142 , 53 , 58 ,26 }
        int whys[] =  { 33 , 74 , 36 , 70 , 108 , 80 , 106 }
        int pts = exes.length;
        g.drawPolygon(exes , whys , pts );
}
```

วงรี (Ovals)

เราใช้ method นี้ในการวาดทั้งวงรีและวงกลม โดยการเรียกใช้ method มีอาร์กิวเมนต์ 4 ตัวคือ ค่า x , y ที่เป็นมุมบนของรูป และความกว้าง-ยาว method ที่ใช้ในการวาดรูปนี้คือ fillOval() และ drawOval() ตัวอย่างของการเรียกใช้ method นี้ ใน paint method คือ

```
public void paint(Graphics g) {
        g.drawOval(20 , 20 , 60 , 60);
        g.fillOval(120 , 20 , 100 , 60 );
}
```

ส่วนของเส้นโค้ง (Arc)

เป็นการวาดส่วนของเส้นโค้งในวงรี โดยในการเรียกใช้ method drawArc() หรือ fillArc() ประกอบด้วยอาร์กิวเมนต์ 6 ตัวคือ จุดเริ่มต้น ความกว้าง ความยาว มุมองศาที่ใช้ในการเริ่มวาดเส้นโค้ง และมุมองศาที่จะวาดก่อนที่การวาดจะสิ้นสุดลง ตัวอย่างของการใช้ method นีแสดงข้างล่าง

```
public void paint(Graphics g) {
        g.drawArc(20 , 20 , 60 , 60 , 90 ,180);
        g.fillArc(120 , 20 , 60 , 60 , 90 ,180 );
}
```

4.4.4การเขียนข้อความ (Drawing Text)

การเขียนข้อความในภาษาจาวา ต้องเขียนใน method paint ในแอพเพล็ท โดยเรียกใช้ method drawString() ที่ประกอบด้วยอาร์กิวเมนต์ 3 ตัวคือ ข้อความที่จะเขียน ค่า x และ y ที่จะใช้เป็นจุดเริ่มต้นในการเขียนข้อความ นอกจากนี้เรายังสามารถเซ็ตรูปแบบของอักษรได้ตามความต้องการโดยใช้ method setFont() ที่มีอาร์กิวเมนต์เป็นรูปแบบของฟอนต์ ดังตัวอย่างการเขียน

```
Font f = new Font("Helvetica", Font.BOLD + Font.ITALIC , 22);
Void setFont(Font font);
Public Font(String name, int style, int size);
```

โดยการเรียกใช้ method setFont และ drawString จะต้องเขียนใน method paint() ดังตัวอย่างนี้

```
public void paint(Graphics g) {
        Font font = new Font("Helvetica",Font.BOLD+Font.ITALIC,22);
```

```
g.setFont(font);
g.drawString("Project" , size().width ,size().height );
}
```

4.4.5การวาดรูปภาพ (Drawing Images)

ภาษาจาวาสามารถสนับสนุนการทำงานร่วมกับรูปภาพขนาด 32 bit นั่นก็หมายความว่าแต่ละพิกเซลในรูปภาพใช้ขนาด 32 บิท และแต่ละสีของภาพก็มีขนาดเป็น 32 บิทเช่นกัน ในการวาดรูปภาพอันดับแรกทราบว่าภาพที่จะวาดเก็บไว้ที่ไหน โดยใช้ method getImage ซึ่งมีอยู่ในคลาส Applet ในการโหลดภาพจาก URL ที่ได้ระบุไว้ ซึ่งมีรูปแบบของการใช้ method 2 แบบด้วยกันคือ

```
Image getImage(URL url)
Image getImage(URL url, String name)
```

ผลลัพธ์จากการเรียกใช้นี้นอกจากจะได้รูปภาพที่จะวาดแล้วยังทราบชนิดของภาพด้วย ซึ่งชนิดของรูปภาพที่คลาส Image ใช้ได้คือ GIF หรือ JPEG เป็นต้น และนอกจากนี้คลาส Image ยังสามารถบอกข้อมูลเกี่ยวกับความกว้างและความยาวของภาพได้อีกด้วย

ส่วนการวาดรูปภาพนั้นใช้ method drawImage ของคลาส Graphics โดยมีวิธีการใช้ method ดังตัวอย่าง

```
boolean drawImage(Image img, int x, int y, ImageObserver observer)
boolean drawImage(Image img, int x, int y,int width , int height, ImageObserver
                  observer)
boolean drawImage(Image img, int x, int y, Color bgColor, ImageObserver observer)
boolean drawImage(Image img, int x, int y , int width , int height , Color bgColor ,
                  ImageObserver observer )
```

โดยที่ ImageObserver คือสิ่งที่ถูกใช้ภายในโดย drawImage สำหรับการรับเอาค่าข้อมูลเกี่ยวกับรูปภาพ

4.4.6 การตรวจสอบรูปภาพ (Tracking Image)

การ Tracking Image เป็นสิ่งที่จำเป็นมากเพราะเนื่องจากการขนถ่ายรูปภาพบนเว็บ

ที่มี bandwidth จำกัด อาจจะประสบปัญหาที่ยังโหลดรูปยังไม่หมด จาวามีการแก้ปัญหาโดยใช้เทคนิค interlacing ซึ่งแต่ละรูปจะถูกทำให้พร่าจนกว่าภาพจะขนถ่ายเสร็จสมบูรณ์ ในการเทคนิค interlacing รูปภาพจะต้องเก็บรูปแบบของ interlace เช่น GIF รุ่น 98a ซึ่งหมายความวาข้อมูลของรูปภาพจะถูกจัดระเบียบในวิธีการใดวิธีการหนึ่งดังนั้นรูปภาพจึงสามารถแสดงออกมาได้แม้ยังโหลดไม่สมบูรณ์ ซึ่งเทคนิคนี้ใช้ได้ดีสำหรับรูปภาพแบบคงที่ (Static Image) เพราะไม่ต้องรอให้รูปโหลดจนเสร็จถึงจะแสดงรูปได้ แต่ถ้าหากเป็นแบบ dynamic เช่น ภาพเคลื่อนไหว เราจำเป็นต้องรอให้รูปโหลดจนเสร็จก่อน ถึงจะแสดงรูปได้ ซึ่งจะใช้เทคนิค Java media tracker สำหรับการตรวจสอบว่ารูปทั้งหมดโหลดเสร็จสิ้นหรือยัง

4.4.7 The MediaTracker Class

คลาสนี้เป็นส่วนหนึ่งของแพกเก็จ AWT ซึ่งมี method มากมายสำหรับการ track มีเดียต่างๆ แต่ MediaTracker ใช้ในการ Track รูปภาพเท่านั้น โดยที่จะมีสถานะบ่งบอกถึงสภาวะการโหลดรูปเหล่านั้น เช่น

LOADING: เป็นการบอกว่ามีเดียกำลังอยู่ในระหว่างการโหลด
ABORTED: เป็นการบอกว่ามีเดียได้ถูกยกเลิกไปแล้ว
ERRORED: เป็นการบอกว่าการโหลดมีเดียเกิดข้อผิดพลาดขึ้น
COMPLETE: เป็นการบอกว่าการโหลดมีเดียเสร็จสมบูรณ์แล้ว

คลาส MediaTracker มี method หลายรูปแบบด้วยกันดังนี้

MediaTracker(Component comp)
Void addImage(Image image, int id)
Synchronized void addImage(Image image, int id , int w , int h)
Boolean checkID(int id)
Synchronized boolean checkID(int id , boolean load)
Boolean checkAll();
Synchronized boolean checkAll(boolean load)
Void waitForID(int id)
Synchronized boolean waitForID(int id, long ms)
Int statusID(int id, boolean load)
Int statusAll(boolean load)
Synchronized boolean isErrorID(int id)
Synchronized boolean isErrorAny()

Synchronized Object[] getErrorsID(int id)

Synchronized Object[] getErrorsAny()

4.4.8 การตอบสนองต่อการป้อนอินพุทของผู้เล่น

การตอบสนองต่อการป้อนอินพุทของผู้เล่น แบ่งเป็น 2 ประเภทคือ

1. การตอบสนองต่อการกดแป้นคีย์บอร์ด (Keyboard events)
2. การตอบสนองต่อเมาส์ (Mouse events)

การตอบสนองต่อการกดแป้นคีย์บอร์ด (Keyboard events)

เหตุการณ์ที่เกี่ยวข้องกับคีย์บอร์ดเกิดขึ้นเมื่อผู้เล่นมีการกดหรือปล่อยคีย์ โดยที่เหตุการณ์พื้นฐานที่สนับสนุนการทำงานโดยคลาส Component นี้มี 2 แบบคือ keyDown และ keyUp ซึ่งมีเม็ทธอดดังข้างล่าง

public boolean keyDown(Event evt , int key)

public boolean keyUp(Event evt, int key)

โดยที่ keyDown จะถูกเรียกใช้เมื่อผู้เล่นกดคีย์ และ keyUp จะถูกเรียกใช้เมื่อผู้เล่นมีการปล่อยคีย์ ทั้งสองเม็ทธอดนี้จะมีการผ่านค่า Event Object และค่าคีย์ โดยที่ค่าคีย์จะเป็นตัวบอกว่าคีย์ใดถูกกดหรือปล่อย และ Event Object จะเก็บข้อมูลพิเศษที่เกี่ยวข้องกับเหตุการณ์ เช่นมีการกด Shift Key พร้อมกับการกดคีย์

ค่าคงที่ (Constant)	คีย์ (Key)
UP	Up arrow
DOWN	Down arrow
LEFT	Left arrow
RIGHT	Right arrow
HOME	Home
END	End
PGUP	Page Up
PGDN	Page Down

ตารางที่ 4 – 2 แสดงค่าคงที่ต่างๆของคีย์บอร์ดที่ใช้ในเกมส์

ในการตรวจสอบการกดคีย์หรือปล่อยคีย์ตามคีย์ข้างต้น จะต้องมีการเปรียบเทียบค่าของคีย์ตามค่าคงที่ที่ได้ระบุไว้ ดังตัวอย่างข้างล่างแสดงการตรวจสอบการกดคีย์ลูกศร

```
public boolean keyDown(Event evt , int key) {
switch (key) {
case Event.LEFT :
        // left arrow key pressed
   break;
case Event.RIGHT:
        // right arrow key pressed
    break;
case Event.UP:
        // up arrow key pressed
     break;
case Event.DOWN:
        // down arrow key pressed
     break;
}
return true;
}
```

นอกจากนี้ถ้าหากมีการกดคีย์ Modifier เช่น Shift และ คีย์Control ก็สามารถทำการตรวจสอบได้โดยใช้เม็ทธอด ข้างล่างนี้

```
public boolean shiftDown()
public boolean controlDown()
public boolean metaDonw()
```

การตอบสนองเหตุการณ์เกี่ยวกับเมาส์ (Mouse Event)

เหตุการณ์ที่เกี่ยวกับเมาส์จะเกิดขึ้นเมื่อปุ่มเมาส์มีการคลิ้ก หรือเคลื่อนที่เมาส์ โดยมีเม็ทธอดที่ช่วยในการตรวจสอบเมาส์ดังนี้

public boolean mouseUp(Event evt , int x , int y) ถูกเรียกเมื่อมีการกดปุ่มเมาส์

public boolean mouseDonw(Event evt, int x , int y) ถูกเรียกเมื่อมีการปล่อยเมาส์

public boolean mouseMove(Event evt, int x, int y) ถูกเรียกเมื่อมีการเคลื่อนเมาส์

public boolean mouseDrag(Event evt, int x, int y) ถูกเรียกเมื่อมีการเคลื่อนและกดเมาส์

public boolean mouseEnter(Event evt, int x , int y) ถูกเรียกเมื่อเมาส์เข้ามาในแอพเพล็ท

public boolean mouseExit(Event evt, int x, int y) ถูกเรียกเมื่อเมาส์ออกนอกแอพเพล็ท

ตัวอย่างการใช้ method ที่เกี่ยวข้องกับเมาส์

```
public boolean mouseMove(Event evt , int x , int y) {
        System.out.println("Mouse position = ( " + x + " , " + y + " ) ");
        Return true;
}
```

4.4.9 การเล่นไฟล์เสียง (Playing Sound)

คลาสที่สนับสนุนการเล่นไฟล์เสียงของจาวาในตอนนี้ อยู่ในคลาส Applet ซึ่งมี AudioClip Class ซึ่งเป็นคลาสหนึ่งในแพกเก็จ applet โดยมีรูปแบบของเสียงเป็นแบบ AU file ซึ่งในการเล่นไฟล์เสียงในจาวา อันดับแรกต้องมีการสร้างและการตั้งค่าเริ่มแรกของ Audio โดยใช้เม็ทธอด getAudioClip ซึ่งมีรูปแบบของการเรียกใช้ 2 แบบคือ

```
public AudioClip getAudioClip(URL url);
public AudioClip getAudioClip(URL url, String name )
```

โดยที่ URLในรูปแบบแรก คือ ชื่อที่เต็มสมบูรณ์ของไฟล์เพลง ส่วนของรูปแบบที่สองนั้น เป็น base URL ของไฟล์เสียงและ name คือชื่อของไฟล์เสียง ซึ่งรูปแบบ 2 นั้นอาจจะใช้ตามตัวอย่างนี้

```
AudioClip clip1 = getAudioClip(getCodeBase(),"sound1.au");
AudioClip clip2 = getAudioClip(getDocumentBase(),"sound2.au");
```

โดยที่ getCodeBase() จะให้ค่ากลับเป็น base URL ของแอพเพล็ท แต่ getDocumentBase() จะให้ค่ากลับเป็น base URL ของไฟล์ HTML ที่แอพเพล็ททำงานอยู่

method ที่มีอยู่ในคลาส AudioClip ที่ใช้เกี่ยวกับไฟล์เสียง มีดังนี้

public abstract void play () สำหรับการเล่นเพลง

public abstract void stop() สำหรับการหยุดเล่นเพลง

public abstract void loop() สำหรับให้เล่นเพลงซ้ำไปเรื่อยๆ

4.5 ระบบเครือข่ายกับจาวา

ในหัวข้อนี้แบ่งการอธิบายเป็นข้อๆดังนี้คือ

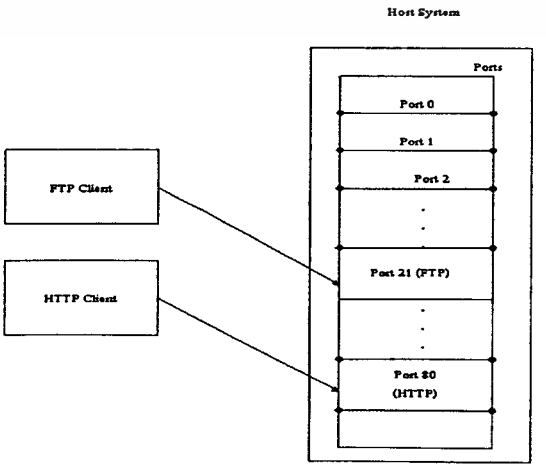
4.5.1 ระบบเครือข่ายพื้นฐาน

ในการที่จะให้จาวาสามารถติดต่อกับระบบเครือข่ายจำเป็นจะต้องมีองค์ประกอบเหล่านี้คือ

1. Address โดยที่แอดเดรส เป็นมาตรฐานบนอินเตอร์เน็ต ที่จะระบุเครื่องคอมพิวเตอร์แต่ละเครื่องที่กำลังเชื่อมต่อกันอยู่ โดยที่แอดเดรสหรือ IP address นี้เป็นตัวเลขขนาด 32 บิท ที่ไม่ซ้ำกันเลย โดยมีรูปแบบของการจัดเรียงตัวเลขคล้ายกับ161.246.4.3ซึ่งถ้าหากเป็นโดเมนเนมเซิร์ฟเวอร์ก็จะเป็น diamond.ce.kmitl.ac.th เนื่องจากความไม่เหมือนกันเลยของตัวเลขนี้ ก็จะทำให้เกิดความง่ายในการสื่อสารไปเครื่องคอมพิวเตอร์ใดๆ

2. Protocols เป็นข้อกำหนดหรือมาตรฐานในการสื่อสารทางอินเตอร์ ซึ่งโปรโตคอลจะเป็นตัวระบุว่าข้อมูลที่กำลังส่งอยู่บนอินเตอร์เน็ตนี้ ส่งอย่างไร จะส่งเมื่อไร และในปลายทางของการสื่อสารจะทราบว่าข้อมูลที่ได้รับมีโครงสร้างอย่างไรและหมายความว่าอย่างไร ตัวอย่างของโปรโตคอลที่ใช้กันในปัจจุบันเช่น HTTP (Hypertext transfer protocol) ซึ่งใช้ในการขนถ่ายข้อมูลจำพวกเอกสาร HTML บนเว็บ หรือ FTP (File transfer protocol) ใช้ในการส่งไบนารีไฟล์บนอินเตอร์ ซึ่งโปรโตคอลเหล่านี้สามารถใช้งานร่วมกับจาวา ได้

3. Port เป็นตัวเลขขนาด 16 บิทที่ใช้ในการระบุแต่ละบริการที่ให้โดยเซิร์ฟเวอร์ของระบบเครือข่าย ซึ่งแต่ละบริการที่มีอยู่บนอินเตอร์เน็ตจะมีหมายเลขพอร์ตที่แตกต่างกัน เช่น ในการ FTP จะใช้พอร์ตเบอร์ 21 และสำหรับ HTTP จะใช้พอร์ต 80 ในการให้บริการ ดังแสดงในรูปที่ 4 – 5



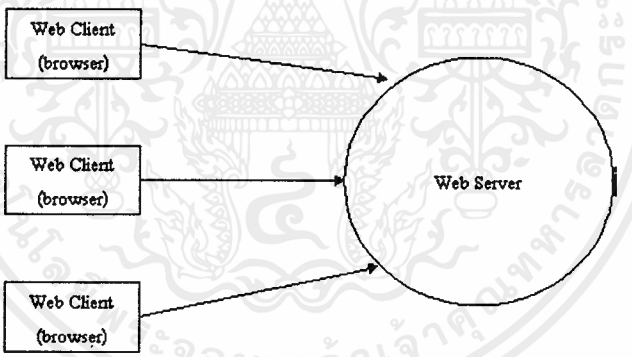
รูปที่ 4 – 5 แสดงความสัมพันธ์ของพอร์ตและโปรโตคอล

นอกจากนี้บริการมาตรฐานต่างก็มีพอร์ตที่อยู่ต่ำกว่าเบอร์ 1024 ซึ่งหมายความว่าพอร์ตที่อยู่เหนือเบอร์ 1024 สามารถใช้ได้ในการสื่อสารตามความต้องการ เช่น การเล่นเกมส์บนอินเตอร์เน็ต ฉะนั้นในการใช้พอร์ตเหนือเบอร์ 1024 จะต้องใช้ที่ยังไม่มีใครใช้

4.5.2 ไคล์เอ็นท์/เซิร์ฟเวอร์

เป็นการใช้แนวความคิดเกี่ยวกับรูปแบบของ Client คือผู้ที่ต้องการข้อมูลบางชนิด และ Server คือผู้ที่มีข้อมูลจำนวนมาก และกำลังคอยที่จะจัดการส่งข้อมูลออกไป เช่น ไคล์เอ็นท์ติดต่อกับเซิร์ฟเวอร์เข้ามาเพื่อต้องการข้อมูลบางอย่าง เซิร์ฟเวอร์จะจัดการหาข้อมูลและส่งข้อมูลกลับไปยังไคล์เอ็นท์

ในทางอินเตอร์เน็ต ไคล์เอ็นท์เป็นเครื่องคอมพิวเตอร์ที่เชื่อมต่อทางอินเตอร์เน็ตที่ต้องการข้อมูล ในขณะที่เซิร์ฟเวอร์เป็นเครื่องคอมพิวเตอร์ที่มีข้อมูลจำนวนมากสำหรับไคล์เอ็นท์ อย่างเช่น Web Server ซึ่งเก็บเอกสาร HTML และข้อมูลสำหรับผู้ที่ต้องการจะเอาไปใช้หรือ browse ส่วน Web Client คือกลุ่มผู้ใช้งานที่เชื่อมต่อไปยัง Web Server และทำการ browse ผ่าน Web Page ดังนั้น Netscape Navigator คือ Client Web Software



รูปที่ 4 – 6 แสดงการติดต่อระหว่าง Web Server กับ Web Client ต่างๆ

4.5.3 Socket

เน็ตเวิร์คซ็อกเก็ต เป็นกรรมวิธีในการติดตั้งสื่อสารผ่านเน็ตเวิร์ก ในการติดต่อผ่านเน็ตเวิร์กจะต้องอาศัยซ็อกเก็ตเพื่อทำให้การติดตั้งสามารถเกิดขึ้นและดำเนินต่อไปได้ เน็ตเวิร์คซ็อกเก็ตจะอนุญาตให้โปรแกรมสามารถเลือกค่าติดตั้งระหว่างคอมพิวเตอร์สองเครื่องได้

เมื่อส่งข้อมูลไปยังซ็อกเก็ต ข้อมูลจะถูกส่งออกไปยังปลายทางโดยอัตโนมัติ โปรแกรมที่ติดต่อกันระหว่างคอมพิวเตอร์สองเครื่องสามารถอยู่บนระบบปฏิบัติการที่แตกต่างกัน แต่สามารถติดต่อกันได้ รวมถึงการติดต่อรูปแบบอื่นๆ ระหว่างคอมพิวเตอร์สองเครื่อง เพราะเน็ตเวิร์กซ็อกเก็ตเป็นเครื่องมือที่ใช้

ในการติดต่อระหว่างจุดสองจุด แบบสองทิศทาง เพื่อความปลอดภัยในการติดต่อสื่อสาร จาวาสามารถเปิดซ็อกเก็ตที่สามารถติดต่อกลับไปยังเว็บไซต์ได้แบบเดียวเท่านั้น

ซ็อกเก็ตในการติดต่อส่งข้อมูลระหว่างสองฝั่งคือ เซิร์ฟเวอร์และไคล์เอ็นท์ โดยที่ฝั่งเซิร์ฟเวอร์จะสร้างซ็อกเก็ตเพื่อรอคอยการติดต่อที่จะเกิดขึ้น ซ็อกเก็ตจะใช้ตำแหน่งและหมายเลขพอร์ตเพื่อสร้างการติดต่อเซิร์ฟเวอร์ จะมีการใช้พอร์ตที่เป็นหมายเลขมาตรฐานในการติดต่อแต่ละแบบ โดยที่ไคล์เอ็นท์จะติดต่อกับเซิร์ฟเวอร์โดยใช้ตำแหน่งและพอร์ตตามที่เซิร์ฟเวอร์กำหนด และเมื่อไคล์เอ็นท์สามารถติดต่อกับเซิร์ฟเวอร์ได้ ระบบจะสร้างซ็อกเก็จเฉพาะสำหรับการติดต่อระหว่างเซิร์ฟเวอร์และไคลเอ็นท์ จากนั้นเซิร์ฟเวอร์ก็จะสามารถรอการติดต่อจากไคล์เอ็นท์ตัวอื่นต่อไป

ที่ฝั่งไคล์เอ็นท์จะใช้คลาส Socket เพื่อสร้างซ็อกเก็ต และระบุข้อมูลในการติดต่อเช่น ชื่อโฮสต์และหมายเลขของพอร์ตที่ใช้ติดต่อ ซึ่งการสร้างซ็อกเก็ตเพื่อใช้ในการติดต่อสร้างโดยใช้รูปแบบดังต่อไปนี้

```
Socket s;
S = new Socket("sunshine09.ce.kmitl.ac.th",80);
```

และทางฝั่งเซิร์ฟเวอร์เองก็จะมีการติดต่อกับซ็อกเก็ตคล้ายคลึงกับทางไคล์เอ็นท์ เพียงแต่เซิร์ฟเวอร์จะเป็นผู้ฟังและรอการติดต่อจากพอร์ต TCP เมื่อไคล์เอ็นท์ซ็อกเก็ตพยายามจะสร้างการติดต่อที่พอร์ต ฟังก์ชัน accept() จะถูกเรียกใช้เพื่อสร้างการติดต่อให้เกิดขึ้น

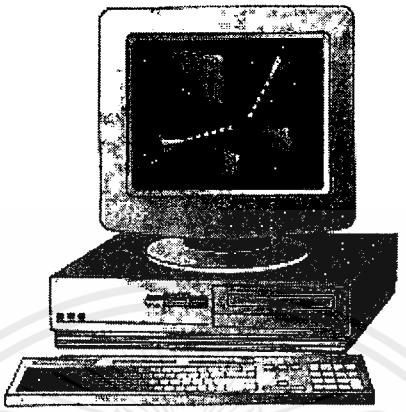
การสร้างเซิร์ฟเวอร์ซ็อกเก็ตจะต้องใช้คลาส ServerSocket เพื่อสร้างซ็อกเก็ตเซิร์ฟเวอร์ และรอการติดต่อจากไคล์เอ็นท์โดยสามารถทำได้ดังนี้

```
ServerSocket ss = new ServerSocket(80);
// เมื่อทำการติดตั้งเรียบร้อยแล้ว
ss.accept();
```

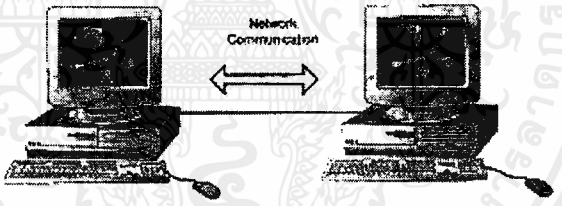
4.6 แนวความคิดพื้นฐานสำหรับการเล่นเกมส์แบบหลายคน

ในหัวข้อนี้จะพิจารณาถึงการสร้างให้เกมส์สามารถเล่นกันได้หลายคนบนเครือข่ายได้ (Network Game) หรืออาจจะไม่ต้องต่อกับระบบเครือข่ายก็สามารถเล่นหลายคนได้ (Non-Network Game) แต่ในการเขียนเกมส์ด้วยภาษาจาวา ผู้เล่นเว็บก็คงจะปรารถนาที่จะเล่นเกมส์แข่งกับผู้อื่นบนเว็บได้

เพื่อความเข้าใจอย่างละเอียดในความสัมพันธ์ระหว่างรูปแบบของเกมส์แบบเครือข่ายกับไม่เครือข่าย ดังแสดงในรูปที่ 4 – 7 และ 4 – 8 โดยที่รูปที่ 4 – 7 แสดงผู้เล่น 2 คนเล่นกันโดยใช้เครื่องเดียวกัน แต่แต่ละคนใช้คีย์ต่างกันในการเล่น และรูปที่ 4 – 8 แสดงผู้เล่น 2 คนเล่นกันโดยต่างคนต่างใช้เครื่องของตนเอง ซึ่งจะเห็นได้ว่ามีอินสแตนซ์ของเกมส์ 2 อันกำลังทำงานอยู่



รูปที่ 4 – 7 แสดงเกมส์แบบสองคนเล่นบนเครื่องที่ไม่ได้ต่อกับระบบเครือข่าย



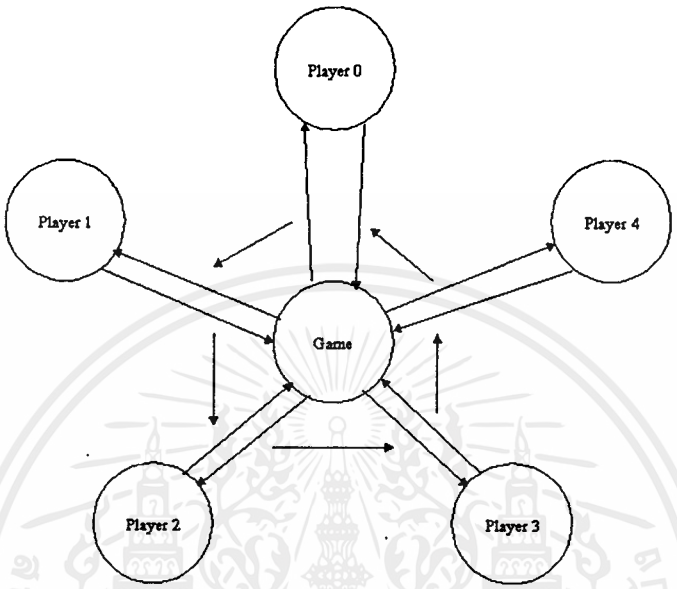
รูปที่ 4 – 8 เกมส์แบบสองผู้เล่นบนระบบเครือข่าย

ชนิดของเกมส์แบบหลายคนเล่น (Type of Multiplayer Game)

ชนิดของเกมส์ที่จะกล่าวถึงนี้เป็นชนิดของเกมส์ที่เล่นเป็นเครือข่าย ซึ่งแบ่งชนิดของเกมส์เป็น 2 ประเภทคือ

เกมส์แบบหมุนรอบ (Turn-Based Games)

คือเกมส์ที่มีรูปแบบของการเล่นที่ผู้เล่นจะมีการสับเปลี่ยนกันเล่นตามรูปที่ 4 – 9 โดยที่เกมส์นี้สามารถทำงานบนเน็ตเวิร์กได้ง่ายดาย เพราะมีเพียงผู้เล่นคนเดียวที่สามารถติดต่อกับเกมส์ในเวลานั้น และจะมีการย้ายผู้เล่นในทุกเวลาหนึ่งๆ เกมส์นี้จะถูกออกแบบให้ทุกผู้เล่นจะต้องรออยู่ในสภาวะการรอจนกระทั่งถึงตาที่จะต้องเล่น จากข้างต้นจะเห็นได้ว่าทิศทางการเล่นเกมส์จะเป็นแบบวงกลม คล้ายกับการเล่นไพ่

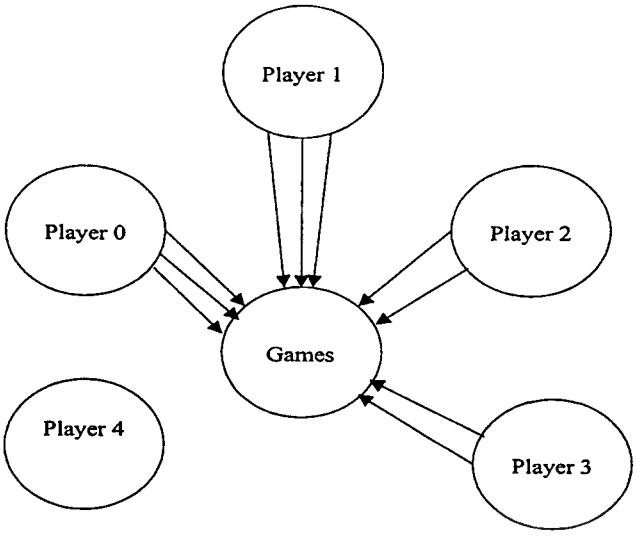


รูปที่ 4 – 9 แสดงเกมส์แบบ turn-based ที่มีผู้เล่น 5 คน

เกมส์แบบเหตุการณ์ (Event-Based Game)

เกมส์ชนิดนี้เป็นเกมส์ที่ถูกบังคับโดยเหตุการณ์ที่เข้ามาในเวลาใดๆ ซึ่งจะเห็นได้ว่ารูปแบบของเกมส์นี้จะไม่มีการหมุนหรือสับเปลี่ยนผู้เล่นเหมือนกับ turn-base game แต่เกมส์จะดำเนินเมื่อมีการป้อนอินพุทมาเท่านั้น เกมส์แบบนี้ที่เห็นกันทั่วไป เช่น Duke Nukem 3D

รูปแบบของเกมส์นี้มีความซับซ้อนและยากกว่าแบบ turn-based game เช่นการสื่อสารของเกต้องการแบนด์วิชที่กว้างกว่า เพราะมีข้อมูลจำนวนมากขนส่ง และนอกจากนี้เกมส์แบบนี้ที่สร้างขึ้นมาก็จะเจอปัญหาที่ต่างกันไป และการเล่นเกมส์จะต้องมีการโต้ตอบกันระหว่างผู้เล่น

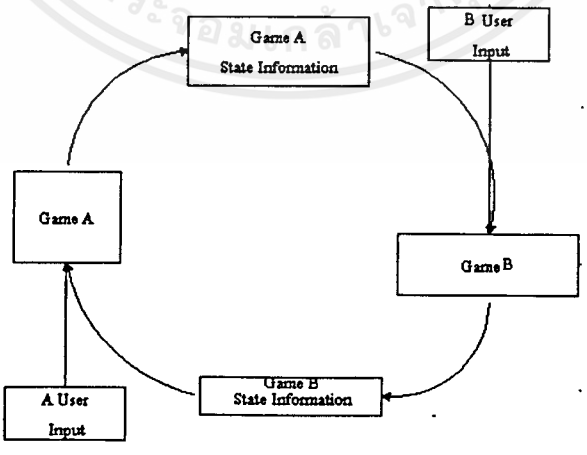


รูปที่ 4 – 10 แสดงเกมส์แบบ event-base ที่มีผู้เล่น 5 คน

4.7 ปัญหาที่อาจเกิดขึ้นในเกมส์แบบเครือข่ายและวิธีการแก้ปัญหา

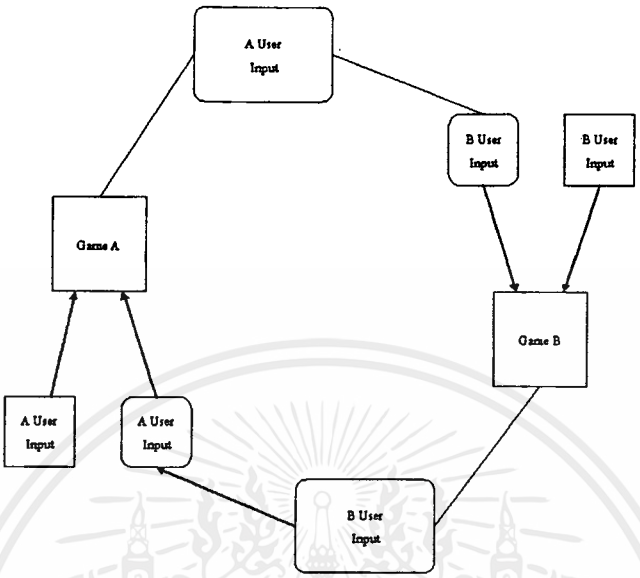
ปัญหาที่เกิดขึ้นส่วนใหญ่จะเป็นการ Synchronization ซึ่งการ Synchronization เป็นการระบุเกี่ยวกับจำนวนของอินสแตนซ์ของเกมส์ที่มีข้อมูลของสภาวะเดียวกัน เพื่อให้ผู้เล่นแต่ละคนทำงานได้เข้าจังหวะกันได้อย่างเหมาะสม

ส่วนการแก้ปัญหานั้น ใช้วิธี State Synchronization ซึ่งจะเป็นวิธีในการสื่อสารโดยที่แต่ละอินสแตนซ์จะสื่อสารสภาวะปัจจุบันของตัวเองแก่อินสแตนซ์อื่น ดังรูปที่ 4 – 11



รูปที่ 4 – 11 แสดง ข้อมูล State synchronization ที่ขนถ่ายกันระหว่างผู้เล่น 2 คนในเกมส์แบบเครือข่าย

หรือนอกจากนี้อาจจะใช้วิธี Input Synchronization ซึ่งเป็นวิธีในการสื่อสารโดยที่แต่ละอินพุทของผู้เล่นจะสื่อสารไปยังอินสแตนซ์อื่นในเกมส์ ดังรูปที่ 4 – 12



รูปที่ 4 – 12 แสดง Input synchronization ในเกมส์แบบเครือข่ายที่เล่น 2 คน

4.8 การเขียนเกมส์โดยใช้ Multithreading

Thread มีลักษณะเหมือนกับโปรแกรมทั่วไป thread 1 ตัวจะมีจุดเริ่มต้น จุดสิ้นสุด โปรแกรม (เป็นเพียงตอนหรือช่วงหนึ่ง) และเวลาใดๆขณะที่ thread กำลังทำงานจะมีจุดประสงค์ของการ execute เพียงอย่างเดียว อย่างไรก็ตาม thread โดยตัวมันเองแล้วไม่ใช่โปรแกรม เพราะไม่สามารถทำงานด้วยตัวเองได้ แต่ต้องทำงานกับโปรแกรม

นิยาม thread เป็นการทำงานอย่างใดอย่างหนึ่งในโปรแกรม

การทำงานของ thread ที่จะพูดถึงไม่ใช่การทำงานของ thread ตัวเดียว แต่เป็นการทำงานแบบ multiple threads ภายในโปรแกรมเดียวกัน โดย thread จะทำงานพร้อมๆกันแต่จะทำหน้าที่คนละอย่าง

ในหนังสือบางเล่มจะใช้คำว่า *lightweight process* แทนคำว่า thread (เนื่องจาก thread มีลักษณะคล้าย process และที่เรียกว่า lightweight เนื่องจาก thread จะทำงานภายใต้สิ่งแวดล้อมเดียวกับโปรแกรมและใช้ resource ที่ถูกจองให้กับโปรแกรม) แต่อย่างไรก็ตาม thread จะต้องมี resource บางอย่างที่เป็นของตัวเอง เช่น stack , program counter เป็นต้น โค้ดที่อยู่ใน thread จะต้องทำงานภายใต้สิ่งแวดล้อมที่กล่าวมาเท่านั้น ฉะนั้นเราอาจเรียก thread อีกอย่างว่า *execution context*

4.8.1 Thread Attributes

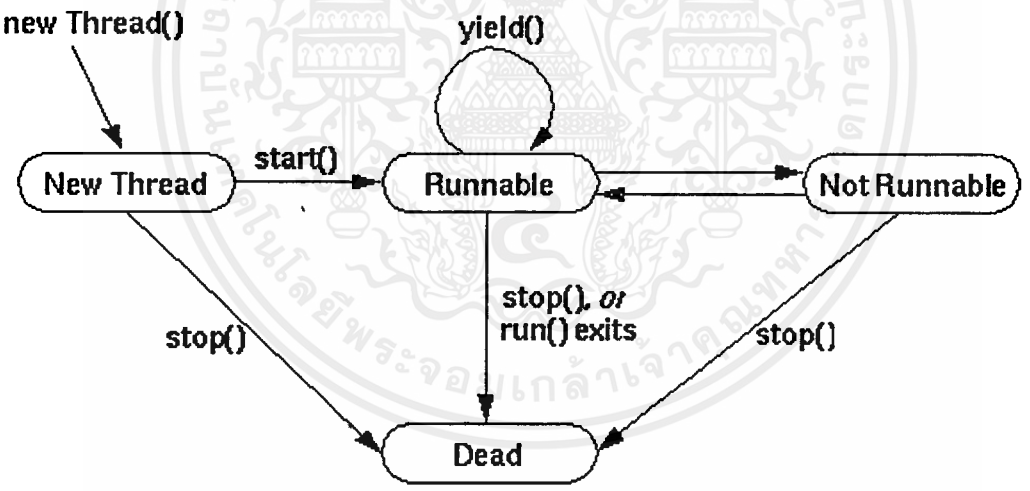
Thread ของจาวาถูก implement โดย Thread class ซึ่งอยู่ใน package java.lang thread class จะ implement ข้อกำหนดที่เป็นอิสระจากระบบของ thread

Thread Body การทำงานทุกอย่างเกิดขึ้นใน thread body (ใน run method) หลังจากที่ thread ถูสร้างและ initial แล้ว runtime system จะเรียก run method ของ thread นั้น โค้ดใน run method จะ implement การทำงานของ thread ที่สร้างขึ้น

บ่อยครั้งที่การทำงานของ thread จะเป็นการทำงานแบบ loop ทำให้บางครั้งการทำงานใช้เวลานาน คุณสามารถเลือกหนึ่งในสองวิธีต่อไปนี้ เพื่อลดขนาดของ run method สำหรับ thread ของจาวา

1. สร้าง subclass ของ Thread class ใน java.lang และ override run method
2. สร้างซึ่ง implement Runable interface ซึ่งอยู่ใน package java.lang เช่นกัน ดังนั้นถ้าคุณสร้าง instance ของ thread (ทั้งทางตรงจาก Thread class และทางอ้อมจาก subclass ของ Thread) ให้สร้าง handle ของ instance ของ Runable class ของคุณให้กับ thread ที่สร้างใหม่ Runable object นี้จะเตรียม run method ให้กับ thread

4.8.2Thread State รูปข้างล่างนี้แสดงสถานะของ thread ในจาวา



รูปที่ 4 - 13 สถานะของ Thread ในจาวา

- New Thread : statement ต่อไปนี้ใช้ในการสร้าง thread ใหม่ แต่ยังไม่มีการทำงานเกิดขึ้น

```
Thread myThread = new myThreadClass ();
```

เมื่อเป็น thread ใหม่ มันจะเป็น thread object ที่ว่างเปล่า ยังไม่มีการจอง resource ของระบบให้ เมื่อ thread อยู่ในสถานะ new thread มันสามารถจะเริ่มหรือหยุด thread ได้เท่านั้น

- Runnable : พิจารณาโค้ดต่อไปนี้

```
Thread myThread = new myThreadClass ( );
myThread . start ( );
```

start method จะทำการจอง resource ที่จำเป็นในการทำงานของ thread ให้ จัดตารางการทำงานของ thread และเรียก run method ของ thread เมื่อถึงจุดนี้ thread จะอยู่ในสถานะ "Runnable" ที่ไม่เรียก running เนื่องจากยังไม่มีการทำงานเกิดขึ้น เนื่องจากคอมพิวเตอร์ส่วนใหญ่มี processor ตัวเดียว ทำให้ไม่สามารถ runnable thread ทุกตัวทำงานพร้อมกันได้ runtime system ของจาวาจะต้อง implement หลักการจัดตาราง ซึ่งจะแบ่งการใช้งาน processor ให้กับ runnable thread ทุกตัว

- Not Runnable : thread จะมาอยู่ที่สถานะ not runnable เมื่อเหตุการณ์ใดเหตุการณ์หนึ่งใน 4 เหตุการณ์นี้เกิดขึ้น
  1. เมื่อมีการอ้าง sleep method ของมัน
  2. เมื่อมีการอ้าง suspend method ของมัน
  3. thread ใช้ wait method ของมันเอง เพื่อรอสถานะของ variable
  4. thread ติดอยู่กับ I/O

  การออกจากสถานะ not runnable (ไม่ว่าจะเข้ามาทางไหน) เกิดขึ้นเมื่อ
  1. ถ้า thread เข้าสู่สถานะ sleep จะออกได้เมื่อเวลาที่กำหนดไว้หมดลง
  2. ถ้า thread เข้าสู่สถานะ suspend จะออกได้เมื่อมีการเรียก resume method
  3. ถ้า thread เข้าสู่สถานะ wait เพื่อรอสถานะ variable จะออกได้เมื่อ object ที่เป็นเจ้าของ variable ปล่อยมันโดยการเรียก notify หรือ notifyAll
  4. ถ้า thread ติดอยู่กับ I/O จะออกได้เมื่อ I/O ทำงานเสร็จ

- Dead : thread สามารถสิ้นสุดได้ 2 วิธี คือ สิ้นสุดด้วยตัวเอง (ออกจาก run method ตามปรกติ) หรือ ถูกทำให้สิ้นสุด(หยุด) ซึ่ง thread สามารถถูกทำให้หยุดได้ทุกเวลาโดยการเรียก stop method stop method จะส่ง ThreadDeath object thread จะสิ้นสุดเมื่อมันได้รับ ThreadDeath exception

4.8.3Thread Priority

การจะให้ multithread ทำงานบน CPU ตัวเดียวนั้น จะต้องมีการทำ *scheduling* โดย runtime system ของจาวาจะมีขั้นตอนในการทำ scheduling เรียกว่า *fixed priority scheduling* ซึ่งขั้นตอนเหล่านี้จะจัด thread ตาม priority (ความสำคัญ) ซึ่งจะสัมพันธ์กับ thread อื่น

thread แต่ละตัวในจาวาจะถูกกำหนดค่าความสำคัญให้ซึ่งจะอยู่ระหว่าง MIN_PRIORITY และ MAX_PRIORITY (เป็นค่าคงที่ซึ่งกำหนดใน Thread class) ณ.เวลาใดๆเมื่อ multithread พร้อมที่จะ

ทำงาน thread ที่มีความสำคัญมากที่สุดจะถูก execute ก่อน เมื่อ thread ที่ทำงานอยู่สิ้นสุดหรือ พักการทำงานเท่านั้น thread ที่มีความสำคัญน้อยกว่าถึงจะสามารถทำงานได้

scheduling ของ CPU จะมีสิทธิในการเลือก thread เต็มที่ ถ้า thread ที่มีความสำคัญมากกว่า thread ที่ทำงานอยู่ในขณะนั้น thread ที่มีความสำคัญมากกว่าจะถูกทำ schedule ในทันที

runtime system จะไม่ใช้สิทธิ์ในการเรียกการทำงานคือเพื่อให้ thread อื่นที่มีความสำคัญเท่ากันทำงาน

ในขณะใดๆ thread อาจยกเลิกสิทธิ์ในการทำงานโดยการเรียก yield method thread สามารถมอบ CPU ให้กับ thread อื่นที่มีความสำคัญเท่ากัน (ไม่สามารถให้กับ thread ที่มีความสำคัญต่ำกว่าได้)

เมื่อ runnable thread ในระบบทุกๆตัวมีความสำคัญเท่ากัน runtime system จะทำ scheduling โดยการเลือก thread ที่อยู่ถัดไป

thread ที่มี while loop อยู่ภายใน run method เมื่อทำงานมันจะไม่ปล่อยการควบคุม CPU ให้กับ thread อื่นจนกว่ามันจะสิ้นสุดด้วยตัวมันเอง (while loop สิ้นสุด) หรือจนกระทั่งถูกอ้างสิทธิ์จาก thread ที่มีความสำคัญมากกว่า thread เช่นนี้เรียกว่า *selfish thread* ในบางระบบหาวิธีแก้ selfish thread ด้วยการทำ *time-slicing* time-slicing จะมีบทบาทเมื่อมี runnable thread หลายตัวที่มีความสำคัญเท่ากัน และ thread เหล่านั้นซึ่งเป็น thread ที่มีความสำคัญสูงที่สุดแข่งกันกันเพื่อแย่ง CPU โดยระบบ time-sliced จะแบ่ง CPU ออกเป็นเวลาออกเป็นช่วงๆและจะแบ่งให้กับ thread ทุกๆตัวที่มีความสำคัญเท่ากันและมีความสำคัญสูงสุดเป็นรอบๆ ในแต่ละรอบจะให้ช่วงเวลา thread ละ 1 ช่วงเพื่อให้ทำงานของมัน จะสังเกตุได้ว่า time-sliced จะไม่รับประกันความถี่หรือลำดับการทำ schedule

Daemon Thread thread ในจาวาสามารถเป็น *daemon thread* ได้ daemon thread เป็น thread ที่ให้บริการ thread อื่นที่กำลังทำงานอยู่ใน process เดียวกัน run method ของ daemon thread ทำซ้ำไม่มีที่สิ้นสุดเพื่อรอการขอบริการจาก thread อื่น เมื่อ thread เดียวที่เหลือใน process เป็น daemon thread interpreter จะออกจากการทำงาน (เนื่องจากไม่มี thread อื่นที่ต้องบริการ)

ในการกำหนดว่าให้ thread เป็น daemon thread ทำได้โดยการเรียก setDaemon method พร้อม argument “true” ส่วนการตรวจสอบว่า thread ใดเป็น daemon thread ทำได้โดยใช้ isDaemon method

Thread Group ทุก thread ในจาวาเป็นสมาชิกของ *thread group* thread group เป็นเครื่องมือในการรวม multithread เข้าด้วยกันเป็น object เดียวและใช้ thread เหล่านั้นทั้งหมดพร้อมกัน thread group ถูก implement ใน ThreadGroup class ใน package java.lang

runtime system จะทำรวม thread เข้า thread group ระหว่างที่thread ถูกสร้าง เมื่อคุณสร้าง thread ใหม่ คุณสามารถให้ runtime system รวม thread ของคุณ เข้ากับ thread group ใดๆที่มีอยู่หรือจะตั้ง thread group ใหม่ก็ได้ เมื่อ thread เป็นสมาชิกของ thread group ใดแล้วไม่สามารถจะย้าย thread group ได้

เมื่อโปรแกรมจาวาเริ่มทำงาน runtime system จะสร้าง thread group ชื่อว่า main ถ้าไม่ได้เจาะจงไว้ thread จะถูกรวมเข้าไว้ใน main thread group (แต่ถ้าคุณสร้าง thread ใน Applet thread group ใหม่อาจใช้ชื่ออื่นที่ไม่ใช่ main)

4.8.4 Multithreaded Programs

Synchronizing Threads บ่อยครั้งที่ thread ต้องการใช้ข้อมูลร่วมกันและต้องพิจารณาถึงสถานะและกิจกรรมที่ thread อื่นทำอยู่ โดยกลุ่มของโปรแกรมที่อยู่ในสถานะการณ์นี้เรียกว่า producer / consumer โดย producer จะสร้างข้อมูล ซึ่งจะถูก consumer นำไปใช้ ซึ่งทำให้การทำงานของทั้ง producer และ consumer เข้าจังหวะกัน (synchronize) ปัญหาของการทำงานไม่เข้าจักหวะกันเรียกว่า *race conditions* เกิดขึ้นเมื่อ multithread ที่ทำงานไม่เข้าจังหวะกันเข้าถึง object ตัวเดียวกันในเวลาเดียวกัน ทำให้ได้ผลลัพธ์ที่ผิดพลาด เพื่อป้องกัน race conditions ไม่ให้เกิดขึ้นสำหรับการทำงานแบบ producer / consumer เรามีวิธีแก้ 2 วิธีคือ monitors และ ใช้ notifyAll กับ wait method

4.8.4.1 Monitors : object ที่ถูกใช้ร่มกันระหว่าง thread สองตัวซึ่งการเข้าถึงจะต้องเข้าจังหวะกันเรียกว่า *condition variable* monitors จะเกี่ยวกับการกำหนดข้อมูล (condition variable) และการทำงานเช่น lock ในข้อมูลเหล่านั้น เมื่อ thread ทำ monitor บนข้อมูลตัวใด แล้วมี thread อื่นมาทำการ lock บนข้อมูลตัว จะทำให้ thread ที่ทำ monitor บนข้อมูลนั้น (thread ที่ไม่ได้เป็นผู้ lock) ไม่สามารถใช้ข้อมูลตัวนั้นได้ ส่วนของโค้ดที่อยู่ในโปรแกรมที่เข้าถึงข้อมูลตัวเดียวกันใน thread เรียกว่า *critical sections* ในจาวาคุณสามารถกำหนด critical sections ในโปรแกรมของคุณได้โดยใช้ synchronized keyword

4.8.4.2 norifyAll และ wait methods : method ทั้ง 2 ตัวนี้เป็นสมาชิกของ java.lang.Object class

Note : notifyAll และ wait method ถูกอ้างโดย thread ที่ถือ lock เท่านั้น

notifyAll method : ใช้ในการบอกให้กับ thread ตัวอื่นๆที่รอ บน monitor โดย thread ปัจจุบันให้ตื่นขึ้น หนึ่งใน thread ที่รออยู่ก็จะทำ monitor และทำงานต่อ

wait method : จะทำให้ thread ปัจจุบันหยุดการทำงานจนกว่า thread อื่นจะบอกมันถึงสถานะที่เปลี่ยนไป

บทที่ 5
เกมส์แบบ Multiuser

เกมส์แบบ Multiuser ที่จะกล่าวถึงนี้คือ เกมส์ Puyo เป็นเกมส์ที่สามารถเล่นกันได้หลายคนผ่านทางอินเตอร์เน็ต โดยเป็นเกมที่มีรูปแบบการเล่นคล้ายคลึงกับเกมส์ Tetris แต่มีข้อแตกต่างกันตรงตัวละคร วิธีการลบตัวละคร ซึ่งจะกล่าวต่อไปในหัวข้อต่อไป

5.1 บทนำ

เกมส์ Multiuser Puyo เป็นเกมส์แบบ Puzzle ที่มีผู้เล่นหลายคน แต่ละคนอาจจะอยู่ตรงจุดไหนก็ได้ หรือใช้เครื่องชนิดไหนก็ได้ ไม่ว่าจะเป็น Sun หรือ PC ธรรมดา เพราะเป็นเกมส์ที่ไม่ขึ้นกับแพลทฟอร์มใดๆ เนื่องจากเขียนด้วยภาษาจาวา มีการควบคุมตัวละครโดยใช้คีย์บอร์ดตามคีย์ที่กำหนดในการเลื่อนไปทางขวา-ซ้าย-ลง หรือหมุนตัวละคร และเป็นเกมส์แบบ event-base game หรือเกมส์ที่ดำเนินตามอินพุทของเหตุการณ์ที่ป้อนเข้ามา คือเกมส์จะคอยตรวจสอบคีย์บอร์ดที่ผู้เล่นป้อนเข้ามา แล้วปฏิบัติงานตามคำสั่งที่สอดคล้องกับคีย์ที่กดนั้นๆ ทำเช่นนี้จนกว่าจะจบเกมส์ (Game Over)

5.2 แนวการออกแบบ (Design)

ในการออกแบบเกมส์ Puyo เริ่มต้นด้วยขั้นตอนดังต่อไปนี้

5.2.1 การออกแบบ Storyline ของเกมส์

ขั้นตอนนี้เป็นการออกแบบภาพโดยรวมของเกมส์ คือ เกมส์มีการดำเนินอย่างไร ตั้งแต่เริ่มต้นเกมส์จนจบเกมส์ จากการออกแบบได้ผลสรุปดังนี้

เกมส์เมื่อมีการเริ่มต้นจะมีตัวละครที่ใช้ในการเล่นปัจจุบัน และตัวละครอนาคต ซึ่งตัวละครปัจจุบันจะแตกต่างกับตัวละครอนาคตตรงที่ตัวละครปัจจุบันสามารถควบคุมโดยคีย์บอร์ดจากผู้เล่นได้ แต่ตัวละครอนาคตจะควบคุมจากผู้เล่นไม่ได้ และรูปแบบของการเรียงตัวละครจะเป็นแถวและคอลัมน์ ซึ่งถ้าหากตัวละครได้ตกมากระทบกับคอลัมน์ใดก็ตามก็วางตัวเองอยู่บนคอลัมน์นั้น และถ้าหากแถวที่ต่ำกว่าของตัวละครเป็นพื้นที่ว่างตัวละครจะต้องตกจนกว่าจะถึงคอลัมน์ที่มีตัวละครอื่นอยู่หรือถึงพื้น และเมื่อตัวละครตกถึงพื้นจะต้องมีการเปลี่ยนแปลงหน้าตาของตัวละคร เช่น ถ้าหากตัวละครใกล้เคียงมีสีเดียวกันจะต้องเปลี่ยนให้เข้ากัน ซึ่งดูเหมือนว่ามีการดึงกันระหว่างตัวละครสีเดียวกัน หรือถ้าหากมีการระเบิดก็จะต้องเปลี่ยนหน้าตาให้เป็นตกใจ และในช่วงที่มีการเกิดเหตุการณ์ต่างๆไม่ว่าจะกดคีย์บอร์ดหรือการระเบิดจะต้องมีเสียงร้องออกมา

หลังจากที่หน้าตาของตัวละครแล้ว จะเข้าสู่ขั้นตอนของการลบตัวละครที่ถูกต้องตามกติกา คือตัวละครจะถูกลบได้ก็ต่อเมื่ออยู่ใกล้กันตั้งแต่สี่ตัวขึ้นไป และหลังจากทำการลบตัวละครตัวละครอื่นๆที่แถว

ล่างของมันมีการลบจะตกลงมาจนกว่าจะถึงคอลัมน์ที่มีตัวละครหรือถึงพื้น จากนั้นจะทำการเปลี่ยนแปลงหน้าตาของตัวละคร และลบตัวละครอีกครั้ง ทำอย่างนี้เรื่อยๆจนกว่าจะไม่มีตัวละครอยู่ใกล้กันตั้งแต่สี่ตัวขึ้นไปอีกแล้ว

เมื่อจบขั้นตอนข้างต้นจะมีการเพิ่มคะแนนและจำนวนตัวละครที่ลบได้ซึ่งจะแสดงผลตรงหน้าจอเหมือนกัน และส่งผลกระทบไปยังผู้เล่นคนอื่นคือผู้เล่นคนอื่นจะได้รับตัวละครสีดำซึ่งไม่สามารถลบได้โดยวิธีปกติ แต่จะลบได้เมื่อตัวละครใกล้เคียงที่ไม่ใช่สีเดียวกันเกิดการลบขึ้น จากนั้นก็จะให้ตัวละครอนาคตเป็นตัวละครเป็นปัจจุบันและสร้างตัวละครอนาคตตัวใหม่แทนทำอย่างนี้ไปเรื่อยๆจนกว่าจะจบเกมส์ คือไม่สามารถเคลื่อนย้ายตัวละครปัจจุบันได้อีกต่อไป

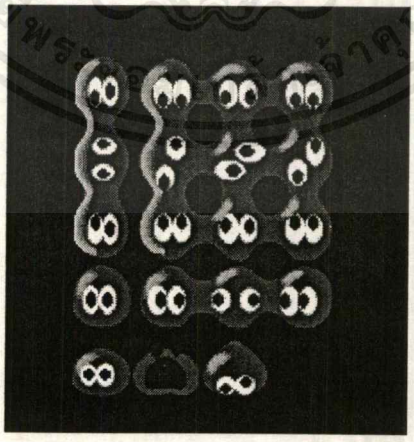
5.2.2 การออกแบบการป้อนอินพุทของผู้เล่น

ซึ่งในขั้นตอนนี้เป็นวิเคราะห์หาคีย์ที่จะใช้ในการควบคุมตัวละครที่ใช้ในเกมส์ ซึ่งผลสุดท้ายจากการวิเคราะห์ ตกลงจะใช้คีย์ดังต่อไปนี้ในการควบคุม

คีย์ลูกศรขวาสำหรับการเลื่อนไปทางขวา
คีย์ลูกศรซ้ายสำหรับการเลื่อนไปทางซ้าย
คีย์ลูกศรลงสำหรับการตกลงอย่างรวดเร็ว
คีย์ลูกศรบนสำหรับการหมุน

5.2.3 การออกแบบตัวละคร

ในขั้นตอนนี้เป็นขั้นตอนออกแบบตัวละครที่จะใช้ในเกมส์ ซึ่งผลที่ได้ตกลงใช้ตัวละครที่มีหน้าตาดังรูปที่ 5 – 1 ซึ่งตัวละครที่สีเดียวกัน จะมีอากัปกิริยาต่างๆกันไปตามสภาวะของตัวละคร



รูปที่ 5 – 1 แสดงหน้าตาต่างๆของตัวละคร

5.2.4 การออกแบบหน้าจอของเกมส์

มีการออกแบบให้หน้าจอประกอบด้วย 3 ส่วนคือ ส่วนการควบคุมซึ่งประกอบด้วยปุ่มButton และ Choice ส่วนที่สองคือส่วนแสดงผลของตัวละคร และส่วนที่สามคือส่วนการแสดงคะแนนและตัวละครถัดไป

5.2.5 การออกแบบการ Interactive ระหว่างผู้เล่น

ขั้นตอนนี้เป็นการออกแบบว่าเกมส์จะส่งผลโต้กลับแก่ผู้เล่นคนอื่นเมื่อใด จากการวิเคราะห์ได้ผลสรุปว่า เมื่อมีการลบตัวละครทิ้งไปเกิน 5 ตัว จะส่งผลโต้กลับแก่ผู้เล่นคนอื่นที่อยู่บนเครือข่าย

5.3 หลักการเขียนโปรแกรม

แนวการดำเนินเกมส์ Multiuser Puyo มีหลักการต่อไปนี้

ที่ฝั่งไคล์เอ็นท์ โปรแกรมประกอบด้วยmethodหลักๆดังนี้

1. CheckMap()

เป็นmethodที่จะถูกเรียกใช้เมื่อตัวละครปัจจุบันได้แตะพื้น

2. CheckDropPlace()

เป็นmethodที่ให้ตัวละครปัจจุบันที่ยังไม่แตะพื้นตกลงมา จนกว่าจะถึงพื้น

3. NowDelete()

เป็นmethodที่ใช้ในการลบตัวละครที่มีค่าสภาวะการอยู่ใกล้กันเกินสามตัว

4. SaveNewMap()

เป็นmethodที่ใช้ในการเก็บตาราง map ตัวละครใหม่เมื่อมีการลบ

5. SaveBlock()

เป็นmethodที่ถูกเรียกใช้เมื่อมีการลบตัวละครแล้วมีตัวละครไม่ได้เตะพื้น

6. CheckLine()

เป็นmethodที่ใช้ในการตรวจสอบว่าสามารถจะลบตัวละครที่อยู่ใกล้กันตัวไหนได้บ้าง

7. CheckDelete()

เป็นmethodที่เรียกหลังจากใช้method CheckLine() เพื่อทำการเช็คอีกครั้งในแนวที่แตกต่างจาก CheckLine()

8. CheckState()

เป็น method ที่ถูกเรียกสำหรับการตรวจสอบสภาวะของตัวละคร เพื่อทำการเปลี่ยนหน้าตาของตัวละครตามสภาวะปัจจุบัน

9. SendLine()

เป็น method ที่เรียกใช้เพื่อทำการส่งผลไปยัง Server เพื่อให้ส่งตัวละครสีดำไปยังผู้เล่นคนอื่น

10. GetNextRandomShape()

เป็น method ที่เรียกใช้เมื่อต้องการรูปแบบของตัวละครสีอื่นๆ

11. AddSuffer()

เป็น method ที่เรียกใช้เมื่อมีการรับข้อมูลจาก Server

12. ส่วนที่ทำการติดต่อกับเซิร์ฟเวอร์

ซึ่งส่วนตรงนี้ประกอบส่วนที่ทำการติดต่อกับเซิร์ฟเวอร์ และส่วนที่ทำการรับส่งข้อมูลกับเซิร์ฟเวอร์

ส่วนที่เซิร์ฟเวอร์จะมีส่วนของโปรแกรมหลักดังนี้

1. ส่วนของการรอการติดต่อจากฝั่งไคล์เอ็นท์

เป็นส่วนที่จะรอเพื่อทำการติดต่อจากฝั่งไคล์เอ็นท์ โดยไคล์เอ็นท์จะเริ่มการติดต่อกับเซิร์ฟเวอร์เมื่อทำการโหลดคลาสนี้ขึ้นทำการรัน

2. ส่วนของการรับข้อมูลที่ส่งมาจากไคล์เอ็นท์

เป็นส่วนที่จะทำงานเมื่อมีการติดต่อกับไคล์เอ็นท์สำเร็จเป็นที่เรียบร้อย โดยจะทำการอ่านค่าจากซ็อกเก็ตอยู่เรื่อยๆ ที่แต่ละไคล์เอ็นท์ติดต่อเข้ามา

3. ส่วนของการส่งข้อมูลที่ได้รับมาไปยังไคล์เอ็นท์

เป็นส่วนที่จะคอยส่งผลต่างๆไปยังไคล์เอ็นท์

4. ส่วนของการยกเลิกการติดต่อจากฝั่งไคล์เอ็นท์

เป็นส่วนที่จะทำการตรวจสอบซ็อกเก็ตของไคล์เอ็นท์ ถ้าหากพบว่าซ็อกเก็ตได้ถูกยกเลิกการใช้หรือไม่สามารถติดต่อกันได้อีกต่อไป จะต้องทำการยกเลิกการติดต่อกับไคล์เอ็นท์นั้นเสีย

5.4 กติกาเกมส์

กติกาการเล่นเกมส์แบ่งออกได้เป็น 2 ส่วนคือกติกาการเล่นแบบเล่นคนเดียว กับเล่นหลายคน โดยที่

5.4.1 กติกาการเล่นแบบเล่นคนเดียว

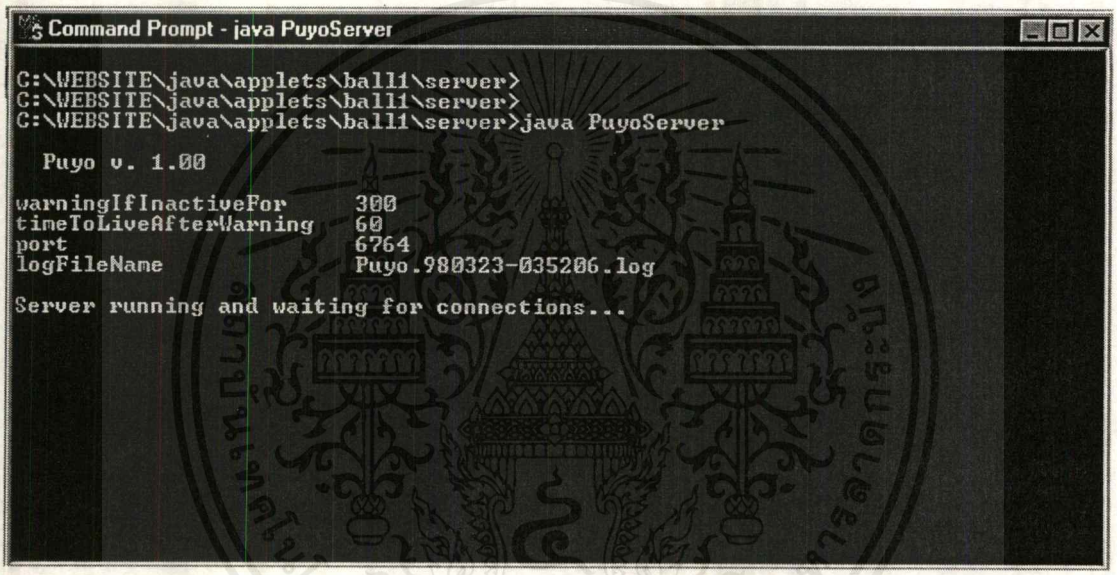
เมื่อผู้เล่นได้ทำการควบคุมตัวละครปัจจุบันจนตกถึงพื้น ก็จะมีการตรวจสอบสถานะปัจจุบันของตัวละครว่ามีการอยู่ใกล้กันกี่ตัวและการตรวจสอบสถานะเพื่อการเปลี่ยนแปลงหน้าตาของตัวละคร ซึ่งถ้าหากตรวจสอบพบว่าตัวละครสีเดียวกันอยู่ใกล้กันตั้งแต่สี่ตัวเป็นต้นไป จะต้องมีการลบตัวละครนั้นออกไปพร้อมกับการเพิ่มจำนวนของตัวละครที่ถูกลบและคะแนนที่แสดงอยู่ในส่วนรายงานผลการเล่น และหลังจากที่มีการลบตัวละครออกไป จะต้องให้ตัวละครที่อยู่เหนือตัวละครที่ถูกลบตกลงมาจนถึงพื้น จากนั้นก็จะมีการตรวจสอบสถานะปัจจุบันของตัวละครและสถานะของการเปลี่ยนแปลงหน้าตาอีกครั้ง ซึ่งถ้าหากพบว่าตัวละครสีเดียวกันอยู่ใกล้กันสี่ตัวขึ้นไปอีกจะต้องลบตัวละครนั้นอีกครั้ง พร้อมกับเพิ่มคะแนนและจำนวนตัวละครที่ลบ ทำตามขั้นตอนข้างต้นเรื่อยๆ แต่ถ้าหากตรวจสอบพบว่าไม่มีตัวละครอยู่ใกล้กันเกินสามตัวอีกแล้ว ก็จะได้รับตัวละครตัวใหม่และดำเนินการเล่นต่อไป

5.4.2 กติกาการเล่นแบบหลายคนเล่น

ในกรณีที่มีการเล่นหลายคน ถ้าหากมีผู้เล่นคนใดคนหนึ่งมีการลบตัวละครตั้งแต่ 5 ตัวขึ้นไป จะต้องมีการส่งผลต่อผู้เล่นคนอื่นให้ได้รับตัวขยะซึ่งเป็นสีดำตามอัตราของจำนวนตัวละครที่ลบไป ซึ่งตัวขยะนี้เมื่อได้รับมากขึ้น และมีจำนวนการอยู่ใกล้ตั้งแต่สี่ตัวขึ้นไป ก็ไม่สามารถลบได้จะคงอยู่ในเกมส์ไปเรื่อยๆ จนกว่าตัวละครสีอื่นที่ไม่ใช่ขยะและอยู่ใกล้กันทางขวาถูกลบ ตัวขยะก็จะถูกลบไปด้วย

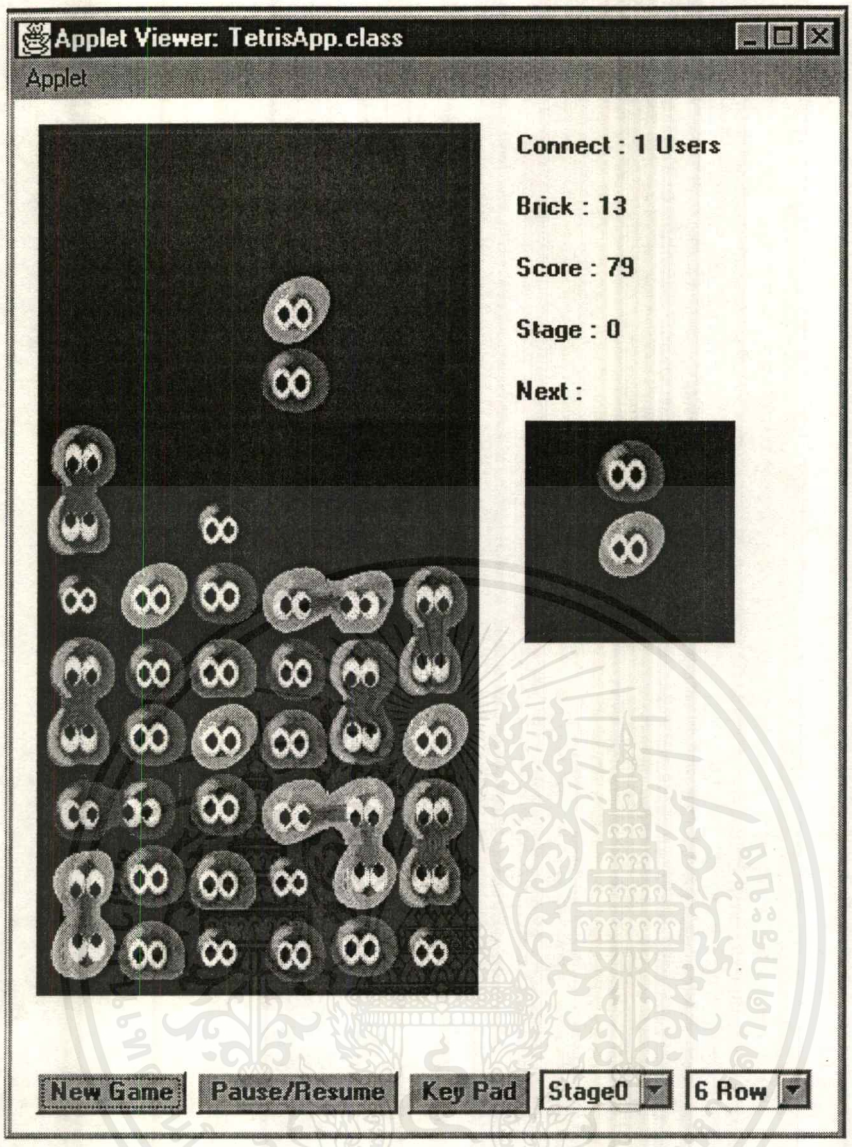
กติกาการจบเกมส์ทั้งแบบเล่นคนเดียวและหลายคนจะเกิดขึ้นเมื่อผู้เล่นไม่มีที่ว่างพอที่จะให้ตัวละครปัจจุบันทำการหมุนหรือเคลื่อนที่ไปทางไหนได้เลย

5.5ตัวอย่างหน้าจอของเกมส์

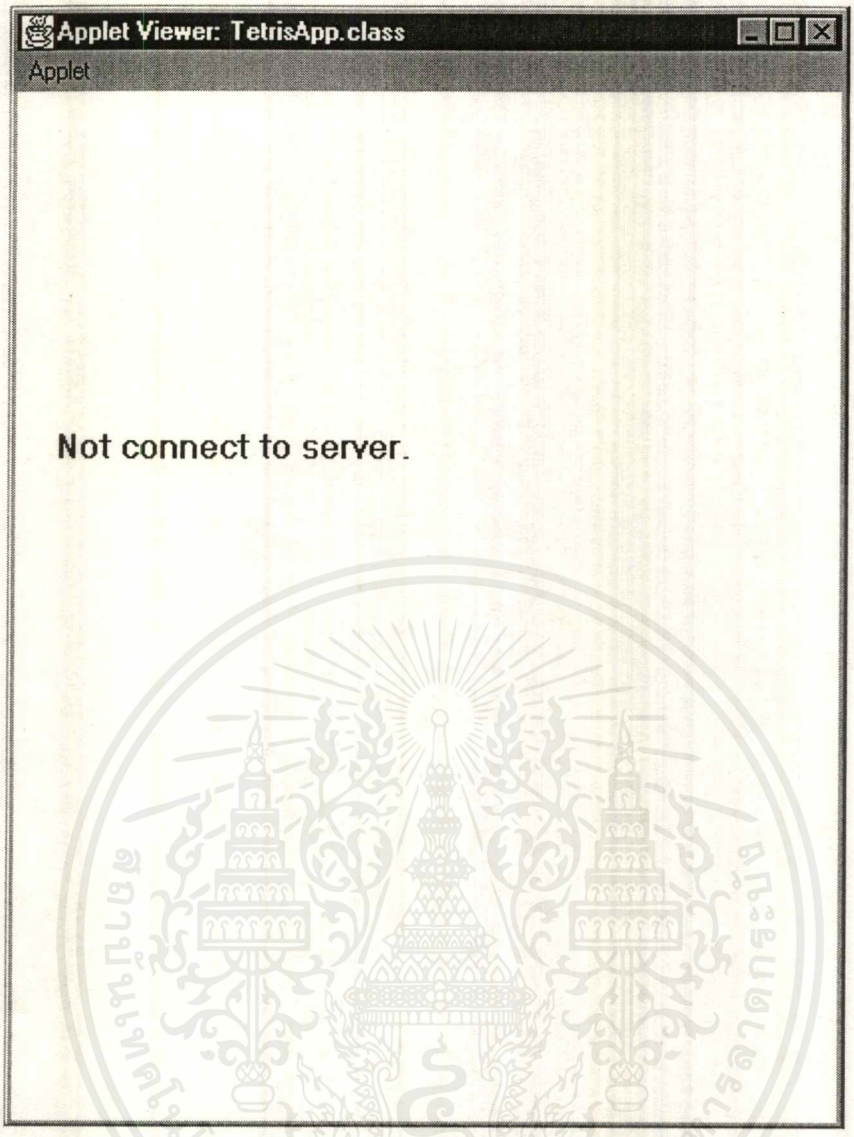


รูปที่ 5 – 2 แสดงหน้าจอการทำงานของ Server ที่กำลังรอการติดต่อจาก Client

รูปที่ 5 – 3 แสดงหน้าจอการทำงานของ Server หลังจากทำการติดต่อกับ Client เรียบร้อยแล้ว



รูปที่ 5 – 4 แสดงผลการทำงานที่ฝั่งไคล์เอ็นท์



รูปที่ 5 – 5 แสดงหน้าจอเมื่อไคล์เอ็นท์ไม่สามารถติดต่อกับ Server ได้

บทที่ 6
คาราโอเกะ

คาราโอเกะเป็นรูปแบบใหม่ของการเล่นเพลงบนเว็บบราวเซอร์จากเดิมที่ส่วนใหญ่ผู้เล่นอินเตอร์เน็ต จะได้รับความบันเทิงจากการฟังเพลงเพียงอย่างได้เดียว ไม่มีโอกาสจะได้ร้องเพลงผ่านเว็บ

6.1 แนวการทำงานของคาราโอเกะ

คาราโอเกะผ่านเว็บมีรูปแบบเหมือนคาราโอเกะโดยทั่วๆไป จะมีข้อแตกต่างก็เพียงแต่เนื้อเพลงของคาราโอเกะทั่วไปจะเป็นการเปลี่ยนสีของเนื้อร้อง แต่คาราโอเกะผ่านเว็บจะมีลูกบอลเคลื่อนที่ตามเนื้อร้องแทน และไม่วีดีโอประกอบการร้องเพลง

6.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนการออกแบบเป็นขั้นตอนแรกในการเขียนโปรแกรม โดยได้มีการออกแบบแต่ละส่วนดังนี้

1. กำหนดรูปแบบหน้าจอของการแสดงผล

จากการวิเคราะห์ได้ผลสรุปของหน้าจอแสดงผลคือ หน้าจอจะประกอบด้วยส่วนที่แสดงเนื้อร้อง ส่วนที่แสดงเวลาในการเล่นเพลง และส่วนที่ควบคุมการทำงานโปรแกรมโดยใช้ปุ่ม Button โดยที่โปรแกรมที่ใช้ในการอัดเพลงให้เป็นไปตามจังหวะเพลง จะประกอบด้วยปุ่มสำหรับ Record , Stop และแสดงเวลาที่ได้ทำการอัดไว้ ส่วนโปรแกรมที่ใช้ในการเล่นเพลงให้เป็นไปตามจังหวะจะประกอบด้วยปุ่มสองปุ่มคือ ปุ่มสำหรับ Play และ Stop

2. กำหนด Algorithm ในการอัดเพลงให้เป็นไปตามจังหวะเพลง

ซึ่งในขั้นตอนนี้ได้คิดค้นวิธีในการอัดจังหวะเพลง ตามทำนองจริง โดยใช้คลาสที่เขียนขึ้นมาสำหรับการอัดเพลง และมีการเขียนเนื้อร้องของเพลงที่อัดตามรูปแบบที่กำหนดไว้ ซึ่งการอัดเพลงจะเกิดขึ้นเมื่อมีการกดปุ่ม Play ในแอพเพล็ทนี้ ซึ่งในการทำงานตรงนี้จะมีคลาสที่คอยทำงานคล้ายนาฬิกาเมื่อผู้อัดเพลงได้กดคีย์ใดๆ ลูกบอลจะถูกแสดง ณ ตำแหน่งของเนื้อร้องถัดไป พร้อมกับเขียนเวลาที่กดคีย์ลงไปใน textArea ที่เก็บไว้ในอีกวินโดวนึง ทำอย่างนี้ไปเรื่อยๆ จนกว่าจะจบเพลง

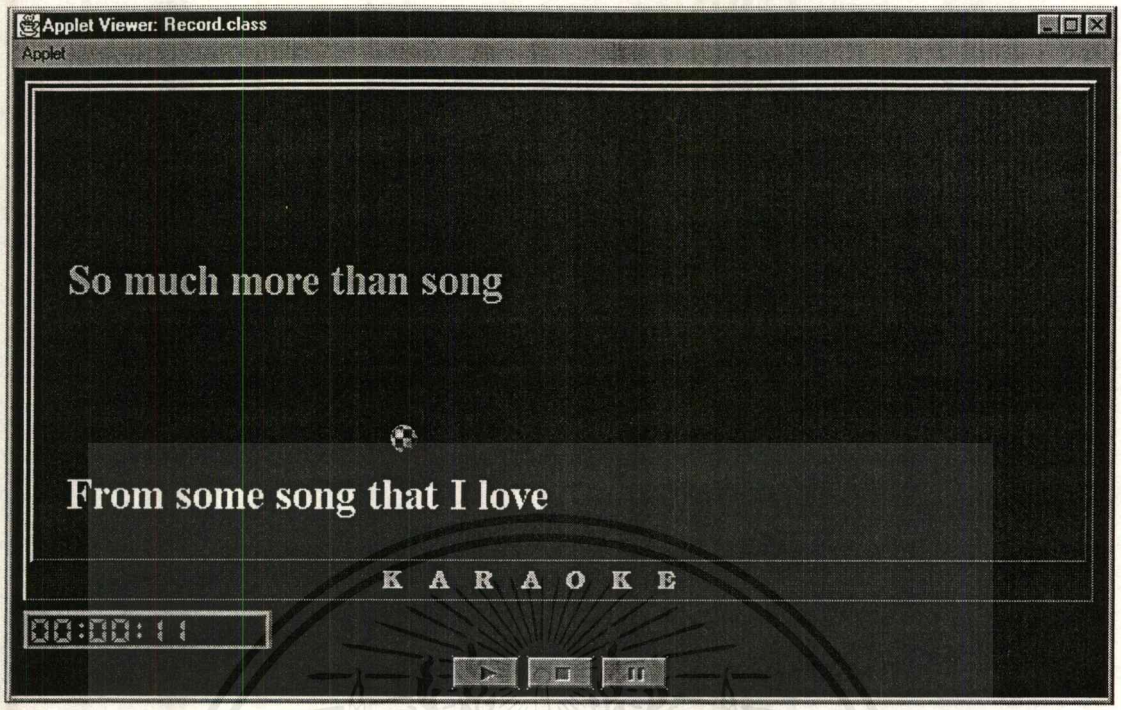
3. กำหนด Algorithm ในการนำจังหวะเพลงที่ได้อัดไว้มาทำการเล่นตามเสียงเพลงจริงๆ

ซึ่งหลังจากเสร็จสิ้นขั้นตอนที่ 2 แล้วเราจะได้จำนวนตัวเลขจำนวนหนึ่งซึ่งเป็นตัวเลขที่ระบุเวลาในการแสดงลูกบอล ณ ตำแหน่งเนื้อร้องนั้น ซึ่งโปรแกรมส่วนนี้จะมีการคำนวณเวลา และระยะทางที่ลูกบอลจะเคลื่อนที่ไปในแต่ละวรรคของเนื้อร้อง โดยลูกบอลจะเคลื่อนที่เป็นแนววิถีโค้ง

หลังจากที่ทำการออกแบบเสร็จสิ้นแล้ว ก็ถึงขั้นตอนการเขียนโปรแกรมตามที่ออกแบบไว้ โดยใช้เครื่องมือในการพัฒนาที่พิจารณาไว้ ส่วนกระบวนการทดลอง และทดสอบการทำงานจะกล่าวในบทต่อไป

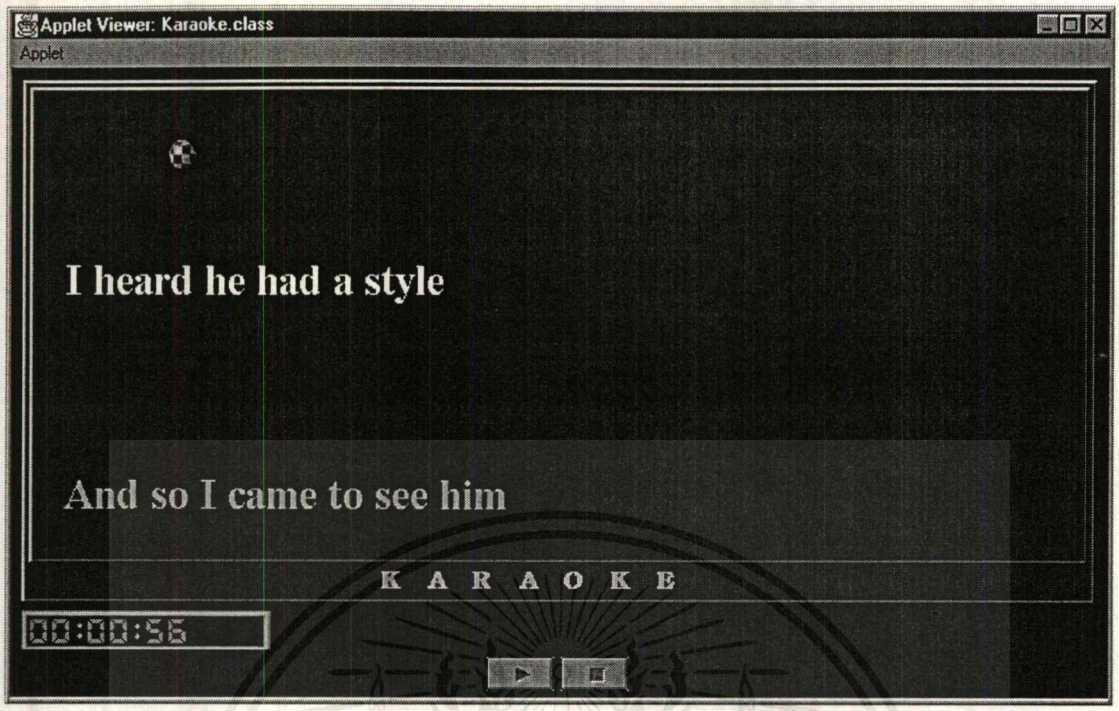
6.3 ตัวอย่างหน้าจอการทำงานโปรแกรมคาราโอเกะ

รูปที่ 6 – 1 แสดงหน้าจอแอพเพล็ทที่ขณะก่อนการอัดเพลง



รูปที่ 6 – 2 แสดงหน้าจอแอพเพล็ทที่ขณะอัดเพลง

รูปที่ 6 – 3 แสดงผลการอัดเพลงที่ได้รับผลลัพธ์เป็นจำนวนตัวเลขที่แทนเวลา



รูปที่ 6 – 4 แสดงหน้าจอแอพเพล็ทที่ทำการเล่นเพลงคาราโอเกะ

บทที่ 7
การทดลองและทดสอบแอพพลิเคชัน

เนื่องจากแอพพลิเคชันที่พัฒนามี 2 แอพพลิเคชัน จึงได้แบ่งหัวข้อการทดลองเป็น 2 หัวข้อดังนี้

7.1 การทดลองและทดสอบเกมส์ Multiuser

ประกอบด้วย 2 ส่วนคือฝั่งไคล์เอ็นท์ และฝั่งเซิร์ฟเวอร์ โดยที่ฝั่งไคล์เอ็นท์ จะเน้นในการตรวจสอบการดำเนินของเกมส์ ซึ่งเกมส์ที่ถูกต้องจะต้องดำเนินไปตาม StoryLine ที่ได้วางไว้ โดยที่สามารถทำการตรวจสอบเป็นส่วนๆไป ซึ่งส่วนที่จะทดสอบในตอนนี้ประกอบด้วย

1. การตรวจการสร้างตัวละครตัวปัจจุบันกับอนาคตว่าเป็นตามรูปแบบที่กำหนดหรือไม่
2. การตรวจการเปลี่ยนสถานะจากอนาคตเป็นปัจจุบันว่าสอดคล้องตรงกันหรือไม่
3. การตรวจสอบการลบตัวละคร ว่าเป็นไปตามกติกาหรือไม่
4. การตรวจสอบการแสดงหน้าตาของตัวละคร ให้เป็นไปตามสถานะการอยู่ใกล้กัน
5. การตรวจสอบการเล่นเสียงตามเหตุการณ์ของเกมส์ ให้เป็นไปตามความถูกต้อง
6. การตรวจสอบส่วนของการควบคุมการเคลื่อนที่ของตัวละครปัจจุบัน
7. การตรวจสอบการตกลงของตัวละครเมื่อมีการลบตัวละครเกิดขึ้น

หลังจากที่การตรวจสอบการทำงานถูกต้อง ก็มาถึงขั้นตอนของการติดต่อกับระบบเครือข่าย ซึ่งทางฝั่งไคล์เอ็นท์จะต้องรับ-ส่งข้อมูลอยู่เรื่อยๆ โดยที่เมื่อมีการลบตัวละครมากกว่าสี่จะต้องส่งผลโต้กลับไปยังเซิร์ฟเวอร์เพื่อที่จะส่งต่อผู้เล่นคนอื่นต่อไป โดยจะมีการตรวจสอบว่าข้อมูลที่ฝั่งไคล์เอ็นท์ส่งไปยังเซิร์ฟเวอร์มีค่าเหมือนกันกับที่เซิร์ฟเวอร์รับหรือไม่ ถ้าผู้ส่งเป็นบุคคลเดียวกัน

สำหรับทางฝั่งเซิร์ฟเวอร์จะต้องตรวจสอบการทำงานให้สามารถรอการเชื่อมต่อจากไคล์เอ็นท์ได้ และจะต้องทำการ synchronization ผู้เล่นแต่ละคนได้ นอกจากนี้จะต้องตรวจสอบส่วนการรับ-ส่งข้อมูลของผู้เล่น และการยกเลิกการติดต่อกับผู้เล่น

และทางฝั่งไคล์เอ็นท์เมื่อมีการรับข้อมูลจากเซิร์ฟเวอร์ จะมีการทดสอบการสร้างตัวละครขยะแก่ผู้เล่น ตามอัตราการลบที่เกิดขึ้นด้วย

จากการทดลองปัญหาที่เกิดขึ้นส่วนใหญ่เกิดจากระบบเครือข่ายที่มีการถ่ายข้อมูลจำนวนมาก ทำให้โปรแกรมทำการส่งข้อมูลไปสู่ผู้เล่นได้ช้ามาก ซึ่งพลอยทำให้เกมส์ทำงานช้าไปด้วย

นอกจากนี้เซิร์ฟเวอร์จะต้องรองรับการทำงานของผู้เล่นหลายคน และเซิร์ฟเวอร์มีขนาดเล็กทำให้การทำงานของโปรแกรมตรงส่วนนี้ก็ช้าไปด้วย

7.2 การทดลองและทดสอบคาราโอเกะ

ในการเขียนโปรแกรมคาราโอเกะประกอบด้วยสองส่วนคือ แอพเพล็ทที่ใช้ในการอัดเพลง และแอพเพล็ทที่ใช้ในการเล่นเพลง ในการทดลองการทำงานของคลาสเหล่านี้ ประกอบด้วยขั้นตอนต่อไปนี้

1. การหาไฟล์เสียงที่ต้องการ และมีเนื้อร้อง
2. จากนั้นเมื่อมีเนื้อร้องแล้วก็ทำการเขียนเนื้อร้องลงในไฟล์ HTML ตามรูปแบบต่อไปนี้

```
<applet code="Record.class"  width=782 height=441>
<Param Name="Song" Value="songname">
<Param Name="Count" Value="n">
<Param Name="Lyrics1" Value="Str1 ">
<Param Name="Lyrics2"  Value="Str2 ">
<Param Name="Lyrics3" Value="Str3 ">
<Param Name="Lyrics4" Value=" Str4  ">
                ...
<Param Name="Lyricsn" Value="Strn ">
</applet>
```

โดย Recode.class คือคลาสที่ใช้ในการอัดเพลง
songname คือชื่อเพลงที่จะใช้อัดจังหวะ
n คือจำนวนบรรทัดของเนื้อเพลงที่จะแสดงในแอพเพล็ท
Strn คือเนื้อร้องที่จะแสดงในแต่ละบรรทัด และจะต้องมีช่องว่าง 1 ช่องหลังตัวอักษร

3. เมื่อทำการเขียนไฟล์ HTML เสร็จแล้วก็จะถึงขั้นตอนของการเรียก HTML และจาวาไฟล์โดยใช้โปรแกรมสำหรับการพัฒนาคือ Visual Café Pro แล้วทำการอัดเพลงตามจังหวะเพลง เมื่อทำการอัดเพลงเสร็จแล้ว ก็จะได้จำนวนตัวเลขจำนวนหนึ่งทำการบันทึกค่านี้ไว้

4. หลังจากนั้นเอาจังหวะเพลงที่ได้มาใส่หลังไฟล์ HTML ที่เขียนไว้แล้วข้างต้น โดยมีรูปแบบของการใส่จังหวะดังนี้

```
<applet code="Karaoke.class"  width=782 height=441>
<Param Name="Song" Value="songname">
<Param Name="Count" Value="n">
<Param Name="Lyrics1" Value="Artist: Michael Jackson /7200/800/900/">
<Param Name="Lyrics2" Value="Album: Dangerous /800/800/">
```

<Param Name="Lyrics3" Value="Song: Heal The World /500/800/300/200/">

...

<Param Name="Lyricsn" Value="The end /200/200/">

</applet>

และเปลี่ยนชื่อคลาสไฟล์เป็น Karaoke.class ซึ่งใช้ในการเล่นไฟล์คาราโอเกะได้

5. ทำการรันไฟล์ HTML และจาวาไฟล์ในโปรแกรมสำหรับการพัฒนาอีกครั้งเพื่อทำการตรวจสอบว่าโปรแกรมทำงานถูกต้องตามที่ออกแบบหรือไม่

6. ถ้าหากเจอข้อผิดพลาดก็ต้องกลับไปแก้ไข โดยปัญหาที่เจอในการทดลอง ทดสอบโปรแกรม คือ การเคลื่อนที่ของลูกบอลไม่สอดคล้องกับจังหวะเพลง เนื่องจากระหว่างการอัดเพลงอาจกดคีย์เร็วเกินไป หรือการเขียนโปรแกรมคำนวณไม่ถูกต้อง และบางครั้งการเคลื่อนที่ของลูกบอกผิดพลาด เช่น เคลื่อนที่เลยผ่านเนื้อร้องไป ซึ่งก็เกิดจากการคำนวณระยะทางผิดพลาด

7. หลังจากแก้ไขเป็นที่เรียบร้อยแล้วก็สามารถนำมารันบนเว็บได้ และเมื่อนำมาแสดงบนเว็บปรากฏว่า การเคลื่อนที่ของลูกบอลทำได้ช้ากว่าเพลงที่ได้เล่นไปแล้ว ซึ่งผลจากการวิเคราะห์เกิดจากระบบเครือข่าย ซึ่งโหลดเสียงได้เร็วกว่าภาพ

บทที่ 8
บทสรุปผลและวิจารณ์

8.1 บทวิจารณ์

ในการทำโครงงานวิจัยครั้งนี้ นอกจากจะได้รับความรู้จากการศึกษาการพัฒนาแอพพลิเคชันโดยการใช้ภาษาจาวาแล้ว ยังสามารถเขียนโปรแกรมที่สามารถทำงานบนเว็บบราวเซอร์ได้ นอกจากนี้ในขณะที่กำลังทำโครงงานอยู่ ก็ได้มีผู้คิดค้นเขียน API ตัวใหม่ของภาษาจาวาขึ้นมาอยู่เรื่อยๆ ซึ่งแสดงให้เห็นว่าภาษาจาวามีการพัฒนาอยู่เรื่อยๆ การเขียนแอพพลิเคชันด้วยภาษาจาวาในอนาคตจะทำได้อย่างมีประสิทธิภาพ และมากด้วยความสามารถกว่านี้ รูปแบบการทำงานซับซ้อนขึ้น เนื่องจาก API ของจาวามีความสามารถในการทำงานเพิ่มขึ้น เช่นการใช้ API Java Media Framework ในการขนถ่ายข้อมูลแบบ On-line Video โดยใช้โปรโตคอล RTP หรืออาจเขียนโปรแกรม Internet Phone ผ่านเว็บโดยใช้ API Java Telephony ทำการพัฒนา โดย API ทั้งสองตัวนี้ยังเป็นรุ่น beta อยู่ และในการพัฒนาจำเป็นต้องใช้อุปกรณ์ที่เป็นของ Sun โดยเฉพาะเช่น การ์ดวีดีโอ

8.2 สรุปผลการทดลอง

ในโครงงานนี้ได้ทำการพัฒนาแอพพลิเคชันที่สามารถทำงานบนระบบเครือข่ายอินเตอร์เน็ตโดยแสดงบนเว็บบราวเซอร์ได้ คือ

1. เกมส์ Multiuser Puyo ที่พัฒนาโปรแกรมด้วยภาษาจาวา
2. คาราโอเกะ ที่พัฒนาโปรแกรมด้วยภาษาจาวา

ซึ่งในการพัฒนาโปรแกรมทำให้ทราบว่า ในการพัฒนาโปรแกรม ต่างก็มีโปรแกรมที่หลากหลายเพื่อใช้ในการพัฒนา เช่น Visual Café Pro หรือ Java Development Kit และโปรแกรมอื่นๆที่ช่วยในการพัฒนาส่วนอื่นๆ ฉะนั้นในการพิจารณาว่าจะเลือกใช้เครื่องมือใดในการพัฒนาโปรแกรมควรจะพิจารณาถึงความคล่องตัวในด้านต่างๆ เพื่อการพัฒนาโปรแกรมที่มีประสิทธิภาพ และสะดวกง่ายดาย

จากการศึกษาและทำโครงงานมาทั้งหมด ทำให้ทราบถึงความสามารถที่มากมายของจาวา ที่ทำให้การค้นหาข้อมูล หรือการนำเสนอข้อมูลบนเว็บมีความน่าสนใจขึ้น

8.3 แนวทางการพัฒนาต่อ

1. เกมส์ Multiuser Puyo ยังประสบปัญหาการทำงานของโปรแกรมช้าเมื่อมีการส่งข้ามเครือข่ายไกล เนื่องจากระบบเครือข่ายมีการส่งข้อมูลจำนวนมาก และเครื่องเซิร์ฟเวอร์ที่ใช้มีการปฏิบัติงานเยอะเมื่อมีผู้ใช้ไคล์เอ็นท์ติดต่อเข้ามาเยอะ ดังนั้นอาจจะปรับปรุงการให้ทำงานโดยใช้เครื่องเซิร์ฟเวอร์ที่มีขนาดใหญ่ขึ้นและหน่วยความจำเยอะกว่านี้ ส่วนโปรแกรมก็สามารถทำการปรับปรุงให้สามารถทำการรัน

โปรแกรมได้เร็วกว่านี้ หรือมีรูปภาพเคลื่อนไหวมากกว่านี้ โดยถ้าหากจะทำให้เร็วขึ้นก็ใช้วิธีจำกัดจำนวนผู้เล่นที่เข้ามาใช้ Server

2. คาราโอเกะ ประสบปัญหาเมื่อส่งข้อมูลข้ามเครือข่าย ลูกบอลจะทำการเคลื่อนที่ได้ช้าลงและไม่ค่อยจะสอดคล้องกับเนื้อร้อง และนอกจากนี้ยังไม่มีการนำภาพเคลื่อนไหวหรือวีดีโอมาใช้ประกอบคาราโอเกะ ดังนั้นถ้าหากจะพิจารณาพัฒนาต่อก็ให้ปรับปรุงให้ลูกบอลวิ่งได้ลงจังหวะกว่านี้ หรือมีวีดีโอประกอบคาราโอเกะ

ภาคผนวก ก

Java Development Kit

1. Java Development Kit

JDK (Java Develpment Kit) คือ เครื่องมือช่วยในการพัฒนา Java ประกอบไปด้วย โปรแกรมที่สำคัญ คือ javac.exe คือ คอมพายเลอร์ ใช้สำหรับแปล Source Code(.java) เป็น Byte Code (.class) java.exe คือ อินเตอร์เพรทเตอร์ ใช้สำหรับปฏิบัติงาน Java Application โดยไม่ต้องปฏิบัติงานผ่าน browser หรือ viewer , appletviewer.exe ใช้สำหรับทดสอบ Java Applet เพื่อดูผลลัพธ์ (view) โดยต้องปฏิบัติงานผ่านไฟล์ .html (HTML : Hyper Text Markup Language) และภายใน JDK ยังประกอบไปด้วย Hotjava browser , Debugger , Library , Document , และ Demo เป็นต้น

2. การติดตั้ง JDK บน Windows 95/NT

ขั้นตอนการติดตั้ง (Installation) โดย JDK ต้องการเนื้อที่บน Hard Disk ประมาณ 6 Mbytes การติดตั้งขั้นแรกให้ปฏิบัติงานหรือรัน (run) ไฟล์ JDK-1_0_1-win32-x86.exe ซึ่งถูกบีบอัด (Compress) หรือลดขนาดไฟล์ไว้ และจะติดตั้งตัวเองโดยอัตโนมัติ โดยสร้าง Subdirectory ต่างๆและขยายไฟล์ต่างๆ ออกเก็บไว้ในแต่ละ Subdirectory ขอแนะนำให้ติดตั้งที่ root เช่น c:\ เมื่อทำการติดตั้งเสร็จเรียบร้อยให้เซ็ตหรือตั้ง PATH ในไฟล์ Autoexec.bat ให้มี c:\java\bin; รวมอยู่ด้วยในกรณีที่ติดตั้งที่ Drive C ก็เป็นอันเสร็จเรียบร้อย

3. การติดตั้ง JDK บน UNIX

JDK จะอยู่ในรูปแบบไฟล์บีบอัดแบบ .tar.Z ซึ่งต้องการเนื้อที่บน Hard disk ประมาณ 9 Mbytes การติดตั้งต้องทำการการขยายไฟล์ออกสองครั้งดังนี้

% uncompress JDK-1_0_1-solaris-sparc.tar.Z

% tar xvf JDK-1_0_1-solaris2-sparc.tar

สมมติว่าเราได้ติดตั้ง JDK อยู่ที่ /usr/local ให้เซ็ตหรือตั้ง PATH ในไฟล์ .profile และ .cshrc ในกรณีที่ใช้ shell csh, tcsh:

% set path=($PATH /usr/local/java/bin)

เท่านี้ JDK ก็พร้อมที่จะถูกใช้งานแล้ว

4. ทดสอบ JDK จาก Demo

เมื่อติดตั้ง JDK เรียบร้อยก็ถึงขั้นตอนทดสอบว่า JDK ที่ติดตั้งนั้นใช้งานได้หรือไม่ โดยจะทดลองกับโปรแกรม Demo ที่ JDK ให้มาด้วย โดยของยกตัวอย่างในรูปแบบของ Windows 95/NT ส่วนบน UNIX นั้นก็ใช้คำสั่งต่างๆเหมือนกันทุกประการ จะแตกต่างกันก็เพียงบน Windows 95/NT จะแปลและปฏิบัติงานบน MS-DOS Prompt ส่วน UNIX นั้นจะทำบน Terminal Windows และสัญลักษณ์แสดงการแบ่ง Subdirectory บน Windows 95/NT ใช้ "\" (Backslash) ส่วน UNIX ใช้ "/" (Slash)

ขั้นตอนแรกให้เปิดวินโดว (Window) MS-DOS Prompt แล้วใช้คำสั่งนี้

```
cd \java\demo\TicTacToe
appletviewer example1.html
```

หลังจากนั้นก็จะมีวินโดวอีกวินโดวหนึ่งปรากฏขึ้น ซึ่งก็คือผลการปฏิบัติงานของโปรแกรม TicTacToe นั่นเอง และสามารถเล่นเกมส์ (Games) TicTacToe นี้สู้กับคอมพิวเตอร์ได้โดยการใช้เมาส์ (Mouse) กดหรือคลิ้ก (Click) ไปที่ช่องต่างๆ ภายในตาราง TicTacToe

กิตติกรรมประกาศ

โครงงานชิ้นนี้สามารถสำเร็จลุล่วงไปได้ด้วยดีนั้นเพราะได้รับความช่วยเหลือ และการสนับสนุนจากบุคคลหลายๆท่านด้วยกัน

ขอขอบพระคุณ บิดา-มารดา ที่ได้อบรมสั่งสอน และให้ความสนับสนุนในทุกๆเรื่อง

ขอขอบพระคุณ ครูบาอาจารย์ทุกท่าน ที่ได้อบรมสั่งสอนทั้งด้านวิชาการและจริยธรรม

ขอขอบพระคุณ ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา ที่ได้ให้คำชี้แนะ คำแนะนำต่างๆ

ขอขอบคุณ Bill Basham และ Juno R Suk ที่ให้ความรู้เรื่อง JAVA

ขอขอบคุณ พี่ๆที่ Nectec ที่ให้ความรู้เกี่ยวกับ JAVA

ขอขอบคุณ พี่ต่าย (อ้วน) , พี่ก้อง , พี่ต่าย (เล็ก) , พี่เอก ที่คอยช่วยเหลือในด้านต่างๆ

ขอขอบคุณ ตี๋ (วินัย) , เต้อ (พิเชียร) และเพื่อนๆทุกคน ที่ให้ความช่วยเหลือ ให้กำลัง

ขอขอบคุณ ป้าร้านข้าวแกง ที่ทำอาหารอร่อยๆ ให้ทานเมื่อหิวยามดึก

บรรณานุกรม

[1] Michael Morrison : "teach yourself Internet Game Programming with Java in 21 days",Sams Publisher 1996

[2] Paul J.Perry เรียบเรียงโดย ณัฐวัฒน์ ลิ้มล้อมวงศ์ "สร้างสรรเว็บเพจสวยๆด้วยภาษาจาวา"

[3] Internet-Intranet ปี 1 และ ปี 2 ฉบับที่ 1 - 10

[4] Steve Simkin , Neil Barlett , Alex Leslie " Java Programming Explorer"