เกมปัญญาประดิษฐ์ผ่านเครือข่าย

Game AI on Internet

นาย เฉลิมรัฐ อุดมผล
นาย ณัฏฐ์ อมรธนานุบาล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

.b.....
.i.....

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระเจ้าเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เกมปัญญาประดิษฐ์ผ่านเครือข่าย

Game AI on Internet

ผู้จัดทำ

1. นายเฉลิมรัฐ อุดมผล รหัสประจำตัว 41014086
2. นายณัฏฐ์ อมรธนานุบาล รหัสประจำตัว 41014127

อาจารย์ที่ปรึกษา

(อ. เกียรติกูล เจียรนัยธนะกิจ)

เกมปัญญาประดิษฐ์ผ่านเครือข่าย

นาย เฉลิมรัฐ อุดมผล 41014086
นาย ณัฐ อมรธนานุบาล 41014127
อ. เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

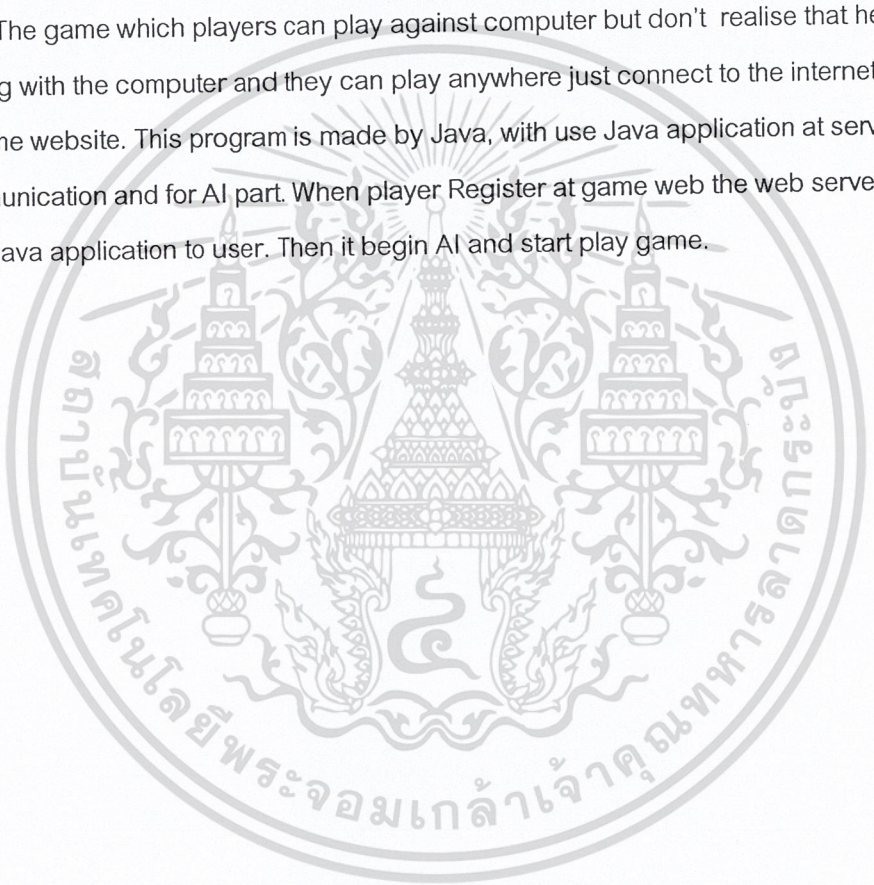
จุดประสงค์ของการ จัดทำเกมปัญญาประดิษฐ์ผ่านเครือข่าย คือ ต้องการทำเกมที่สามารถคิดและเล่นได้เอง โดยที่ผู้เล่นไม่รู้สึกว่าตนกำลังเล่นอยู่กับคอมพิวเตอร์และเป็นเกมที่สามารถเล่นได้หลายคน โดยที่ผู้เล่นแต่ละไม่จำเป็นต้องมีโปรแกรมเพียงแต่ เชื่อมต่อกับเครือข่ายอินเตอร์เน็ตและเข้าเวบไซต์ของเกมก็พอ โปรแกรมถูกสร้างด้วยภาษาจาวาโดยจะมีจาวาเอปพลิเคชั่นทำงานที่เครื่องเซิร์ฟเวอร์ สำหรับการติดต่อกับผู้ใช้และประมวลผลในส่วนของ AI โดยเมื่อมีผู้เล่นเข้ามาก็จะทำการสร้างซ็อกเก็ตสำหรับติดต่อ และประเมินแผนการเล่นในส่วนของ AI และทำการเล่นเกมร่วมกับผู้เล่นคนอื่น

Game AI on Network

Chalermrat	Udompol	41014086
Nut	Amontananuban	41014080
Kietkul	Gairanaitanakij	Advisor

ABSTRACT

Main purpose of Game AI on Network is to make game that can play and think by itself. The game which players can play against computer but don't realise that he is playing with the computer and they can play anywhere just connect to the internet and go to game website. This program is made by Java, with use Java application at server for communication and for AI part. When player Register at game web the web server will send java application to user. Then it begin AI and start play game.



กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงก็คือ อาจารย์ เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ให้ความเอาใจใส่ แนะนำและช่วยเหลือเสมอมา ซึ่งต้องขอบคุณเป็นอย่างสูง

และต้องขอกราบขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้พวกข้าพเจ้ามีวันนี้ ก็คือ บิดามารดา ที่ช่วยเลี้ยงดูเรามาและยังสนับสนุนทุนทรัพย์ด้วยดีเสมอมา

อีกทั้งเพื่อนๆ ทุกคนที่คอยช่วยเหลือ และเปิดเพลงให้ฟังตลอดช่วงทำงาน

ขอบคุณป้าหน่อยที่ทำก๋วยเตี๋ยวที่เผ็ดแต่อร่อยให้เป็นพลังงานกับเราเสมอมา

นาย เฉลิมรัฐ อุดมผล

นาย ณัฐ อมรธนานุบาล

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญรูป	VI
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของโครงงาน	2
1.3 ขอบเขตของโครงงาน	2
1.4 กติกาเกมจับหมู	3
1.5 ตัวอย่างการเล่นเกมจับหมูและอธิบายการเล่น	4
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง	7
2.1 โปรโตคอล TCP	7
2.2 หมายเลขพอร์ต	7
2.3 ภาษาจาวา	7
2.4 ซ็อกเก็ต (socket)	25
2.5 ไคลเอนต์/เซิร์ฟเวอร์(Client/Server)	26
2.6 คลาส URL	32
2.7 คลาส InetAddress	32
2.8 คลาส Socket	32
2.9 คลาส ServerSocket	33
2.10 ดาตาแกรม (Datagram)	34
2.11 Minnimax Game	34
2.12 Alpha Beta Plunning	35
บทที่ 3 การสร้างและการออกแบบ	36
3.1 เกมออนไลน์	36
3.2 สิ่งที่ควรพิจารณาเมื่อเขียนเกมด้วยจาวา	36
3.3 กราฟฟิกและแอนิเมชัน	36
3.4 การป้อนอินพุต	36
3.5 เสียง	37

3.6 ระบบเครือข่าย 37
3.7 การออกแบบเกม 38
3.8 แนวความคิดพื้นฐาน 38
3.9 แนวการดำเนินเกม 38
3.10 โหมดการเล่นเกม 39
3.11 แนวความคิดพื้นฐานสำหรับการเล่นเกมแบบหลายคน 39
3.12 ปัญหาที่อาจเกิดขึ้นในเกมแบบเครือข่ายและวิธีการแก้ปัญหา 40
3.13 การออกแบบส่วนปัญญาประดิษฐ์ 43
3.14 โครงสร้างการทำงานของโปรแกรม 47
บทที่ 4 4.1 ปัญหาที่พบและแนวทางแก้ไข 53
4.2 แนวทางในการพัฒนาเพิ่มเติม 54
4.3 สรุปผลที่ได้จากการทำโครงงาน 55
4.4 ประโยชน์ที่ได้รับจากการทำโครงงาน 55
บรรณานุกรม 56

สารบัญรูปภาพ

หน้าที่

รูปที่ 1.1 ตัวอย่างเกม 4
รูปที่ 1.2 การเริ่มเล่นเกม 4
รูปที่ 1.3 กรณีที่ไม่มีไพ่ที่สามารถลงได้(1) 5
รูปที่ 1.4 กรณีที่ไม่มีไพ่ที่สามารถลงได้(2) 6
รูปที่ 1.5 การคิดแต้มที่ได้ 6
รูปที่ 2.1 โครงสร้างการทำงานของคลาสแอพแพล็ต 11
รูปที่ 2.2 แสดงโครงสร้างแบบรากต้นไม้ของระบบเชื่อมต่อ 16
รูปที่ 2.3 แผนภาพแสดงประเภทของคอนเทนเนอร์ 16
รูปที่ 2.4 ตารางแสดงคุณสมบัติคลาสคอนเทนเนอร์ทั้ง 4 ประการ 17
รูปที่ 2.5 แสดงถึงการทำงานของเธรดแบบมีโปรเซสเซอร์1 ตัว และแบบที่มีโปรเซสเซอร์หลายตัว 22
รูปที่ 2.6 แสดงโครงสร้างระดับล่างของ Transport stack 30
รูปที่ 2.7 แสดงการใช้ โพรโตคอล ทีซีพี/ไอพี (TCP/IP) ในระบบ ไคลเอนต์/เซิร์ฟเวอร์ 30
รูปที่ 2.8 แสดงถึงการทำงานของ ServerSocket 33
รูปที่ 2.9 ตัวอย่าง Mininmax Game 35
รูปที่ 2.10 การทำ Plunning 35
รูปที่ 3.1 แผนภาพการเข้าระบบของผู้เล่น 40
รูปที่ 3.2 แผนภาพการออกจากระบบ 41
รูปที่ 3.3 แผนภาพการส่งข้อความ 42
รูปที่ 3.4 แผนภาพการเล่นเกม 42
รูปที่ 3.5 แผนภาพแสดงการทำงานส่วนปัญญาประดิษฐ์ 43
รูปที่ 3.6 ตัวอย่าง Tree 44
รูปที่ 3.7 การ search แบบ Minimax 45
รูปที่ 3.8 การ Search แบบ Minimax เมื่อทำสมบูรณ์แล้ว 46
รูปที่ 3.9 ตัวอย่างการทำ Alpha Beta Plunning 46
รูปที่ 3.10 การทำงานของการ generate point 48

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

1.1.1 สิ่งจูงใจ

จากการพัฒนาไปอย่างรวดเร็วของเทคโนโลยีทางด้านคอมพิวเตอร์และอินเทอร์เนต ทำให้การติดต่อสื่อสารกันข้ามซีกโลก หรือข้ามทวีปนั้นเป็นได้โดยง่าย จากประสิทธิภาพของอินเทอร์เน็ตสามารถสื่อสารได้เหมือนอยู่ในที่เดียวกัน ไม่ว่าจะเป็นรูปแบบข้อความ ภาพ และเสียง ด้วยความสามารถนี้ทำให้เกิดเกมบนเครือข่ายอินเทอร์เนตขึ้นมา ซึ่งสามารถเล่นได้เหมือนผู้เล่นแต่ล่ะคนมานั่งอยู่ ณ.ตำแหน่งเดียวกัน ทำให้เกิดการพบปะผู้คนกันขึ้นมา

1.1.2 ปัญหา

ในปัจจุบันเกมบนเครือข่ายนั้นจะกำหนดว่าแต่ล่ะเกมนั้นต้องใช้ตัวผู้เล่นกี่คน ถ้าผู้เล่นไม่ครบจะไม่สามารถ เล่นต่อไปได้ต่อไปได้ หรือบางกรณีผู้เล่นต้องการฝึกฝนกับผู้เล่นที่มีความชำนาญ จึงมีการใช้ปัญญาประดิษฐ์มาช่วยในการเล่นกับผู้คน ในปัจจุบันปัญญาประดิษฐ์ ในเกมเครือข่ายนั้นมีน้อยหรือถ้ามีก็ขาดประสิทธิภาพ เช่น ใช้เวลานานหรือความสามารถต่ำ เล่นไปแล้วเกิดการล็อกเป็นต้น

1.1.3 แนวทางแก้ไข

จากปัญหาที่ขาดผู้เล่น หรือขาดผู้เล่นที่มีความชำนาญเพื่อฝึกฝน ทำให้ทางผู้จัดทำ จะสร้างปัญญาประดิษฐ์ มาใช้แทนตัวผู้เล่น แต่ทว่าตัวปัญญาประดิษฐ์นั้นต้องมีความสามารถ และประสิทธิภาพสูงคือ ข้อแรกสามารถเล่นเกมกับผู้เล่นโดยถูกกติกา และเก็บคะแนนเฉลี่ยได้เกินกว่าหรือเท่ากับค่าเฉลี่ยการเล่นเกมเท่าไปของผู้เล่น ข้อสอง ระยะเวลาในการคิด และประมวลผลนั้นต้องไม่นานเกินไปจนผู้ใช้รับไม่ได้ และต้องไม่เกิดการล็อกเกิดขึ้น คือสามารถเล่นได้จนจบไม่ว่าจะเกิดกรณีใดๆ (ยกเว้น เกิดการผิดพลาดทางฮาร์ดแวร์หรือ การสื่อสาร) ปัญญาประดิษฐ์นี้จะทำงานได้โดยไม่ต้องมีผู้ดูแลระบบมาคอยควบคุมดูแล

1.2 วัตถุประสงค์ของโครงงาน

ระบบเกมออนไลน์ที่มีปัญญาประดิษฐ์ (Game AI on Internet) สร้างขึ้นมาเป็นเว็บไซต์ที่ให้บริการห้องเกมและเน้นที่มีผู้เล่นปัญญาประดิษฐ์ โดยจะต้องสร้างเกมพื้นฐานเข้ามารองรับซึ่งเป็นเกมที่ง่ายต่อความเข้าใจ และมีกฎที่ง่าย แต่สามารถเล่นได้หลากหลายรูปแบบ ซึ่งสรุปได้เป็นหัวข้อได้ดังนี้

- เพื่อศึกษาการเขียนโปรแกรมบนเครือข่าย ด้วยภาษาจาวา
- เพื่อศึกษาการสร้างและการตอบสนองเหตุการณ์ในการทำเกมบนเครือข่าย
- เพื่อศึกษาการออกแบบปัญญาประดิษฐ์ตามที่เรียนมา
- เพื่อนำโครงงานนี้ไปใช้ได้ในระบบจริง

1.3 ขอบเขตของโครงงาน

ระบบเกมบนเครือข่ายนี้นี้สามารถแบ่งได้เป็นสามส่วนประกอบใหญ่ คือ

1.3.1 ส่วนของผู้ใช้งานทั่วไป

- ใช้จาวาแอพเพล็ต ในการเขียนและสร้าง พร้อมทั้งวาง และตรวจสอบกฎการเล่น
- เก็บข้อมูลของผู้ใช้ที่เข้ามาในระบบ และให้ผู้ใช้สามารถเห็นผู้ใช้คนอื่นๆในระบบ พร้อมทั้งส่งข้อความให้กับผู้ใช้ทั้งแบบประกาศและแบบตัวต่อตัว
- ใช้ GUI ในการติดต่อกับผู้ใช้ทั้งหมด ทำให้ผู้ใช้ไม่ต้องมีความรู้ในการใช้คำสั่งต่างๆในโปรแกรม
- ส่งและรับแพคเกทข้อความให้แก่เซิฟเวอร์ได้โดยไม่มีความผิดพลาด และแสดงผลออกมาได้ถูกต้องโดยไม่ใช่เวลามากเกินไป

1.3.2 ส่วนของเซิร์ฟเวอร์

- ส่งและรับแพ็กเกจข้อความให้แก่ผู้ใช้ได้โดยไม่มีความผิดพลาด และประมวลผลออกมาได้ถูกต้องโดยไม่ใช่เวลามากเกินไป
- เก็บและรักษาข้อมูลในระบบได้อย่างถูกต้อง และไม่มีการรั่วไหลออกไปข้างนอกระบบ
- ใช้ภาษาจาวาในการสร้างแต่ล่ะเทร็ดและตัวโปรแกรมให้ทำงานได้โดยไม่มีข้อบกพร่อง

1.3.3 ส่วนของปัญญาประดิษฐ์

- ศึกษาและสร้างกฎในการเดิน 6 ตาแรกเพื่อไม่ให้การสร้างโหนดสำหรับทรี มีมากเกินไป
- ศึกษาและใช้การค้นหาการเดินใน 6ตาหลังด้วยทรี
- ศึกษาและสร้างกฎในการตัดโหนดที่ไม่ต้องการทิ้ง
- ลดเวลาการหาเส้นทางการเดินให้มากที่สุด

1.4 กติกาเกมจับหมู

1. เล่นครั้งละ 4 คน โดยแต่ละคนจะได้รับไพ่ 13 ใบในตอนแรกโดยเมื่อเริ่มเกมผู้เลนทุกคนต้องมีไพ่หน้าหัวใจอย่างน้อย 1 ใบ
2. การลงไพ่ครั้งแรกผู้เล่นที่มีไพ่ 2 ดอกจิกต้องเป็นคนเริ่มลงก่อนโดยสามารถเริ่มเล่นไพ่ใดก็ได้
3. การลงไพ่จะลงครั้งละ 1 ใบ โดยวนเรียงลำดับไปในทิศทางเดียวกันตลอด
4. การลงไพ่ถ้าผู้เล่นนั้นยังมีไพ่หน้านั้นอยู่ก็ต้องลงไพ่หน้านั้นตาม ในกรณีที่ไพ่ในหน้านั้นหมดแล้วก็สามารถลงหน้าใดก็ได้ ผู้ที่ลงไพ่ที่ใหญ่สุดที่เป็นหน้าเดียวกันกับไพ่ใบแรกที่ลงคนนั้นก็จะได้เป็นผู้เริ่มเล่นในเทิร์นถัดไป
5. การคิดคะแนนจะมีแต้ม คือ

   ไพ่หน้าหัวใจจะมีแต้มติดลบ คือ A = -50, K = -40, Q = -30, J = -20, 10 = -10 , 9 = -9, 8 = -8 , 7 = -7, 6 = -6, 5 = -5, 4 = -4, 3 = -3 และ 2 = -2

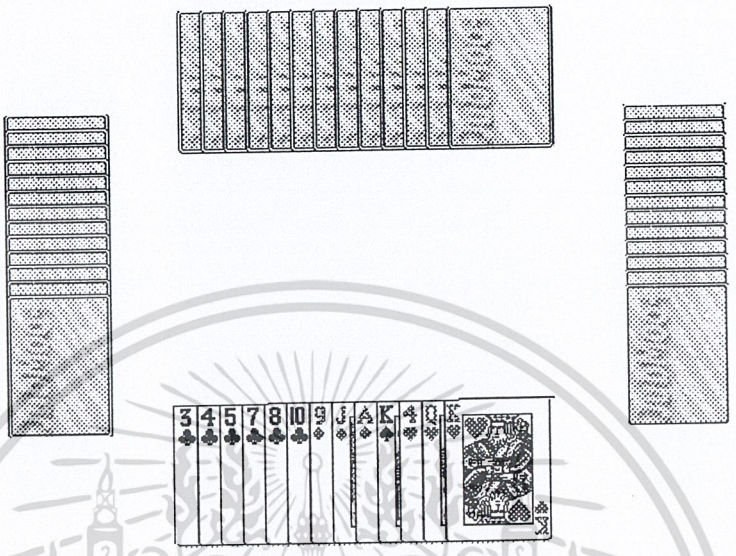
   ไพ่ J ข้าวหลามตัดจะมีค่า +100 แต้ม

   ไพ่ Q โพดำจะมีค่า -100 แต้ม

   ไพ่ 10 ดอกจิกจะทำให้แต้มที่มีอยู่เพิ่มขึ้น 2 เท่า แต่ถ้าไม่สามารถเก็บไพ่แต้มได้เลย แต่มี 10 ดอกจิก จะได้แต้ม +50
6. เมื่อเล่นจบเกมแล้วก็จะนำแต้มที่เก็บได้มาคำนวณผู้ที่ไม่สามารถเก็บไพ่แต้มใดๆได้เลยจะ ได้แต้มเป็น 0

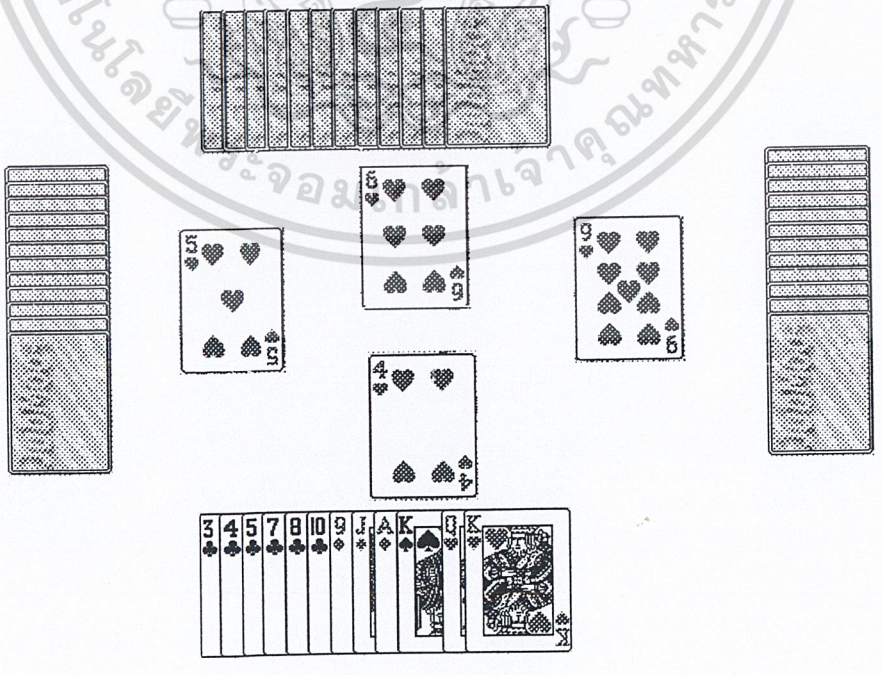
กรณีพิเศษ คือ เมื่อมีผู้ใดเก็บไพ่หน้าหัวใจได้คนเดียวครบทุกใบก็จะได้แต้มพิเศษเป็น +194 แต้ม

1.5 ตัวอย่างการเล่นเกมจับหมูและอธิบายการเล่น



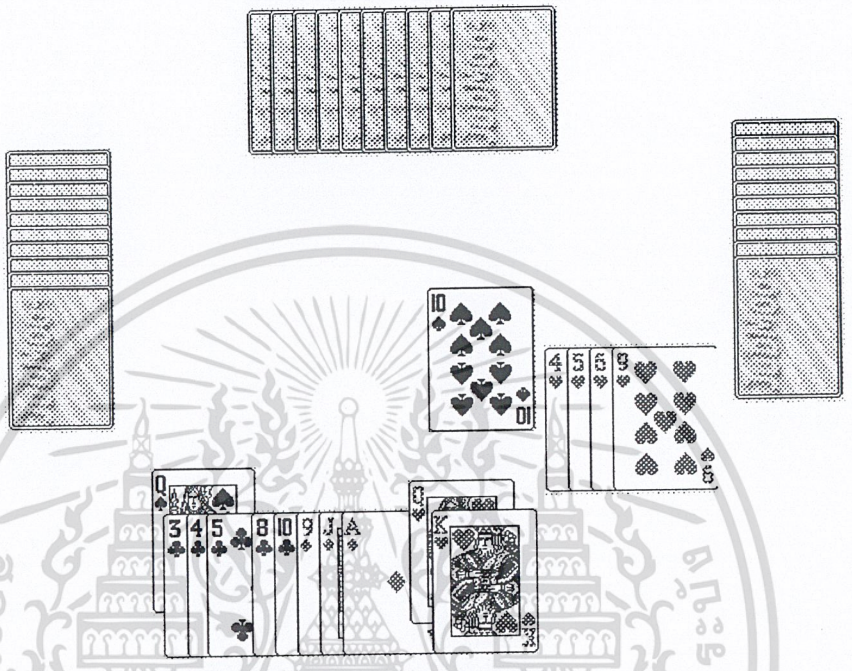
รูปที่ 1.1 ตัวอย่างเกม

เริ่มแรกทุกคนจะได้ไพ่คนละ 13 ใบ โดยตนเองจะสามารถเห็นได้เฉพาะไพ่ที่ตนได้รับเท่านั้น ผู้เล่นคนอื่นจะไม่อนุญาตให้ผู้อื่นดูไพ่ยกเว้นไพ่ที่ตนเองจะลง



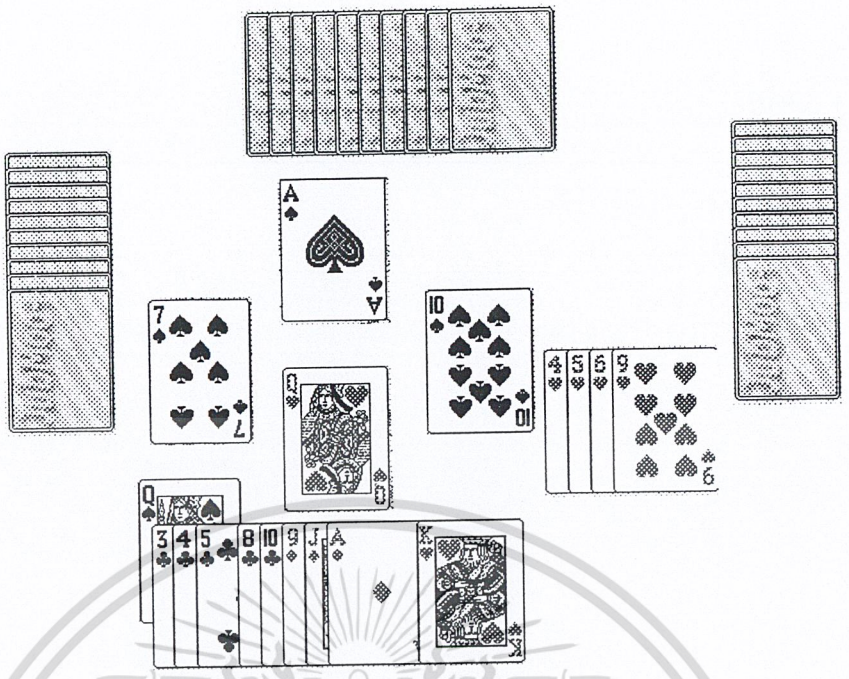
รูปที่ 1.2 การเริ่มเล่นเกม

จากภาพเราเป็นคนเริ่มเล่นเกมก่อน โดยเริ่มลงไพ่คือ 4 หัวใจทุกคนที่มีไพ่หน้าหัวใจก็ต้องลงหน้าหัวใจตาม โดยในตานี้ คนที่ลงไพ่ 9 หัวใจซึ่งสูงที่สุดในรอบนี้จึงได้ไพ่กองนี้ไป



รูปที่ 1.3 กรณีที่ไม่มีไพ่ที่สามารถลงได้(1)

หลังจากที่ได้ไพ่กองที่เป็นหัวใจไปแล้วผู้เล่นก็จะโชว์ไพ่ที่มีแต้มให้ผู้อื่นเห็น และเป็นคนเริ่มลงไพ่ต่อไป โดยลงไพ่ 10 โพธิ์ดำ แต่เราไม่มีไพ่โพธิ์ดำจึงสามารถลงไพ่ใบใดก็ได้ในมือ



รูปที่ 1.4 กรณีที่ไม่มีไพ่ที่สามารถลงได้(2)

เนื่องจากเราไม่มีหน้าโพธิ์แล้วและได้เลือกลง Q โพธิ์แดงแทน ผู้เล่นที่ลงไพ่ในหน้าเดียวกับไพ่เริ่มแรกและสูงที่สุดจะเป็นผู้ได้ไพ่กองนี้ไป (เริ่มเล่นด้วยหน้าโพธิ์ดำ ดังนั้นผู้ที่ลง A โพธิ์ดำจึงได้ไป

เมื่อเล่นจนจบเกมแล้วก็จะนำไพ่ที่มีแต้มในมือมาคิดคะแนน เช่น

รูปที่ 1.5 การคิดแต้มที่ได้

จะได้แต้ม คือ (-2) + (-3) + (-4) + (-5) + (-7) + (-8) + (-30) + (-40) + (-50) = -149 แต้ม

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 โปรโตคอล TCP

โปรโตคอลนี้อยู่ในระดับทรานส์สปอร์ตเลเยอร์เหมือนกับโปรโตคอล UDP แต่มีลักษณะที่ตรงข้ามกันคือ เป็นโปรโตคอลแบบคอนเนคชั่นออเรียนเต็ด โดยที่มีความน่าเชื่อถือในการรับส่งข้อมูลและลำดับของข้อมูลจะมีลำดับเหมือนกับต้นทางและเนื้อหาข้อมูลไม่ผิดพลาด จึงทำให้เกิดความสิ้นเปลืองในการเชื่อมต่อของการส่งข้อมูลมากกว่าโปรโตคอล UDP

2.2 หมายเลขพอร์ต

เนื่องจากในเวลาใดๆ สามารถมีโพรเซสของผู้ใช้สามารถใช้ UDP หรือ TCP ได้พร้อมกัน ดังนั้นจะต้องมีวิธีแยกแยะว่าข้อมูลเป็นของผู้ใช้คนใด ซึ่งวิธีที่ TCP และ UDP ใช้คือ การใช้หมายเลขพอร์ต

เมื่อโพรเซสไคลเอ็นต์ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ ไคลเอ็นต์จะต้องเจาะจงที่เซิร์ฟเวอร์ที่ต้องการติดต่อแต่ละพังรู้แอดเดรสอินเทอร์เน็ต 32 บิตเพียงอย่างเดียวนั้นไม่พอ เพราะว่าสามารถติดต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโปรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้ง TCP และ UDP ได้มีการกำหนดหมายเลขพอร์ตมาตรฐาน (well-known ports) ซึ่งไคลเอ็นต์เลิกใช้หมายเลขพอร์ตนี้แล้ว สามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอ็นต์ที่อื่นได้ โปรเซสที่ได้รับมอบหมายเลขพอร์ตชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าไหร่ แต่เป็นหน้าที่ของอีกโปรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ใน TCP และ UDP นั้นหมายเลขพอร์ตตั้งแต่ 1-1023 เป็นพอร์ตที่สงวนไว้สำหรับหมายเลขพอร์ตมาตรฐาน

2. 3 ภาษาจาวา

ทุกวันนี้อินเทอร์เน็ตกำลังเติบโตเป็นอย่างมากซึ่งนำไปใช้งานในด้านต่างๆ ทั้งธุรกิจการค้า การเรียนการสอน การติดต่อสื่อสาร และอื่นๆ อีกมาก ซึ่งในการพัฒนาแอพพลิเคชันบนอินเตอร์เนตนั้นมีโปรแกรมต่างๆ มากมายที่สนับสนุน แต่โปรแกรมที่มีความสามารถมากที่สุดตัวหนึ่งบนอินเทอร์เน็ตคือ โปรแกรมที่สร้างขึ้นมาจากภาษาจาวา ซึ่งถูกพัฒนาโดยบริษัทซันไมโครซิสเต็มซึ่งจุดประสงค์เริ่มแรกคือพัฒนาขึ้นมาเพื่อทำการควบคุมอุปกรณ์อิเลคทรอนิกส์ได้โดยไม่ขึ้นอยู่กับแพลตฟอร์ม (platform) ที่ใช้คือ การพัฒนาซอฟต์แวร์ให้สามารถทำงานได้โดยไม่ต้องสนใจว่า

ฮาร์ดแวร์ที่ใช้มีลักษณะอย่างไร นอกจากนี้จาวายังสนับสนุนความสามารถในการเชื่อมต่อเครือข่าย การติดต่อสื่อสารผ่านทางระบบเครือข่ายคอมพิวเตอร์ด้วย

2.3.1 คุณสมบัติทั่วไปของภาษาจาวามีดังนี้

1. เขียนง่าย เนื่องจากจาวามีลักษณะเป็นภาษาคอมพิวเตอร์ระดับสูงแต่ได้ตัดบางคุณลักษณะที่ไม่จำเป็นในภาษาระดับสูงออกไป เช่น ไม่สนับสนุนพอยเตอร์ (point math), การโหลดเกินของตัวดำเนินการ (operator overloading), การสืบทอดจากคลาสหลายคลาส (multiple inheritance)
2. มีชนิดเป็นสแตติก คือ ออบเจ็กต์ทั้งหมดที่ถูกใช้ในโปรแกรมตั้งถูกกำหนดก่อนที่มันจะถูกใช้งานทำให้ตัวคอมไพเลอร์ของจาวาสามารถรายงานชนิดที่ขัดแย้งกันได้
3. เมื่อคอมไพล์โปรแกรมแล้วผลลัพธ์จะเป็นไฟล์ข้อมูลแบบไบต์โค้ด (ซึ่งเหมือนกับ machine code) ซึ่งมันสามารถถูกเอ็กซิคิวต์ภายใต้ระบบปฏิบัติการด้วย Java Interpreter ซึ่งตัว Interpreter จะอ่านไฟล์ที่เป็นไบต์โค้ด แล้วจะแปลงคำสั่งที่เป็นไบต์โค้ดไปเป็นภาษาเครื่อง (machine-language Commands) ซึ่งจะถูกเอ็กซิคิวต์ได้โดยตรง โดยเครื่องที่กำลังรันโปรแกรมจาวาอยู่ เพราะฉะนั้นเราจะกล่าวได้ว่าจาวาเป็นทั้งตัวคอมไพล์และ อินเทอร์พรีต
4. สนับสนุนการทำงานได้หลายๆ งานในเวลาเดียวกัน
5. โปรแกรมจาวาจะจัดการกับขยะที่เกิดจากโปรแกรมด้วยตัวมันเอง ซึ่งหมายความว่าโปรแกรมจะไม่ต้องการที่จะลบออบเจ็กต์ซึ่งอยู่ในหน่วยความจำ
6. ความมั่นคง (Robust) เพราะว่า Java Interpreter จะตรวจสอบระบบทั้งหมดที่จะถูกเข้าถึงโดยโปรแกรม ดังนั้นจึงทำให้มั่นใจได้ว่าโปรแกรมจาวาจะไม่ทำให้ระบบพัง และถ้าเกิดความผิดพลาดขึ้นจะมี throw exception ออกมา ซึ่งตัว exception นี้จะถูกจับและจัดการโดยโปรแกรมเพื่อทำให้ไม่เกิดความเสี่ยงกับระบบ
7. ความปลอดภัย ระบบจาวาไม่ใช่แค่รับรองการทำงานในการเข้าถึงหน่วยความจำเท่านั้น แต่ยังทำให้แน่ใจได้ว่า จะไม่มีไวรัสเกิดขึ้นในการรันโปรแกรม เพราะว่าพอยเตอร์ไม่ถูกใช้ในภาษาจาวา
8. มีความยืดหยุ่น โปรแกรมจาวาสนับสนุน native method (ซึ่งเป็นฟังก์ชันที่ถูกเขียนขึ้นโดยภาษาอื่นๆ เช่น C++) โดยจะทำให้โปรแกรมเมอร์นั้นเขียนฟังก์ชันซึ่งอาจจะถูกเอ็กซิคิวต์ได้เร็วกว่าฟังก์ชันเดียวกันในการเขียนภาษาจาวา โดย native method เชื่อมต่อแบบไดนามิก(dynamically link) กับโปรแกรมจาวา แต่เมื่อใดที่ภาษาจาวาแก้ไขในเรื่องความเร็วได้ก็ไม่มีความจำเป็นต้องใช้ native method อีกต่อไป
9. เข้าใจง่ายเนื่องจากเป็นภาษาชั้นสูงและ เขียนเหมือนภาษาอ่านทั่วไป

2.3.2 การทำงานของจาวา

เมื่อเราทำการสร้างไฟล์จาวาขึ้นมาได้แล้ว เราต้องทำให้ไฟล์นี้สามารถทำงานได้ โดยการนำไฟล์จาวาที่สร้างมานี้ไปทำการคอมไพล์ หลังจากคอมไพล์แล้วจะได้ไฟล์ที่เรียกว่า คลาสไฟล์ ซึ่งเป็นไฟล์ที่เก็บไบต์โค้ดไว้(ไบต์โค้ดจะมีลักษณะใกล้เคียงคำสั่งเครื่องมากที่สุด๗ ไบต์โค้ดที่ได้นี้จะถูกรันบน Java Virtual Machine(JVM) ซึ่งจำทำการตรวจสอบไบต์โค้ดที่ได้ จากนั้นจะอ่านไบต์โค้ดและดำเนินการตามคำสั่ง

2.3.3 Java's Virtual Machine

ภาษา Java นำความคิดในการสร้างเครื่องจักรสมมติมาใช้ เพื่อให้มีคุณสมบัติ platform independent แต่แยกการแปลภาษาออกไปโดยใช้คอมไพเลอร์ แล้วนำโปรแกรมของเครื่องจักรสมมติที่ได้มาทำงานโดย interpreter วิธีการนี้จะทำงานได้เร็วขึ้นกว่าการใช้อินเทอร์พรีเตอร์ เพียงอย่างเดียว ไฟล์ของโปรแกรม source code ภาษา Java จะต้องมีสกุลเป็น .java คอมไพล์เลอร์ของภาษา Java จะทำหน้าที่คอมไพล์โปรแกรมภาษา Java ไปเป็นโปรแกรมที่ประกอบด้วยคำสั่งของ Java Virtual Machine(JVM) และจะถูกเก็บในไฟล์ที่มีสกุลเป็น .class เรียกว่าไฟล์ Java class

ปัจจุบันบริษัท Javasoft ซึ่งเป็นส่วนหนึ่งของบริษัท Sun Microsoft ได้กำหนดมาตรฐานของภาษา Java และเป็นกำหนดชุดคำสั่งของ JVM ที่เป็นฮาร์ดแวร์ยังอยู่ในระหว่างการพัฒนา ดังนั้น JVM เกือบทั้งหมดที่ใช้กันอยู่ในปัจจุบันจึงเป็นโปรแกรมที่จะจำลองการทำงานของ JVM บนเครื่องคอมพิวเตอร์ทั่วไป โดยทั่วไป virtual processor ของ JVM ที่จำลองขึ้นบนคอมพิวเตอร์เครื่องหนึ่ง จะแปลคำสั่งของ JVM เป็นคำสั่งของหน่วยประมวลผลในคอมพิวเตอร์เครื่องนั้นทำงานคำสั่งนั้นคำสั่ง (opcode) ของ JVM มีขนาด 1 ไบต์ ทุกคำสั่ง (บางครั้งเราเรียกโปรแกรม Java class ว่าโปรแกรม byte code) จำนวนคำสั่งของ JVM จึงมีได้สูงสุดเพียง 256 คำสั่ง เปรียบกับหน่วยประมวลผลทั่วไปแล้วคล้ายกับว่าคำสั่งของ JVM มีมากมายยิ่ง แต่จริงๆแล้วคำสั่งของ JVM แบ่งออกได้เป็นไม่กี่ประเภท โดยที่แต่ละประเภทนั้นจะทำหน้าที่คล้ายๆกัน เพียงแต่ทำกับ operands ต่างชนิดข้อมูลกัน

ชุดคำสั่งของ JVM ถูกออกแบบมาเพื่อสนับสนุนการทำงานของโปรแกรมเชิงวัตถุจึงมีคำสั่งเกี่ยวกับการสร้าง instance และการอ้างถึงสมาชิกใน instance ซึ่งไม่มีในหน่วยประมวลผลทั่วไป ภาษา Java เป็นภาษาที่เน้นความถูกต้องเกี่ยวกับชนิดข้อมูล จึงมีคำสั่งสำหรับคำนวณชนิดข้อมูล จึงมีคำสั่งสำหรับคำนวณชนิดข้อมูลพื้นฐานแต่ละชนิด บางคำสั่งของ JVM จะเหมือนกับคำสั่งที่มีในหน้วยประมวลผลทั่วไป แต่ก็มีหลายคำสั่งที่ไม่มีในหน่วยประมวลผลทั่วไป จึงต้องสร้างขึ้นเป็นโปรแกรมของคำสั่งนั้น

JVM ถูกออกแบบให้สามารถจำลองได้บนเครื่องทั่วไปไม่ว่าจะเป็นหน่วยประมวลผลของบริษัทใด แต่หน่วยประมวลผลต่างรุ่นต่างบริษัทมีจำนวนรีจิสเตอร์ไม่เท่ากัน บางรุ่นมีรีจิสเตอร์หน้าที่

พิเศษที่รุ่นอื่นไม่มี ผู้ออกแบบจึงตัดปัญหานี้โดยให้ JVM ไม่มีรีจีสเตอร์และทำการคำนวณทั้งหมดบนแสต็กชุดคำสั่งของ JVM จึงเป็น stacked operations หรือกล่าวได้ว่า JVM เป็น stack machine Virtual processor ใน JVM จะทำการ แมพปิ้งจากคำสั่ง byte code ในไฟล์ Java class ไปเป็นโปรแกรมของเนทีฟโค้ด ที่ทำหน้าที่ของคำสั่งนั้นแล้วส่งนั้นให้หน่วยประมวลผลทำงาน สังเกตว่าเนทีฟโค้ดนั้นอาจเป็น Application Program Interface(API) ของระบบปฏิบัติการที่ใช้หรืออาจจะเป็น standdard classes ที่สร้างขึ้นสำหรับหน่วยประมวลผลนั้น ทำให้ JVM หนึ่งอาจใช้งานได้ในระบบต่างกัน โดยเปลี่ยนแปลงแต่ standard class เท่านั้น

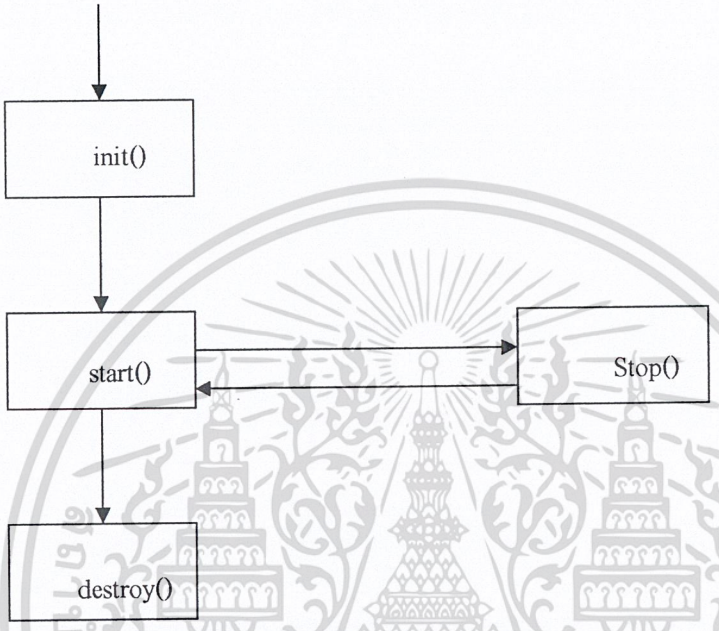
2.3.4 จาวาแอพเพล็ต

Java เป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่มีพื้นฐานมาจากภาษา C++ ซึ่งเป็นภาษาเขียนโปรแกรมเชิงวัตถุ หรือ Object-Oriented โดยออกแบบมาให้ไม่ขึ้นอยู่กับเครื่องหรือฮาร์ดแวร์ใดๆ คือสามารถทำงานได้บนเครื่องทุกรุ่นทำแบบโดยไม่จำกัด การเอ็กซีคิวต์โปรแกรมภาษา Java นี้ไม่ได้ใช้วิธีการคอมไพล์เป็นภาษาเครื่องของคอมพิวเตอร์แต่ละแบบ แต่ใช้สภาพแวดล้อม(Environment) ของ Java แทน โดยอ่านโปรแกรมต้นฉบับของ Java ขึ้นมา แล้วทำการแปลและเอ็กซิคิวต์ใน Java Virtual Machine(JVM) ทั้งนี้ในภาษา Java เองได้จัดเตรียม API ในแพลทฟอร์มต่างๆ ไว้ให้ใช้งาน ซึ่งประกอบด้วยฟังชันพื้นฐานต่างๆ ที่จำเป็นในการพัฒนาโปรแกรม เช่น การจัดวินโดว์ การจัดหน่วยความจำ และการรับข้อมูลจากผู้ใช้เป็นต้น

Java Applet เป็นรูปแบบหนึ่งของการใช้ภาษา Java ที่ไม่เอ็กซิคิวต์เป็นคำสั่งเดี่ยวๆ แต่จะถูกฝังและทำงานภายใต้โปรแกรมอื่น เช่น โปรแกรมบราวเซอร์ เป็นต้น โดยจะเก็บ Java Appet เป็นไฟล์ไว้ที่เซิร์ฟเวอร์ และให้ไคลเอ็นต์โหลดมาทำงานด้วยโปรโตคอล HTTP เหมือนว่าแอพแพล็ต เป็นส่วนหนึ่งของคำสั่งHTML ของเวบเพจนั้น(โดยมี Tag Applet เป็นตัวกำหนดขอบเขตของ Java Applet ใน HTML) Java Apllet นั้นจะไม่สามารถติดต่อกันกับระบบภายนอกโดยตรงได้เลย เช่น การติดต่อไปหาเวบเซิร์ฟเวอร์ แต่จะต้องติดต่อผ่านบราวเซอร์อีกทีหนึ่ง การทำงานของ Java Applet ในลักษณะนี้จะช่วยให้บราวเซอร์ สามารถทำงานแบบไดนามิค เช่น รับข้อมูลที่เป็นเสียง วีดีโอ หรือคาตาเบสต่างๆ รวมทั้งข้อมูลในประเภทอื่นๆได้

2.3.4.1 คลาสแอพเพล็ต

จาวาแอพเพล็ตต้องทำการรวมเอา java.applet.Applet โดยที่แอพเพล็ตจะจัดหาโครงสร้างการทำงานที่สำคัญให้กับเราในการรันบนบราวเซอร์ จาวาแอพพลิเคชันจะมีเมธอด main() โดยมีเมธอดในแอพเพล็ตจะขึ้นอยู่กับการทำงานของบราวเซอร์ ซึ่งมีโครงสร้างการทำงานดังรูป



รูปที่ 2.1 โครงสร้างการทำงานของคลาสแอพเพล็ต

2.3.4.1 การทำงานของเว็บบราวเซอร์ที่ควบคุมการทำงานของเมธอดต่างๆ ในแอพเพล็ต

บราวเซอร์จะเป็นตัวควบคุมการเรียกเมธอด init(), start(), stop(), และ destroy() โดยปกติแล้วเมธอดเหล่านี้จะไม่เกิดการทำงานใดๆ เว้นแต่ต้องการทำงานเฉพาะบางอย่าง ซึ่งทำได้โดยการเขียนโค้ดลงให้เหมาะสมกับเมธอดที่ต้องการทำงานด้วย

2.3.4.2 วงจรชีวิตของแอพเพล็ต (Applet Life Cycle)

วงจรชีวิตในที่นี้จะหมายถึง วงจรที่แอพเพล็ตต้องทำงานกับเมธอดต่างๆ ที่ตัวแอพเพล็ตอ้างอิงด้วย ซึ่งปกติจะมีการทำงานวนเวียนดังนี้

1. ตั้งค่าเริ่มต้นให้ตัวเอง (initialize)
2. เริ่มรันแอพเพล็ต (start)
3. หยุดรันแอพเพล็ต (stop)
4. ทำลายตัวเองเพื่อเตรียมจะหยุดการทำงานจนกว่าจะมีการเรียกใช้ใหม่ (destroy)

ซึ่งอธิบายได้ว่าแอพเพล็ตจะเริ่มต้นจากการตั้งค่าเริ่มต้นให้ตัวเอง แล้วจบลงด้วยการทำลายตัวเอง แต่ในขณะที่ทำงานบางครั้งเราทิ้งหน้าจอไว้ซักพัก แล้วกลับมาดูอีกครั้ง เราจะพบว่าแอพเพล็ตไม่ทำงานตัวเอง เพียงแต่จะหยุดการทำงานเท่านั้น และเมื่อเรากลับมาดูใหม่ ก็ไม่จำเป็นต้องทำการตั้งค่าเริ่มต้นใหม่อีก ซึ่งจะทำให้การทำงานเร็วขึ้น แต่อย่างไรก็ตาม ก็จะมีการใช้ทรัพยากรสิ้นเปลืองเช่นกัน

วงจรชีวิตที่กล่าวมาข้างต้นอาศัยเมธอดพื้นฐานดังนี้

1. init()

เมธอด init() จะถูกเรียกว่าเมื่อแอพเพล็ตเริ่มทำงานครั้งแรกเท่านั้นส่วนมากแล้วเราจะใช้เมธอดนี้ เพื่อจัดการเตรียมค่าองค์ประกอบเริ่มต้นต่างๆ ให้พร้อมต่อการใช้งาน เช่น การกำหนดค่าเริ่มต้นให้กับตัวแปร การโหลดไฟล์ภาพหรือไฟล์เสียง ซึ่งกิจกรรมเหล่านี้บางครั้งถ้าไปทำขณะที่กำลังทำงานอยู่อาจจะทำให้การทำงานช้าลง ดังนั้นกิจกรรมใดที่เราคิดว่าจะต้องทำเพียงครั้งเดียวหรือควรจัดการก่อนเข้าสู่การทำงานจริงก็ควรจะจัดการในเมธอดนี้ ซึ่งมีรูปแบบ

```
public void init()
```

2. start()

เมธอด start() จะถูกเรียกหลังจากเมธอด init() และจะทำงานใหม่ทุกครั้งเมื่อแอพเพล็ตถูกเรียกกลับทำให้ทำงาน เช่น เมื่อเราสั่งให้ Internet Explorer ย้อนกลับไปโหลดโฮมเพจก่อนหน้านี้ (Back) แล้วเราสั่งให้กลับมาที่หน้าเดิม (Forward) ที่มีแอพเพล็ตของเราทำงานอยู่ แอพเพล็ตจะได้รับเหตุการณ์ start อีกครั้งหนึ่งมีรูปแบบ

```
public void start()
```

3. stop()

เมธอด stop() จะทำงานตรงข้ามกับเมธอด start() กล่าวคือ จะถูกเรียกเมื่อออกจากโฮมเพจหน้าที่มีแอพเพล็ตนั้นๆทำงานอยู่เพื่อไปทำงานที่โฮมเพจหน้าอื่นๆและเมื่อเรากลับมาทำงานที่แอพเพล็ตหน้าเดิม เมธอด start() ก็จะถูกเรียกใช้งานอีกครั้งหนึ่ง และสุดท้ายเมธอด stop() ก็จะถูกเรียกใช้งานก่อนเมธอด destroy() เสมอซึ่งมารูปแบบ

```
public void stop()
```

4. destroy()

เมธอด destroy() จะถูกเรียกใช้งานหลังจากเมธอด ทุกครั้งเพื่อทำการสะสางทรัพยากรต่างๆ ที่จองไว้ใช้งานก่อนหน้านี้ เช่น ยกเลิกการติดต่อกับระบบเครือข่ายใดๆ ที่เรากำลังติดต่ออยู่ หากเรามีกิจกรรมใดๆ ที่จะต้องทำก่อนจบการทำงานของแอพเพล็ตก็ควรจะจัดการในเมธอดนี้ มีรูปแบบ

public void destroy()

เมธอดที่ใช้ในการวาดภาพและเมธอดที่ตอบสนองต่อเหตุการณ์

1 สำหรับเมธอดที่ใช้ในการแสดงผลหน้าจอประกอบด้วย

paint(Graphics g)

เมธอด paint(Graphics g) ใช้ในการวาดภาพภายในพื้นที่แสดงผลของแอพเพล็ต ซึ่งเมธอดนี้เป็นเมธอดเดียวกับที่กราฟฟิกในการวาดภาพหรือวาดตัวอักษรบนพื้นที่แสดงผลของแอพเพล็ต มีแบบ

public void paint(Graphics g)

update()

เมธอด update() ทำหน้าที่คล้ายเมธอด paint(Graphics g) คือใช้จัดการกับการแสดงผลในแอพเพล็ต แต่ใช้ในลักษณะปรับปรุงการแสดงผลของภาพ กรณีการทำ Double Buffered และขณะที่มีการวาดภาพใหม่เมื่อเรียกเมธอด update() จะไม่ทำการลบพื้นที่แสดงผลภาพใหม่ โดยเมื่อเมธอด update() ถูกเรียกใช้งาน ก็จะไปเรียกให้เมธอด paint(Graphics g) ให้ทำงานอีกครั้งต่อหนึ่ง ดังนั้น เมื่อเราต้องการที่จะปรับปรุงส่วนใดๆ บนพื้นที่แสดงผล เราก็จะมาจัดการกับเมธอดนี้ มีรูปแบบ

public void update()

สำหรับเมธอดที่ใช้ตอบสนองต่อเหตุการณ์จะประกอบไปด้วยหลักๆ คือ เมธอดที่ตอบสนองเหตุการณ์จากเมาส์และเมธอดที่ตอบสนองเหตุการณ์จากคีย์บอร์ด

2 สำหรับเมธอดที่ตอบสนองเหตุการณ์จากคีย์บอร์ด

1. mouseDnterrd(MouseEvent e)

เมธอดนี้จะทำงานเมื่อเราเลื่อนเมาส์จากภายนอกพื้นที่แสดงผลของแอพเพล็ตเข้าสู่พื้นที่การแสดงผลของแอพเพล็ต มีรูปแบบ

public void mouseDntered(MouseEvent e)

2. mouseExited(MouseEvent e)

เมธอดนี้จะทำงานตรงข้ามกับเมธอด mouseEntered(MouseEvent e) คือ จะทำงานเมื่อตำแหน่งของเมาส์หลุดออกนอกพื้นที่แสดงผลของแอพเพล็ต มีรูปแบบ

public void mouseExited(MouseEvent e)

3. mousePressed(MouseEvent e)

เมธอดนี้จะทำงานเมื่อมีการกดปุ่มบนเมาส์ภายในพื้นที่แสดงผลของแอพเพล็ต มีรูปแบบ

public void mousePressed(MouseEvent e)

4. mouseReleased(MouseEvent e)

เมธอดนี้จะทำงานตรงข้ามกับเมธอด mousePressed(MouseEvent e) กล่าวคือ จะทำงานเมื่อปุ่มของเมาส์ถูกปล่อยหลังจากการกด มีรูปแบบ

public void mouseReleased(MouseEvent e)

5. mouseMoved(MouseEvent e)

เมธอดนี้จะทำงานเมื่อมีการเลื่อนตำแหน่งของเมาส์ภายในพื้นที่แสดงผลของแอพเพล็ต โดยที่ไม่ได้มีการกดปุ่มเมาส์ มีรูปแบบ

public void mouseMoved(MouseEvent e)

6. mouseDragged(MouseEvent e)

เมธอดนี้ทำงานคล้ายกับเมธอด mouseMoved(MouseEvent e) แต่ต่างที่เมธอดนี้จะตอบสนองต่อเหตุการณ์ที่มีการกดปุ่มเมาส์ควบคู่กันไปด้วย มีรูปแบบ

public void mouseDragged(MouseEvent e)

7. mouseClicked(MouseEvent e)

เมธอดนี้จะทำการนับจำนวนการกดปุ่มเมาส์ภายในพื้นที่แสดงผลของแอพเพล็ต มีรูปแบบ

public void mouseClicked(MouseEvent e)

3 สำหรับเมธอดที่ตอบสนองเหตุการณ์จากคีย์บอร์ดประกอบด้วย

keyPressed(KeyEvent e)

เมธอดนี้จะทำงานเมื่อมีการกดปุ่มใดๆ บนคีย์บอร์ดในขณะที่แอพเพล็ตยังคงทำงานอยู่ มีรูปแบบ

public void keyPressed(KeyEvent e)

keyReleased(KeyEvent e)

เมธอดนี้จะทำงานตรงข้ามกับเมธอด กล่าวคือ จะทำงานเมื่อมีการปล่อยคีย์ ซึ่งเกิดหลังจากที่มีการกดคีย์ และอยู่ในระหว่างที่แอพเพล็ตยังทำงานอยู่ มีรูปแบบ

public void keyReleased(KeyEvent e)

keyTyped(KeyEvent e)

เมธอดนี้ทำงานคล้ายกับ 2 เมธอดข้างต้น กล่าวคือ เมธอดนี้จะทำงานเมื่อมีการกดปุ่มบนคีย์บอร์ดพร้อมทั้งต้องปล่อยปุ่มจากคีย์บอร์ดด้วย จึงจะเริ่มทำงาน มีรูปแบบ

public void keyTyped(KeyEvent e)

2.3.5 ระบบติดต่อผู้ใช้ (User Interface)

ระบบติดต่อผู้ใช้ คือ ส่วนประกอบหนึ่งของโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่เป็นสื่อกลางสำหรับโต้ตอบกับผู้ใช้ ในความหมายของการโต้ตอบก็คือ การรับคำสั่งและการแสดงผลลัพธ์ ระบบติดต่อผู้ใช้มีอยู่ด้วยกันหลายรูปแบบตั้งแต่อดีตจนถึงปัจจุบัน แต่สามารถแบ่งได้ใหญ่ๆ อยู่ 2 รูปแบบ คือ แบบรับคำสั่งทางคีย์บอร์ดแล้วแสดงผลเป็นตัวอักษร (Command-line interface) และอีกแบบคือแบบชี้แล้วกด (Point – and - click) โดยการใช้เมาส์สั่งงานร่วมกับการแสดงผลแบบกราฟฟิก (Graphic User Interface หรือ GUI) ซึ่งระดับติดต่อผู้ใช้แบบนี้เป็นที่นิยมกันในโปรแกรมสมัยใหม่ เพราะค่อนข้างเป็นมาตรฐาน ใช้งานได้ง่ายกว่าแบบแรก และไม่เสียเวลาในการเรียนรู้มากนัก

การทำงานในระดับล่างของระบบติดต่อผู้ใช้ระบบปฏิบัติการ (Operating System) จะส่งข้อมูลอินพุตจากคีย์บอร์ดและเมาส์ไปให้แก่โปรแกรม เพื่อประมวลผลและแสดงผลข้อมูลที่เอาท์พุทในรูปของจุดสีหรือบิตแมพ (Bitmap) ซึ่งประกอบกันขึ้นเป็นตัวอักษรหรือรูปภาพบนจอภาพในจาวา AWT ได้ถูกออกแบบให้โปรแกรมเมอร์หรือผู้พัฒนาโปรแกรม ไม่ต้องเป็นกังวลกับงานในระดับล่าง เช่น การจัดการกับอินเตอร์รับท์ของเมาส์หรือคีย์บอร์ด แม้กระทั่งรายละเอียดการเขียนจุดสี (pixel) ลงบนจอภาพ

ด้วยเหตุที่ว่า จาวาเป็นภาษาคอมพิวเตอร์ที่ไม่ขึ้นอยู่กับระบบใดๆ ดังนั้นการทำงานของ AWT ได้ถูกออกแบบให้มีเครื่องไม้เครื่องมือที่ใช้สำหรับติดต่อผู้ใช้ ซึ่งสามารถปฏิบัติงานกับระบบปฏิบัติการหลายๆ ระบบได้ โดยการใช้เครื่องมือดั้งเดิมของระบบปฏิบัติการนั้นๆ เพื่อรักษาหน้าตาของเครื่องมือ ซึ่งจะแตกต่างกันบ้างเมื่อทำงานอยู่บน OS ที่ต่างระบบกัน แต่ก็ไม่ถือเป็นอุปสรรคสำคัญในการพัฒนาโปรแกรม

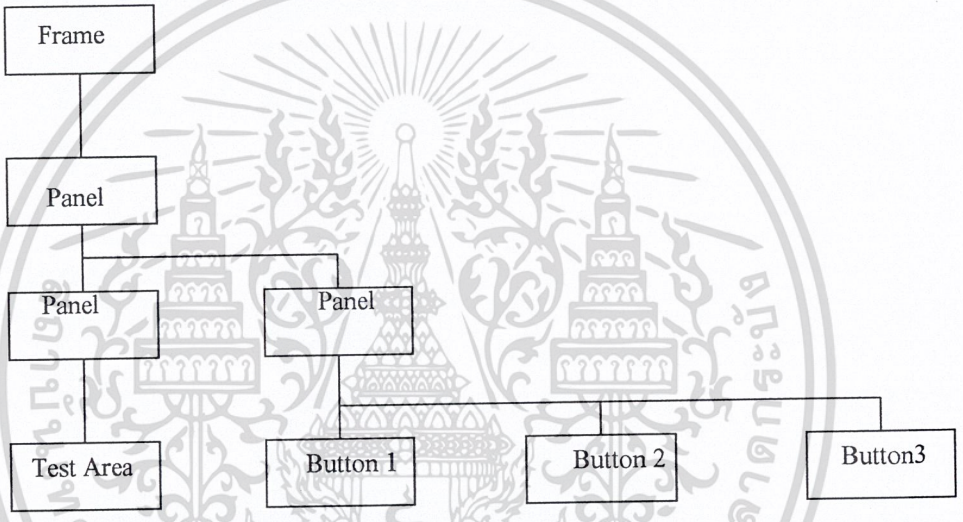
2.3.6 คอมโพเน็นต์และคอนเทนเนอร์

ระบบติดต่อผู้ใช้ถูกสร้างขึ้นจากระบบย่อยๆ ที่เรียกว่า คอมโพเน็นต์ (Component) ตัวอย่างของคอมโพเน็นต์ที่พบเห็นทั่วๆ ไป เช่น ปุ่มกด แถบเลื่อน ช่องเติมอักษร กล่องรายการ ฯลฯ กล่าวคือ คอมโพเน็นต์เป็นส่วนที่ทำให้ผู้ใช้สามารถทำการโต้ตอบกับโปรแกรม (สั่งงาน) และคือ ส่วนที่ทำให้ผู้ใช้ทราบสถานะการปฏิบัติงานของโปรแกรม (การแสดงผล) ภายใน AWT คอมโพเน็นต์ทุกคอมโพ

เน็นต์ก็คือ อินสแตนซ์ (instance) หรือตัวแทนตัวหนึ่งของคลาส Component หรือเป็นสับไทป์ (sub type) ของคลาส Component

คอมโพเน็นต์ใช้เครื่องมือที่ทำงานโดดเดี่ยวอย่างอิสระ แต่จะอยู่ภายใต้การควบคุมจากคอนเทนเนอร์ (Container) ซึ่งคือที่สำหรับบรรจุคอมโพเน็นต์หรือแม้กระทั่งบรรจุคอนเทนเนอร์ด้วยกันอันเนื่องมาจาก ตัวคอนเทนเนอร์เองบางครั้งก็ทำหน้าที่เป็นคอมโพเน็นต์เช่นกัน และคอนเทนเนอร์ทุกๆ คอนเทนเนอร์ก็เป็นอินสแตนซ์ของคลาส Container หรือเป็นสับไทป์ของ Container

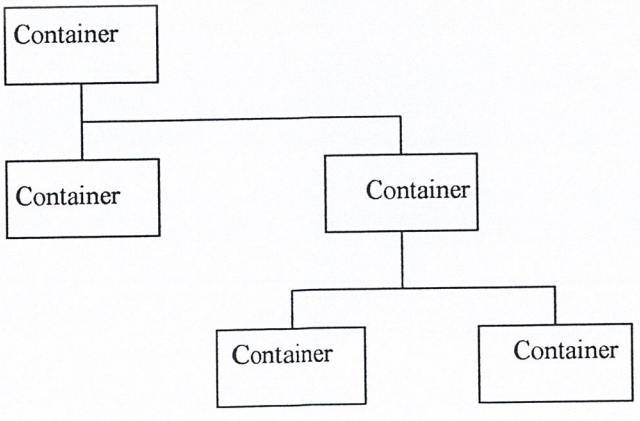
คอมโพเน็นต์จำเป็นต้องอยู่ภายในคอนเทนเนอร์ซึ่งกลุ่มของคอนเทนเนอร์ (หรือคอนเทนเนอร์ด้วยกัน) ที่บรรจุภายในคอนเทนเนอร์เหล่านี้ เปรียบเสมือนการสร้างโครงสร้างแบบรากต้นไม้ของคอมโพเน็นต์ โดยที่คอมโพเน็นต์บนสุดเปรียบเสมือนรากแก้วซึ่งจะแตกแขนงรากย่อยๆ ออกไป ในที่นี้คือ คอมโพเน็นต์ ตามตัวอย่างดังรูป



รูปที่ 2.2 แสดงโครงสร้างแบบรากต้นไม้ของระบบเชื่อมต่อ

2.3.6.1 ประเภทของคอมเทนเนอร์

คอนเทนเนอร์มีอยู่ 4 ประเภท คือ Window ซึ่งประกอบไปด้วยซับไทป์อีก 2 คลาส คือ Frame และ Dialog สำหรับ panel เป็นคลาสที่อยู่ระดับเดียวกันกับ Window ซึ่งแสดงได้ตามรูป



รูปที่ 2.3 แผนภาพแสดงประเภทของคอนเทนเนอร์

คลาส	คุณสมบัติ
Window	เป็นพื้นที่การแสดงผลระดับบนสุด อินสแตนซ์ของคลาส Window จะไม่สามารถถูกเพิ่มเติมหรือฝังตัวภายในคอนเทนเนอร์ใดๆ ได้ รูปร่างของคลาส Window จะไม่แสดงกรอบและหัวเรื่องของวินโดว์
Frame	เป็นพื้นที่การแสดงผลระดับบนสุด มีการแสดงกรอบและแถบชื่อของวินโดว์ อินสแตนซ์ของคลาส Frame อาจมีเมนูก็ได้ การทำงานโดยทั่วไปจะเหมือนกับอินสแตนซ์ของคลาส Window
Dialog	เป็นพื้นที่การแสดงผลระดับบนสุด มีการแสดงกรอบและแถบชื่อของวินโดว์ อินสแตนซ์ของคลาส Dialog จะเกิดขึ้นไม่ได้ถ้าปราศจากการเชื่อมต่อกับอินสแตนซ์ของคลาส Frame
Panel	คอนเทนเนอร์ทั่วไปสำหรับบรรจุคอมโพเน็นต์ โดยการใช้เมธอด add() เพื่อเพิ่มคอมโพเน็นต์ให้แก่อินสแตนซ์ของคลาส Panel

รูปที่ 2.4 ตารางแสดงคุณสมบัติคลาสคอนเทนเนอร์ทั้ง 4 ประการ

2.3.6.2 การสร้างคอมโพเน็นต์

การที่จะเพิ่มเติมคอมโพเน็นต์ใดๆ เพื่อสร้างระบบติดต่อกับผู้ใช้นั้น เราควรสร้างคอนเทนเนอร์เพื่อเป็นที่เก็บคอมโพเน็นต์เสียก่อน สำหรับจาวาแอพพลิเคชันสิ่งแรกที่เราต้องทำคือ การสร้างอินสแตนซ์ของคลาส Window หรือคลาส Frame แต่สำหรับจาวาแอพเพล็ต คลาส Frame จะมีอยู่แล้ว ซึ่งเป็นวินโดว์ของเว็บบราวเซอร์นั่นเอง และเนื่องจากคลาส Applet เป็นสับไทป์ของคลาส Panel ดังนั้นเราจึงสามารถที่จะเพิ่มเติมคอมโพเน็นต์ต่างๆ ให้กับแอพเพล็ตได้เลย

2.3.6.3 การเพิ่มคอมโพเน็นต์ให้กับคอนเทนเนอร์

คอมโพเน็นต์สามารถนำมาเพิ่มให้กับคอนเทนเนอร์ได้โดยการใช้เมธอด add() การใช้งานเมธอด add() นั้นสามารถทำได้ 3 วิธี ขึ้นกับโครงร่างของการวางตำแหน่งคอมโพเน็นต์ (Component Layout)

2.3.6.4 การวางตำแหน่งคอมโพเน็นต์

โครงร่างหรือการวางตำแหน่งคอมโพเน็นต์ ภายในคอมโพเน็นต์นั้นจะไม่ได้ถูกควบคุมจากตัวคอนเทนเนอร์เอง แต่จะถูกควบคุมจากตัวจัดการโครงร่าง (Layout Manager) ซึ่งทำหน้าที่โดยตรงกับตัวคอนเทนเนอร์นั้น ตัวจัดการโครงร่างเป็นตัวจะตัดสินให้กับตัวคอมโพเน็นต์วางอยู่ในตำแหน่งไหนในคอนเทนเนอร์

การจัดการโครงร่างของตัวจัดการโครงร่างมี 5 รูปแบบง่ายๆ ธรรมดาไปจนถึงรูปแบบที่ซับซ้อนมากๆ

1. FlowLayout

โครงร่างนี้จะจัดวางคอมโพเน็นต์จากซ้ายไปขวาในแนวนอนหรือแนวคอลัมน์ก่อน แล้วจึงเริ่มลงมาแถวใหม่ เมื่อคอมโพเน็นต์เกิบขอบเขตในแถวนั้นๆ สำหรับการเพิ่มคอมโพเน็นต์ใหม่เข้าไปจะใช้เมธอด add() ซึ่งมีอาร์กูเมนท์เป็นชื่อของคอมโพเน็นต์ที่จะทำการเพิ่มเข้าไป

2. BorderLayout

โครงร่างนี้จะจัดแบ่งสัดส่วนพื้นที่ของแอพเพล็ต คือ North (ด้านบน), South (ด้านล่าง), West (ด้านซ้าย), East (ด้านขวา) และ Center(กึ่งกลาง) แต่ละคอมโพเน็นต์จะถูกจัดวางตำแหน่งภายในพื้นที่เหล่านี้ จุดเด่นของโครงร่างแบบนี้คือ แต่ละพื้นที่จะคงสัดส่วนของตัวเองไว้เสมอ แม้จะมีการเปลี่ยนแปลงขนาดของคอมโพเน็นต์ไปในรูปแบบใดก็ตาม ซึ่งพื้นที่ที่จะมีขนาดกว้างที่สุด คือ บริเวณ Center และอาร์กูเมนท์ที่ป้อนเข้าในเมธอด add() เพื่อที่จะทำการเพิ่มคอมโพเน็นต์จะมี 2 ตัว คือ ชื่อของออบเจ็กต์ (คอมโพเน็นต์) และชื่อของพื้นที่ที่จะมีการวางตำแหน่งของคอมโพเน็นต์

ข้อสังเกต *คลาสที่เป็นคอนเทนเนอร์แต่ละแบบจะมีโครงร่างหรือเลย์เอาท์แบบเดิม (default) อยู่แล้ว เช่น คลาส Frame และคลาส Dialog จะใช้ตัวจัดการโครงร่างแบบ Borderlayout ส่วนตัวจัดการโครงร่างตั้งต้นของคลาส Panel และ Applet เป็นแบบ FlowLayout*

3. GridLayout

เป็นส่วนแสดงผลของแอพเพล็ตภายในกริดของคอลัมน์และแถว แต่ละส่วนที่ควบคุม User Interface จะถูกแทนที่ในเซลล์ภายในกริดจากซ้ายไปขวา และจากบนลงล่าง ตัวจัดการโครงร่างจะจัดส่วนควบคุมที่ถูกต้องไว้ โดยส่วนประกอบของ User Interface จะปรากฏภายในแอพเพล็ต

4. CardLayout

โครงร่างนี้ใช้จัดที่ว่างบนจอได้ดีกว่าตัวจัดการโครงร่างอื่น ใช้เพื่อเลือกลำดับของส่วนควบคุมที่จะแสดงในเวลาต่างๆ กันเมื่อใส่ส่วนควบคุมเพิ่มไปยังโครงร่างแบบการ์ด ตัวจัดการโครงร่างจะไม่แสดงส่วนควบคุมทั้งหมดออกมาพร้อมกัน แต่จะเลือกแสดงส่วนควบคุมได้โดยใช้ตัวจัดการโครงร่างเป็นตัวจัดการซึ่งนำมาใช้ในกรณีที่ต้องการ User Interface แบบซับซ้อน

5. GridBagLayout

เป็นตัวจัดการโครงร่างแบบพิเศษที่อนุญาตให้แอพเพล็ตใส่ระดับความสำคัญให้แต่ละส่วนควบคุมภายในแอพเพล็ต กล่าวคือ ถ้าแอพเพล็ตมีการเปลี่ยนขนาดใหม่อีกครั้ง ส่วนควบคุมที่มีระดับความสำคัญสูงที่จะครอบครองพื้นที่หน้าจอมากขึ้น

เมธอดพื้นฐานที่ใช้ในการเรียนใช้งานระบบติดต่อผู้ใช้

- add() ใช้เพิ่ม UI ต่างๆ เข้าในคอนเทนเนอร์
- remove() เป็นการนำ UI ออกจากคอนเทนเนอร์
- setLayout() เป็นการเรียกใช้งานตัวจัดการโครงร่าง (Layout Manager)

การใส่แอพเพล็ตลงในเอกสาร HTMLสามารถทำได้โดยใส่แทก <APPLET> ลงในโค้ดของ HTML ซึ่งมีรูปแบบของการเขียนโค้ดดังนี้

<APPLET CODE = Applet.class WIDTH = widthInt HEIGHT = heightInt ><APPLET>

ทุกรูปแบบข้างต้น แสดงถึงองค์ประกอบพื้นฐานต่อไปนี้

1. เริ่มต้นด้วยการประกาศเป็น APPLET-tag คือ <APPLET>
2. บอกถึงไฟล์ที่จะใช้ในการรัน ซึ่งต้องเป็นไฟล์ที่คอมไพล์มาแล้ว นั่นคือเป็นไฟล์ *.class ตามด้วยการบอกถึงขนาดความกว้างและความยาวของพื้นที่ที่ต้องการแสดงแอพเพล็ตนั้น โดยใส่กับ WIDTH และ HEIGHT ตามลำดับ
3. ปิดท้ายด้วย <APPLET>

นอกจากองค์ประกอบข้างต้นแล้ว ยังมีฟังก์ชันที่บอกถึงที่อยู่ของไฟล์ไบท์โค้ดนั้นด้วย ซึ่งมีรูปแบบ

CODEBASE = URL

ในส่วน URL ของ CODEBASE นี้ นอกจากนี้จะใช้เป็น URL แล้วยังสามารถแทนที่ด้วยที่อยู่ได้ เร็กทอรีได้ด้วยและกรณีที่มีการรับข้อมูลเข้าสู่แอพเพล็ตก็สามารถทำได้โดยผ่านแทก <PARAM> ตามรูปแบบ

<APPLET CODE = Applet.class WIDTH = 600 HEIGHT = 400>

<PARAMNAME = param 1 VALUE = value 1>

<PARAMNAME = param 2 VALUE = value 2>

</APPLET>

แต่สำหรับการเขียนโค้ด HTML นั้นไม่มีความจำเป็นต้องเป็นตัวใหญ่ทั้งหมดและกรณีที่เบราเซอร์นั้นไม่สามารถรันแอพเพล็ตได้เราจะใส่แทรก <blockquote> เพื่อให้การทำงานทำข้ามบล็อกนี้ไป

2.3.7 การติดต่อสื่อสารกับโปรแกรมอื่น

แอพเพล็ตสามารถติดต่อสื่อสารกับโปรแกรมอื่นได้ 3 ทาง คือ

1. โดยการเรียกเมธอดของแอพเพล็ตอื่นบนเว็บเดียวกัน
2. โดยการใช้ API ซึ่งจะมีเมธอดที่ช่วยให้แอพเพล็ตสามารถติดต่อกับบราวเซอร์หรือแอพเพล็ตวิวเวอร์ที่เก็บแอพเพล็ตนั้น
3. โดยการใช้ API ช่วยให้สามารถติดต่อกับโปรแกรมอื่นๆ ผ่านเครือข่าย โดยจุดสำคัญก็คือ จะต้องอยู่ในเครือข่ายที่ใช้โฮสท์เดียวกัน

สำหรับการติดต่อสื่อสาร จะมีหัวข้อย่อยๆ ที่จะต้องพิจารณาดังที่กล่าวข้างต้นคือ การส่งเมสเสจไปยังแอพเพล็ตอื่นในเพจเดียวกันซึ่งปกติแอพเพล็ตจะพยายามค้นหาแอพเพล็ตอื่นๆ และสามารถส่งเมสเสจไปยังแอพเพล็ตนั้นได้โดยอยู่บนพื้นฐานของความปลอดภัยดังนี้

1. แอพเพล็ตจะต้องรันอยู่บนเพจเดียวกัน และบนบราวเซอร์ตัวเดียวกัน
2. มีแอพเพล็ตวิวเวอร์ (Appletviewer) หลายตัวที่ต้องการให้แอพเพล็ตเหล่านั้นเกิดจากเซิร์ฟเวอร์เดียวกันด้วยการค้นหาแอพเพล็ตอื่น สามารถทำได้โดยใช้เมธอด getApplet() โดยอาจหาตามรายชื่อของแอพเพล็ต หรือจะค้นหาทั้งหมดเลยก็ได้ ซึ่งปกติแล้วแอพเพล็ตจะไม่มีชื่อ การที่จะทำให้แอพเพล็ตมีชื่อได้จะต้องมีการกำหนดไว้ในไฟล์ HTML โดนการกำหนดทำได้ 2 แบบ คือ

- โดยการใส่ชื่อไว้ใน APPLET – tag

<applet codebase = example/ code = Sender.class width = 450 height = 250 name = **“share”**>

…

</applet>

- โดยการใส่ PARAM - tag

<applet codebase = example/ code = Receiver.class width = 450 height = 350

<param name = “name value = “value”>

…

</applet>

การใช้เมธอด getApplet() เพื่อหาแอพเพล็ตบนเพลทนั้น เมื่อมีการเรียกใช้เมธอดนี้แล้วก็รีเทิร์นค่าของแอพเพล็ตทั้งหมดที่มีอยู่บนเว็บนั้นออกมาให้

2.3.8 Java Application

จาวาแอพพลิเคชันสามารถรันได้โดยตัวมันเอง ซึ่งต่างจากจาวาแอพเพล็ตตรงที่แอพเพล็ตต้องการบราวเซอร์ที่สนับสนุน และจาวาแอพพลิเคชันจะรันโดยเรียกคำสั่ง java ตามด้วยชื่อโปรแกรมที่มีชนิดเป็นคลาส

จาวาแอพพลิเคชันจะเริ่มต้นรันที่เมธอด main() ซึ่งแสดงดังนี้

```
public static void main(String args[])
(
...
)
```

public หมายถึง เมธอดนี้เป็นประโยชน์ต่อคลาสและออบเจ็กต์ต่างๆ ดังนั้นเมธอด main() จึงสามารถประกาศเป็น public

static หมายถึง main() เป็นเมธอดของคลาส

void หมายถึง เมธอด main() ไม่ส่งค่าใดๆ กลับ

main() ใช้พารามิเตอร์เดียวเป็นอะเรย์ของสตริง พารามิเตอร์นี้จะรับอากิวเมนต์ที่ส่งผ่านมาจากบรรทัดคำสั่งของ DOS (DOS command line)

ส่วนตัวลำดับของเมธอด main() สามารถบรรจุโค้ดใดๆ ก็ได้ตามที่ต้องการเช่น

- การกำหนดค่าเริ่มแรกให้ตัวแปร
- การสร้าง instance ต่างๆ ของคลาสใดๆ ที่ประกาศ

เมื่อจาวาประมวลผลเมธอด main() คลาสจะยังไม่ถูกสร้าง instance โดยอัตโนมัติ ดังนั้นเมื่อดำเนินการคลาสในลักษณะออบเจ็กต์แล้วจะต้องสร้าง instance ในเมธอดของ main() ด้วยตัวเราเอง

เพราะว่าแอพพลิเคชันของจาวาเป็นโปรแกรมเดี่ยว มันใช้ประโยชน์ในการผ่านอาร์กิวเมนต์หรือตัวเลือกให้โปรแกรม เพื่อพิจารณาวิธีการที่โปรแกรมจะรันหรือช่วยโปรแกรมดำเนินการอินพุทที่แตกต่างกันมากมาย อาร์กิวเมนต์ของ command–line ซึ่งเราสามารถใช้ตามวัตถุประสงค์ที่เราต้องการได้ เช่น ใช้อินพุตของ debug เพื่อระบุชื่อไฟล์ที่จะอ่านหรือบันทึก หรือสารสนเทศอื่นๆ ที่เราต้องการให้โปรแกรมของจาวารับทราบ

2.3.9 การผ่านอาร์กิวเมนต์ต่างๆ ให้โปรแกรมของจาวา

การผ่านอาร์กิวเมนต์ให้โปรแกรมของจาวา เราจะต้องใส่ค่าของอาร์กิวเมนต์ที่ command-line ของ MS-DOS ในโปรแกรมของจาวาดังนี้

java myProgram arg1 arg2 arg3

บนบรรทัดคำสั่ง (command line) นี้ได้ใช้อาร์กิวเมนต์ 3 ค่า คือ arg1, arg2 และ arg3 แต่ละอาร์กิวเมนต์จะแยกจากกันโดยช่องว่าง เช่น

java myProgram java is best

ซึ่งในตัวอย่างจะส่งไป 3 อาร์กิวเมนต์ แต่ถ้ากลุ่มของอาร์กิวเมนต์ล้อมรอบด้วยเครื่องหมาย "" แล้วจะถือว่าเป็น 1 อาร์กิวเมนต์ เช่น

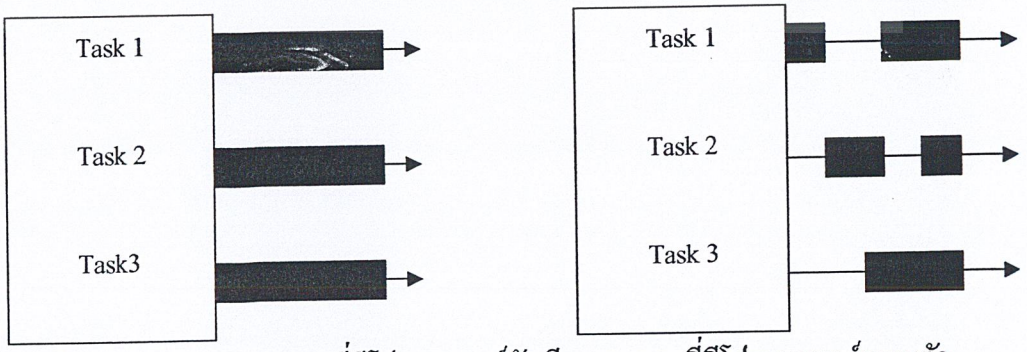
java myProgram "java is best"

2.3.10 การเชื่อมโยงอาร์กิวเมนต์ต่างๆ ในโปรแกรม

จาวาสามารถเชื่อมโยงอาร์กิวเมนต์โดยบรรจุไว้ในอาร์เรย์ของสตริง ซึ่งจะถูกส่งผ่านไปให้เมธอด main() ซึ่งก็คือ args ภายในเมธอด main() เราสามารถเชื่อมโยงอาร์กิวเมนต์ args[] เป็นตัวรับอาร์กิวเมนต์จาก command-line ได้

2.3.11 การทำมัลติเธรด (Multitherading program)

เธรด (thread) มีลักษณะเหมือนโปรแกรมทั่วๆ ไป คือ มีทั้งจุดเริ่มต้นและจุดสิ้นสุดอย่างชัดเจน ในช่วงเวลาใดเวลาหนึ่งและเวลาใดๆ ขณะที่เธรดกำลังทำงานจะมีจุดประสงค์เพื่อทำการเอ็กซีคิวท์ (execute) โปรแกรมเพียงอย่างเดียว แต่อย่างไรก็ตามโดยตัวมันเองแล้ว เธรดไม่ใช่โปรแกรม เพราะไม่สามารถทำงานด้วยตัวเองได้ แต่ต้องทำงานร่วมกับโปรแกรม



รูปที่ 2.5 แสดงถึงการทำงานของเธรดแบบที่มีโปรเซสเซอร์ตัวเดียวและแบบที่มีโปรเซสเซอร์หลายตัว

การทำงานของเธรดโดยทั่วไปแล้วไม่ได้เป็นการทำงานของเธรดเพียงอย่างเดียว แต่จะเป็นการทำงานของหลายๆ เธรดภายในโปรแกรมเดียวกัน โดยให้เธรดทำงานพร้อมๆ กันแต่จะให้ทำหน้าที่คนละอย่างกัน ในหนังสือบางเล่มจะใช้คำว่า lightweight process แทนคำว่า เธรด และเหตุที่เรียกว่า lightweight ก็เพราะเธรดทำงานภายใต้สภาวะแวดล้อมเดียวกันกับตัวโปรแกรมและใช้ทรัพยากรที่โปรแกรมมีการจองไว้ แต่อย่างไรก็ตามเธรดจะต้องมีทรัพยากรบางอย่างที่เป็นของตัวเอง เช่น สแตก (stack) โปรแกรมเคาน์เตอร์ (program counter) เป็นต้น โค้ดที่อยู่ในเธรดจะทำงานภายใต้สิ่งแวดล้อมที่กล่าวมาเท่านั้น ซึ่งอาจเรียกเธรดได้อีกชื่อหนึ่งว่าexecuting context

2.3.11.1 คุณสมบัติของเธรด

ภายในเธรดจะมี Thread body การทำงานทุกอย่างจะเกิดขึ้นใน Thread body ซึ่งอยู่ในเมธอด run() หลังจากเธรดถูกสร้างและทำการตั้งค่าแล้ว รันไทม์ซิสเต็ม (runtime system) จะเรียกเมธอด run() ของเธรดขึ้นมาทำงานโดยโค้ดในเมธอด run() จะเป็นตัวทำงานทุกอย่างของเธรดนั่นเองและบ่อยครั้งการทำงานของเธรดจะมีการทำงานแบบลูป (loop) ทำให้การทำงานใช้เวลานาน เพื่อลดเวลาและขนาดของเมธอด run() เราสามารถกระทำได้วิธีใดวิธีหนึ่งจากวิธีต่อไปนี้

1. สร้างสับคลาส (sub class) ของคลาส Thread ใน java.lang.Thread และทำการโอเวอร์ไรด์เมธอด run()
2. ทำการ implements Runnable interface ในแพ็กเก็ต java.lang เช่นกัน แล้วใช้อินสแตนซ์ของเธรดเป็นตัวจัดการงานต่างๆ ที่ทำให้เมธอด run() เช่นกัน

2.3.11.2 สถานะของเธรด

เธรดสามารถอยู่ในสถานะใดสถานะหนึ่งใน 5 สถานะต่อไปนี้คือ new, ready, running, inactive หรือfinishedเมื่อเธรดถูกสร้างขึ้นเธรดจะเข้าสู่สถานะ new และหลังจากที่เธรดถูกสั่งให้เริ่มต้นด้วยเมธอด start() เธรดจะเข้าสู่สถานะ ready ซึ่งที่สถานะ ready นี้เธรดพร้อมที่จะทำงานแล้วแต่ยังไม่มีการทำงานใดๆ เกิดขึ้น ขึ้นกับระบบปฏิบัติการจะเป็นผู้จอง CPU timeให้กับเธรดนั้นๆ

เมื่อเธรดที่พร้อมจะทำงานถูกสั่งให้ทำงาน (execute) เธรดจะเข้าสู่สถานะ running ซึ่งเธรดที่กำลังทำงานอยู่นี้อาจเข้าสู่สถานะ ready ได้อีกครั้งหนึ่ง เมื่อหมดเวลาหรือ CPU time ที่ระบบปฏิบัติการจัดมาให้หรืออาจเกิดจากเมธอด yeild() ถูกเรียกให้ทำงานก็ได้

เธรดสามารถเข้าสู่สถานะ inactive ได้จากหลาย ๆ เหตุผล กล่าวคือ อาจเรียกใช้เมธอด sleep(), wait() หรือ Suspend() ก็ได้ แต่กรณีที่เธรดมีการติดต่อทำงานกับระบบ I/O เธรดก็จะเข้าสู่สถานะ inactive ได้เช่นกัน เธรดที่อินแอกทีพ (inactive thread) อยู่จะกลับมาแอกทีพได้อีกครั้งหนึ่งเมื่อกิจกรรมที่ทำให้เกิดการอินแอกทีพเสร็จสิ้น ยกตัวอย่างเช่นเธรดถูกสั่งให้ sleep แล้วเวลาดังกล่าวก็หมดเวลา ในกรณีนี้เองเธรดก็จะกลับมาเป็นแอกทีพเธรด และเข้าสู่สถานะ ready อีกครั้งนั่นเอง ท้าย

สุดเธรดจะเข้าสู่สถานะ finished เมื่อทำการรันจนเสร็จสิ้นในเมธอด run() หรือมีการเรียกใช้เมธอด stop()

2.3.11.3 ระดับความสำคัญของเธรด (Thread Priority)

เธรดทุกตัวในจาวามีระดับความสำคัญ โดยปกติแล้วเธรดจะได้รับความสำคัญมาจากโปรแกรมที่แตกเธรดออกมา แต่เราก็สามารถเปลี่ยนระดับความสำคัญของเธรดได้โดยใช้เมธอด setPriority() และสามารถหาระดับความสำคัญของเธรดได้จากเมธอด getPriority() โดยที่ระดับความสำคัญของเธรดนั้นจะเป็นตัวเลขที่มีค่าจาก 1 ถึง 10 แต่ในคลาส Thread เองจะมีค่าคงที่ MIN_PRIORITY, NORM_PRIORITY และ MAX_PRIORITY ที่แทนด้วยค่า 1, 5 และ 10 ตามลำดับ โดยระดับความสำคัญของเธรดหลักจะเป็น Thread_NORM_PRIORITY

ในจาวารันไทม์ซิสเต็ม (Java runtime system) จะนำเธรดที่มีระดับความสำคัญที่สุดมารันก่อนแต่ถ้าเธรดเหล่านั้นมีลำดับความสำคัญเท่ากัน CPU จะถูกจองให้เธรดเหล่านั้นแบบ round-robin และเธรดที่มีระดับความสำคัญต่ำกว่าจะถกนำมารันได้ก็ต่อเมื่อเธรดที่มีระดับความสำคัญสูงกว่าถูกรันไปจนหมดแล้วเท่านั้น

2.3.11.4 เธรดที่ทำงานอยู่เบื้องหลัง (Daemon Thread)

เธรดในจาวาสามารถเป็นเธรดที่ทำงานอยู่เบื้องหลังหรือเดมอนเธรดได้ เดมอนเธรดเป็นเธรดที่ให้บริการกับเธรดที่กำลังทำงานอยู่ในโปรเซส (process) เดียวกัน เมธอด run() ของเดมอนเธรดจะทำงานไปเรื่อย ๆ ไม่สิ้นสุดเพื่อรอการขอบริการจากเธรดอื่น และเมื่อมีเธรดเพียงเธรดเดียวเหลืออยู่ในโปรเซสคือเดมอนเธรด ตัวอินเตอร์พรีเตอร์ (interpreter) ก็จะจบการทำงาน (เนื่องจากไม่มีเธรดอื่นที่ต้องบริการ)

ในการกำหนดให้เธรดเป็นเดมอนเธรดทำได้โดยใช้เมธอด setDaemon() พร้อมใส่อาร์กิวเมนต์ “true” ส่วนการตรวจสอบว่าเธรดใดเป็นเดมอนเธรดจะใช้เมธอด isDaemon()

2.3.11.5 กลุ่มของเธรด

เธรดทุกเธรดในจาวาเป็นสมาชิกของกลุ่มเธรด (thread group) กลุ่มเธรดเป็นเครื่องมือในการรวมมัลติเธรดเข้าด้วยกันเป็นออบเจ็กต์เดียวกัน และใช้เธรดเหล่านั้นทั้งหมดพร้อมกัน กลุ่มเธรด implements คลาส Thread Group ในแพ็กเก็ต java.lang

รันไทม์ซิสเต็มจะทำการรวมเธรดทุกเธรดของเราเข้ากับกลุ่มของเธรดที่มีอยู่แล้ว หรือจะตั้งกลุ่มเธรดขึ้นมาใหม่ก็ได้และเมื่อเธรดเป็นสมาชิกของกลุ่มเธรดใดแล้วก็ไม่สามารถย้ายไปยังเธรดกลุ่มอื่นได้

2.3.11.6 โปรแกรมแบบมัลติเธรด

การซิงโครไนซ์เธรด (Synchronizing Thread) บ่อยครั้งจะมีการใช้ข้อมูลร่วมกันและจำเป็นต้องพิจารณาถึงสถานะและกิจกรรมที่เธรดอื่นทำอยู่ โดยกลุ่มของโปรแกรมที่อยู่ในสถานการณ์นี้เรียกว่า producer และ consumer เข้าจังหวะกัน (synchronize) ปัญหาที่เกิดจากการทำงานไม่เข้าจังหวะกันเรียกว่า race conditions ซึ่งเกิดขึ้นได้เมื่อเธรดทำงานไม่เข้าจังหวะกันและทำการเข้าถึงข้อมูลหรือออบเจ็กต์ตัวเดียวกัน ทำให้ได้ข้อมูลที่ไม่ถูกต้อง เราสามารถแก้ปัญหานี้ได้ 2 วิธี คือ monitors และใช้ notifyAll คู่กับเมธอด wait()

1. Monitors

ออบเจ็กต์ที่ถูกใช้ร่วมกันระหว่างเธรดสองตัวที่จะทำการเข้าถึงจะต้องทำให้เข้าจังหวะกัน ซึ่งเรียกว่า ข้อมูลควบคุม (condition variable) monitor จะเกี่ยวข้องกับการกำหนดข้อมูล(condition variable) และการล็อก (lock) ข้อมูลเหล่านี้เมื่อเธรดทำการ monitor บนตัวข้อมูลตัวใด แล้วมีเธรดอื่นมาทำการล็อกข้อมูลจะทำให้เธรดที่ทำการ monitor บนข้อมูลนั้น (แต่ไม่ได้เป็นคนล็อกข้อมูล) ไม่สามารถใช้ข้อมูลตัวนั้นได้ ส่วนของโค้ดที่อยู่ในโปรแกรมที่มีการเข้าถึงข้อมูลตัวเดียวกันในเธรดเรียกว่า critical sections ซึ่งในจาวาสามารถกำหนดตัวโปรแกรม (เมธอดหรือตัวแปรให้เป็น critical sections ได้โดยใช้คีย์เวิร์ด sychronized

2. notifyAll กับเมธอด wait()

เมธอดทั้งคู่นี้เป็นสมาชิกของ java.lang.Object notifyAll และเมธอด wait() ถูกใช้งานโดยเธรดที่เป็นผู้ถือล็อกอยู่

- notifyAll ใช้ในการบอกให้กับเธรดอื่นที่รอบน monitor โดยให้เธรดปัจจุบันตื่นขึ้นหนึ่งในจำนวนที่รออยู่ ส่วนตัวที่เหลือก็ monitor และทำงานไป
- เมธอด wait() จะทำให้เธรดปัจจุบันหยุดการทำงานและรอจนกว่าเธรดอื่นจะบอกถึงการเปลี่ยนแปลงสถานะ

2.4 ซ็อกเก็ต (socket)

ซ็อกเก็ตเป็นกลไกระดับล่างที่ใช้ในการแลกเปลี่ยนข้อมูล ซึ่งใช้รับส่งข้อมูลข้ามเครือข่าย TCP/IP โดยข้อมูลที่ใช้ในการรับส่งเรียกว่าดาตาแกรม (Datagram) ซึ่งซ็อกเก็ตจะมี 2 แบบ คือ

อินเตอร์เน็ตโปรโตคอลซ็อกเก็ต (IP socket) กับยูสเซอร์คาตาแกรมซ็อกเก็ต (UDP socket) โดยไอพีซ็อคเกตใช้รับส่งไอพีคาตาแกรมในขณะที่ยูดีพีซ็อคเกตใช้รับส่งยูดีพีคาตาแกรม โดยส่วนใหญ่แล้วไอพีซ็อกเก็ตจะนำมาใช้งานแพร่หลายมากกว่ายูดีพีซ็อกเก็ต

ในจาวาจะมีแพ็กเก็ต java.net ที่สนับสนุนการใช้อินพุตและเอาท์พุตซ็อกเก็ต และเนื่องจาก java.net ถือเป็นแพ็กเก็ตหลักที่อยู่ในจาวา ดังนั้นจึงนำมาใช้ได้ในทุกมาตรฐานของเครื่องจำลองการทำงานจาวา (java virtual machine)

2.4.1 ข้อดีข้อเสียของซ็อกเก็ต

ซ็อกเก็ตเปรียบเหมือนการแลกเปลี่ยนข้อมูลข้ามเครือข่ายกันในระดับล่าง ซึ่งโดยหลักแล้วจะทำงานได้เร็วและใช้หน่วยความจำได้มีประสิทธิภาพมากกว่า ซึ่งพอสรุปได้ว่าซ็อกเก็ตเหมาะสมสำหรับการรับส่งข้อมูลง่ายๆ จำนวนไม่มากนัก เช่น ข้อความ Ascii แต่จะไม่เหมาะสมสำหรับการส่งข้อมูลที่มีขนาดใหญ่และมีความซับซ้อน เช่นออบเจ็กต์

2.5 ไคลเอนต์/เซิร์ฟเวอร์(Client/Server)

จากอดีตการเราใช้ระบบแบ่งปันเวลา (Time Sharing) ซึ่งมีเครื่องเมนเฟรมเป็นโฮสต์ในการจัดการงานทุกอย่าง ทั้งเก็บข้อมูล ประมวลผล การคำนวณต่างๆ โดยที่ดั๊มเทอร์มินัลมีหน้าที่เพียงแค่แสดงผลลัพธ์ของข้อมูลเท่านั้น ต่อมามีการออกแบบระบบเครือข่ายท้องถิ่น (LAN) มีเซิร์ฟเวอร์อยู่ 1 ตัว ทำหน้าที่เป็นไฟล์เซิร์ฟเวอร์คอยเก็บข้อมูลและแอปพลิเคชั่นไว้ จึงต้องมีฮาร์ดดิสก์ความจุหลายจิกะไบต์อยู่บนไฟล์เซิร์ฟเวอร์เครื่องนี้ เพราะเครื่องไคลเอนต์ไม่มีฮาร์ดดิสก์ จะมีเพียงแค่ดิสก์ไดร์ฟสำหรับใส่แผ่นบูตเข้าสู่ระบบเน็ตเวิร์กหรือก๊อปปี้ข้อมูลเท่านั้น การทำงานส่วนใหญ่จะยังอยู่ที่เซิร์ฟเวอร์ เมื่อมีผู้เข้ามาขอใช้ไฟล์และแอปพลิเคชั่นมากขึ้นและไฟล์มีขนาดใหญ่ขึ้นก็จะทำให้ประสิทธิภาพการทำงานของไฟล์และแอปพลิเคชั่นลดลงทันที

ระบบไคลเอนต์/เซิร์ฟเวอร์ได้ถูกพัฒนาขึ้นมาเพื่อตอบสนองแนวความคิดการดาวน์ไซท์ซิ่ง (Downsizing) ให้มีประสิทธิภาพและค่าใช้จ่ายที่ต่ำกว่าระบบแบ่งปันเวลา (Time Sharing) ของเครื่องเมนเฟรม ไคลเอนต์/เซิร์ฟเวอร์ เป็นระบบการประมวลผลแบบกระจาย (Distributed Processing) โดยจะแบ่งการประมวลผลระหว่างเซิร์ฟเวอร์และไฟล์ แทนที่โปรแกรมแอปพลิเคชั่นจะรันอยู่เฉพาะบนเครื่องเซิร์ฟเวอร์ ก็จะมีการแบ่งการทำงาน การคำนวณของโปรแกรมแอปพลิเคชั่นให้มาทำงานบนเครื่องไคลเอนต์ด้วย และเมื่อใดที่เครื่องไคลเอนต์ต้องการผลลัพธ์บางส่วนของข้อมูล จะมีการร้องขอไปยังเครื่องเซิร์ฟเวอร์ เพื่อให้ส่งเฉพาะข้อมูลบางส่วนเท่านั้นกลับมาให้เครื่องไคลเอนต์เพื่อคำนวณข้อมูลนั้นอีกทีหนึ่ง อาจกล่าวได้ว่าทศวรรษที่ 90 เป็นระบบการทำงานของไคลเอนต์/เซิร์ฟเวอร์ ซึ่งจะช่วยให้องค์กรต่างๆ ลดค่าใช้จ่ายค่าบำรุงรักษา สำหรับระบบเมนเฟรมและมินิลงได้มาก

รูปแบบของไคลเอนต์/เซิร์ฟเวอร์ที่ใช้งานมีอยู่ 4 ชนิดด้วยกันคือ

1. Stan alone ไคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบนี้ผู้ให้บริการหรือเซิร์ฟเวอร์จะอยู่บนเครื่องเดียวกับผู้ขอใช้บริการหรือไคลเอนต์ ทำให้มีความเร็วในการติดต่อสื่อสารระหว่างผู้ให้บริการและผู้ขอใช้บริการสูงมาก แต่ประสิทธิภาพในการประมวลผลระบบฐานข้อมูลจะลดลงบ้าง ระบบนี้เรียกอีกอย่างว่า Tiny ไคลเอนต์/เซิร์ฟเวอร์
2. Department ไคลเอนต์/เซิร์ฟเวอร์ หรือ LAN based single เซิร์ฟเวอร์ การทำงานแบบนี้จะมีผู้ให้บริการเกี่ยวกับฐานข้อมูล แอปพลิเคชั่น ฯลฯ อยู่บนเครื่องเซิร์ฟเวอร์ และผู้ขอใช้บริการทั้งหลายจะอยู่บนเครื่องไคลเอนต์ โดยจะเชื่อมต่อกันด้วยระบบเครือข่ายท้องถิ่น (LAN) และมิดเดิลแวร์ (Middleware) เป็นตัวกลางที่ทำงานอยู่ระหว่างไคลเอนต์และเซิร์ฟเวอร์ การติดต่อระหว่างผู้ให้บริการและผู้ขอใช้บริการจะช้ากว่าแบบ Stan alone เพราะจะต้องติดต่อผ่านระบบเครือข่าย ยิ่งถ้ามีผู้ขอใช้บริการเข้ามาดึงข้อมูลกันครั้งละมากๆ หลายๆ เครื่อง ประสิทธิภาพจะลดลงอย่างเห็นได้ชัด วิธีเพิ่มประสิทธิภาพก็คือการเพิ่มเครื่องเซิร์ฟเวอร์ขึ้นในระบบ
3. Workgroups ไคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบเวิร์กกรุ๊ปนี้จะเป็นกลุ่มของเซิร์ฟเวอร์ที่หลากหลายแพลต ฟอร์ม หลายผู้ผลิต มีความแตกต่างกันของเซิร์ฟเวอร์ แต่ทั้งหมดนี้จะเชื่อมต่อกันทางระบบเครือข่าย ระบบเครือข่ายท้องถิ่น และ WAN และใช้มิดเดิลแวร์มาตรฐานในการทำงาน
4. Enterprise ไคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบเอ็นเทอร์ไพท์หรือระดับองค์กร จะทำให้มีการเชื่อมโยงเครื่องเซิร์ฟเวอร์หรือโฮสต์ต่างแพลตฟอร์มเข้าด้วยกัน ทำให้มีการใช้ทรัพยากรบนระบบได้อย่างมีประสิทธิภาพสูงสุด โดยที่ไคลเอนต์สามารถที่จะเลือกใช้ทรัพยากรบนระบบได้อย่างมีประสิทธิภาพสูงสุดโดยที่ไคลเอนต์สามารถจะเลือกใช้ทรัพยากร ฐานข้อมูลจากเซิร์ฟเวอร์เครื่องใดก็ได้ผ่านทางมิดเดิลแวร์

2.5.2 โครงสร้างพื้นฐานของ ไคลเอนต์/เซิร์ฟเวอร์

จากการพัฒนาของระบบปฏิบัติการเน็ตเวิร์คทั้งทางค่ายไมโครซอฟท์ คือ วินโดว์ เอ็นที เซิร์ฟเวอร์ และโนเวล คือ เน็ตแวร์ (Netware) ทำให้การทำงานบนระบบเน็ตเวิร์คเป็นไปอย่างสะดวกรวดเร็ว และกว้างขวางขึ้น โดยเฉพาะโปรโตคอลแบบอีเทอร์เน็ต (Ethernet)ที่นิยมใช้ในปัจจุบันซึ่งเหมาะกับระบบไคลเอนต์/เซิร์ฟเวอร์ อีกทั้งระบบปฏิบัติการที่รันอยู่บนเครื่องไคลเอนต์ก็มีความสามารถทางเน็ตเวิร์ค และการทำงานแบบมัลติเธรด วินโดว์ เอ็นที เซิร์ฟเวอร์ เป็นระบบปฏิบัติการ 32 บิต ที่ทำงานแบบ Preemptive Multithreading ซึ่งจะแกงานของ แอปพลิเคชั่นแต่

ละตัวออกจากกันทำให้ไม่มีการรบกวนการทำงานระหว่างกัน เพราะโมเดลในการออก วินโดว์ เอ็นที เซิร์ฟเวอร์ ก็เป็นสถาปัตยกรรมไคลเอนต์/เซิร์ฟเวอร์อยู่แล้ว

2.5.2.1 ไคลเอนต์ เป็นส่วนที่จะรันแอปพลิเคชั่นบนไคลเอนต์โดยใช้ระบบจียูไอ (GUI (Graphical ผู้ใช้งาน Interface)) หรือ โอโอยูไอ (OOUI (Object Oriented ผู้ใช้งาน Interface)) หรือ ดีเอสเอ็ม (DSM (Distributed System Management)) เป็นการติดต่อกับ ผู้ใช้งาน ผ่านระบบกราฟฟิกซึ่งทำงานแบบเชิงวัตถุ

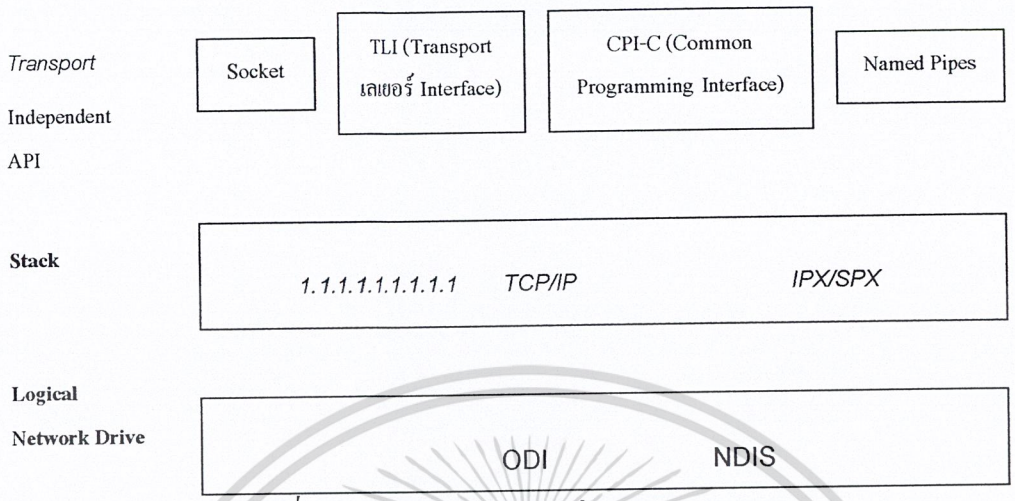
2.5.2.2 มิดเดิลแวร์ (Middleware) เป็นส่วนที่ทำงานอยู่ระหว่างไคลเอนต์และเซิร์ฟเวอร์เป็นเสมือนสะพานเชื่อมการทำงานสามารถแล่นออกเป็น 4 แบบ คือ Service Specific, DSM, NOS และ Transport stack

- Service Specific หรือการบริการโดยเฉพาะจะขึ้นอยู่กับการใช้แอปพลิเคชั่นในการทำงาน เช่น แอปพลิเคชั่นของ object แบบกระจายจะใช้ มิดเดิลแวร์ ORB (Object Reguest Broker) แอปพลิเคชั่นกรุ๊ปแวร์ จะใช้มิดเดิลแวร์ เมล์ และ ทรานส์แอคชั่น (TP monitor)จะใช้มิดดิลแวร์ TxRPC (Transactional Remote Procedure Call) ส่วนระบบฐานข้อมูลซีเควล (SQL) จะใช้ ODBD (Open Database Connection) DRDA (Distribute Relational Database Architecture) ของ ไอบีเอ็ม, RDA (Remote Database Access), Oracle Glue, CLI (Call-Level Interface)
- DSM (Distributed System Management) จะรันบนทุกโหนดของระบบเน็ตเวิร์คที่เป็นไคลเอนต์/เซิร์ฟเวอร์ จะมีมิดเดิลแวร์ SNMP (Simple Network Management โพรโตคอล), CMIP (Common Management Information โพรโตคอล) และ DME
- NOS (Network Operation System) เป็นระบบปฏิบัติการเน็ตเวิร์คซึ่งให้บริการทั่วๆ ไป โดยจะมีทั้ง Directory Services, Naming Service, Security/Authentication Service, Messaging Service, Distributed file, RPC, Peer to Peer ฯลฯ ระบบปฏิบัติการเหล่านี้เช่น วินโดว์ เอ็นที เซิร์ฟเวอร์, Netware, Banyan Vines, OSF DCE
  - NOS จะช่วยให้การใช้ชื่อ (namespace) เพียงชื่อเดียวสามารถเข้าถึงทรัพยากรต่างๆ บนระบบเน็ตเวิร์ครวมได้
  - NOS จะทำให้ผู้ใช้งาน (User) ไม่ต้องรับรู้เกี่ยวกับความผิดพลาดที่เกิดขึ้น เช่น การรับ-ส่งข้อมูลผิดพลาด ระบบเน็ตเวิร์คมีปัญหาหรือมีการเคลื่อนย้ายทรัพยากรจากไดเรกทอรี่ไปยังเซิร์ฟเวอร์ NOS จะแก้ไขและอัพเดตข้อมูลต่างๆ ให้เป็นหนึ่งเดียว
  - NOS สามารถใช้รหัสผ่านเพียง 1 ชุด เข้าสู่ระบบเน็ตเวิร์คจากเครื่องใด ที่ไหนก็ได้ โดยจะใช้ระบบรักษาความปลอดภัยแบบ DEC (Distributed Computing Environment) ในการตรวจสอบ
  - NOS จะมีระบบโกลบอลไดเรกทอรี่ (Global Directory) ซึ่งจะนำคน แอปพลิเคชั่น โปรแกรม สิ่งต่างๆ เข้ามาทำงานร่วมกัน ทำให้ไม่ต้องขึ้นกับสถานที่ สามารถจะเปลี่ยนสถานที่ในการเข้าใช้ทรัพยากรได้

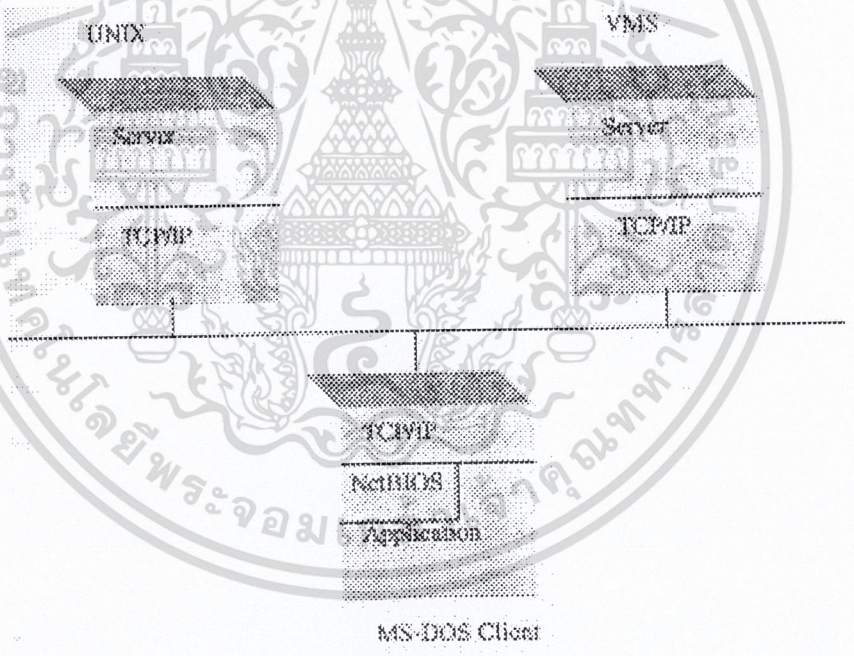
- NOS จะจัดการในเรื่องการกระจายเวลา (Distributed time) ให้ทั้งระบบคือจะมีการซิงโครไนซ์ในเรื่องเวลาระหว่าง เซิร์ฟเวอร์และไคลเอนต์ทุกตัว
- NOS จะจัดการในเรื่องการกระจายความปลอดภัย (Distributed Security) อย่างต่ำอยู่ในระดับ C2 ซึ่งจะต้องมีการแสดงตน (Authentication) มีการเข้ารหัสผ่าน (Encrypt) ใช้มาตรฐาน Kerboros และแอปพลิเคชั่นเซิร์ฟเวอร์จะมีอำนาจ (Authentication) ในการใช้ ACLs (Access ควบคุม List) เพื่อควบคุมการเข้าใช้ทรัพยากรของ ผู้ใช้งาน
- NOS สามารถที่จะใช้มิดเดิลแวร์ MOM (Message Oriented Middleware) ในการช่วยจัดคิวข้อความ (Message queue) เพื่อให้ทั้งไคลเอนต์และเซิร์ฟเวอร์ยังคงทำงานได้อย่างต่อเนื่อง แม้จะมีปัญหาทางระบบสื่อสาร ลักษณะนี้อาจเรียกว่า Loosely-Coupled queue based และอีกรูปแบบคือ การใช้ RPCs (Remote Procedure Calls) ซึ่ง NOS เหล่านี้คือ OSF/DCE, ONC/SUN, เน็ตแวร์ 4.xx/โนเวล

- Transport stack เป็นบริการพื้นฐานในการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์ บนระบบ LAN และ WAN โพรโตคอล หลักๆ ในส่วนของ Transport stack มีอยู่ด้วยกัน 4 ตัวด้วยกันคือ NetBIOS, ทีซีพี/ไอพี (TCP/IP), ไอพีเอกซ์/เอสพีเอกซ์ (IPX/SPX) และเอสเอ็นเอ (SNA)
  - NetBIOS เป็น โพรโตคอล ที่ออกแบบโดยบริษัท ไอบีเอ็ม ให้ใช้งานกับเครือข่ายขนาดเล็ก ต่อมาพัฒนาเป็น NetBEUI (NetBIOS Extended ผู้ใช้งาน Interface) สามารถจะใช้งานกับระบบเครือข่ายที่มีเครื่องคอมพิวเตอร์ตั้งแต่ 20-200 เครื่อง ไม่สามารถใช้งานกับเครือข่ายขนาดใหญ่ได้ และไม่สามารถค้นหาเส้นทางได้ จะเห็นว่าใช้กับงานเวิร์คกรุ๊ป เช่น วินโดว์ส สำหรับ เวิร์คกรุ๊ป (Microsoft LAN Manager) โพรโตคอล NetBIOS จะทำงานอยู่ในชั้นของ เซสชัน เลเยอร์ ตามมาตรฐานโอเอสไอ (OSI-7 เลเยอร์)
  - ทีซีพี/ไอพี (TCP/IP (Transmission Control โพรโตคอล/ อินเทอร์เน็ต โพรโตคอล)) เป็น โพรโตคอล ที่ใช้งานบนระบบยูนิกซ์ (UNIX) พัฒนาขึ้นในปี 2512 โดยกระทรวงกลาโหมของสหรัฐอเมริกามีเครือข่ายชื่อ อัลปาเน็ต (ARPANET (Advanced Research Project Agency Network)) สำหรับใช้งานกับเครือข่ายขนาดใหญ่อย่าง WANs มีความสามารถในการค้นหาเส้นทาง และมีความยืดหยุ่นในการทำงานสูง
  - ไอพีเอกซ์/เอสพีเอกซ์ (IPX/SPX (Internet work Packet Exchange/Sequenced Packet Exchange)) เป็น โพรโตคอล หลักของระบบปฏิบัติการเน็ตเวิร์ค เน็ตแวร์ มีความฉลาดในการทำงานกว่า NetBIOS คือสามารถค้นหาเส้นทางได้ทำให้โพรโตคอล ไอพีเอกซ์/เอสพีเอกซ์ สามารถจะทำงานบนระบบเครือข่ายท้องถิ่น และ WAN ได้

- เอสเอ็นเอ (SNA (Systems Network Architecture)) เป็น โพรโตคอล ที่ออกแบบโดย บริษัท ไอบีเอ็มเพื่อใช้งานบนระบบเครือข่ายเครื่องเมนเฟรมของไอบีเอ็ม



รูปที่ 2.6 แสดงโครงสร้างระดับล่างของ Transport stack



รูปที่ 2.7 แสดงการใช้ โพรโตคอล ทีซีพี/ไอพี (TCP/IP) ในระบบ ไคลเอนต์/เซิร์ฟเวอร์

2.5.2.3 เซิร์ฟเวอร์ เป็นส่วนที่จะรันแอปพลิเคชั่นในการจัดการทรัพยากรต่างๆ สำหรับระบบไคลเอนต์/เซิร์ฟเวอร์สามารถแบ่งออกได้เป็น 4 แบบด้วยกันคือ

- ระบบฐานข้อมูลซีเควล (DBMS)
- ระบบจัดการทรานส์แอคชั่น (TP monitor)
- ระบบกรุ๊ปแวร์ (Groupware)
- ระบบออบเจ็กต์แบบกระจาย (Distributed Objects)

- ระบบบานข้อมูล ซีเควล การประมวลผลฐานข้อมูลในระบบไคลเอนต์/เซิร์ฟเวอร์ จะมีสองส่วนคือ ส่วนหลัง และ ส่วนหน้า ส่วนหลัง จะเป็นส่วนของเซิร์ฟเวอร์ซึ่งจะมีระบบฐานข้อมูล ซีเควล เช่น ไมโครซอฟท์ ซีเควล เซิร์ฟเวอร์ ทำหน้าที่เก็บข้อมูล จัดเรียงลำดับ ค้นหา เรียกใช้ ป้องกันข้อมูล ฯลฯ และมีส่วนหน้า มีวินโดว์ส เช่น ซีเควล วินโดว์ส, เดลไฟ(Delphi) ฯลฯ ด้วยเหตุที่ ซีเควล เป็นภาษาในเชิงสอบถาม อธิบายแบบมีโครงสร้าง คำสั่งที่ใช้อ่านเข้าใจง่าย Process การทำงานเริ่มจาก ผู้ใช้งาน ส่งคำสั่งเข้าไปเพื่อขอใช้บริการผู้ให้บริการรับคำสั่งมาทำการประมวลผลเสร็จแล้วส่งผลลัพธ์กลับไปให้ผู้ใช้งาน ผู้ขอใช้บริการ จะเห็นว่ามีเพียงผลลัพธ์ที่ ผู้ใช้งาน ต้องการเท่านั้นที่ถูกส่งออกไปบนระบบเน็ตเวิร์ค เป็นการลดทราฟฟิก (Traffic) ของระบบลงด้วย มีส่วนของชุดคำสั่งภาษา ซีเควล เรียกว่า Stored Procedure สำหรับจัดการข้อมูล นอกจากนี้ยังมี Triggers, Rules, Views และ Scroll cursor ช่วยทำให้ข้อมูลมีความถูกต้องและสมบูรณ์ที่สุด
- ระบบการจัดการแอปพลิเคชั่น การทำงานบนเครื่องเมนเฟรมทั้งระบบ จะมีความสลับซับซ้อนของโปรแกรมมากจึงต้องมี ทรานส์แอคชั่น (TP monitor) (Transaction Processing monitor) ซึ่งเป็นระบบติดตามการประมวลผลทรานส์แอคชั่นอยู่ในระบบด้วย ซอฟท์แวร์ประเภท ทรานส์แอคชั่น (TP monitor)ที่นิยมใช้กันคือ CICS (IBM), Tuxedo (BEA) และ Encina สำหรับระบบไคลเอนต์/เซิร์ฟเวอร์ การนำ ทรานส์แอคชั่นมาใช้งานนับว่ามีประโยชน์ และเพิ่มประสิทธิภาพในการประมวลผลแบบกระจาย (Distributed Processing) ทรานส์แอคชั่น จะจัดการกับทรานส์แอคชั่น โดยการจัดเส้นทางเดินในระบบให้จากไคลเอนต์ไปยังเซิร์ฟเวอร์ตัวใดตัวหนึ่งบนระบบแล้วกลับมาที่เดิม ถ้าไม่ประสบความสำเร็จ ก็จะเริ่มต้นทำงานใหม่ งานที่ ทรานส์แอคชั่น ทำคือ การจัดการทรัพยากร และ ผู้ใช้งาน Request จัดการเรื่องของ Two phase commit เก็บ Log ของทรานส์แอคชั่นและทรัพยากรในระบบด้วย สำหรับระบบ NOS ของ วินโดว์ เอ็นที เซิร์ฟเวอร์ ของบริษัทไมโครซอฟท์ได้อกแบบ MTS (Microsoft Transaction เซิร์ฟเวอร์) ให้ทำงานร่วมกับบานข้อมูล ไมโครซอฟท์ ซีเควล เซิร์ฟเวอร์ อย่างมีประสิทธิภาพ นอกจากนี้เรายังสามารถจะใช้ซอฟท์แวร์ ทรานส์แอคชั่น ของเมนเฟรม เช่น Tuxedo ให้มาทำงานบนแพลตฟอร์ม วินโดว์ เอ็นที เซิร์ฟเวอร์ ได้อีกด้วย
- ระบบกรุ๊ปแวร์ กรุ๊ปแวร์เป็นเทคโนโลยีบนระบบเมนเฟรมซึ่งมีหลากส่วนทำงานร่วมกัน เช่น ระบบ อิเล็กทรอนิคเมล์ ระบบเวิร์คโฟล ระบบจัดการเอกสารแบบมัลติมีเดีย ระบบจัดการรูปภาพ ระบบจัดตารางเวลา ระบบการประชุม แต่สามารถนำมาใช้งานกับระบบไคลเอนต์/เซิร์ฟเวอร์ได้ ซอฟท์แวร์ประเภทนี้คือ โลตัสโน๊ต (Lotus Notes), DCA OpenMind, Image Plus/2
- ระบบออบเจ็กต์ต์แบบกระจาย เทคโนโลยีออบเจ็กต์ต์แบบกระจาย (Distributed Object) ช่วยให้ระบบไคลเอนต์/เซิร์ฟเวอร์แบบนี้นำหน้าระบบไคลเอนต์/เซิร์ฟเวอร์ทั้ง 3 แบบ

(ระบบฐานข้อมูล ซีเควล , ระบบ ทรานส์แอคชั่น และ ระบบกรุ๊ปแวร์) ด้วยการรวบรวมขั้นตอนในการทำงานและข้อมูลไว้ในออบเจ็กต์ซึ่งมันจะเดินทางไปบนระบบเน็ตเวิร์ค (ได้ทุกหนทุกแห่ง) และยังทำงานแบบมัลติแพลตฟอร์มอีกด้วยเช่น AIX, SUN-solaris, HP-UX, IBM-MVS, Digital UNIX Open VMS, ลินุกซ์(Linux) ฯลฯ กลุ่ม โอเอ็มจี (OMG (Object Management Group))ได้ออกแบบออบเจ็กต์ต์สถาปัตยกรรม COBRA (Common Object Reguest Broker Architecture) ให้ใช้งานนานร่วม 8 ปี พร้อมทั้งมีบริษัทที่ใช้งาน COBRA อยู่ร่วม 600 บริษัท ทางด้านไมโครซอฟท์ก็ได้พัฒนา ดีคอม (DCOM (Distributed Common Object Model)) ออกมาโดยเริ่มจากแพลตฟอร์มบนวินโดว์ส 95 และ วินโดว์ส เอ็นที นอกจากนี้กำลังจะพอร์ตข้ามไปยัง MVS, UNIX, Solaris, AIX, SCO UNIX, HP-UX, ลินุกซ์(Linux), Open VMS

2.6 คลาส URL

คลาส URL เป็นคลาสที่ห่อ URL ไว้ใช้บอกที่อยู่ของเอกสารที่เราจะทำการดาวน์โหลดมาจากเซิร์ฟเวอร์ สามารถสร้างคลาสนี้ได้จาก constructor

public URL(String url) throws MalformedURLException

อาร์กิวเมนต์สตริงเป็นตัวบอกที่ตั้งของเอกสารที่ URL อ้างถึง เช่น ถ้าเราจะอ้างถึงเว็บไซท์ เราสามารถสร้างคลาส URL ได้โดย

URL kmitl = new URL ("http://www.ce.kmitl.ac.th");

ซึ่งจะ throws exception ขึ้นถ้ารูปแบบตัวแปรไม่ถูกต้องตามรูปแบบของ URL ซึ่งเราสามารถจัดการกับ exception ได้โดยใช้ try-catch block หรือให้เมธอดที่มีการเรียกใช้ constructor นี้ throws MalformedURLException ก็ได้ (หรืออาจใช้คลาสบรรพบุรุษ Exception แทนก็ได้)

เมื่อเราทำการสร้าง URL ที่ชี้ไปยังที่ตั้งของเอกสารแล้วเราสามารถที่จะดึงเอกสารออกมาโดยใช้ เมธอด OpenStream ซึ่งจะรีเทิร์น InputStream ที่เก็บเนื้อหาของเอกสารไว้ออกมาซึ่งสามารถอ่าน InputStream เพื่อเข้าถึงข้อมูลในเอกสารได้อย่างง่ายดาย เมธอด getContent รีเทิร์นออบเจ็กต์ที่ห่อเนื้อหาของเอกสารเอาไว้ และทำงานได้สำเร็จเมื่อเครื่องจำลองการทำงานจาวามีการดูแลและจัดการชนิดของเอกสารที่ทำการเข้าถึงอย่างเหมาะสม

2.7 คลาส InetAddress

คลาส InetAddress ใช้แสดงถึงแอดเดรสของ internet host เราใช้คลาสนี้ในการที่จะสร้างซ้อกเก็ตหรือออบเจ็กต์ดาตาแกรม ซึ่งคลาส InetAddress ไม่มี constructor แต่ใช้เมธอดสแตติกเพื่อที่จะรีเทิร์นอินสแตนซ์ของ InetAddress ออกมาให้

2.8 คลาส Socket

ใน internet host หรือเครื่องเจ้าบ้านจะมีกลุ่มหมายเลขพอร์ตที่จะใช้สื่อสารกับเครื่องอื่น ซึ่งคลาสของซ็อคเกตเป็นตัวห่อหมายเลขพอร์ตทั้งยังจัดหาเมธอดในการรับส่งข้อมูลผ่านพอร์ตดังกล่าวด้วย เราสามารถสร้างซ็อกเก็ตโดยใช้ constructor

Socket(String host, int port) หรือ

Socket(InetAddress address, int port)

เช่นต้องการจะเปิดซ็อกเก็ตบนพอร์ตหมายเลข 80 ที่เครื่อง www.ce.kmitl.ac.th สามารถเขียนได้โดย

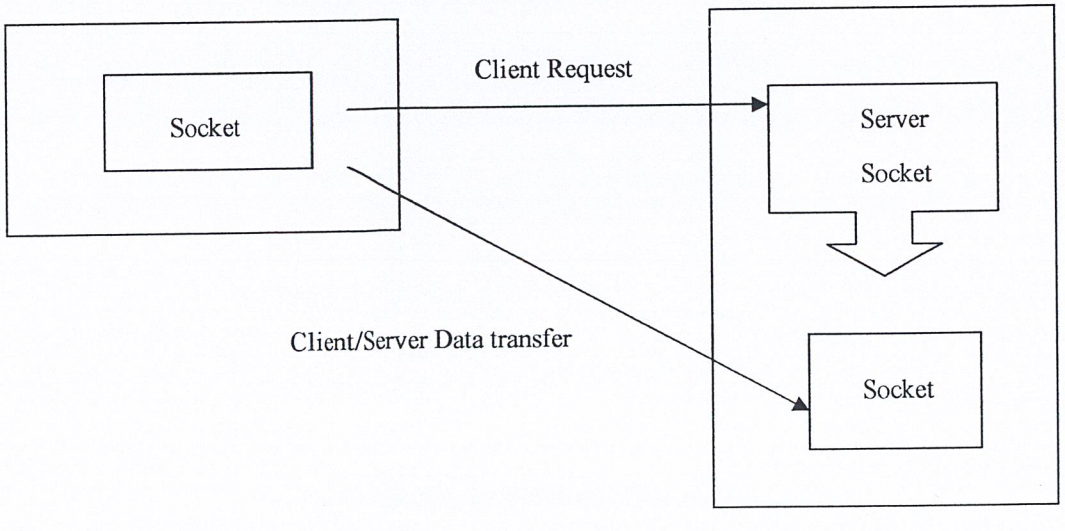
Socket kmitl = new Socket ("www.ce.kmitl.ac.th",80)

คลาส Socket ยังมี constructor ที่เฉพาะมากกว่าแต่เรานำมาใช้งานน้อยจึงไม่ขอกล่าวถึง ซ็อกเก็ต constructor จะ throws exception ขึ้นเมื่อมีข้อผิดพลาดในการสร้างซ็อกเก็ต ดังนั้นจึงจำเป็นต้องใช้ try-catch block หรือ throws clause เพื่อจัดการกับข้อผิดพลาดดังกล่าว

เมื่อเราได้สร้างซ็อกเก็ตขึ้นมาแล้ว เราสามารถใช้เมธอด getInputStream เพื่อใช้ติดต่อกับ InputStream และเมธอด getOutputStream เพื่อติดต่อกับ OutputStream โดยการใช้เมธอดที่ทำการอินพุต/เอาท์พุต ธรรมดาในการที่จะอ่านและเขียนผ่านสายข้อมูล (Stream) นี้ เปรียบเหมือนทำการอินพุต/เอาท์พุตลงบนไฟล์ ในกรณีที่ใช้ InputStream และ OutputStream เราจะพบว่าคลาส BufferedInputStream และ PrintStream มีความสะดวกในการใช้งานมากกว่า

2.9 คลาส ServerSocket

คลาส ServerSocket ถูกใช้โปรแกรมของเซิร์ฟเวอร์ ซึ่งใช้เมธอด accept เพื่อรอการร้องขอการติดต่อจากไคล์เอนท์และรีเทิร์นซ็อคเกตที่ใช้ในการติดต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์เพื่อแลกเปลี่ยนข้อมูลระหว่างกัน ข้อมูลสำคัญคือ ซ็อกเก็ตออบเจ็กต์ที่ฝั่งไคลเอนท์จะทำการสวิตช์โดยอัตโนมัติ โดยใช้ซ็อกเก็ตตัวใหม่เพื่อทำการติดต่อสื่อสาร



รูปที่ 2.8 แสดงถึงการทำงานของ ServerSocket

ในการที่จะสร้างเซิร์ฟเวอร์ซ็อกเก็ตเราใช้

constructor ServerSocket(int port)

เมื่อ constructor เกิดข้อผิดพลาดก็จะ throw IOException แต่ถ้าไม่มีข้อผิดพลาดใดๆ เซิร์ฟเวอร์ซ็อกเก็ตก็พร้อมที่จะใช้งาน เมื่อเราให้เซิร์ฟเวอร์ซ็อกเก็ตทำการรับข้อมูล (accept) การทำงานของโปรแกรมจะหยุดรอโดยไม่ทำงานใดๆ จนกว่าจะมีการร้องขอการติดต่อจากไคล์เอนท์ แล้วหลังจากนั้นเซิร์ฟเวอร์ซ็อกเก็ตจะรีเทิร์นซ็อกเก็ตให้ เช่น เพื่อจะสร้างเซิร์ฟเวอร์ซ็อกเก็ตที่รออยู่บนพอร์ตหมายเลข 1234 ในการติดต่อกับไคล์เอนท์ทำได้โดย

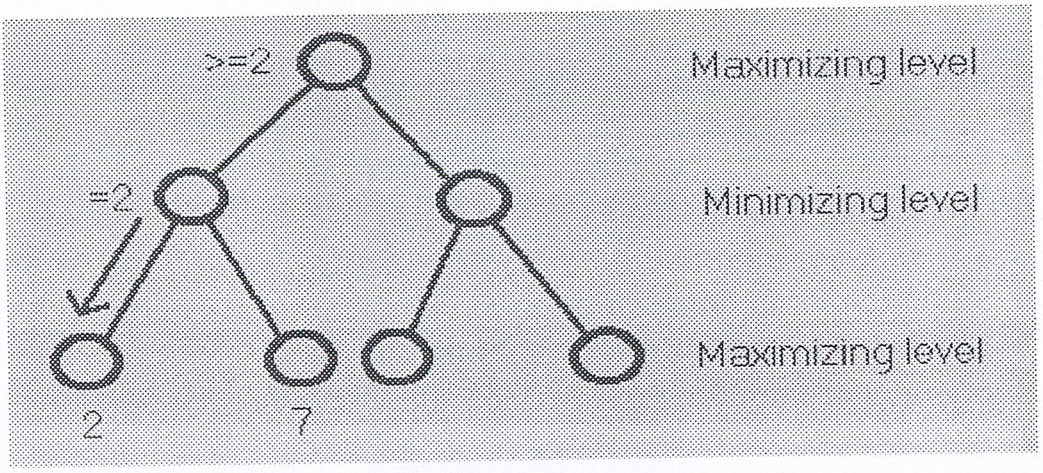
```
ServerSocket server = new ServerSocket(1234);
Socket client = server.accept();
```

2.10 ดาตาแกรม (Datagram)

ดาตาแกรมใช้ยูดีพีซ็อคเกตในการติดต่อสื่อสาร ซึ่งยูดีพีซ็อคเกตนี้จะไม่ทำการติดต่อให้อัตโนมัติเหมือนในไอพีซ็อคเกตและยิ่งกว่านั้น TCP/IP ไม่รับประกันว่าเครื่องเจ้าบ้านทำการติดต่อกับยู ดีพีแพ็กเก็ตได้ (บางครั้งเราเรียกดาตาแกรม) ตามลำดับที่ถูกต้องเหมือนตอนที่ทำการส่งมาก ดังนั้นยูดีพี แพ็กเก็ตจึงใช้งานหลักๆ อยู่เพียงการส่งข้อมูลสั้นๆ ซึ่งมีโครงสร้างข้อมูลไม่ซับซ้อน อีกทั้งผู้ดูแลระบบเครือข่ายในบางองค์กรได้ทำการปรับไฟล์วอลเพื่อทำยูดีพีแพ็กเก็ตจากนอกองค์การ ดังนั้นเมื่อเราต้องการที่จะใช้ยูดีพีแพ็กเก็ตจึงควรที่จะติดต่อสอบถามกับผู้ดูแลระบบก่อนที่จะทำการลงมือเขียนโปรแกรม เพื่อเป็นการประหยัดค่าใช้จ่ายด้านเวลา

2.11 Minnimax Game

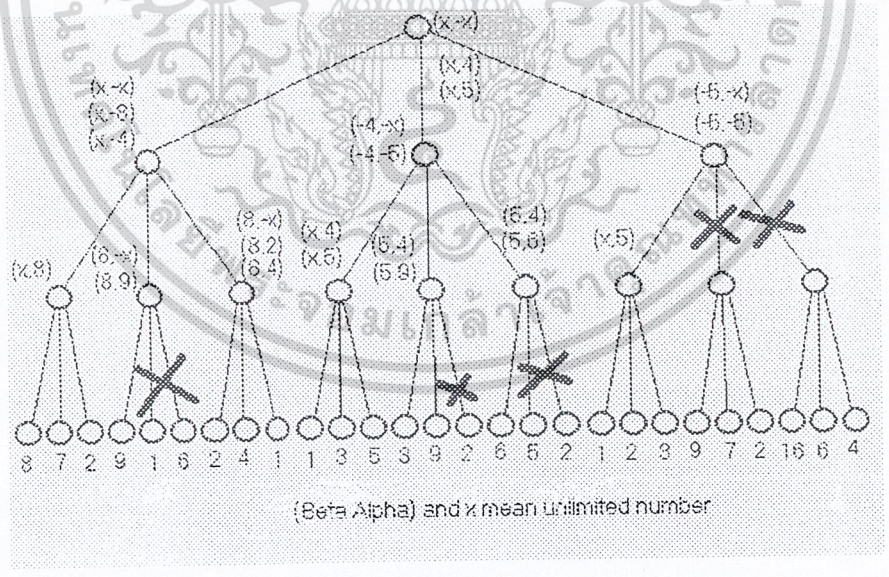
Max, Min คือ ผู้เล่น 2 คนที่มีเป้าหมายสวนทางกัน โดย Max ต้องการหาทางเล่นโดยที่ตนเองจบเกมโดยมีแต้มสูงสุด ซึ่งตรงกันข้ามกับ Min โดยอัลกอริทึมนี้มีสมมติฐานว่า โปรแกรมมีเวลาเหลือเฟือในการหาเส้นทางที่ไม่ปกติต่างๆ ดังนั้นเมื่อเราสามารถสร้างโหนดต่างๆ ของ tree มาได้แล้วก็จะต้องทำการกำหนดค่าน้ำหนักให้แต่ละโหนดในโหนดสุดท้ายเพื่อที่จะทำการคำนวณแผนการเล่นได้ ตัวอย่างเช่น



รูปที่ 2.9 ตัวอย่าง Mininmax Game

2.11 Alpha Beta Plunning

การทำ Plunning ก็เพื่อลดเวลาในการค้นหา เนื่องจากว่าเรามีจำนวนโหนดมาก ดังนั้นเราจะจึงเสียเวลาในการ Search มาก ทั้งๆ ที่เรารู้อยู่ว่าต้องการได้ค่าที่มากที่สุดดังนั้น เราจึงสามารถจะ plun โหนดที่มีค่าน้อยเมื่อเทียบกับ กิ่งติดกันออกได้ เมื่อทำเช่นนี้แล้วจะพบว่าเราจะสามารถทำการค้นหาได้รวดเร็วยิ่งขึ้น



รูป 2.10 การทำ Plunning

บทที่ 3

การสร้าง และการออกแบบ

3.1 เกมออนไลน์

เกมออนไลน์เป็นเกมที่ถูกพัฒนาขึ้นมาเพื่อสร้างความบันเทิงให้กับผู้เล่นบนระบบเครือข่ายอินเตอร์เน็ต ซึ่งสามารถเล่นได้กับทุกคนที่กำลังใช้งานบนอินเตอร์เน็ตอยู่

3.2 สิ่งที่ควรพิจารณาเมื่อเขียนเกมด้วยจาวา

ในการเขียนเกมด้วยจาวาจำเป็นต้องพิจารณาถึงองค์ประกอบเหล่านี้ คือ

1. **กราฟฟิกและแอนิเมชันหรือภาพเคลื่อนไหว**
2. **การป้อนอินพุท**
3. **เสียง**
4. **ระบบเครือข่าย**
5. **การจัดการกับมิเดีย**

3.3 กราฟฟิกและแอนิเมชันหรือภาพเคลื่อนไหว

เนื่องจากกราฟฟิกและภาพเคลื่อนไหวถือเป็นจุดสำคัญที่จะทำให้ตัวเกมน่าเล่นหรือน่าเบื่อ ดังนั้นจึงถือเป็ฯจุดสำคัญที่ควรจะพิจารณาเป็นอันดับแรกในการเขียนเกม จาวามีเอพีไอ (API) ที่สนับสนุนการทำงานเกี่ยวกับกราฟฟิกต่างๆ เช่น รูปภาพ กราฟฟิก 2 มิติ เป็นต้น ในส่วนของภาพเคลื่อนไหวนั้น ถึงแม้ว่าจาวาจะไม่มีเอพีไอที่สามารถจัดการได้โดยตรงแต่ก็มีแนวทางในการเขียนที่สามารถทำให้ง่ายขึ้น

3.4 การป้อนอินพุต

การป้อนหรือการรับอินพุตก็เป็นจุดสำคัญอีกจุดหนึ่งในการเล่นเกมเพราะเป็นสิ่งที่ใช้บ่งบอกความรู้สึกระหว่างผู้เล่นกับเกม จาวาได้สนับสนุนการทำงานของอุปกรณ์ที่ใช้รับอินพุทจากผู้เล่นไว้

สองแบบ คือ คีย์บอร์ดและเมาส์ โดยในขณะที่เขียนโปรแกรมอยู่ ผู้เขียนสามารถทำการตรวจสอบการทำงานของอุปกรณ์เหล่านี้ได้ โดยการตรวจสอบเหตุการณ์ที่เกิดขึ้นจากอุปกรณ์เหล่านี้

3.5 เสียง

เสียงก็นับว่าเป็นจุดสำคัญในเกมเหมือนกัน จาวาสนับสนุนการทำงานกับเสียงได้ไม่ดีเท่าที่ควร เนื่องจากทำงานได้กับรูปแบบของเสียงเพียงสองรูปแบบ คือ *.au และ *.midi

3.6 ระบบเครือข่าย

ระบบเครือข่ายเป็นส่วนที่สำคัญที่ทำให้จาวาได้รับความนิยมอย่างมาก ระบบเครือข่ายของจาวาเป็นรูปแบบที่สนับสนุนการใช้งานด้วย API อย่างง่ายมาก และจาวาก็เป็นภาษาที่ไม่ขึ้นกับระบบปฏิบัติการใดๆ ซึ่งทำให้สมารถเล่นเกมกับผู้อื่นที่ใช้เครื่องชนิดอื่น เช่น Sun หรือ Macintoshes ได้อย่างไม่เป็นปัญหา

การจัดการกับมีเดียร์

การจัดการกับมีเดียร์ต่างๆ คือ การคอยตรวจสอบมีเดียร์ต่างๆ เช่น กราฟฟิก เสียงและอื่นๆ ว่าถ่ายโอนข้อมูลสมบูรณ์หรือยัง เพื่อป้องกันปัญหารูปภาพแสดงไม่เต็มจอขณะที่เล่นเกมอยู่

3.7 การออกแบบเกม

เป็นการพิจารณาว่าเกมจะออกมาในรูปแบบไหน ซึ่งหัวข้อที่ต้องพิจารณาเป็นหัวข้อย่อย ดังนี้

แนวความคิดพื้นฐาน (Basic Idea)

แนวการดำเนินเกม (Story Line)

โหมดการเล่นเกม (Play Mode)

3.8 แนวความคิดพื้นฐาน

เป็นแนวความคิดเกี่ยวกับรูปแบบของเกมว่าจะเป็นแบบผจญภัย เกมแอ็คชั่น เกมยิงหรือรวมกันหลายๆ รูปแบบ

3.9 แนวการดำเนินเกม

เป็นสิ่งที่ช่วยให้ขอบเขตของเกมแคบขึ้น ทำให้มองภาพรวมของเกมทั้งหมดได้ชัดเจนขึ้น และแนวทางดำเนินเกมจะช่วยในการเขียนหรือพัฒนาโปรแกรมได้ง่ายขึ้น

3.9 โหมดการเล่นเกม

เป็นรายละเอียดของเกมในลักษณะที่ว่ามีจำนวนตัวผู้เล่นเท่าใด เล่นคนเดียวหรือสองคน เล่นคนเดียวต่อหน้าจอหรือเล่นสองคนต่อหน้าจอเป็นต้น

3.10 แนวความคิดพื้นฐานสำหรับการเล่นเกมแบบหลายคน

สำหรับการพัฒนาเกมบนเครือข่ายสามารถเล่นได้พร้อมกันได้หลายคนบนเครือข่าย (Network Game) หรืออาจจะไม่ต้องการติดต่อกับเครือข่ายแต่ก็สามารถเล่นแบบหลายคนได้ (Non-Network Game)

เกมแบบหมุนรอบเป็นเกมที่มีรูปแบบการเล่นที่ผู้เล่นจะมีการสับเปลี่ยนกันเล่นตามรูปด้านล่างโดยที่เกมนี้สามารถทำงานบนเน็ตเวิร์คได้ไม่ยากนัก เพราะมีผู้เล่นเพียงคนเดียวที่สามารถติดต่อกับเกมในเวลานั้น และจะผลัดเปลี่ยนผู้เล่นทุกช่วงเวลาค่าหนึ่ง เกมประเภทนี้จะออกแบบให้ผู้เล่นส่วนใหญ่ต้องรอจนกว่าจะถึงตาที่ตัวเองเป็นผู้เล่น

3. 11 ปัญหาที่อาจเกิดขึ้นในเกมแบบเครือข่ายและวิธีการแก้ปัญหา

ปัญหาที่เกิดขึ้นส่วนใหญ่ จะเป็นปัญหาที่เกี่ยวกับการซิงโครไนซ์เซชัน (Synchronization) ซึ่งการซิงโครไนซ์เซชันเป็นการระบุเกี่ยวกับจำนวนของอินสแตนซ์ของเกมที่มีข้อมูลอยู่ในสภาวะเดียวกัน เพื่อให้ผู้เล่นแต่ละคนทำงานได้เข้าจังหวะกันได้อย่างเหมาะสม

ส่วนการแก้ปัญหานั้น ใช้วิธีสเตทซิงโครไนซ์เซชัน (State Synchronization)ซึ่งจะเป็นวิธีในการสื่อสารโดยที่แต่ละอินสแตนซ์จะสื่อสารสภาวะปัจจุบันของตัวเองให้กับอินสแตนซ์

ในที่นี้เราได้เลือกเกมในแนวการวางแผนขึ้นมา เนื่องจากเกมในแนวนี้จะแบ่งการเล่นออกเป็นรอบ และสามารถวางแผน หรือคาดคะแนไว้ล่วงหน้าได้ซึ่งทำให้ง่ายต่อการเขียนโปรแกรมสำหรับส่วนติดต่อ และในส่วนของ AI ที่จะสามารถคาดเหตุการณ์ในอนาคตได้แม่นยำขึ้น

เกมที่นำมาใช้คือเกมไพ่จับหมู เนื่องจากมีการเล่นเป็นรอบ โดย 1 รอบมี 4 คน ทำให้ง่ายต่อการติดต่อ โดยทุกคนจะติดต่อผ่านเซิร์ฟเวอร์ แล้วตัวเซิร์ฟเวอร์ก็จัดการติดต่อกับผู้ใช้คนอื่นอีกที จากรูปแบบการทำงานแบบนี้ทำให้ตัว AI(ปัญญาประดิษฐ์) จะประมวลผลที่เซิร์ฟเวอร์และมีความสามารถในการดึงข้อมูลมาใช้ในการประมวลผล เพราะข้อมูลทั้งหมดก็เก็บไว้ที่เซิร์ฟเวอร์

โปรแกรมนี้เขียนด้วยภาษา JAVA ซึ่งมีความสามารถเปิดด้วยเวบบราวเซอร์ที่สามารถเล่นจาวาได้ ทำให้ผู้เล่นไม่จำเป็นต้องคอมไพล์หรือติดตั้งตัวโปรแกรมนี้ในเครื่องของผู้ใช้ ตัวโปรแกรมนี้อนุญาตให้ผู้ใช้หลายคนสามารถเล่นและเก็บข้อมูลของผู้ใช้แต่ละคน การออกแบบตั้งอยู่ในระบบไคลเอ็นต์-เซิร์ฟเวอร์ โดยตัวเซิร์ฟเวอร์เป็นคนจัดการทุกอย่างทั้งตัวผู้ใช้ และการเล่น ส่วนผู้ใช้ติดต่อผ่าน

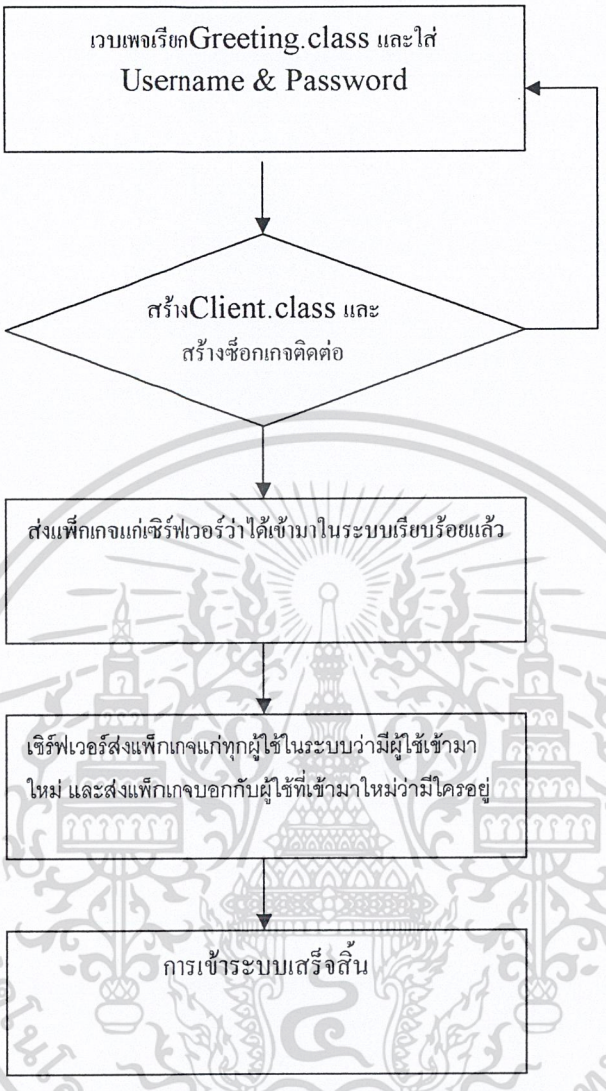
ไคลเอ็นต์โปรแกรมโดยผ่าน GUI ทำให้ผู้ใช้สามารถติดต่อกับผู้ใช้คนอื่น เข้าร่วมเกม และกิจกรรมอื่นๆ ในระบบนี้ การทำงานในระบบนั้นจริงเป็นเพียงการการส่งเมสเซสระหว่างไคลเอ็นต์ กับเซิร์ฟเวอร์ ตัวเมสเซจจะเป็นตัวบอกให้ตัวเซิร์ฟเวอร์ และตัวไคลเอนทำกิจกรรมใดบ้าง

จากที่กล่าวมาข้างต้น เราสร้างคลาสเพื่อใช้สำหรับทั้งส่วนไคลเอ็นต์และเซิร์ฟเวอร์ ในส่วนของเซิร์ฟเวอร์นั้นมีคลาสเซิร์ฟเวอร์ (Server)ที่คอยรอรับข้อมูล โดยมีคลาสผู้เล่น (Player) เป็นตัวคอยอ่านข้อมูลให้เซิร์ฟเวอร์สำหรับแต่ล่ะผู้ใช้ และต้องมีคลาสไพ่(Card) และคลาสมือ(Hand) เพื่อให้เกมดำเนินต่อไป ในการอ่านข้อมูลนั้นต้องมีส่วนที่เหมือนโปรโตคอลของควบคุมการทำงาน ทั้งส่วนของเซิร์ฟเวอร์และไคลเอน ตัวโปรโตคอลนี้เราก็สร้างคลาสแพ็กเกจ(Packet) มารองรับซึ่งทำหน้าที่สร้างแพ็กเกจข้อความ สำหรับแต่ล่ะข้อมูลที่ส่งออกมา เราต้องการแพ็กเกจหลากหลายรูปแบบเพื่อนำข้อมูลเดินทางไปและบอกว่าควรทำอย่างไรกับข้อมูลนี้ จาวามีคลาสมากมายสำหรับระบบเครือข่ายแต่เราได้เลือก ServerSocket และ Socket เพราะเราต้องการส่งข้อมูลแบบจุดต่อจุด และเราไม่ต้องการกำหนดรูปแบบของข้อมูล และเราส่งข้อมูลในรูปแบบของไบต์(Byte) เนื่องจากมีประสิทธิภาพสูง

นอกจากคลาสที่กล่าวด้านบนแล้วเซิร์ฟเวอร์ต้องการเธรด (Threads) อีก 2 เพื่อ ในการอ่านข้อมูลจากไคลเอ็นต์ คือ “listener” ซึ่งอยู่ในคลาสผู้เล่น และอีกตัวก็คือใช้เพื่อการส่งข้อมูล “replier” ซึ่งเก็บไว้ในคลาสส่งข้อมูลออก (PlayerOutput) ที่ส่งข้อมูลออกไปหาไคลเอ็นต์ ที่เราใช้ 2 เธรดแทนที่ใช้แค่ 1 เพราะว่าจะทำให้ไคลเอ็นต์สามารถส่งข้อมูลมาได้เรื่อยๆแล้วเซิร์ฟเวอร์ก็สามารถอ่านได้พร้อมส่งข้อมูลออกไปได้ในรูปแบบขนาน ในส่วนของไคลเอ็นต์นั้นก็เหมือนกันมีเธรดคอยฟังข้อมูล ที่เก็บไว้ในคลาสไคลเอ็นต์ (Client) แต่ในส่วนการส่งข้อมูลนั้นจะใช้AWT แทนคือเมื่อมีเหตุการณ์เกิดขึ้นในแต่ละเหตุการณ์นั้นจะส่งข้อมูลออกไปเอง คลาสไคลเอ็นต์นอกจากจะใช้รับส่งข้อมูลแล้ว ยังเป็นตัวที่สร้างซ็อกเก็ตเพื่อติดต่อกับเซิร์ฟเวอร์ด้วย

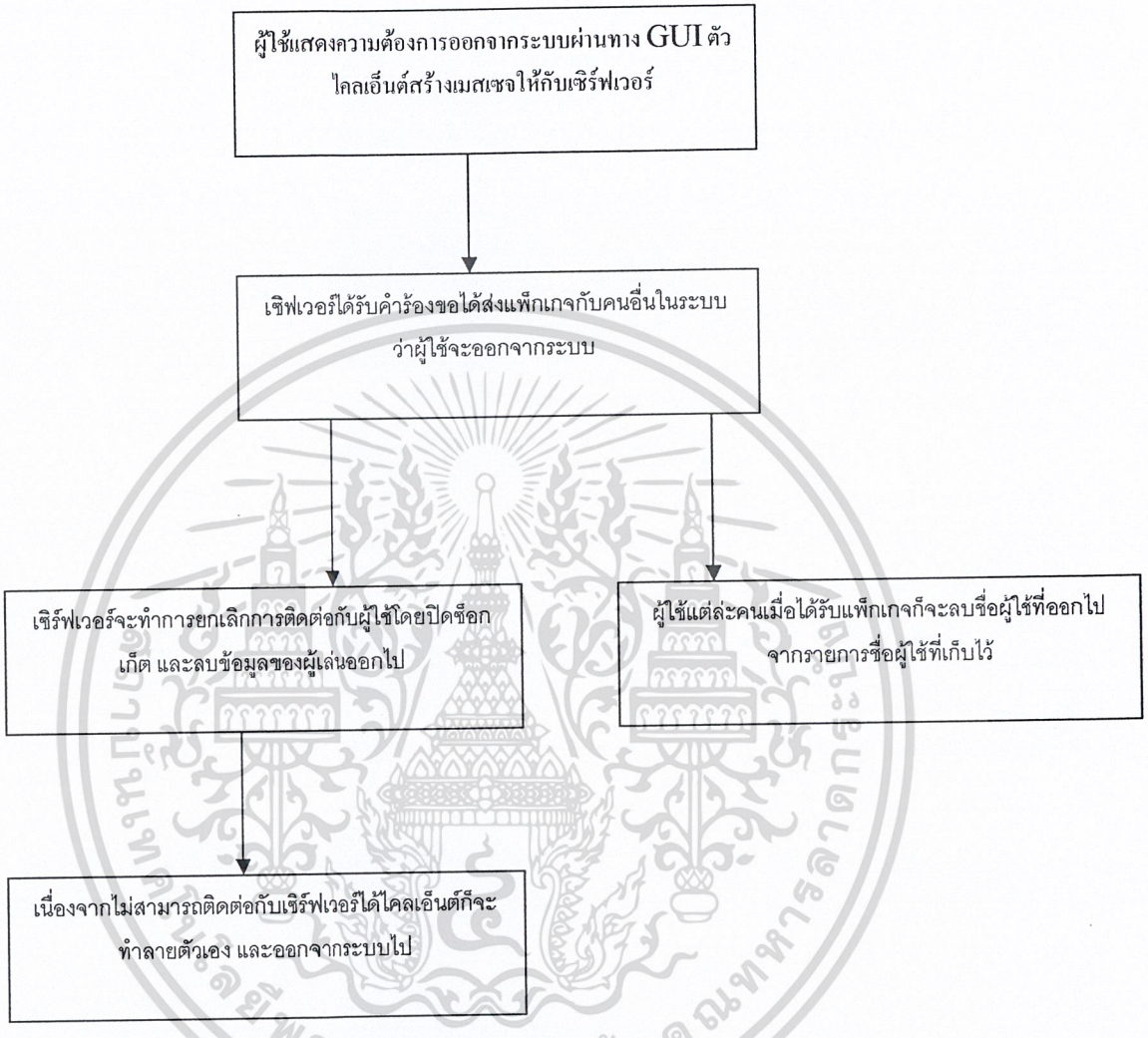
ในการใช้ผู้ใช้จะเข้าเวบเพจที่โปรแกรมเราบริการอยู่ ซึ่งจะต้องมีคลาสที่คอยสร้างคลาสไคลเอ็นต์เพื่อติดต่อกับเซิร์ฟเวอร์ พร้อมทั้งรับชื่อที่ใช้เข้าระบบ กับรหัสผ่านอีกด้วยคลาสนี้เราให้ชื่อว่า Greeting.class

จากที่กล่าวมาข้างต้นสามารถเขียนแผนภาพของระบบได้ดังนี้



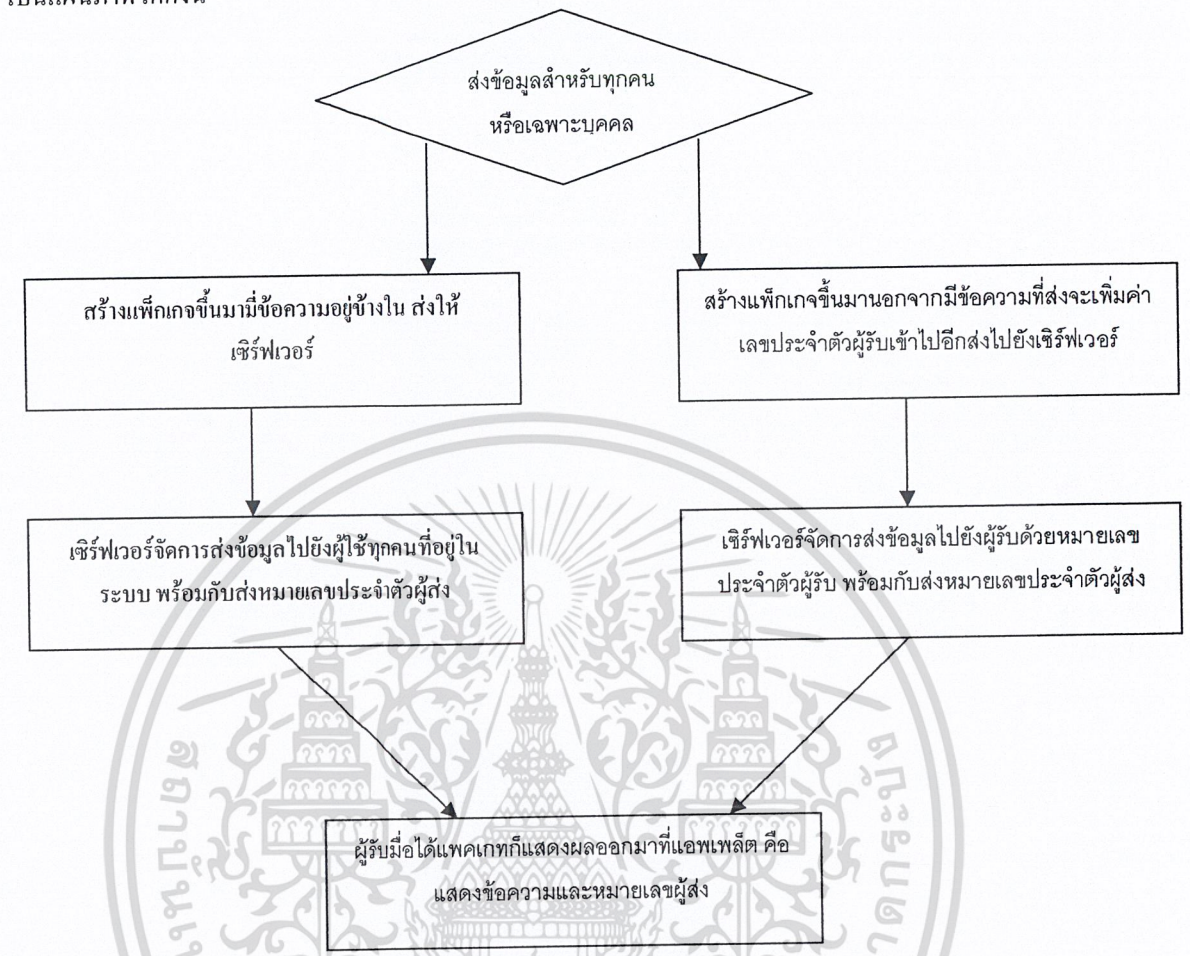
รูปที่ 3.1 แผนภาพการเข้าระบบของผู้เล่น

เมื่อทำการเข้าระบบได้แล้วตัวผู้ใช้สามารถทำการส่งข้อความ เข้าห้องเกม และออกจากระบบได้โดยสามารถแสดงเป็นแผนภาพได้ดังนี้

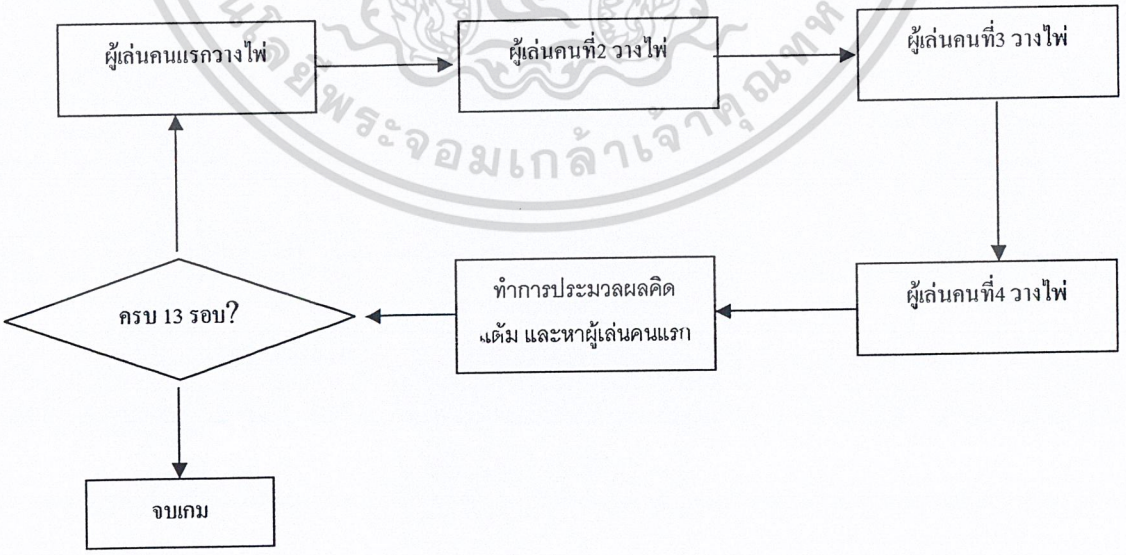


รูปที่ 3.2 แผนภาพการออกจากระบบ

การส่งข้อความการส่งข้อความแบ่งได้เป็น2แบบคือส่งให้ทั้งระบบ หรือส่งให้เฉพาะคน ซึ่งแสดงให้เป็นแผนภาพได้ดังนี้



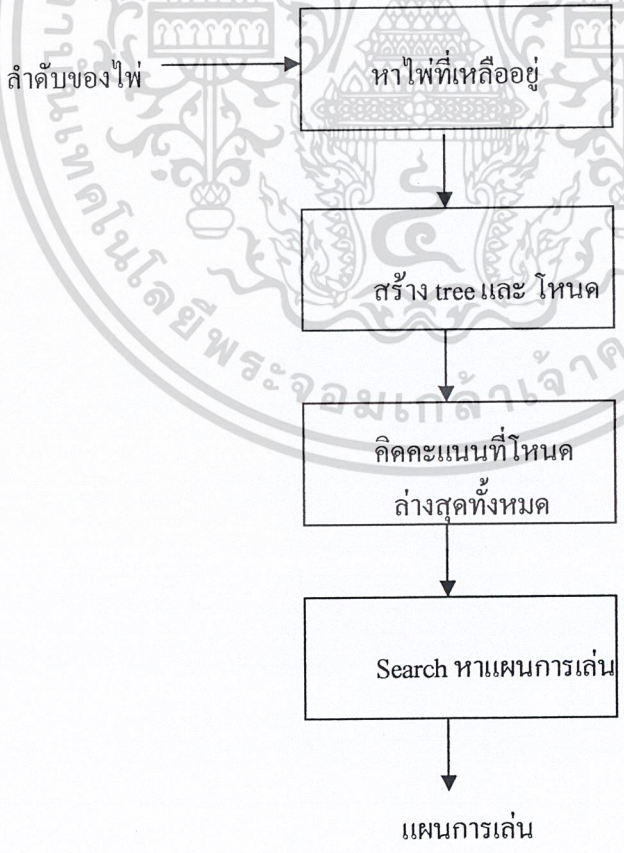
รูปที่ 3.3 แผนภาพการส่งข้อความ



รูปที่ 3.4 แผนภาพการเล่นเกม

3. 12 การออกแบบส่วนปัญญาประดิษฐ์

ในส่วนนี้โดยจะอาศัยหลักการของการทำ Mini-Max เนื่องจากเป็นวิธีที่มีโอกาสในการชนะสูงที่สุด แต่การเล่นจะมีวิธีการเล่นได้หลายรูปแบบมาก เช่น ในครั้งแรกถ้าเราเป็นเริ่มเล่นคนแรก (เป็นคนที่เริ่มลงไพ่ก่อน) กับการที่เราลงไพ่เป็นคนที่ 3 (หลังจากมีคนลงไพ่มา 2 ใบแล้ว) ก็จะมีวิธีการลงต่างกันมากอีก โดยอาจจะทำให้โหนดมีจำนวนมากเกินไป เช่น ถ้าเราเป็นคนเริ่มเล่นแล้วลงไพ่ใบแรกสำหรับตาแรก ไพ่ที่ คนที่ 2 สามารถลงได้ คือ ไพ่ 39 ใบที่เหลือ เช่นเดียวกันกับคนที่ 3 และ 4 ส่วนในตาที่ 2 ก็จะต้องมีไพ่อีก ให้คิดคือ 1 คน สามารถมีไพ่ที่มีอากาสจะลงเช่นกัน ซึ่งถ้าคิดเช่นนี้แล้วจะพบว่ามีจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้มากเกินไป ทำให้การสร้างและการ search แต่ละโหนดจะช้ามาก ผู้เล่นย่อมไม่ต้องการรอนานมากเช่นนั้น ดังนั้นจึงแก้ปัญหาด้วยโดย ใน 6 ตาแรกของการเล่นจะเป็นการใช้ กฎ(rule) เข้ามาช่วยเพื่อที่จะลดจำนวนโหนดที่เกิดขึ้นให้มีน้อยลงเพื่อให้การสร้างและการ search โหนดมีน้อยลงด้วย โดยเมื่อสร้างโหนดโดยโหนดที่สร้างจะคิดเฉพาะจากไพ่ที่เหลืออยู่เท่านั้น โดยถ้าทำการ search เฉพาะโหนดจำนวนเท่านี้ก็จะเหลือโหนดจำนวนไม่มากเท่านั้น เมื่อลงไปยังโหนดล่างสุดแล้วก็จะทำการเจนเนอเรทคะแนนจากไพ่ที่เหลือเพื่อนำมาเทียบกับโหนดอื่นๆ ในการทำแผนการเล่นโดยการ search จากโหนดที่ดีที่สุดเพื่อที่จะเล่นตามแผนการนั้นๆ

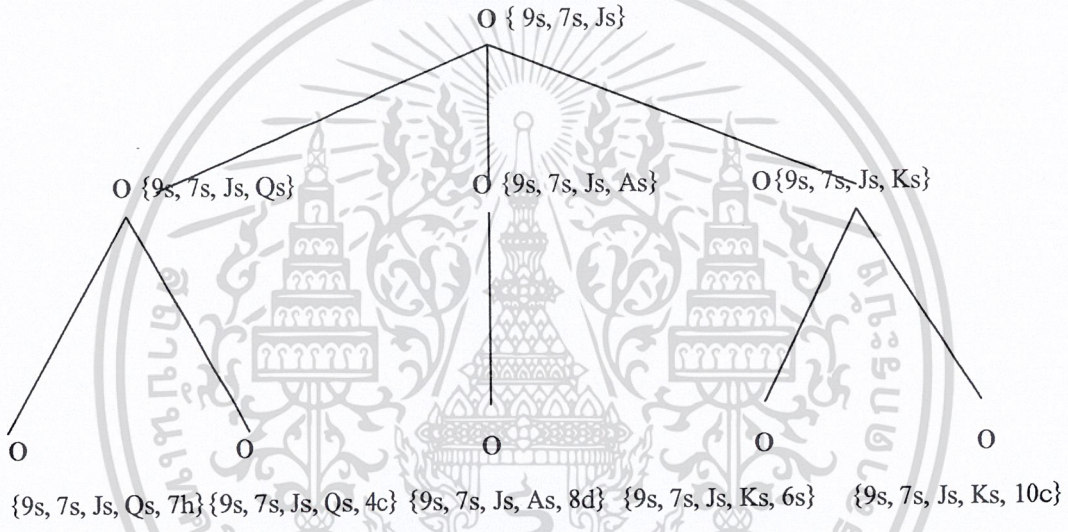


รูปที่ 3.5 แผนภาพแสดงการทำงานส่วนปัญญาประดิษฐ์

3.12.1 การสร้าง Tree

สำหรับการสร้าง Tree นั้นจะเริ่มสร้างเมื่อพบว่ามีไพ่มากกว่า 24 ใบแล้วเมื่อถึงเทินที่ตนเองต้องลงไพ่ก็จะทำการเริ่มคิดโดยเราจะนำเฉพาะไพ่ที่เหลืออยู่จากการเล่นมาคิดโดยในแต่ละโหนดจะประกอบด้วย ชื่อผู้ที่เริ่มลงไพ่คนแรกของเกมและ sequence ของไพ่ที่ลงในแต่ละตา เช่น {John, Ad, 9d, 7d, 2d, Js, 7s, 8s, As}

เมื่อได้โหนดแรกแล้วก็จะทำการสร้างความเป็นได้ที่จะมีเหตุการณ์ต่างๆเกิดต่อไปโดยที่ไพ่ที่จะสามารถเกิดขึ้นได้ก็จะลดลงไปเรื่อยๆตามแต่ละชั้น ทำให้ชั้นล่างสุดมีจำนวนเหตุการณ์น้อยที่สุด



รูปที่ 3.6 ตัวอย่าง Tree

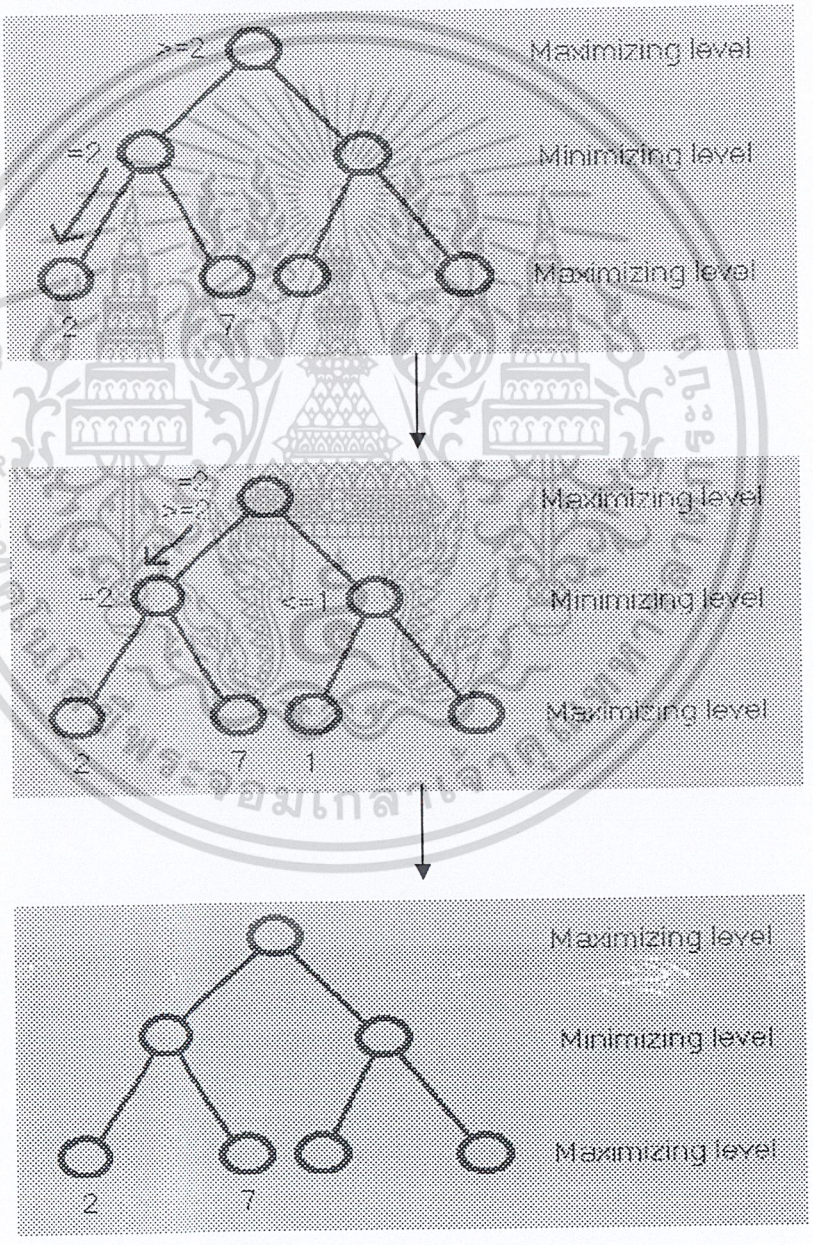
3.12.2 การ Search

การ search จะทำการ search แบบ Minimax search เพราะเหมาะกับการใช้กับเกมในลักษณะเช่นนี้เนื่องจากเรารู้ว่าคะแนนของแต่ละโหนดเป็นค่า min หรือ ค่า max ของโหนดข้างล่างได้

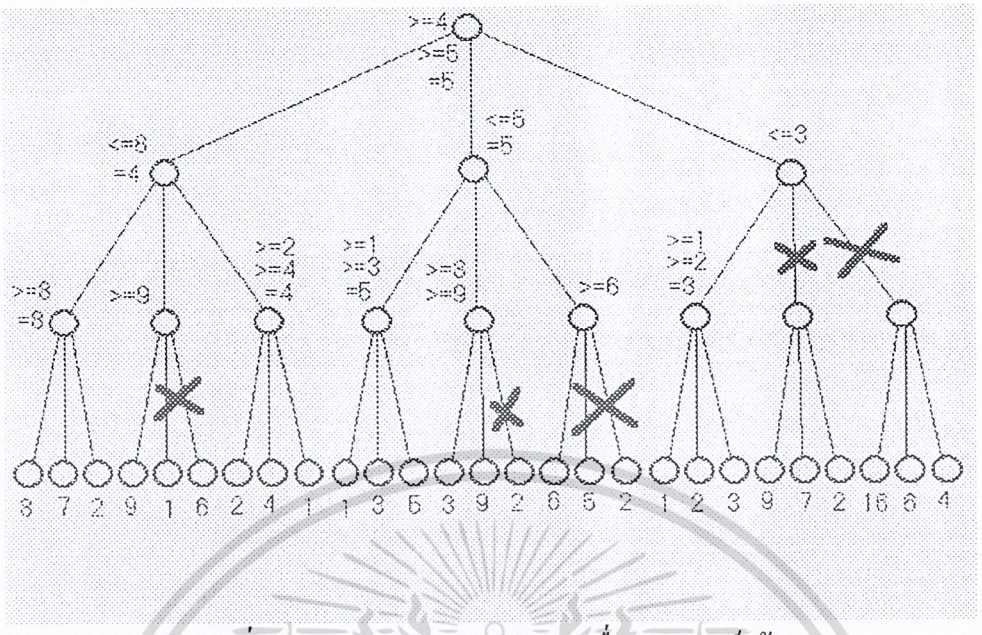
วิธีการ search คือ

- ถ้าพบว่าถึงโหนดล่างสุดแล้วให้คำนวณค่าของโหนดนั้นๆ ให้สอดคล้องกับเกม
- ถ้าต้องได้ค่าต่ำสุดให้หาค่าต่ำที่สุดของระดับชั้นนั้น
- ถ้าต้องได้ค่าสูงสุดให้หาค่าสูงที่สุดของระดับชั้นนั้น

ตัวอย่าง



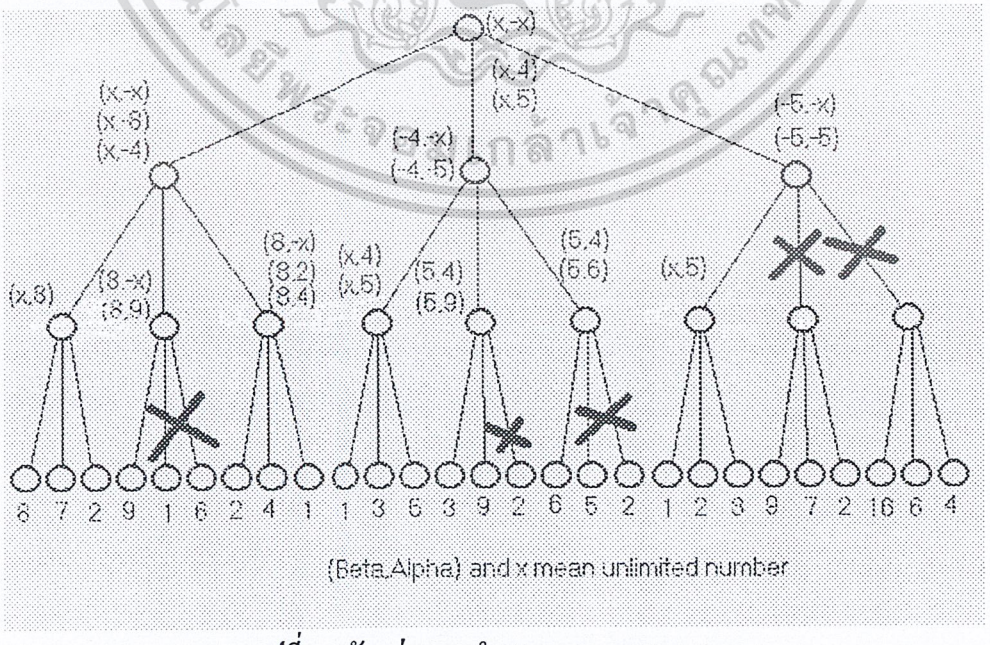
รูปที่ 3.7 การ search แบบ Minimax



รูปที่ 3.8 การ Search แบบ Minimax เมื่อทำสมบูรณ์แล้ว

3.13.3 Alpha Beta Plunning

การทำ Plunning ก็เพื่อลดเวลาในการค้นหา เนื่องจากว่าเรามีจำนวนโหนดมาก ดังนั้นเราจะจึงเสียเวลาในการ Search มาก ทั้งๆ ที่เรารู้อยู่ว่าต้องการได้ค่าที่มากที่สุดดังนั้น เราจึงสามารถจะ plun โหนดที่มีค่าน้อยเมื่อเทียบกับ กิ่งติดกันออกได้ เมื่อทำเช่นนี้แล้วจะพบว่าเราจะสามารถทำการค้นหาได้รวดเร็วยิ่งขึ้น



รูปที่ 3.9 ตัวอย่างการทำ Alpha Beta Plunning

3.12.4 การ คำนวณคะแนนจาก Sequence ของโหนด

การคำนวนคะแนนจะพิจารณาจากแต้มที่ตนเองได้โดยถ้าแต้มที่ติดลบเยอะสุดก็จะทำให้โหนดนั้นได้คะแนนต่ำ ซึ่งถ้าได้คะแนนสูงมากก็หมายถึงการที่เกมนั้นเราได้แต้มลบน้อยหรืออาจจะเป็นแต้มบวกซึ่งได้ผลดีสำหรับการเล่นเกมนี้ เช่น

โหนดสุดท้ายได้ ลำดับ(sequence) ของไพ่เป็น {Note, 4h, 6h, 9h, Qh, 10c, 9c, Jc, JA, …} ลำดับของไพ่ที่ได้มาจะไม่ได้เรียงว่าไพ่ชุดนั้นใครเก็บไพ่กองใดไปบ้างเราจึงต้องทำการแยกก่อนว่าผู้เราได้เก็บไพ่กองใดไปบ้างโดยสามารถหาได้จากการดูว่าผู้เล่นคนใดเป็นคนเริ่มเล่นคนแรกและเรียงลำดับการเรียงว่าวนไปทางใดและใครเป็นผู้ที่ลงไพ่ใหญ่ที่สุดในเทร์นนั้น ก็จะสามารถหาได้ว่าใครได้ไพ่กองใดไป เช่น

การเรียงลำดับผู้เล่นเป็น A → B → C → D

ลำดับไพ่ที่ได้เป็น {A, 6h, 9h, Qh, Jh, 10c, 9c, Jc, Ac 10d, 8d, Qd, Jd, 8h, 2h, 10h, 5d, …}

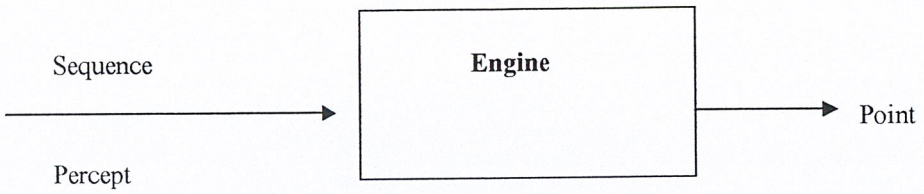
จากลำดับไพ่นี้เราสามารถรู้ได้ว่าการเล่นไพ่เป็นในรูปแบบนี้ คือ

A	*B*	*C*	*D*	
6h	9h	Qh	Jh	→ Aเป็นคนเริ่มก่อน(ดูจากข้อมูลของโหนด)และผู้ที่เก็บไป คือ C (เนื่องจาก C ลง Q หัวใจซึ่งใหญ่ที่สุดในรอบนี้)
Jc	Ac	10c	9c	→ C เป็นผู้เริ่มก่อนดูจากเหตุการก่อนหน้านี้และผู้ที่เก็บไปคือ B (เนื่องจาก B ลง A ดอกจิกซึ่งใหญ่ที่สุดในรอบนี้)
Jd	10d	8d	Qd	→ B เป็นผู้เริ่มก่อนดูจากเหตุการก่อนหน้านี้และผู้ที่เก็บไปคือ D (เนื่องจาก D ลง Q ข้าวหลามตัดซึ่งใหญ่ที่สุดในรอบนี้)
8h	2h	10h	5d	→ D เป็นผู้เริ่มก่อนดูจากเหตุการก่อนหน้านี้และผู้ที่เก็บไปคือ C (เนื่องจาก C ลง 10 หัวใจตัดซึ่งใหญ่ที่สุดในรอบนี้)

เมื่อเราได้ลำดับไพ่เช่นนี้แล้วเราก็จะสามารถรู้ได้ว่าเราเก็บไพ่ใดมาบ้าง (ในการทำงานในส่วนนี้เราจะสนใจเฉพาะไพ่ที่เราเก็บได้เพื่อหาคะแนนที่เราลบน้อยสุด) แล้วนำไพ่ที่เราเก็บได้มาคิดคะแนน

เช่น ไพ่ที่เก็บได้มี 8h, 9d, 7s, 5c, 10c, 9s, Qh, Kc ไพ่ที่มีแต้มได้แก่ 8h ,10c และ Qh แต้มที่ได้ คือ -76 แต้ม (มาจาก (8+30) *2))

ในขณะที่อีกโหนดเป็น Jd, 9d, Kd, 7d, Ah, 4h, 7h, 3h แต้มที่ได้ คือ 100 – 50 – 4 – 7 – 3 = +36 แต้ม



รูปที่ 3.10 การทำงานของการ generate point

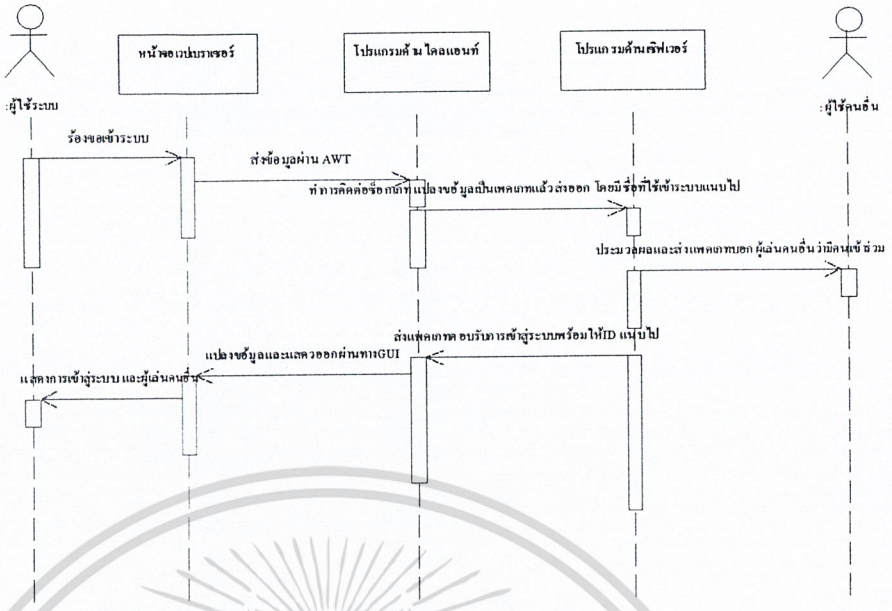
13.13 โครงสร้างการทำงานของโปรแกรม

โครงสร้างขอระบบเป็นแบบ ไคลเอ็นต์-เซิร์ฟเวอร์ โดยจะติดต่อกับผู้ใช้ด้วยการส่งข้อความระหว่างกัน เพื่อบอกถึงสถานะของผู้ใช้ การเรียกใช้บริการ หรือ รับส่งข้อมูล

รูปแบบในการออกแบบตอนแรก ผู้ใช้แต่ละคนมีสิทธิที่จะเข้าระบบ ออกจากระบบ ทำการส่งข้อความระหว่างผู้เล่นด้วยกัน เลือกเข้าสู่ห้องเกม ดำเนินการเล่นเกมและสุดท้ายออกจากระบบ แสดงได้ดังรูปข้างล่างนี้

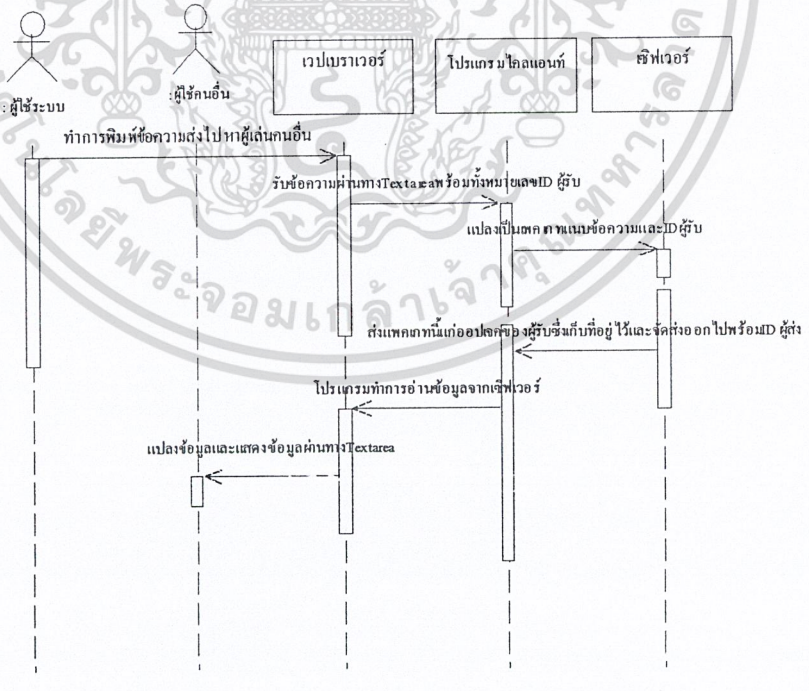
รูปที่ 3.12 ฟังก์ชันการทำงานของที่ผู้ใช้สามารถขอใช้บริการได้

จากรูปด้านบน ได้แสดงความสารถในการทำงานของผู้ใช้ในระบบ ซึ่งแสดงรายละเอียดในแต่ล่ะการทำงานด้วย Seqence Diagram ข้างล่าง



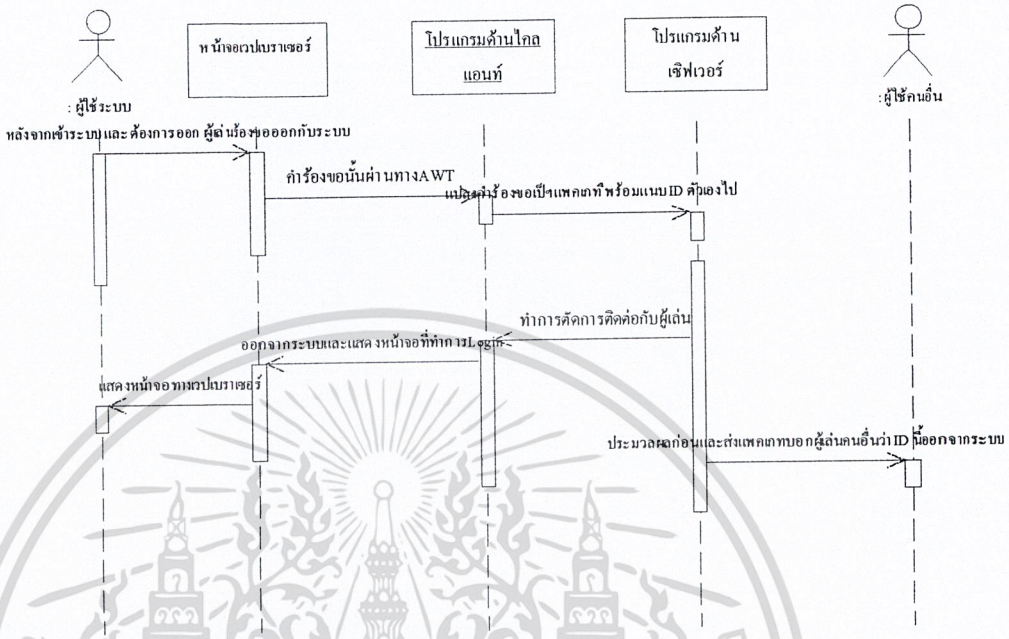
รูปที่ 3.13 แผนภาพแสดงการทำงานของการเข้าสู่ระบบ

ในการเข้าสู่ระบบนั้นทำการโดยการติดต่อโดยซ็อกเก็ต โปรแกรมจากคลาสไคลเอ็นต์จะทำการสร้างซ็อกเก็ต ติดต่อกับทางเซิร์ฟเวอร์เมื่อเซิร์ฟเวอร์ทำการติดต่อ ได้แล้วจะสร้างออบเจ็กต์มารองรับการติดต่อของผู้ใช้ พร้อมทั้งให้ค่าID กับไคลเอ็นต์เพื่อใช้ในการติดต่อ



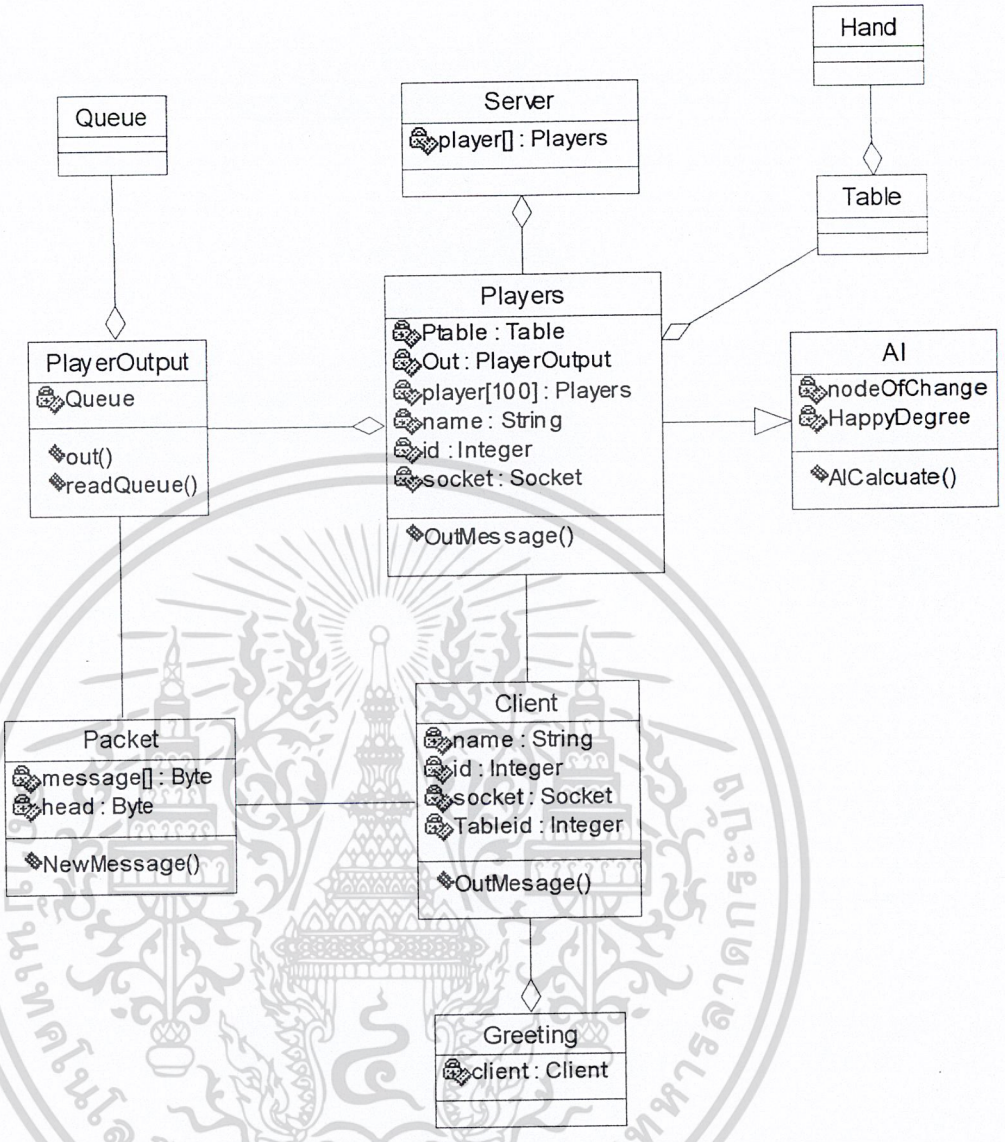
รูปที่ 3.14 แผนภาพแสดงการทำงานการส่งข้อความให้กับผู้ใช้คนอื่น

การส่งข้อมูลระหว่างผู้ใช้ทำได้โดยข้อความที่ส่งจะถูกแปลงเป็นแพคเกท ด้วยไคลเอ็นต์คลาส และส่งต่อให้กับเซิร์ฟเวอร์ เซิร์ฟเวอร์จะเรียกออบเจ็กของผู้เล่นนั้นเป็นตัวส่งข้อความให้กับผู้รับ และแสดงผลออกหน้าจอโดยGUI



รูปที่ 3.15 แผนภาพแสดงการออกจากระบบ

การออกจากระบบจะทำได้หลังจากที่เข้าสู่ระบบแล้วเมื่อเซิร์ฟเวอร์ได้รับการร้องขอออกจากระบบจะทำการทำลายซ็อกเก็ตที่ติดต่อกัน ลบข้อมูลผู้ใช้ออกจากออบเจ็กต์ที่ใช้ในการติดต่อเพื่อใช้กับคนอื่นต่อ พร้อมทั้งบอกผู้เล่นคนอื่นว่าผู้เล่นนี้ออกไปจากระบบแล้ว



รูปที่ 3.16 คลาสทั้งหมดที่ใช้ในโปรแกรมนี้

ภาพด้านบนแสดงรูปแบบของคลาสในระบบโดยแบ่งเป็น2ฝ่ายคือ

1.ฝั่งเซิร์ฟเวอร์

2.ฝั่งไคลเอ็นต์

1.ฝั่งเซิร์ฟเวอร์คลาสหลักนั้นคือ เซิร์ฟเวอร์ทำหน้าที่รอรับการร้องขอของไคลเอ็นต์และทำการสร้างออบเจ็กต์ Player เก็บรายละเอียดของผู้ใช้ที่เข้ามาและใช้ออบเจ็กต์นี้ในการติดต่อกับผู้ใช้ หน้าที่ของคลาสPlayer คือ การอ่านข้อมูลที่ไคลเอ็นต์ประมวลผลและส่งข้อมูลกลับไปคลาสนี้ ประกอบไปด้วยคลาสต่างๆ ซึ่งมีหน้าที่ดังนี้

PlayerOutput : ทำหน้าที่ในการส่งข้อมูลออกไปสู่ไคลแอนท์โดยทำการเก็บข้อมูลเข้าไปในคลาส Queue โดยคลาส Queue ทำหน้าที่เก็บข้อความโดยเป็นลักษณะ FIFO

Table:เป็นห้องเกมที่เก็บรายชื่อผู้เล่นไว้และทำการดำเนินเกม

ปัญญาประดิษฐ์นั้นเป็นเหมือนกับผู้เล่น 1 คนที่ทางเซิร์ฟเวอร์สร้างขึ้นมาเพื่อใช้สำหรับการเล่นเกมกับผู้เล่น โดยมีสิทธิเท่ากับผู้ใช้1คน

2.ฝั่งไคลเอ็นต์ประกอบด้วยคลาสหลักคือ Greeting ซึ่งเก็บไว้ในเว็บเพจเมื่อมีการเข้าระบบฝ่ายทางเว็บนี้ Greeting จะไปสร้าง ไคลเอ็นต์ ขึ้นมาเพื่อทำการติดต่อกับเซิฟเวอร์โดยเมื่อติดต่อกับเซิร์ฟเวอร์ได้ ทางเซิร์ฟเวอร์จะเอาออบเจ็กต์ Player มารอรับและเก็บข้อมูลผู้เล่นนั้นไว้

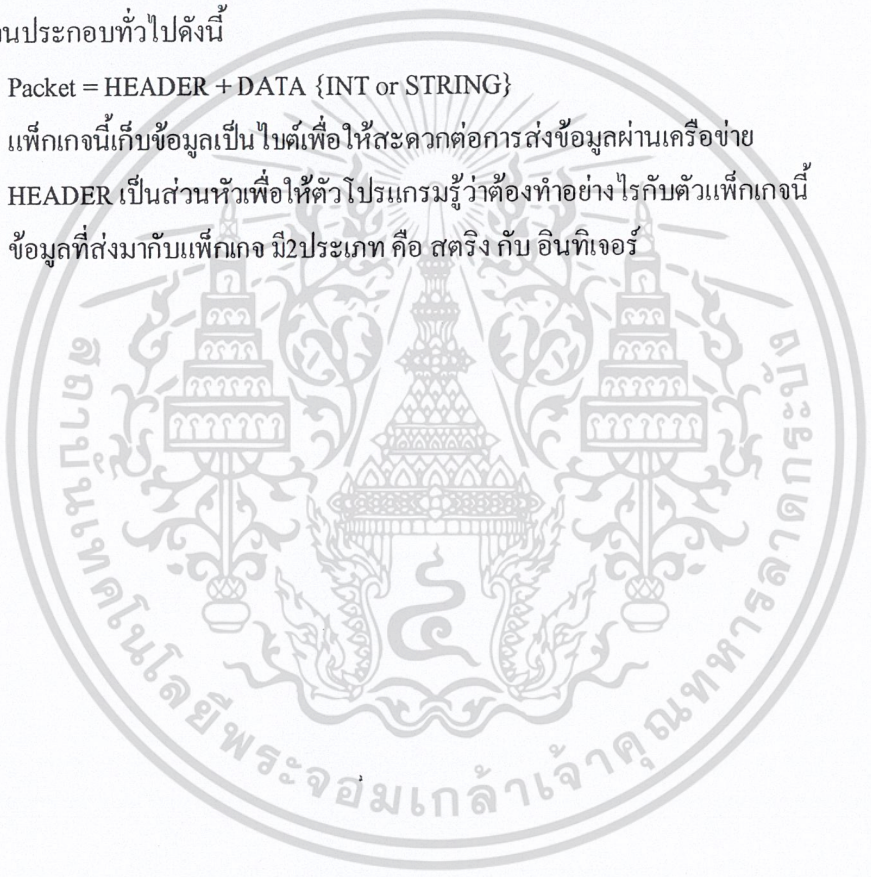
นอกจากนี้ยังมีคลาสPacket ในการสร้างแพ็กเกจเพื่อส่งข้อมูลระหว่างไคลแอนท์-เซิฟเวอร์ โดยมีส่วนประกอบทั่วไปดังนี้

Packet = HEADER + DATA {INT or STRING}

แพ็กเกจนี้เก็บข้อมูลเป็นไบต์เพื่อให้สะดวกต่อการส่งข้อมูลผ่านเครือข่าย

HEADER เป็นส่วนหัวเพื่อให้ตัวโปรแกรมรู้ว่าต้องทำอย่างไรกับตัวแพ็กเกจนี้

ข้อมูลที่ส่งมากับแพ็กเกจ มี2ประเภท คือ สตริง กับ อินทิเจอร์



บทที่ 4
บทสรุป และวิจารณ์

โครงงานนี้จัดทำขึ้นเพื่อศึกษาการสร้างเกมที่สามารถเล่นร่วมกันหลายคนได้โดยผ่านเครือข่ายอินเตอร์เน็ต โดยมีการเข้ามาร่วมแข่งขันของคอมพิวเตอร์ในส่วนปัญญาประดิษฐ์ ซึ่งวิธีการของส่วนปัญญาประดิษฐ์ คือ การใช้ Minimax Game และการทำ Prunnig โดยโครงงานนี้ได้พัฒนาขึ้นโดยใช้ภาษาจาวา โดยใช้หลักการติดต่อกันแบบ ไคลเอนต์—เซิร์ฟเวอร์ โดยผู้เล่นเพียงแต่ทำการเข้ามายังเวบเพจของเกมก็สามารถจะเล่นเกมนี้ได้

4.1 ปัญหาที่พบและแนวทางแก้ไข

1. การคำนวณการสร้าง Tree ค่อนข้างนานเกินไป คือ ถ้าทำการสร้างทรีทั้งหมดตั้งแต่เริ่มเกมในตาแรก จะใช้เวลาในการสร้างทรีประมาณ 10 นาทีซึ่งนานเกินไป ผู้เล่นไม่อาจจะรอได้ดังนั้นจึงทำการแก้ปัญหาด้วยการเริ่มสร้างทรีเมื่อมีการเล่นผ่านไปล้ว 8 ตาดังนั้นจำนวนโหนดของทรีก็จะลดได้จำนวนมาก จึงลดเวลาในการสร้างทรีได้มาก ซึ่งเหลือเพียงไม่ถึง 15 วินาทีซึ่งเป็นเวลาที่สามารถยอมรับได้ โดยหาได้จากผลการทดลองดังนี้

จำนวนตาที่สร้างทรี	จำนวนหน่วยความจำ (เฉลี่ย)	จำนวนโหนดของทรี (เฉลี่ย)	จำนวนเวลาที่ใช้ (เฉลี่ย)
1ตา	2680 ไบต์	5โหนด	น้อยกว่า1/1000วินาที
2ตา	15,000 ไบต์	30โหนด	10 ms
3ตา	120,000 ไบต์	200โหนด	30 ms
4ตา	15,000,000 ไบต์	22,500 โหนด	1,500 ms
5ตา	140,000,000 ไบต์	227,500 โหนด	16,000 ms

โดยค่าต่างๆ จะเปลี่ยนแปลงตามลักษณะของไพ่ที่ได้รับเข้ามา ยิ่งหน้าไพ่ของแต่ล่ะผู้เล่นเหมือนกันจะทำให้ การสร้างทรีลดจำนวนโหนดลงไปอีก และขึ้นอยู่กับความสามารถของเครื่องที่ทดสอบด้วย จากตารางด้านบน ทำการทดสอบด้วยเครื่องคอมพิวเตอร์ Celeron 666 แรม 128 เมื่อขึ้นไปถึงลำดับ 6 ตา การทดสอบสามารถทำสำเร็จได้เป็นบางครั้งเนื่องจากหน่วยความจำไม่เพียงพอ (ทำการ จองที่หน่วยความจำไว้ที่ 384 เมกะไบต์) และใช้เวลามากเกินไปคือ มากกว่า 3 นาที จึงทำการสร้างทรีเพียงแค่ 5 ตาเท่านั้น

2. ในตาแรกๆ คอมพิวเตอร์ไม่ฉลาดเท่าที่ควรเนื่องจาก rule ใช้เล่นครอบคลุมได้ไม่ครบทุกกรณี เพราะมีเหตุการณ์จำเพาะในกรณีพิเศษ ในบางกรณีที่คาดไม่ถึง

3. จำนวนโหนดที่สามารถสร้างได้มีจำกัด คือ ประมาณ 140,000 โหนด ซึ่งการทรีของเกมประเภทไพ่ต้อง การจำนวนโหนดมากกว่านี้ดังนั้น การสร้างทรีจึงทำได้ไม่มากระดับเท่าที่ควร
4. การเก็บข้อมูลไว้ในส่วนความจำชั่วคราวของเวบราวเซอร์ ทำให้เมื่อมีการเปลี่ยนตัวโปรแกรมที่เซิร์ฟเวอร์ แต่ทว่า ไคลเอ็นต์ยังเรียกตัวโปรแกรมที่เก็บไว้ในหน่วยความจำของเวปบราวเซอร์ ทำให้เกิดการผิดพลาดในการประมวลผลขึ้นมาได้
5. ข้อจำกัดในการแสดงผล เนื่องจากภาษาจาวาในต้องมีการ นำคลาสพื้นฐานที่ให้มาทำการสร้างคลาสใหม่ แต่ทว่าเวบราวเซอร์มีข้อจำกัดของคลาสพื้นฐาน คือมีคลาสไม่ครบทุกคลาส หรือมีรุ่นที่ต่ำกว่าคลาสพื้นฐานจาวาที่ออกมา ทำให้การแสดงผลนั้นต้องขึ้นอยู่กับคลาสพื้นฐานของเวบบราวเซอร์อีกด้วย
6. การส่งข้อมูลที่ผิดพลาด เนื่องจากสาเหตุใดๆก็ตาม เมื่อข้อมูลมีการผิดพลาดเกิดขึ้นทำให้ตังวโปรแกรมมีการทำงานที่ผิดพลาดขึ้นมา เพราะข้อมูลที่ส่งไปมากันนั้นบางครั้ง เป็นข้อมูลสำคัญที่ใช้ประมวลผลและส่งผลต่อสถานะของระบบอีกด้วย
7. จากปัญหาข้อ 4 5 และ 6 นั้นเราสามารถแก้ไขได้โดยการเขียนโปรแกรมดักจับข้อมูลที่ผิดพลาดขึ้นมาได้ แต่ทว่าบางครั้งก็ไม่สามารถเขียนโปรแกรมดักข้อผิดพลาดได้ทั้งหมด

4.2 แนวทางในการพัฒนาเพิ่มเติม

4.2.1 ส่วนของเน็ตเวิร์ก

1. ทำการเพิ่มความปลอดภัยในข้อมูลที่ส่งติดต่อกันระหว่างไคลเอ็นต์ และเซิร์ฟเวอร์โดยเราสามารถทำการเข้ารหัสข้อมูล เพื่อกันคนดักจับข้อมูล หรือเขียนข้อมูล ขึ้นมาและส่งเข้าเซิร์ฟเวอร์โดยไม่ผ่านโปรแกรม
2. ตรวจเช็คข้อมูลว่าถูกต้องตามที่ต้องการหรือไม่ โดยอาจทำการส่งพร้อมมีข้อมูลตรวจสอบแนบไปด้วยอีกชุดหนึ่งโดยที่ข้อมูลตรวจสอบนี้จะสามารถหาได้โดยการนำข้อมูลที่จะส่ง หรือรับไปผ่านกระบวนการ เพื่อใช้ตรวจสอบข้อมูลว่าถูกต้องหรือไม่
3. ในส่วนของอินเทอร์เฟซนั้นสามรถปรับปรุงให้สวยงาม และ มีฟังก์ชันการทำงานที่มีลูกเล่นมากกว่านี้ได้เช่น การส่งข้อความเชิญผู้เล่นเข้าสู่ห้องเกม , การสร้างกฎพิเศษ หรืออื่นๆอีก

4.2.2 ส่วนของปัญญาประดิษฐ์

1. ควรมีการปรับปรุงส่วนของกฎการเล่นในช่วงตาแรกๆ โดยให้กฎครอบคลุมวิธีการเล่นเพิ่มขึ้น โดยการเพิ่มกฎขึ้นและทำให้กฎละเอียดและมีประสิทธิภาพขึ้น เพื่อให้การเลนในช่วงแรกเก่งยิ่งขึ้น

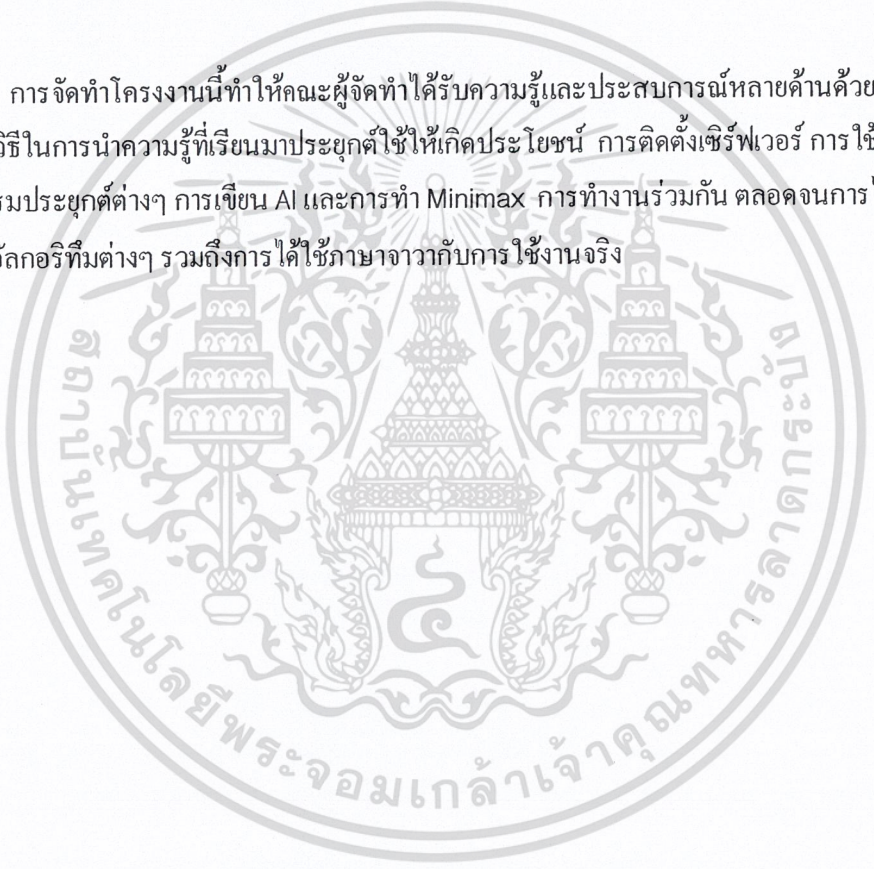
2. ปรับปรุงอัลกอริทึมในการสร้าง tree และ การ search tree ให้มีประสิทธิภาพเพิ่มขึ้นคือ กำจัดการสร้างโหนดที่ไม่มีโอกาสเกิดซึ่งเป็นการสร้างที่เสียเวลาเปล่าเป็นเวลานาน

4.3 สรุปผลที่ได้จากการทำโครงงาน

โครงงานที่จัดทำขึ้นนี้จัดได้ว่าประสบความสำเร็จในการดำเนินการ และตรงตามวัตถุประสงค์ที่ตั้งไว้ คือ สามารถทำโครงงานไปใช้งานได้จริงและผู้ที่ได้เล่นเกมได้รับความเพลิดเพลินด้วย

4.4 ประโยชน์ที่ได้รับจากการทำโครงงาน

การจัดทำโครงงานนี้ทำให้คณะผู้จัดทำได้รับความรู้และประสบการณ์หลายด้านด้วยกัน ไม่ว่าจะเป็น วิธีในการนำความรู้ที่เรียนมาประยุกต์ใช้ให้เกิดประโยชน์ การติดตั้งเซิร์ฟเวอร์ การใช้โปรแกรมประยุกต์ต่างๆ การเขียน AI และการทำ Minimax การทำงานร่วมกัน ตลอดจนการได้ศึกษาการคิดอัลกอริทึมต่างๆ รวมถึงการได้ใช้ภาษาจาวากับการใช้งานจริง



บรรณานุกรม

[1] อ.บัณฑิต จามรภูติ : “การประยุกต์ใช้ระบบ ไคลเอนต์ เซิร์ฟเวอร์”

[2] ดร.วีระศักดิ์ ซึงถาวร : “Java Programming Volume 1”

[3] ดร.วีระศักดิ์ ซึงถาวร : “Java Programming Volume 2”

[4] อ.กิตติ ภักดีวัฒนะกุล: “Java ฉบับโปรแกรมเมอร์”

[4] Merlin Hughes, Michael Shoffner, Derek Hamner : “ java Network Programming”

[5] http://www.java.sun.com

[6] http://www.jcreator.com