การพัฒนาโปรแกรมหุ่นยนต์รบที่ควบคุมด้วยภาษาจาวา

A DEVELOP MUST OF ROBOT CONTROL GAME USING JAVA

ชนณิกานต์ ธรรมบริสุทธิ์
พรชัย แก้วสถิตพรชัย
พุธพงษ์ พงษ์ธรรมรักษ์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

A DEVELOP MUST OF ROBOT CONTROL GAME USING JAVA

CHONNIKARN THUMBORISUTH

PORNCHAI KAEWSATIDPORNCHAI

PUTTAPONG PONGTHUMMARAK

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LARDKRABANG

ACADAMIC YEAR 2002

หัวข้อปัญหาพิเศษ การพัฒนาโปรแกรมเกมหุ่นยนต์รบที่ควบคุมโดยภาษาจาวา

A DEVELOP MUST OF ROBOT CONTROL GAME USING JAVA

ชื่อนักศึกษา นางสาวชนณิกานต์ ธรรมบริสุทธิ์ 42050377

นายพรชัย แก้วสลิตพรชัย 42050412

นายพุธพงษ์ พงษ์ธรรมรักษ์ 42050418

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษา วิสันต์ ตั้งวงษ์เจริญ

ชาญชัย ดีอ่วม

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ปีการศึกษา 2545

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ไพรบูลย์ พันธรักษ์พงษ์	
กรรมการ	อ.ศังกรณ์ศรัณย์ ลองชูผล	
กรรมการและอาจารย์ที่ปรึกษา	อ.วิสันต์ ตั้งวงษ์เจริญ	
กรรมการและอาจารย์ที่ปรึกษา	อ.ชาญชัย ดีอ่วม	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)

หัวหน้าภาควิชาวิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ การพัฒนาโปรแกรมเกมหุ่นยนต์รบที่ควบคุมโดยภาษาจาวา
ชื่อนักศึกษา นางสาวชนณิกานต์ ธรรมบริสุทธิ์ 42050377
นายพรชัย แก้วสถิตพรชัย 42050412
นายพุธพงษ์ พงษ์ธรรมรักษ์ 42050418
ปริญญา วิทยาศาสตรบัณฑิต
ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2545
อาจารย์ที่ปรึกษา วิสันต์ ตั้งวงษ์เจริญ
ชาญชัย ดีอ่วม

บทคัดย่อ

ในปัจจุบันเทคโนยีการโปรแกรมเชิงวัตถุได้รับความนิยมแพร่หลายเป็นอย่างสูง เนื่องจากช่วยในการพัฒนาโปรแกรม โดยที่การพัฒนาต่อจากเดิมสามารถทำได้โดยง่าย แต่ขั้นตอนในการศึกษาการเขียนโปรแกรมเชิงวัตถุเข้าใจยากและต้องใช้เวลาในการศึกษานาน

ดังนั้นจึงได้ทำการพัฒนาโปรแกรมเกมหุ่นยนต์รบที่ควบคุมด้วยภาษาจาวา โดยที่ภาษาจาวาเป็นภาษาที่มีแนวความคิดแบบการโปรแกรมเชิงวัตถุ เพื่อใช้เป็นเครื่องมือแก่ผู้ที่ทำการศึกษาการโปรแกรมเชิงวัตถุ ทำให้ผู้เล่นเกิดความสนุกสนานและเกิดแรงจูงใจในการการเขียนโปรแกรม โดยที่เกมหุ่นยนต์รบที่ควบคุมด้วยภาษาจาวาโดยมีส่วนประกอบหลักคือ ส่วนของการสร้างหุ่นยนต์ และส่วนการแข่งขันของหุ่นยนต์ โดยในการสร้างหุ่นยนต์จะต้องทำการเลือกส่วนประกอบของหุ่นยนต์และเขียนโปรแกรมที่ใช้บังคับ ส่วนการแข่งขันของหุ่นยนต์จะแบ่งการเล่นออกเป็น 2 แบบ คือ Story mode และ Fighting mode โดยที่การเล่นแบบ Story mode จะเป็นการแข่งขันกับหุ่นยนต์ที่ได้กำหนดไว้แล้ว ส่วน Fighting mode จะเป็นการแข่งขันกับหุ่นยนต์ตัวใดก็ได้ที่มีอยู่ที่เซิร์ฟเวอร์โดยทำการเลือกหุ่นยนต์ของตนเอง 1 ตัวและเลือกหุ่นยนต์ที่ต้องการแข่งขันด้วยอีก 1 ตัว โดยการแข่งขันของหุ่นยนต์จะมีสนามรบเป็นตัวควบคุมการทำงานของหุ่นยนต์ให้ดำเนินได้อย่างถูกต้อง

Special Project Title	A DEVELOP MUST OF ROBOT CONTROL GAME USING JAVA
Students	Miss.Chonnikarn Thumborisuth 42050377
	Mr.Pornchai Kaewsatidpornchai 42050412
	Mr.Puttapong Pongthummarak 42050418
Degree	Bachelor of Science
Department	Mathematics and Computer Science, Faculty of Science
Programme	Computer Science
Academic Year	2002
Special Project Advisor	Wisan Tungwongjarean
	Chanchai Dee-uam

ABSTRACT

Today, object oriented programming technology plays an important role in computer programming. Developer, student and many people who is interest in object oriented programming use more object oriented technics because of their benefits such as time to develop, module developing, cost effective. To understand the concept of object oriented programming is quite difficult and take much time.

So, this project have developed JavaRobot Creater which is controled by Java Language in order to be a intermediary of education. Which use the entertainment to enjoy the learners. It has two major components. One is a part of building the robot and the second is the robot fighting which is controled by the fighting field. The compatition is separated into two types; Story mode and Fighting mode

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่อง สามารถสำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำขอขอบพระคุณ อ.วิสันต์ ตั้งวงศ์เจริญ และอ.ชาญชัย ดีอ่วม อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำ และเป็นที่ปรึกษาในการแก้ปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้ และขอขอบพระคุณ อาจารย์ทุกๆท่านที่ให้คำแนะนำและให้ข้อคิดดีๆในการทำปัญหาพิเศษนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ การทำปัญหาพิเศษครั้งนี้สำเร็จด้วยดี รวมทั้งเพื่อนๆพี่ๆและน้องๆทุกคนที่ให้ความช่วยเหลือในด้านต่างๆเกี่ยวกับปัญหาพิเศษ ไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2546

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ปัจจุบันโปรแกรมภาษาเชิงวัตถุ (Object Oriented)ได้มีความแพร่หลายและมีความนิยมเป็นอย่างมาก เพราะโปรแกรมภาษาเชิงวัตถุเป็นภาษาที่สามารถนำกลับมาปรับปรุงเป็นรุ่น (Version) ใหม่ได้อย่างรวดเร็ว แต่โปรแกรมภาษาเชิงวัตถุเป็นภาษาที่ศึกษาทำความเข้าใจยาก เพราะมองเห็นภาพจริงของการทำงานของวัตถุ (Object) ได้ยาก ดังนั้นน่าจะมีเครื่องมือที่ทำให้ผู้เริ่มพัฒนาโปรแกรมภาษาเชิงวัตถุมองภาพได้ง่ายขึ้น โดยใช้เกมเป็นสื่อกลางในการเรียนรู้และทำความเข้าใจในภาษาเชิงวัตถุโดยเฉพาะภาษาจาวา (Java) ซึ่งเป็นภาษาที่ได้รับความนิยมสูงมากในขณะนี้ โดยเป็นการปรับปรุงพัฒนาโปรแกรมเกม การเล่น และความสนุกให้มากขึ้นจากรุ่นที่แล้ว

1.2 วัตถุประสงค์ของการทำ

1) เพื่อปรับปรุงเกม Online JavaRobot จากเดิมให้มีความน่าสนใจและมีประสิทธิภาพมากยิ่งขึ้น

2) เพื่อให้ผู้ที่สนใจเทคโนโลยีเชิงวัตถุโดยเฉพาะภาษาจาวา มีแนวทางการศึกษาให้ง่ายยิ่งขึ้น และยังลดเวลาการเรียนรู้ภาษาเชิงวัตถุและสามารถนำไปประยุกต์ใช้กับงานที่เกี่ยวข้องได้ โดยใช้เกมเป็นเครื่องมือในการเรียนรู้

3) เพื่อให้ผู้เล่นได้เรียนรู้เทคนิคการทำงานของภาษาเชิงวัตถุจากเกมที่เล่น

4) เพื่อเป็นแรงจูงใจทำให้ผู้เล่นสนใจศึกษาเนื้อหาภาษาเชิงวัตถุจากความสนุกในการเล่น

1.3 ขอบเขตของปัญหา

1) ปรับปรุงเนื้อหาของเกมให้มากขึ้น โดยเพิ่มโหมดการแข่งขันให้สามารถเลือกเล่นได้มากขึ้นคือ โหมดการแข่งขันกับคอมพิวเตอร์ และโหมดการแข่งขันระหว่างผู้เล่น โดยในโหมดการแข่งขันกับคอมพิวเตอร์มีการแบ่งระดับตามความยากง่ายในการเล่น

2) ปรับปรุงส่วนประกอบของหุ่นยนต์ให้มีมากขึ้น เช่น อาวุธ และบ่า โดยอาวุธนั้นได้การเปลี่ยนแปลงประเภทของอาวุธเป็น อาวุธหนักและอาวุธเบา ซึ่งอาวุธหนักจะติดได้เฉพาะที่บ่า และอาวุธเบาจะติดได้เฉพาะที่แขน

3) ปรับปรุงวิธีการเล่นเพื่อให้ผู้เล่นมีความรู้สึกเป็นเจ้าของมากยิ่งขึ้น โดยผู้เล่นสามารถจัดการกับหุ่นยนต์ของตนเองโดยผ่านทางเวบแอปพลิเคชัน (Web Application)

4) สามารถเก็บสถานะการเล่นของหุ่นยนต์ของผู้เล่นผู้นั้น โดยที่ผู้เล่น 1 คน สามารถสร้างหุ่นยนต์ได้มากกว่า 1 ตัว (History Robot)

5) ปรับปรุงในส่วนการทำงานของเทรด (thread) และเหตุการณ์ (Event) เพื่อให้การทำงานของเกมเป็นไปได้อย่างถูกต้อง รวมถึงการปรับปรุงโปรแกรมให้มีประสิทธิภาพมากขึ้น

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้โปรแกรมเกม JavaRobot ที่มีประสิทธิภาพและมีความสนุกมากขึ้นกล่าวคือ ได้มีการปรับปรุงการทำงานของเทรด และเหตุการณ์ ให้การทำงานของเกมเป็นไปอย่างถูกต้อง มีโหมดการแข่งขันให้ผู้เล่นได้เลือกเล่นมากขึ้น มีการเพิ่มส่วนประกอบของหุ่นยนต์

2) ได้ความรู้ในและเข้าใจเทคโนโลยีเชิงวัตถุได้ง่ายขึ้น โดยใช้เกมเป็นเครื่องมือในการเรียนรู้

3) ผู้เล่นสามารถลดระยะเวลาในการเรียนรู้ภาษาเชิงวัตถุ

4) เพื่อให้ผู้เล่นความสนุกสนาน ผ่อนคลาย และให้ความบันเทิง

1.5 ขั้นตอนในการดำเนินงาน

1) ศึกษาและทำความเข้าใจในโปรแกรมภาษาเชิงวัตถุ ภาษาจาวา และเว็บโปรแกรมมิ่ง (Web Programming)

2) เก็บรวบรวมข้อมูลและตัวอย่างที่เกี่ยวข้องกับเกม

3) วิเคราะห์และออกแบบโครงสร้างหลักของเกม และออกแบบในรายละเอียดของตัวเกมเพิ่มขึ้น

4) ออกแบบอินเตอร์เฟส (Interface) ในส่วนติดต่อกับผู้ใช้ทั้งส่วนแอปพลิเคชัน (Application) และเว็บแอปพลิเคชัน

5) เขียนโค้ดโปรแกรม (Code Program) ในส่วนต่างๆที่ได้ออกแบบไว้

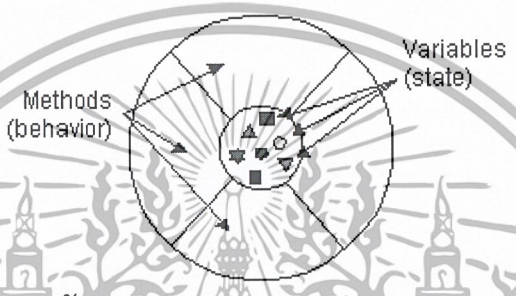
6) ทดสอบตัวเกมเพื่อดูว่าการทำงานของเกมส์ว่าเป็นไปตามจุดประสงค์ที่วางไว้หรือไม่

7) แก้ไขข้อบกพร่องที่เกิดขึ้นและปรับปรุงเกมให้มีความสมบูรณ์

บทที่ 2
ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

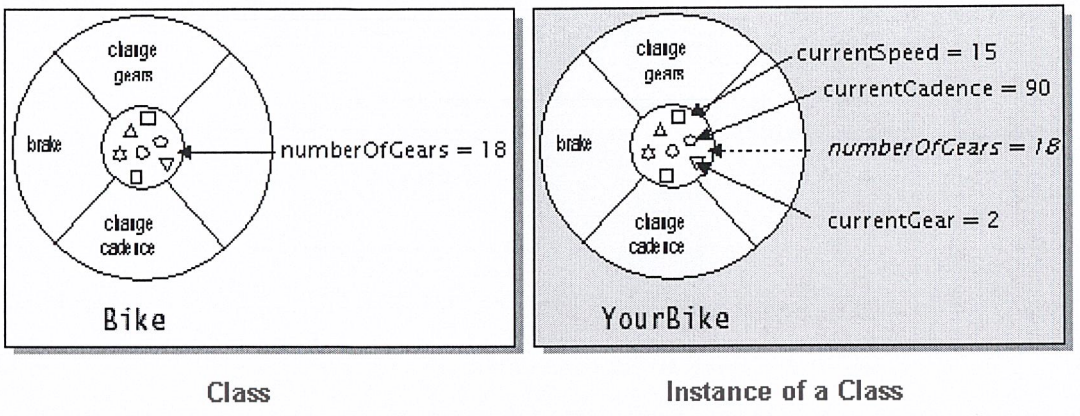
2.1 ทฤษฎีของการโปรแกรมเชิงวัตถุ (Object Oriented Methodology)

คำว่า "Object" (Real World Object) โดยทั่วไปจะหมายถึง วัตถุใด ๆ จะมีสิ่งสำคัญคือ ลักษณะ,สถานะ และ พฤติกรรม (Attribute, State, Behavior) ส่วน Software Object จะพยายามเลียนแบบ Real World Object โดยแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นสถานะ (Data and variable) และพฤติกรรม (Method, Process, Operation)



รูปที่ 2.1 แสดงองค์ประกอบของวัตถุ

Object An object เป็นคอลเลคชัน (collection) ของ ข้อมูล (Data) (Attribute, properties) และ Function logic ซึ่งข้อมูลจะบอกถึงคุณสมบัติหรือสถานะของวัตถุและเมธตอด (Method) จะบอกถึงพฤติกรรมต่างๆของวัตถุนั้นๆ โดยเมธตอดจะแบ่งเป็น 2 ประเภท คือ interface method ซึ่งเป็นเมธตอดที่ได้ถูกใช้ได้จากวัตถุอื่น และ internal method ซึ่งเป็นเมธตอดที่จะถูกเรียกใช้ได้เฉพาะภายในวัตถุที่เป็นเจ้าของเท่านั้น และยังมีอีกคำหนึ่งที่ควรทราบคือ "คลาส (Class) " เป็นพิมพ์เขียวของวัตถุที่ไม่สามารถนำมาใช้ได้โดยตรง โดยจะมีการบอกถึงเมธตอดที่ใช้ได้โดยวัตถุและมีการแสดงชนิดของข้อมูล (data type) ที่บอกถึงสถานะของวัตถุโดยยังไม่ระบุค่าในคุณลักษณะ (Attribute) แต่ละตัวซึ่งถ้าเป็นวัตถุ จะมีการระบุค่าของคุณลักษณะ และเมธตอดเหมือนคลาสของมัน



รูปที่ 2.2 แสดงความสัมพันธ์ระหว่างคลาส (Class) และวัตถุ (Object)

2.1.1 **หลักการพื้นฐานของโปรแกรมภาษาเชิงวัตถุ** (Object- Oriented Programming)

วัตถุทั่วไปในธรรมชาติจะเก็บสถานะ (state) เป็นข้อมูลของตัวเองได้และต้องสามารถทำกิจกรรมบาง อย่างได้ เช่นการคำนวณและการเปลี่ยนแปลงข้อมูลของตัวเองหรือทำการติดต่อโต้ตอบกับวัตถุอื่น หากเราสมมุติว่านักเรียนคนหนึ่งเป็นวัตถุ เขาก็ควรจะมีชื่อ มีที่อยู่ มีผลการเรียนเป็นข้อมูลของตัวเอง และเขาน่าจะสามารถตอบคำถามอย่างเช่น ชื่ออะไร ผลการเรียนอยู่ระดับใดหรือเมื่อมีการประกาศผลสอบ เข้าก็น่าจะจำผลสอบได้ เป็นต้น ตัวแปลในภาษาอิมเพอเรทีฟ (imperative) ไม่สามารถจำลองพฤติกรรมและการทำงานของวัตถุอย่างนี้ ภาษาเชิงวัตถุจึงเสนอคลาสขึ้นเป็นกลไกสำหรับกำหนดโครง สร้างและพฤติกรรมของวัตถุเมื่อต้องการจำลองพฤติกรรมของวัตถุนั้น เราจะต้องสร้างสิ่งที่เรียกว่าอินสแตนซ์ (instances) ของคลาสนั้น ซึ่งจะมีหน่วยความจำสำหรับเก็บข้อมูลของวัตถุ และมีเมธอดสำหรับกระทำต่อข้อมูลของมันเองหรือตอบโต้แลกเปลี่ยนข้อมูลกันอินสแตนซ์อื่น อาจกล่าวได้ว่าอินสแตนซ์เป็นเหมือนกับตัวแปรพื้นฐานที่เก็บข้อมูลได้พร้อมกับมีการดำเนินการ (operations) ที่เกี่ยวข้องกับหน้าที่ของมันด้วย แต่อินสแตนซ์มีช่วงชีวิตต่างจากตัวแปรในภาษาอิมเพอเรทีฟ

คลาส คือโครงสร้างและพฤติกรรมของวัตถุประเภทหนึ่ง หากเปรียบอินสแตนซ์ในภาษาเชิงวัตถุก็เหมือนกับตัวแปลในภาษาอิมเพอเรทีฟแล้วคลาสก็เหมือนกับชนิดของข้อมูล (type) นั้นเองคลาสถูกสร้างขึ้นเพื่อกำหนด โครงสร้างของอินสแตนซ์ของวัตถุประเภทหนึ่ง เราจึงมีคลาสก่อนจะสร้างอินสแตนซ์ของคลาสนั้น และเมื่อมีคลาสหนึ่งแล้วจะสร้างอินสแตนซ์ของคลาสนั้นกี่อินสแตนซ์ ก็ได้ โดยที่อินสแตนซ์ของคลาสเดียวกันจะมีโครงสร้างเหมือนกันแต่อาจจะมีข้อมูลต่างกันไป ในการกำหนดคลาส เราต้องระบุว่าคลาสนั้นมีสมาชิก (member) ใดบ้างสมาชิกของคลาสมี 2 ประเภทคือ

- สมาชิกที่เป็นข้อมูลเรียกว่า data members หรือ field อาจเป็น ค่าคงที่ ตัวแปรของชนิดข้อมูลพื้นฐานอาร์เรย์ (array) หรือแม้กระทั้งอินสแตนซ์ของคลาส

- สมาชิกที่เป็นฟังก์ชันเรียกว่า methods members หรือเรียกสั้นๆว่าเมธตอดซึ่งอาจเป็นได้ทั้งโพรซีเดอร์ (procedures) คือฟังก์ชันที่ไม่ส่งค่าออกมาหรือฟังก์ชันที่ส่งค่าออกมา

ในยุค 1970s มีผู้พบข้อสังเกตว่าชนิดข้อมูลที่สร้างขึ้นโดยผู้เขียนโปรแกรม (user define data type) โดยใช้อาร์เรย์ หรือโครงสร้าง (structures) ที่มีในภาษาอิมเพอเรทีฟ ทั้งหลายนั้นไม่สามารถใช้งานได้อย่างปลอดภัยเหมือนกับชนิดของข้อมูลพื้นฐานที่มากับภาษาเช่น int หรือ float จึงมีการศึกษาและเสนอว่าชนิดของข้อมูลที่ใช้งานได้อย่างปลอดภัยนั้นควรมีลักษณะดังนี้คือ รายละเอียดของการเก็บแสดงข้อมูลนั้นต้องไม่มีความสำคัญต่อการใช้งานชนิดข้อมูลนั้นเนื่อง จากข้อมูลภายในจะไม่ถูกอ้างถึงได้โดยตรง แต่ต้องการจัดการผ่านทางการดำเนินการที่กำหนดไว้เท่านั้น ชนิดของข้อมูลที่มีลักษณะดังกล่าวนี้จะถูกเรียกว่า abstract data type และจากการศึกษาก็พบว่าภาษาที่จะสามารถสร้างชนิดข้อมูลแบบนี้ได้ต้องมีกลไกทางภาษาสองอย่างดังต่อไปนี้

- Encapsulation สำหรับนำข้อมูลกับการดำเนินการมาผูกติดกันเป็นหน่วยหนึ่ง

- Information hiding เพื่อกำหนดให้การอ้างถึงข้อมูลหรือการดำเนินการภายในชนิดข้อมูลหนึ่งเป็นไปได้หรือไม่ได้

กลไกทั้งสองถูกเสนอขึ้นใช้งานในภาษา Modula-2 และภาษา Ada ซึ่งยังไม่ถือว่าเป็นภาษา เชิงวัตถุ และเมื่อมีการพัฒนาภาษาเชิงวัตถุขึ้นก็ต้องมีข้อกำหนดว่าภาษาเชิงวัตถุต้องมีอย่างน้อย 4 กลไกคือ encapsulation, information hiding, inheritance, dynamic binding

หลักการพื้นฐานของการโปรแกรมเชิงวัตถุที่สำคัญและอาจจะเคยรู้มาบ้างแล้วในเรื่องของ ทฤษฎีของการโปรแกรมเชิงวัตถุแต่กล่าวโดยสรุปได้ดังนี้

1) Object : วัตถุคล้ายกับกล่องๆหนึ่ง ซึ่งในกล่องมีข้อมูล (instance variable) และเมธตอดรวมกันอยู่ภายใน และวัตถุนี้จะทำการรับและการส่งข้อความ (message) ระหว่างวัตถุเพื่อทำการใช้งานข้อมูลและเมธตอดที่อยู่ภายใน โดยการรับและการส่งข้อความนี้จึงเปรียบเสมือนอินเตอร์เฟสของวัตถุนั้นๆด้วย

2) Class : เป็นการจัดกลุ่มของวัตถุตามคุณสมบัติและผู้ใช้สามารถติดต่อคลาสโดยผ่านทางเมธตอดภายในคลาส หรือเมธตอดในซุปเปอร์คลาส (super class หรือ parent class) เท่านั้น หรืออาจจะกล่าวได้ว่าเป็นพิมพ์เขียวของวัตถุ

3) Method : เป็นการทำงานที่สำคัญหรือเป็นวิธีการกระทำที่สามารถทำงานกับวัตถุได้ แต่ละคลาสจะมีเมธตอดของตัวเอง โดยการทำงานจะเริ่มจากการส่งข้อความไปยังวัตถุที่ต้องการ (Receiver Object) เพื่อใช้เรียกเมธตอดที่อยู่ภายในวัตถุนั้น

ตัวอย่าง การส่งข้อความไปยังวัตถุที่ต้องการ

receiver.message_name(a1,a2,a3);

4) Encapsulation : คือการรวมกันของโครงสร้างข้อมูล (Data Structure) กับฟังก์ชัน (Method, Action) เกิดเป็นวัตถุใหม่ที่มีความสามารถในการซ่อนข้อมูลจากระบบภายนอกได้ (เหมือนเป็นยาเม็ดแคปซูลที่เราทานแก้หวัดซึ่งจะมองไม่เห็นยาภายในรู้แต่เพียงว่ายาเม็ดนี้ใช้รักษาโรคอะไรเท่านั้น) ทำให้ข้อมูลมีความมั่นคงขึ้น

5) Inheritance : คือการสร้างคลาสใหม่ขึ้นมา โดยมีการสืบทอดคุณสมบัติพื้นฐานที่มาจากคลาสเดิมแต่จะมีข้อมูลหรือเมธตอด เพิ่มขึ้นจากคลาสเดิม ถ้าคลาส B เป็นสับคลาส (Subclass) ของคลาส A วัตถุที่เป็นอินสแตนซ์ของคลาส B จะมีคุณสมบัติพิเศษกว่าอินสแตนซ์ของคลาส A แต่อย่างน้อยที่สุดจะต้องมีคุณสมบัติเหมือนอินสแตนซ์ของคลาส A

การถ่ายทอดคุณสมบัตินี้จะรวมถึงข้อมูลและเมธตอดของคลาส A และคลาส A จะถูกเรียกเป็นซุปเปอร์คลาสของคลาส B

6) Polymorphism : คือการที่เราส่งข้อความที่เหมือนกันไปในวัตถุที่ต่างกัน แต่ละวัตถุจะตอบสนองออกมาไม่เหมือนกัน ตามแต่ละชนิดและหน้าที่ของวัตถุ ความสามารถที่ใช้ในข้อความเหมือนกัน

สำหรับการกระทำที่เหมือนกัน ไปยังวัตถุต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหาหนึ่งๆ

ตัวอย่าง การ polymorphism

AB + CD ได้ผลลัพธ์ ABCD ซึ่ง concatenation โดยการดำเนินการจากคลาสสตริง (String)

5 + 3 ได้ผลลัพธ์ 8 ซึ่งรวมโดยตัวดำเนินการ (Operator) จากคลาสอินทิเจอร์ (Integer)

ทั้ง 2 แบบ เป็นการส่งข้อความ '+' เข้าไปยังวัตถุภายในคลาสสตริงและอินทีเจอร์ตามลำดับ

7) Dynamic Binding : คือ การนำโปรแกรมย่อยๆมาประกอบให้ใช้งานในขณะรันไทม์ (run time) โดยในขณะคอมไพล์ไทม์ (Compile time) นั้นจะเก็บโปรแกรมในรูปแบบของคลาสต่างๆไว้ เพื่อไม่ให้เกิดความยุ่งยาก ซับซ้อน ต่อจากนั้น เมื่อนำมาใช้ในขณะรันไทม์เมื่อมีการเรียกใช้คลาสนั้นมาไว้ในส่วนของโปรแกรม และเมื่อใช้งานเสร็จแล้วจะถูกลบออกจากหน่วยความจำ

8. Override : คือ การที่เราสร้างสับคลาสขึ้นมาใหม่ และมีการสร้างเมธตอดที่ซ้ำกับเมธตอดที่เป็นของซูเปอร์คลาส

9) Overload : คือ การที่เราสร้างเมธตอดชื่อเหมือนกัน แต่รับพารามิเตอร์ (Parameter) ต่างกัน ภายในคลาสเดียวกัน

2.1.2 คุณสมบัติของวัตถุ

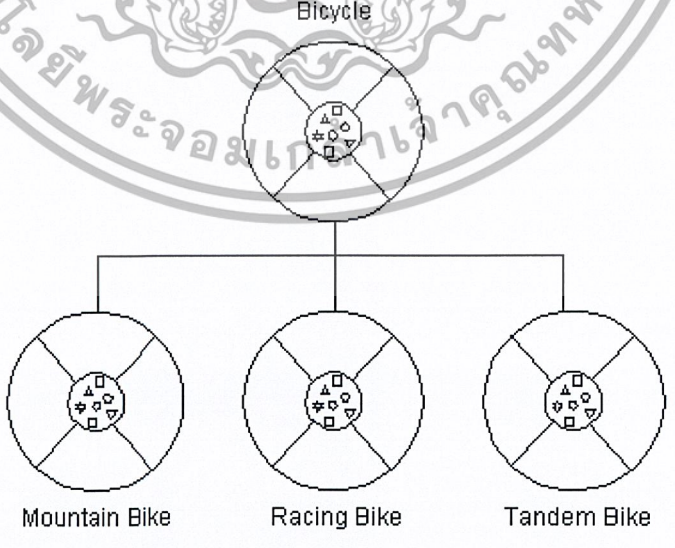
2.1.2.1 การปกป้องข้อมูลภายใน (Encapsulation/Information hiding)

ทฤษฎีนี้ถูกคิดค้นโดย James Rambaugh ซึ่งหมายถึง การแยกลักษณะภายนอก (Interface) ซึ่งสามารถติดต่อกับวัตถุอื่น ๆ ออกจากส่วนที่อิมพลีเมนต์ (implement) ภายในของวัตถุซึ่งจะถูกใช้ได้เฉพาะตัววัตถุนั้น ๆ มีข้อดี คือ เพิ่มความสามารถในการปกป้องข้อมูลโดยไม่ให้วัตถุอื่นเข้ามาทำการเปลี่ยนแปลงข้อมูลก่อนได้รับอนุญาตทั้งนี้ตัววัตถุควรรู้และทำการเปลี่ยนค่าด้วยตนเองซึ่งเป็นผลให้ดูแลรักษาวัตถุ (maintain object) ง่ายขึ้น

2.1.2.2 การถ่ายทอดคุณสมบัติ (Inheritance)

คือหลักการในการที่วัตถุทุก ๆตัวใน generalized collection ได้มีการใช้ข้อมูล (structure) และพฤติกรรม (behavior) ร่วมกันซึ่งจะมีความสัมพันธ์แบบ is- a relationship โดยเรียกคลาสที่อยู่เหนือกว่าว่าซูเปอร์คลาสซึ่งจะถ่ายทอดคุณสมบัติทั้งคุณลักษณะ และเมธตอดมายังคลาสที่ต่ำกว่าเรียกว่าสับคลาสซึ่งจะมีคุณลักษณะ และเมธตอดเพิ่มเติมจากซูเปอร์คลาสมีข้อดีคือ

- เพิ่ม Consistency เพียงแค่เปลี่ยนเป็นเมธตอดและคุณลักษณะที่ซูเปอร์คลาสซึ่งเป็นผลให้ค่าที่สับคลาสเปลี่ยนไปด้วย
- เป็นการส่งเสริมการนำวัตถุกลับมาใช้ใหม่



รูปที่ 2.3 แสดงการสืบทอดคุณสมบัติ (Inheritance)

2.1.2.3 **การแสดงคุณสมบัติของวัตถุ** (Polymorphism)

อาจหมายถึง "having many forms" โดยการส่งข้อความเดียวกันให้กับวัตถุที่ต่างกันเป็นผลทำให้วัตถุแสดงพฤติกรรมที่แตกต่างกัน มีข้อดีคือ เป็นการสนับสนุนนำโปรแกรมกลับมาใช้ใหม่และสามารถเปลี่ยนแปลงซอฟต์แวร์ได้ในระหว่างการพัฒนา

2.1.2.4 **การพิจารณาคุณลักษณะ** (Identity)

คือตัววัตถุโดยจะดูว่าในระบบมีวัตถุอะไรบ้าง ให้ดูคุณลักษณะและเมธตอดจะเป็นตัวแบ่งแยกวัตถุและการพิจารณาคุณลักษณะต้องพิจารณาไปถึงรูปแบบของมัน เช่น Attribute Object หรือเป็นตัวแปร ถ้าคุณลักษณะเป็นตัวแปรต้องรู้ชนิดของตัวแปร

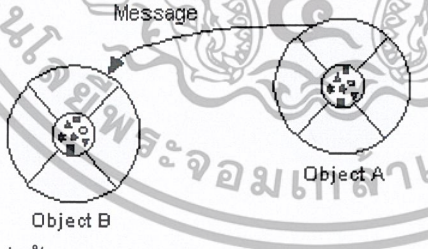
2.1.2.5 **การเป็นคลาสต้นแบบ** (Classification)

คือคลาสที่เป็นต้นแบบในการสร้างวัตถุใดๆที่อยู่ในคลาสนั้นๆ ให้มีคุณลักษณะและคุณสมบัติตามคลาสที่ได้สร้างวัตถุนั้นขึ้นมา เราจะกล่าวได้ว่า Set Of Object = Instance Of The Class

2.1.3 **การติดต่อกันระหว่างวัตถุ** (Message Passing)

คือการที่วัตถุติดต่อกันด้วยการส่งข้อความถึงกันและกันซึ่งข้อความจะประกอบด้วยจุดหมายปลายทาง (destination) ของข้อความนั้นและข้อมูลที่สำคัญ (argument หรือ parameter)

การติดต่อกันระหว่างวัตถุเปรียบได้กับ function call หรือ procedure call ที่มีใน structured programming โดยผ่าน Interface method ของวัตถุนั้น ๆ มีผลทำให้วัตถุที่เป็นผู้รับข้อความนั้นกระทำการอย่างใดอย่างหนึ่ง



รูปที่ 2.4 แสดงการส่งข้อความ (message) จาก Object A ไปยังObject B

2.1.4 **การวิเคราะห์และออกแบบเชิงวัตถุ** (Object Oriented Analysis and Design: OOAD)

ปัจจุบันเราอยู่ในโลกที่ซับซ้อนงานแต่ละงานจะมีความยุ่งยาก วัตถุหนึ่ง ๆ อาจเกิดจากการรวมของวัตถุต่าง ๆ ได้ วัตถุต่าง ๆก็สามารถมีความสัมพันธ์ต่อกันได้ ซึ่งวัตถุเหล่านี้อาจจะเป็นคน, สัตว์, สิ่งของ งานต่างๆ หรือจะเป็นสิ่งที่เราสนใจในการวิเคราะห์และออกแบบระบบอาจทำให้เกิดความสับสน ยังไม่มีเทคนิคในการวิเคราะห์และออกแบบระบบใดที่จะช่วยให้สามารถวิเคราะห์ลักษณะที่สำคัญจากโลกรอบตัวเรา ให้อยู่ในรูปของนามธรรม (Abstract) ให้มากที่สุด และทำให้

สามารถจำแนกวัตถุต่าง ๆ ออกเป็นลำดับชั้นตามความสัมพันธ์และรูปแบบของมัน ซึ่งการวิเคราะห์และออกแบบระบบเชิงวัตถุจะเป็นวิธีที่ช่วยให้สามารถแปลงโลกแห่งความเป็นจริงเป็นซอฟต์แวร์ได้ง่ายมากขึ้น

การวิเคราะห์และออกแบบระบบเชิงวัตถุ คือการกำหนดแนวทางการปฏิบัติ (process) และสัญลักษณ์ (Notation) ที่ใช้ในการวิเคราะห์และออกแบบซอฟต์แวร์ ซึ่งจะอ้างอิงตามหลักการของวัตถุ คือ การรวมคุณลักษณะและหน้าที่การดำเนินการไว้ด้วยกัน การวิเคราะห์และออกแบบระบบเชิงวัตถุนั้น มีการกำหนดมาตรฐานขั้นตอนการปฏิบัติในการวิเคราะห์และออกแบบระบบ ซึ่งในมาตรฐานจะต้องมีการนิยามคำศัพท์ต่าง ๆ เพื่อความเข้าใจที่ตรงกัน สร้างมาตรฐานของโมเดล (Model) และสัญลักษณ์ที่ใช้ในการวิเคราะห์และออกแบบระบบ มีนักวิชาการเป็นจำนวนมากได้กำหนดมาตรฐานต่าง ๆ ซึ่งเป็นที่รู้จักดี เช่น Object Oriented Software Engineering : OOSE โดย Jacobson, Object Modeling Technique : OMT โดย Rumbaugh, OO analysis and design โดย Coad และYourdon เป็นต้น มาตรฐานต่าง ๆ ก็จะมีความเหมือนหรือแตกต่างกันไป แต่โดยรวมแล้วก็ต้องประกันได้ว่าทำงานได้จริงและถูกต้อง ในแต่ละเฟส (Phase) การทำงานจะต้องสามารถคาดคะเนได้ว่าผลลัพธ์ที่ได้คืออะไร การทำงานถ้ามีความผิดพลาดเกิดขึ้นจะต้องกลับไปหาได้ว่าความผิดพลาดที่เกิดขึ้นอยู่ที่ขั้นตอนใด มาตรฐานที่ใช้ต้องสามารถรองรับได้ทั้งระบบงานเล็ก ๆ จนถึงระบบงานที่มีความซับซ้อนมาก ๆ ได้ โดยสรุปแล้วการวิเคราะห์และออกแบบระบบเชิงวัตถุมีโมเดลที่ใช้อยู่ 3 แบบ คือ

2.1.4.1 **การวิเคราะห์และออกแบบระบบเชิงวัตถุแบบฟังก์ชันนอลโมเดล (Functional Model)** ฟังก์ชันนอลโมเดล เป็นโมเดลที่ใช้ในการแสดงความต้องการของระบบทั้งหมดช่วยในการอธิบายรายละเอียดหลัก ๆ ภายในวัตถุ แสดงให้เห็นการไหลของข้อมูลในแต่ละการทำงานโดยจะสนใจเพียงแค่ว่ามีงานอะไรบ้างที่ต้องทำ จะยังไม่สนใจว่างานนั้น ๆ ทำอย่างไร เช่นในการพิมพ์รายงานสรุปประจำเดือน จะต้องรู้ว่างานนี้ใครเป็นผู้รับผิดชอบรายงาน รูปร่างหน้าตาของรายงานเป็นอย่างไร มีข้อมูลอะไรบ้างที่สนใจ และเมื่อออกรายงานแล้ว ใครเป็นผู้ที่รับรายงานนั้นเป็นต้น เครื่องมือที่ใช้ในการแสดงความต้องการของระบบทั้งหมดในลักษณะที่ผู้ใช้งานสามารถเข้าใจได้ง่าย โดยจะถูกนำไปใช้ต่อไปในเฟสต่าง ๆ ของการวิเคราะห์และออกแบบระบบ

2.1.4.2 **การวิเคราะห์และออกแบบระบบเชิงวัตถุแบบออบเจ็กโมเดล (Object Model)** ออบเจ็กโมเดลเป็นโมเดลที่ใช้ในการแสดงโครงสร้างของระบบ โดยจะแสดงในรูปของคลาสต่าง ๆ พิจารณาจากความต้องการของระบบที่แสดงอยู่ในฟังก์ชันนอลโมเดลเครื่องมือที่ใช้ในการแสดงโครงสร้างของระบบจะมีคอมโพเนนท์ไดอะแกรม (Component Diagram) และคลาสไดอะแกรม (Class Diagram) ซึ่งคอมโพเนนท์ไดอะแกรมแสดงให้เห็นถึงความสัมพันธ์ระหว่างคอมโพเนนท์ต่างๆ ภายในระบบ คลาสไดอะแกรมแสดงให้เห็นถึงคุณลักษณะคือ ข้อมูล

หรือตัวแปร การดำเนินการคือ เมธตอดภายในคลาส และความสัมพันธ์ระหว่างคลาสต่าง ๆ ภายในระบบ

2.1.4.3 **การวิเคราะห์และออกแบบระบบเชิงวัตถุแบบไดนามิคโมเดล** (Dynamic Model) ไดนามิคโมเดล เป็นโมเดลที่ใช้ในการแสดงถึงการทำงานระหว่างวัตถุต่าง ๆ ตามการส่งข้อความหรือเมื่อเหตุการณ์ต่าง ๆ ได้เกิดขึ้น วัตถุในที่นี้หมายถึงอินสแตนซ์ที่สร้างขึ้นจากคลาสที่ได้ออกแบบไว้ใน ออบเจคโมเดล โดยที่แต่ละวัตถุมีคุณสมบัติ และพฤติกรรมเช่นเดียวกับคลาสต้นแบบ ในการทำงานของระบบจะประกอบขึ้นจากการส่งข้อความไปมาระหว่างวัตถุเหล่านั้น เมื่อมีการทำงานไปเรื่อย ๆแล้ววัตถุอาจจะมีการเปลี่ยนสถานะ (State) ไปตามเหตุการณ์ที่เกิดขึ้นได้ เพื่อให้เป็นตามที่กำหนดไว้ในฟังก์ชันนอลโมเดล เครื่องมือที่ใช้คือซีเควนซ์ไดอะแกรม (Sequence Diagram) และสเตทไดอะแกรม (State Diagram) ซีเควนซ์ไดอะแกรมแสดงให้เห็นถึงลำดับขั้นตอนการทำงาน เมื่อมีเหตุการณ์หนึ่ง ๆ ขึ้นแล้ววัตถุต่าง ๆ มีการทำงานต่อไปอย่างไร วัตถุประสงค์หลักของซีเควนซ์ไดอะแกรม คือเพื่อให้ผู้เขียนโปรแกรมสามารถนำไปศึกษาและพัฒนาโปรแกรมหรือขยายขีดความสามารถของโปรแกรมต่อไป ส่วนสแตทไดอะแกรม เป็นไดอะแกรมที่แสดงให้เห็นถึงสถานะทั้งหมดที่เป็นไปได้ และเหตุการณ์ที่ทำให้เกิดการเปลี่ยนสถานะของวัตถุแต่ละตัว

2.2 หลักการพื้นฐานของภาษาจาวา

2.2.1 ลักษณะหรือคุณสมบัติของภาษาจาวา

ภาษาจาวาถูกออกแบบมาให้มีลักษณะดังต่อไปนี้

1) จาวาเป็นภาษาที่เป็นเชิงวัตถุ กล่าวคือมีการออกแบบให้มีโครงสร้างเป็นเชิงวัตถุ เพื่อให้การพัฒนาโปรแกรมเป็นไปโดยง่ายและรวดเร็ว และสามารถนำส่วนต่างๆ มาใช้ได้ใหม่โดยไม่ต้องแก้ไขหรือมีการแก้ไขที่น้อยที่สุด

ในภาษาจาวา เกือบทุกสิ่งทุกอย่างจะกำหนดในรูปเชิงวัตถุ ยกเว้นเฉพาะตัวแปรบางชนิด เช่น ตัวเลขและค่าทางตรรกะ จาวามีการจำแนกออกเป็นคลาสแต่ละคลาสจะมีชุดของเมธตอดซึ่งแสดงถึงคุณสมบัติเฉพาะของเชิงวัตถุนั้น คลาสสามารถถ่ายทอดคุณสมบัติจากคลาสอื่นได้ โดยที่คลาสเริ่มต้นจะเป็นคลาสออบเจคเสมอ เช่นคลาสรถบรรทุกมีการถ่ายทอดคุณสมบัติมาจากคลาสของรถ ซึ่งรถถ่ายทอดคุณสมบัติมาจากคลาสยานพาหนะและยานพาหนะถ่ายทอดคุณสมบัติมาจากวัตถุอีกทอดหนึ่ง

2) เป็นภาษาที่ง่ายในการเรียนรู้และใช้งาน ซึ่งตีความหมายได้หลายอย่างในแง่ต่างๆ ต่อไปนี้

- ภาษาจาวานำหลักไวยากรณ์ภาษาส่วนใหญ่มาจากภาษา C และ C++ ผู้ที่คุ้นเคยกับภาษา CหรือC++อยู่แล้ว จะเข้าใจไวยากรณ์ภาษาจาวาได้ง่าย หรือใช้เวลาศึกษาไม่นานนัก

- ภาษาจาวาตัดคุณสมบัติบางอย่างของภาษา C และ C++ ที่ยุ่งยากออกไปเช่น pointer arithmetic, default argument, scope resolution, protect and private inheritance และ operator overloading เมื่อจาวาเป็นภาษาเชิงวัตถุแล้วกลไกอย่างเช่น structures, unions, bit fields และ enumerated types รวมทั้ง typedef ก็ไม่มีความจำเป็นจึงถูกตัดออกไป นอกจากนี้กลไกที่ทำให้เกิดความกำกวม เช่น multiple inheritance และ copy constructor และกลไกที่อาจทำลายแบบแผนการเขียนโปรแกรมเชิงวัตถุที่ดีเช่น friend methods ก็ถูกตัดออกไปด้วย ซึ่ง จาวาเป็นภาษาที่มีการถ่ายทอดแบบ Single inheritance กล่าวคือ แต่ละคลาสจะถ่ายทอด คุณสมบัติจากคลาสใดคลาสหนึ่งเพียงคลาสเดียว ทำให้ความซับซ้อนของภาษาลดลง

- ภาษาที่ถูกเรียกว่า typed language จะทำการตรวจสอบเกี่ยวกับชนิดของข้อมูล เพื่อช่วยให้โปรแกรมทำงานโดยไม่มีความผิดพลาดเกี่ยวกับชนิดของข้อมูล แต่การเขียนโปรแกรมภาษาเช่นนี้มีข้อยุ่งยาก เช่น ตัวอักษรบวกกับเลขจำนวนเต็มไม่ได้ ภาษาจาวาเป็น typed language แต่เพื่อให้ใช้งานง่าย จึงสร้างกฎเกณฑ์เกี่ยวกับ automatic type coercion ซึ่งเป็นการเปลี่ยนแปลงค่าระหว่างชนิดของข้อมูลที่ต่างกัน ช่วยให้โปรแกรมง่ายขึ้นโดยลดภาระการเปลี่ยนชนิดของข้อมูลไปจากโปรแกรมเมอร์

3) โปรแกรมที่สร้างขึ้นด้วยภาษาจาวาจะไม่มีความผิดพลาดจากข้อบกพร่องของภาษา ซึ่งไม่เกี่ยวกับตรรกะของโปรแกรม คุณสมบัติของภาษาอย่างนี้เรียกว่าความคงทน (robust) ภาษาจาวาถูกออกแบบให้มีความคงทนด้วยวิธีการดังนี้

- ภาษาจาวามีการใช้กลไก exception handling เพื่อให้โปรแกรมสามารถจัดการกับความผิดปกติบางอย่างที่เกิดขึ้นในขณะโปรแกรมทำงานให้โปรแกรมทำงานต่อไปได้โดยไม่ต้องหยุดลง ถึงแม้ว่าโปรแกรมที่คุณเขียนขึ้นจะมีข้อบกพร่องแฝงอยู่ก็ตาม ถ้าเป็นข้อบกพร่องที่ระบบไม่รู้จักหรือคาดหวังว่าจะเกิดขึ้น โปรแกรมจาวาจะค้นหาในเอกเซฟชัน (exception) ต่าง ๆ และดำเนินการก่อนที่จะเกิดปัญหาในภายหลัง ที่เป็นเช่นนั้นเนื่องมาจากโปรแกรมในรูปแบบเดิมอนุญาตให้สามารถดำเนินการกับหน่วยความจำทั้งหมดของเครื่องได้ โปรแกรมสามารถเปลี่ยนค่าต่างๆในหน่วยความจำซึ่งอาจก่อให้เกิดปัญหาในภายหลัง โปรแกรมจาวาอนุญาตให้โปรแกรมดำเนินการกับหน่วยความจำเพียงบางส่วนเท่านั้น ทำให้โปรแกรมจาวาไม่สามารถแก้ไขค่าในส่วนที่ไม่ควรแก้ไขได้

จาวาใช้เอกเซฟชันในการบ่งถึงข้อผิดพลาดเมื่อคุณรันโปรแกรม แต่ในบางครั้งเอกเซฟชันเป็นการแจ้งถึงเหตุการณ์พิเศษที่โปรแกรมของคุณควรให้ความสนใจ

ถ้าคุณพยายามที่จะจัดการกับข้อผิดพลาดทุกอย่างที่อาจเกิดขึ้นในโปรแกรมของคุณ โครงสร้างของโปรแกรมคุณก็จะซับซ้อนมากและยากต่อการเข้าใจ ข้อดีของเอกเซฟชัน คือมีการแยกโค้ดที่จัดการกับความผิดพลาดออกจากโค้ดที่จัดการกับความผิดพลาดออกจากโค้ดที่ทำงานเมื่อไม่มีข้อผิดพลาดเกิดขึ้น ข้อดีอีกข้อหนึ่งก็คือ มีการเตรียมการเป็นพิเศษเพื่อจัดการกับเอกเซฟชันชนิดต่าง ๆ โปรแกรมของคุณจะต้องจัดการกับเอกเซฟชันนั้น ๆ มิฉะนั้นโปรแกรมของคุณก็จะไม่สามารถคอมไพล์ได้

เอกเซฟชันในจาวาก็คือ วัตถุซึ่งสร้างขึ้นเมื่อมีเหตุการณ์ผิดปกติเกิดขึ้นในโปรแกรม วัตถุนี้จะมีข้อมูลเกี่ยวกับธรรมชาติของปัญหาเอกเซฟชันจะถูกโทร์ว (thrown) นั่นก็คือ วัตถุที่บ่งถึงรายละเอียดเกี่ยวกับเอกเซฟชันจะถูกส่งไปเป็นอาร์กิวเมนต์ไปให้ส่วนของโปรแกรมที่รับผิดชอบในการจัดการกับปัญหานั้น ส่วนของโปรแกรมที่รับวัตถุของเอกเซฟชันเข้ามานั้นเราเรียกว่าแคช (catch)

- ภาษาจาวาไม่มีกลไกสำหรับคืน (de-allocation) หน่วยความจำที่ขอมาในขณะโปรแกรมทำงาน (dynamic memory allocation) อย่างที่มีในภาษา C และ C++ คือ free() และ delete() ภาษาจาวาอาศัย automatic garbage collector ทำหน้าที่เก็บหน่วยความจำที่ไม่สามารถอ้างถึงได้แล้ว กลับไปใช้งานใหม่โดยอัตโนมัติ

- ภาษาจาวาเป็นภาษาประเภท strongly typed หมายถึง ภาษาที่เน้นความถูกต้องของชนิดข้อมูลที่ใช้ในโปรแกรม คอมไพเลอร์ของภาษาประเภทนี้จะทำการตรวจสอบว่าโปรแกรมการจัดการกับชนิดข้อมูลของตัวแปรถูกต้องหรือไม่ เรียกกิจกรรมของคอมไพเลอร์นี้ว่า type checking ความผิดพลาดเกี่ยวกับชนิดข้อมูลทั้งหมดจะถูกปฏิเสธตั้งแต่ตอนคอมไพล์โปรแกรมจึงไม่มีความผิดพลาดเกี่ยวกับชนิดข้อมูลเกิดขึ้นในระหว่างที่โปรแกรมทำงาน นอกจากนี้ยังมีการตรวจสอบอีกว่า ในระหว่างโปรแกรมทำงาน มีการเปลี่ยนชนิดข้อมูล (casting) ระหว่างค่าต่างชนิดของข้อมูลถูกต้องหรือไม่ และการอ้างถึงสมาชิกในอาร์เรย์ หรือสตริง (string) อยู่ในขอบเขตที่ถูกต้องหรือไม่

4) โปรแกรมภาษาจาวามักจะถูกส่งผ่านระบบเครือข่ายไปทำงานบนเครื่องคอมพิวเตอร์ ดังนั้นภาษาจาวาต้องมีหลักประกันว่าจะไม่ก่อให้เกิดอันตรายต่อเครื่องหรือระบบ ภาษาจาวาจึงมีข้อกำหนดหลายอย่างเพื่อให้โปรแกรมไม่สามารถทำอันตรายต่อระบบที่รับโปรแกรมนั้นไปทำงาน โดยแบ่งการป้องกันออกเป็นหลายระดับดังนี้

- จาวาอินเตอร์พรีเตอร์ มี byte-code verifier ทำหน้าที่ตรวจสอบโปรแกรมที่จะถูกทำงานว่า มีคำสั่งที่ผิดปกติ หรือมีการทำงานที่ไม่สมควรหรือไม่ หากพบก็จะปฏิเสธที่จะทำงานโปรแกรมนั้น

- ภาษาจาวามีระบบรักษาความปลอดภัยที่เรียกว่า sandbox model นั่นคือ โปรแกรมที่ถูกนำมาจากเครื่องอื่นผ่านทางระบบเครือข่ายจะถือว่าเป็นโปรแกรมที่ไม่น่าไว้ใจ (un-trusted codes) และถูกเก็บอยู่ในภาวะที่เรียกว่าแซนด์บอกซ์ (sandbox) โปรแกรมที่อยู่ในแซนด์บอกซ์จะ

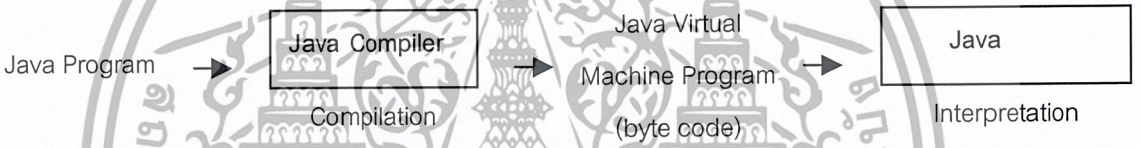
มีข้อจำกัดในการทำงานหลายอย่าง ซึ่งถูกควบคุมโดย security manager เช่น ไม่สามารถอ่านหรือเขียนไฟล์ เป็นต้น

5) จุดมุ่งหมายสำคัญของการออกแบบภาษาจาวาคือ โปรแกรมต้องสามารถทำงานบนเครื่องต่างระบบกันได้ เรียกคุณสมบัตินี้ว่า "ไม่ขึ้นกับระบบ" (architecture neutral หรือ platform independent) เพื่อให้ได้คุณสมบัตินี้ ภาษาจาวาต้องใช้วิธีการแปลภาษาแบบทั้ง compilation และ interpretation รวมทั้งการกำหนด Java Virtual Machine

2.2.2 การทำงานของจาวา

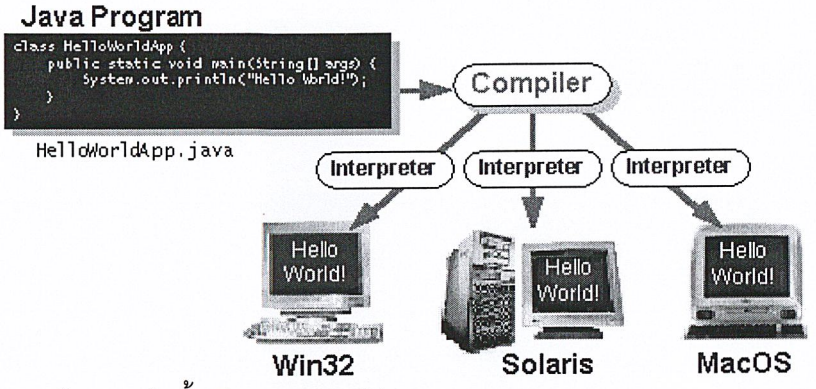
2.2.2.1 การทำงานของ Java Virtual Machine (JVM)

ภาษาจาวานำความคิดการสร้างเครื่องจักรสมมติ (Virtual Machine) มาใช้ เพื่อให้โปรแกรมไม่ขึ้นกับระบบ โดยมีคอมไพเลอร์ทำการแปลภาษาให้เป็นโปรแกรมของ JVM แล้วนำโปรแกรมนั้นมาทำงานด้วยเครื่องจักรสมมติที่จำลองขึ้นโดย จาวาอินเตอร์พรีเตอร์ (Java interpreter) สรุปได้ดังรูป



รูปที่ 2.5 แสดงการแปลงจากโปรแกรมจาวาไปเป็นโปรแกรมของ Java Virtual Machine

ในวิธีนี้โปรแกรมภาษาจาวาจะถูกคอมไพล์โดย Java compiler ได้เป็นโปรแกรมของ JVM แล้วสามารถนำไปทำงานบนเครื่องใดๆที่มีจาวาอินเตอร์พรีเตอร์ได้ จึงมีคุณสมบัติไม่ขึ้นกับระบบ โปรแกรมของ JVM จะทำงานได้เร็วกว่าการใช้อินเตอร์พรีเตอร์ (interpreter) เพียงอย่างเดียวเพราะขั้นตอนการทำ compilation ถูกแยกออกไปจากการ execution และด้วยการออกแบบคำสั่งของ JVM ให้ใกล้เคียงกับคำสั่งของหน่วยประมวลผลทั่วไป จาวาอินเตอร์พรีเตอร์ จึงเปลี่ยนคำสั่งของ JVM ไปสู่คำสั่งของหน่วยประมวลผลที่ใช้งานได้ง่าย การทำอินเตอร์พรีเตอร์โปรแกรมของ JVM จึงเร็วกว่าการ interpretation ของภาษาระดับสูงอื่นๆ ดูรูปประกอบ



รูปที่ 2.6 แสดงถึงความไม่ขึ้นกับระบบของโปรแกรมภาษาจาวา

ภายใน JVM จะมีหน่วยประมวลผลสมมติ ที่เรียกว่า virtual processor ทำหน้าที่ประมวลผลคำสั่งของ JVM โดยปกติหน่วยประมวลผลสมมติ ของ JVM ที่จำลองขึ้นบนคอมพิวเตอร์เครื่องหนึ่ง จะแปลคำสั่งของ JVM เป็นคำสั่งของหน่วยประมวลผลในคอมพิวเตอร์เครื่องนั้น เรียกว่าเนทีฟโค้ด (native code) ที่ทำหน้าที่เดียวกันแล้วให้หน่วยประมวลผลทำงานคำสั่งเนทีฟโค้ดนั้น สังเกตว่าเนทีฟโค้ดนั้นอาจเป็น Applicaiton Program Interface (API) ของระบบปฏิบัติการที่ใช้หรืออาจจะเป็น standard library ที่สร้างขึ้นสำหรับหน่วยประมวลผลนั้น ทำให้ JVM หนึ่งอาจใช้งานได้ในระบบต่างๆ โดยเปลี่ยนแปลงแต่ standard library เท่านั้น คำสั่ง (opcode) ของ JVM มีขนาด 1 byte ทุกคำสั่ง จึงเรียกโปรแกรม JVM ว่าโปรแกรม byte code

นอกจากที่กล่าวมาแล้ว ยังมีลักษณะหรือคุณสมบัติอื่นๆ ของภาษาจาวาอีกดังนี้

- จาวามีการทำงานแบบมัลติเทรด (Multi-thread)

ระบบคอมพิวเตอร์สมัยใหม่ เช่น Unix และ Windows 95 การทำงานแบบมัลติทาสกิ้ง (muti – tasking) ซึ่งหมายความว่าคอมพิวเตอร์สามารถทำงานได้มากกว่า 1 งานในขณะเดียวกัน จาวาเป็นภาษาที่ถูกออกแบบให้เป็นมัลติทาสกิ้ง

โปรแกรมจาวาสามารถมีได้มากกว่า 1 เทรดของการทำงานเช่นเทรดหนึ่งอาจทำงานคำนวณบางอย่างที่ใช้เวลานานและอีกเทรดหนึ่งก็ทำการติดต่อกับผู้ใช้ ทำให้ผู้ใช้ไม่ต้องหยุดทำงานเพื่อรอให้จาวาเสร็จงานที่ใช้เวลาในการทำงานนานก่อน

โปรแกรมในลักษณะมัลติเทรดโดยทั่วไปจะยุ่งยากซับซ่อน เนื่องจากเหตุการณ์หลายเหตุการณ์สามารถเกิดขึ้นพร้อม ๆ กัน แต่จาวาได้ตระเตรียมระบบเพื่อจัดการในการเกิดเหตุการณ์พร้อมกัน ช่วยให้การเขียนโปรแกรมเป็นไปได้โดยง่าย

- จาวามีขนาดเล็ก

เนื่องจากจาวาได้ถูกออกแบบเพื่อใช้กับคอมพิวเตอร์ขนาดเล็ก ระบบของจาวาจึงค่อนข้างเล็ก สามารถรันได้อย่างมีประสิทธิภาพบนเครื่องคอมพิวเตอร์ที่มีตั้งแต่ 4 Mb RAM ตัวแปลภาษาจาวาใช้เนื้อที่หน่วยความจำเพียงไม่กี่ร้อยกิโลไบต์

โปรแกรมที่สร้างขึ้นด้วยภาษาจาวาแบ่งได้เป็น 2 ประเภท คือ Java Application และ Java Applet

2.2.2.2 การทำงานของจาวาแอปพลิเคชัน (Java Application)

จาวาแอปพลิเคชัน คือ โปรแกรมที่สร้างขึ้นเพื่อถูกทำงานโดย จาวาอินเตอร์พรีเตอร์ ต้องเป็นโปรแกรมที่สมบูรณ์ มี main() เป็นจุดเริ่มต้น และสามารถควบคุมการดำเนินไปของตัวเองได้ ทำงานภายใต้ จาวาอินเตอร์พรีเตอร์ โดยไม่ต้องมีโปรแกรมอื่นช่วย บางคนจึงเรียกว่า stand alone program

ขั้นตอนการสร้างและทำงานจาวาแอปพลิเคชันมีดังนี้ เริ่มจากใช้ editor เขียนโปรแกรมสำหรับแอปพลิเคชันเก็บไว้ในไฟล์ที่เอกซ์เทนชัน (extension) เป็น .java จากนั้นใช้คอมไพเลอร์ javac.exe ทำการคอมไพล์ จะได้ผลลัพธ์เป็นไฟล์ที่มีเอกซ์เทนชันเป็น .class ซึ่งเป็นโปรแกรม JVM เมื่อต้องการทำงานก็ส่งไฟล์ที่มีเอกซ์เทนชันเป็น .class นั้นให้แก่ java.exe

2.2.2.3 การทำงานของจาวาแอปเพล็ต (Java Applet)

แอปเพล็ต คือ โปรแกรมที่สร้างขึ้นเพื่อส่งไปกับหน้า HTML (HTML page) ให้ไปทำงานภายใต้เว็บบราวเซอร์ (web browser) ที่มีจาวาอินเตอร์พรีเตอร์ โดยสามารถติดต่อกับผู้ใช้และแสดงผลผ่านทางเว็บเพจ (web page) นั้น หรือกล่าวได้ว่าเป็นโปรแกรมที่ฝากมากับหน้าHTML นั่นคือ จากในหน้า HTML จะสามารถเรียกแอปเพล็ตมาทำงานและแสดงผลในพื้นที่ของเพจ (page) นั้น โดยที่เว็บบราวเซอร์จะเป็นผู้โหลดแอปเพล็ตที่ถูกเรียกมาจากเซิร์ฟเวอร์ (server) ที่ให้หน้า HTML นั้นมาและจัดการทำงานให้โดยอัตโนมัติ วิธีนี้ทำให้เราสามารถส่งโปรแกรมไปทำงานบนเครื่องใดๆในระบบอินเตอร์เน็ต โดยใช้หน้า HTML เป็นตัวกลาง

ขั้นตอนการสร้างและทำงานแอปเพล็ตมีดังนี้ ใช้ editor เขียนโปรแกรมสำหรับแอปเพล็ตเก็บไว้ในไฟล์ที่เอกซ์เทนชันเป็น .java จากนั้นใช้คอมไพเลอร์ javac.exe ทำการคอมไพล์ จะได้ผลลัพธ์เป็นไฟล์ที่มีเอกซ์เทนชันเป็น .class ซึ่งเป็นโปรแกรม JVM ของแอปเพล็ตนั้น (สังเกตว่า java.exe จะไม่สามารถทำงานกับไฟล์ที่มีเอกซ์เทนชันเป็น .class ที่ได้นั้นได้เนื่องจากไม่ใช่ จาวาแอปพลิเคชัน จากนั้นก็ต้องแทรกแท็ก (Tag) <applet code="file.class" width=100 height=100> </applet> ลงไปในเอกสาร HTML (file.class คือไฟล์ applet ที่เราต้องการ) ซึ่งไฟล์ของแอปเพล็ตนั้นจะถูกมองหาในไดเรกทอรีและเครื่องเดียวกันกับหน้า HTML นั้นหากเป็นกรณีที่แอปเพล็ตอยู่ที่เครื่องอื่นก็ต้องอ้างถึงไฟล์ของแอปเพล็ตนั้น และเมื่อต้องการดูผลของแอปเพล็ตนี้ก็ใช้เว็บบราวเซอร์ที่มี จาวาอินเตอร์พรีเตอร์ เช่น Netscape หรือ Microsoft Internet Explorer เรียกดูไฟล์ HTML ที่มีแอปเพล็ตอยู่ เมื่อไฟล์นี้ถูกจัดการถึงประโยคที่เรียกแอปเพล็ตนั้น ไฟล์แอปเพล็ตนั้นก็จะถูกโหลดมาทำงาน นอกจากนี้ยังสามารถใช้โปรแกรม appletviewer.exe ดูผลของแอปเพล็ตนี้ได้ด้วย

2.2.3 การทำงานแบบเทรด

2.2.3.1 ความหมายของเทรด

ในระบบปฏิบัติการแบบมัลติทาสกิ้งผู้ใช้ซึ่งอาจจะมีมากกว่าหนึ่งคน สามารถส่งโปรแกรมให้แก่ระบบปฏิบัติการทำงานมากกว่าหนึ่งโปรแกรมในเวลาหนึ่งได้ โดยโปรแกรมเหล่านี้จะไปเข้าคิวเพื่อรอเข้าทำงานในหน่วยประมวลผล บางระบบปฏิบัติการอาจกำหนดเวลาที่แต่ละการประมวลผลจะได้ทำงาน หากการประมวลผลใดใช้เวลาทำงานเกินเวลาที่กำหนดให้ ก็จะถูกหยุดและออกมาเข้าคิวรอรอบต่อไป และโปรแกรมที่รอคิวจะเข้าไปทำงานแทน ความเร็วของคอมพิวเตอร์ทำให้การทำงานลักษณะนี้ดูคล้ายกับว่าโปรแกรมเหล่านั้นทำงานไปพร้อมๆ กัน ทั้งที่จริงแล้วเป็นการจัดคิวให้เข้าทำงานทีละโปรแกรม โปรแกรมส่วนใหญ่มักต้องทำงานหลายอย่างไปพร้อมๆ กัน เช่น ในขณะหนึ่งอาจอ่านข้อมูลจาก disk พร้อมกับแสดงตัวอักษรที่จอภาพ หากงานทั้งสองนี้ถูกแยกกันทำเป็นสองการประมวลผล และทำเป็นเวลานาน จะเห็นการพิมพ์ตัวอักษรที่จอภาพสะดุดเป็นช่วงๆ เนื่องจากมีการสลับกันทำงานระหว่างการประมวลผลที่อ่านข้อมูลกับการประมวลผล ที่พิมพ์ตัวอักษร การเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผล เรียกว่า Context Switching ในขั้นตอนนี้ต้องมีการเก็บสถานะการคำนวณของการประมวลผลเดิมไว้ เพื่อให้สามารถกลับมาทำงานต่อจากจุดนั้นได้ และต้องโหลดสถานะการคำนวณของการประมวลผล ใหม่เข้าไปทำงานแทนที่ภาระการเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผลนี้เป็นโอเวอร์เฮด (Overhead) ที่ทำให้โปรแกรมงานทำงานช้าลง ผู้ออกแบบระบบปฏิบัติการและภาษาสำหรับโปรแกรมจึงหาวิธีทำให้ โอเวอร์เฮดของการเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผลลดลง ด้วยการเสนอกลไกใหม่ที่เรียกว่าเทรด

2.2.3.2 การทำงานของเทรด

ระบบปฏิบัติการแบบมัลติเทรดดิ้ง (Multi-Threading) จะสามารถแบ่งการประมวลผลออกเป็นหน่วยที่เล็กๆซึ่งเรียกว่าเทรด หรือบางทีเรียกว่า light-weight Process ระบบปฏิบัติการแบบนี้จะสามารถสร้างหลายเทรดในหนึ่งการประมวลผล โดยปกติเทรดหนึ่งคือ execution flow ของทำงานอย่างหนึ่ง หากนำหลายๆเทรดมารวมกันในการประมวลผลหนึ่ง จะช่วยให้การประมวลผลนั้นสามารถทำงานได้มากกว่าหนึ่งอย่างไปพร้อมๆ กันโดยไม่ต้องมีการทำการเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผลโปรแกรมจึงทำงานได้เร็ว และเป็นการใช้งานหน่วยประมวลผลได้อย่างมีประสิทธิภาพมากขึ้น

ในระบบมัลติทาสกิ้งโดยปกติหนึ่งการประมวลผลจะมีหนึ่ง PCB ส่วนในระบบมัลติเทรดดิ้งหนึ่งการประมวลผลมีหนึ่ง PCB เช่นกัน แต่อาจมีหลายเทรดเหล่านั้นมีส่วนที่เป็นโปรแกรม (code segment) ร่วมกันและใช้หน่วยความจำสำหรับเก็บข้อมูล (data segment) บางส่วนร่วมกัน หากเทรดในการประมวลผลเดียวกันใช้ตัวแปรบางตัวร่วมกัน เราเรียกตัวแปรเหล่านั้นว่า

shared variable และถ้าเทรดหนึ่งมีตัวแปรของมันเองที่ไม่ถูกใช้โดยเทรดอื่น ก็เรียกตัวแปรนั้นว่า local variable โดยปกติเทรดหนึ่งจะทำงานในบางส่วนของโปรแกรมเท่านั้น แต่ละเทรดจะมีโปรแกรมของตัวเองมีจุดเริ่มต้นและจุดสิ้นสุดแตกต่างกันไปเทรดแต่ละตัวมักจะทำงานเป็นอิสระต่อกันทำให้เทรดหลายๆ ตัวสามารถทำงานพร้อมกันได้ ถ้าเครื่องคอมพิวเตอร์นั้นมีหน่วยประมวลผลหลายตัวก็อาจแบ่งให้หน่วยประมวลผลแต่ละตัวทำงานแต่ละเทรดได้พร้อมกันแบบขนาน (Parallel) ซึ่งจะช่วยให้โปรแกรมทำงานเร็วขึ้น

การจัดการให้เทรดใดถูกทำงานเมื่อใด และโดยหน่วยประมวลใด นั้นเป็นหน้าที่ของระบบปฏิบัติการ ไม่อยู่ในการควบคุมของผู้เขียนโปรแกรมโดยตรง ผู้เขียนโปรแกรมทำได้แต่เพียงสร้าง เทรดให้ทำหน้าที่ต้อง และควบคุมให้เทรดเหล่านั้นเริ่มต้นทำงาน และหยุดการทำงาน ในระหว่างที่ เทรดทำงาน โปรแกรมยังสามารถออกคำสั่งให้เทรดนั้นหยุดรอ และเริ่มต้นทำงานเมื่อถึงเวลาที่กำหนดได้

ข้อแตกต่างระหว่างเทรดกับการประมวลผล (Process) พอสรุปได้ดังนี้คือ

1) Dependency แม้ว่าแต่ละเทรดของการประมวลผลเดียวกันจะทำงานที่แตกต่างกัน แต่มีความเกี่ยวข้องกันบางอย่างและต้องทำงานอยู่ภายในสภาพแวดล้อม (environment) เดียวกัน จึงอาจใช้ตัวแปรบางตัวร่วมกัน หรืออาจใช้โปรแกรมบางส่วนร่วมกัน แต่สำหรับการประมวลผลจะไม่มีความเกี่ยวข้องเลย เพราะแต่ละการประมวลผลจะมีสภาพแวดล้อมของตัวเอง

2) Communication การติดต่อระหว่างเทรดทำได้ง่าย โดยอาจใช้ shared variables แต่สำหรับการประมวลผลไม่มี shared variable เนื่องจาก เมื่อมีการทำการเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผลจะเปลี่ยนสภาพแวดล้อมของการประมวลผลด้วย วิธีเดียวที่การประมวลผลจะส่งค่าให้แก่กันได้คือทำผ่านทางไฟล์ เรียกว่า pipe ซึ่งเป็นวิธีที่สิ้นเปลืองมากและช้ากว่าการอ่านหรือกำหนดค่าตัวแปรอย่างมาก

3) Context Switching เมื่อการประมวลผลหนึ่งถูกนำเข้าทำงาน หากการประมวลผลนั้นมีมากกว่าหนึ่งเทรดก็จะมีการสลับกันทำงานระหว่างเทรดเหล่านั้น โดยที่ไม่ต้องเปลี่ยนสภาพแวดล้อมและ PCB ของการประมวลผลทำให้การสลับการทำงานระหว่างเทรดเป็นภาระต่อเครื่องน้อยกว่าการทำการเปลี่ยนการทำงานของการประมวลผลหนึ่งไปสู่อีกการประมวลผลของการเปลี่ยนการประมวลผลส่งผลให้เทรดเหล่านั้นมีเวลาทำงานมากขึ้น

2.2.3.3 เทรดในภาษาจาวา

ภาษาจาวาสนับสนุนการโปรแกรมมัลติเทรดดิ้งด้วยกลไกที่กำหนดขึ้นภายในภาษา ต่างกับภาษา C และ C++ ที่สนับสนุนเทรดด้วยไลบรารี (library) ที่เขียนขึ้นต่างหากสำหรับแต่ละคอมไพเลอร์และระบบปฏิบัติการ โปรแกรมภาษาจาวาทุกๆ โปรแกรมจะต้องมีเทรดอย่างน้อยหนึ่งเทรดเสมอ คอมไพเลอร์จะเป็นผู้สร้างเทรดให้แก่ main() ของทุกๆ โปรแกรม เรียกว่าเทรดหลัก

(main Thread) และจาวาอินเตอร์พรีเตอร์ จะเป็นผู้ควบคุมเทรดหลักที่หลังจากทำงานแล้วอาจจะมีการสร้างเทรดอื่นๆ ได้อีก

คลาสเทรดมี data members สำหรับเก็บค่าคุณสมบัติของเทรดดังต่อไปนี้

private char name [] ;

คือชื่อของเทรดเราสามารถตั้งชื่อเทรดโดยส่งสตริงเป็นพารามิเตอร์ให้แก่ คอนสตรักเตอร์ (constructor) หากไม่มีชื่อเทรดนั้นจะถูกตั้งชื่อให้โดยอัตโนมัติเป็น “Thread-1”+n โดย n เป็นเลขจำนวนเต็มที่เพิ่มขึ้นตามจำนวนเทรดที่ถูกสร้างขึ้น

private int priority ;

คือค่าระดับความสำคัญของเทรดในการถูกระบบปฏิบัติการเลือกไปทำงาน

private boolean daemon = false;

เป็นค่าที่ระบุว่าเทรดนั้นเป็นเดมอนเทรด (Daemon Thread) หรือไม่

หากเทรดหนึ่งถูกสร้างขึ้นและถูกเริ่มการทำงานด้วย start() แล้ว จะถือว่ามีชีวิต ซึ่งเมื่อเรียก isAlive() จะเป็น true จนกว่าเมื่อสิ้นสุดการทำงานแล้ว isAlive() จะเป็น false

2.2.3.4 **รายละเอียดและการทำงานของคลาสเทรด** (Class Thread)

คลาสเทรดถูกกำหนดใน package java.lang ใช้สำหรับสร้างอินสแตนซ์ซึ่งเราจะเรียกสั้นๆว่า เทรด คลาสเทรดมีโปรแกรมสำหรับการทำงานของเทรดตัวหนึ่ง ที่มีการดำเนินของโปรแกรมแยกออกไปจากโปรแกรมที่สร้างมันเทรดที่สร้างขึ้นจะยังไม่เริ่มต้นทำงานจนกว่า start() ของมันจะถูกเรียก เมื่อ start() ถูกเรียกแล้ว มันจะไปเข้าคิว จนเมื่อมันถูกเลือกไปทำงาน โปรแกรมใน run() ก็จะถูกทำงาน

คลาสเทรดมีเมธตอดสำหรับควบคุมให้เทรดเข้าสู่สถานต่างๆ เช่น start(), yield(), sleep (), suspend(), resume() และ stop() โดยปกติเราจะไม่โอเวอร์ไรด์ (override) เมธตอดเหล่านี้ นอกจากนี้ยังมี

public void run()

ซึ่งเป็นเมธตอดที่ว่างเปล่า ที่โดยปกติเราจะต้องโอเวอร์ไรด์พื่อระบุว่าเมื่อเทรดนั้นอยู่ในสถานะ running จะทำอะไรบ้าง หากไม่ทำการโอเวอร์ไรด์ run() ก็จะได้เทรดที่ไม่ทำอะไรเลย

2.2.3.5 **รายละเอียดและการทำงานของอินเตอร์เฟสรันเอเบิล** (Interface Runnable)

วิธีที่สองในการสร้างเทรดคือสร้างคลาสที่อิมพลีเมนต์รันเอเบิลอินเตอร์เฟส (implement Runnable interface) ในอินเตอร์เฟสนี้มีเพียง run() เป็น abstract method คลาสที่ อิมพลีเมนต์รันเอเบิลอินเตอร์เฟสจะต้องมี run() เพื่อระบุการทำงานของเทรดในสถานะ running

อินเตอร์เฟสรันเอเบิลถูกกำหนดไว้ใน java.lang ดังนี้

```
Public abstract interface Runnable {
        Public abstract void run();
    }
```

การใช้รันเอเบิลสร้างเทรดอาจทำได้สองวิธี

1) สร้างคลาสที่อิมพลีเมนต์รันเอเบิลโดยมีเทรดตัวหนึ่งเป็นสมาชิก

2) สร้างอินสแตนซ์ของคลาสที่อิมพลีเมนต์รันเอเบิลแล้วส่งให้เป็นพารามิเตอร์ของคอนสตรักเตอร์ของคลาสเทรด

ตัวอย่าง แสดงการสร้างคลาสที่อิมพลีเมนต์รันเอเบิลแล้วส่งให้เป็นพารามิเตอร์ของคอนสตรักเตอร์ของคลาสเทรด

```
// RunnableTest2.java
Class   MyThread implements Runnable  {
        public void run()  {
            For (int i = 0 ; i < 100 ; i++)
                System.out.print(Thread.currentThread( ).getName( )) ;
        }
}
Class   RunnableTest2  {
        public static void main (String args[ ]  {
            new Thread (new MyThread ( ) , “A” ) .start ( ) ;
            new Thread (new MyThread ( ) , “B”)  .start ( ) ;
        }
}
```

คลาส My Thread ในตัวอย่างนี้อิมพลีเมนต์รันเอเบิล แต่ไม่มีเทรด t เป็นสมาชิกเหมือนในตัวอย่างที่แล้ว ดังนั้นเมื่อจะเรียก getName() ต้องเรียก Thread.currentThread() ก่อนเพื่อได้รีเฟอร์เรนซ์ (reference) ของเทรดที่กำลังทำงานอยู่ ส่วนใน main() มีการสร้างอินสแตนซ์ของคลาสเทรดขึ้น โดยคอนสตรักเตอร์มีพารามิเตอร์ตัวแรกเป็นรันเอเบิลซึ่งมี run() ที่จะถูกทำงาน และตัวที่สองเป็นสตริงที่จะเป็นชื่อของเทรดนั้น

2.2.3.6 เมธตอดที่สำคัญของเทรดในภาษาจาวา

เมธตอดของ java.lang.Thread	
currentThread()	คืนค่ารีเฟอร์เรนซ์ของเทรดที่กำลัง execute
destroy()	ทำลายเทรดโดยไม่ clean up

getName()	คืนชื่อของเทรด
getPriority()	คืนค่าความสำคัญของเทรด
getThreadGroup()	คืนค่า ThreadGroup
isAlive()	เทรดตอยู่หรือไม่
isDaemon()	เทรดเป็นเดมอนเทรดหรือไม่
join(long millis)	คอยให้เทรดตาย
resume()	เรียกเทรดที่ถูก suspend() ให้กลับมาทำงาน
run()	จะถูกเรียกทำงานเมื่อเทรดเข้าสู่สถานะ ready
setDaemon()	กำหนดให้เทรดเป็นเดมอนเทรด
setName(String name)	กำหนดชื่อเทรด
setPriority()	กำหนดความสำคัญของเทรด
sleep(long millis)	สั่งให้เทรดหลับ
start()	สั่งให้เทรดเริ่มต้นทำงาน JVM จะเรียก run() มาทำงานให้เอง
stop()	สั่งให้เทรดหยุดทำงาน อาจจะก่อให้เกิดเอกเซฟชัน
suspend()	สั่งให้เทรดหยุดการทำงานชั่วคราว
yield()	สั่งให้เทรดเปลี่ยนสถานะจาก running เป็น ready
Methods inherited from class java.lang.Object	
wait()	สั่งให้เทรดเข้าสู่สถานะบล็อค(blocked state)
notify()	ใช้เรียกให้เทรดอื่นที่ถูก wait() กลับสู่สถานะ ready
notifyAll()	สั่งให้ทุกเทรดที่ถูก wait มีโอกาสถูกเลือกเข้าทำงานเท่าๆกัน

2.2.3.7 วิธีการใช้ฟังก์ชันเดมอนเทรด (Daemon Threads)

มีหลายกรณีที่เทรดหนึ่งต้องสร้างเทรดลูกขึ้นมาเพื่อคอยสนับสนุนการทำงานของมัน โดยปกติเทรดลูกพวกนี้จะถูกสร้างขึ้นมาในระหว่างเทรดแม่ทำงาน และจะทำงานร่วมกับเทรดแม่ไปจนเมื่อเทรดแม่จะสิ้นสุดการทำงานก็สั่ง stop() เทรดเหล่านั้น แต่ในบางกรณีอาจยุ่งยากที่จะเก็บ รีเฟอร์เรนซ์ของเทรดลูกทั้งหมดที่สร้างขึ้น และ การตรวจสอบว่าเทรดใดตายไปแล้วหรือยัง หรือจะมีปัญหาถ้าเทรดแม่จบการทำงานอย่างไม่ปกติโดยไม่มีโอกาสได้เรียก stop() ของเทรดลูกที่มันสร้างขึ้น เราเรียกเทรดลูกที่ยังทำงานอยู่ แต่เทรดแม่ตายของมันไปแล้วว่า orphan Threads ซึ่งเทรดแบบนี้มักจะใช้ทรัพยากรเครื่องอย่างไม่ก่อให้เกิดประโยชน์

โดยปกติ หากยังมีเทรดใดเทรดหนึ่งยังทำงานอยู่ JVM ก็จะปล่อยให้ทำงานต่อไปเรื่อยๆ แต่ ภาษาจาวาเสนอเทรดพิเศษขึ้นเรียกว่าเดมอนเทรดหาก JVM พบว่าเทรดที่ยังทำงานอยู่นั้นมีแต่ เดมอนเทรดเท่านั้น ไม่มีเทรดธรรมดาทำงานอยู่เลย มันก็จะหยุดตัวเองลง ส่งผลให้เดมอนเทรดเหล่านั้นตายโดยอัตโนมัติ ดังนั้นหากเทรดแม่ตัวหนึ่งมีเทรดลูกที่ทำงานสนับสนุนเป็นเดมอนเทรดทั้งหมด สมมติว่ามีเพียงเทรดแม่ตัวเดียวที่เป็นเทรดปกติเมื่อ เทรดแม่ตายเทรดลูกทั้งหมดก็จะตายลงเองด้วย

2.2.3.8 **วิธีการใช้ฟังก์ชันเรซิ่งคอนดิชัน** (Racing Condition)

หากมีเทรดตั้งแต่สองตัวขึ้นไป แย่งกันใช้ทรัพยากรหนึ่งอย่างไม่ถูกควบคุมก็อาจจะเกิดปัญหาขึ้นได้ เนื่องจากลำดับการใช้งานทรัพยากรนั้นของแต่ละเทรดอาจมีความสำคัญต่อผลลัพธ์ของโปรแกรมในบางกรณีอาจทำให้ โปแกรมเดียวกันให้ผลลัพธ์แตกต่างกันในการทำงานแต่ละครั้ง ปรากฏการณ์นี้เรียกว่าเรซิ่งคอนดิชันของเทรดในโปรแกรมนั้น

2.2.3.9 **วิธีการใช้ฟังก์ชันซิงโครไนเซชัน** (Synchronization)

ภาษาจาวามีกลไกสำหรับการควบคุมให้ที่เวลาหนึ่ง ทรัพยากรนั้นถูกใช้งานได้โดยเทรดเดียวเท่านั้น โดยนำกลไกมูทัวลี่เอกซ์คลูซีฟ (mutually exclusive) ของระบบปฏิบัติการมาใช้ นั่นคือ หากนำมูทัวลี่เอกซ์คลูซีฟไปครอบไว้กับทรัพยากรใด จะจำกัดให้ที่เวลาหนึ่งมีเพียงเทรดเดียวเท่านั้นที่สามารถเข้าครอบครองและใช้งานทรัพยากรนั้น ระหว่างที่ทรัพยากรนั้นมีเทรดหนึ่งครอบครองอยู่ หากมีเทรดอื่นต้องการใช้งานทรัพยากรนั้นก็จะหยุดรอจนกว่าเทรดที่ใช้งานอยู่จะยอมปล่อยทรัพยากรนั้นออกจากการครอบครองได้เพียงผู้เดียวในเวลาใดเวลาหนึ่งว่ามูทัวลี่เอกซ์คลูซีฟ

ภาษาจาวาสามารถทำให้เกิดมูทัวลี่เอกซ์คลูซีฟขึ้นกับอินสแตนซ์หนึ่งได้ หากคลาสของอินสแตนซ์นั้นมีเมธตอดหนึ่งที่ถูกระบุด้วย keyword “synchronized” และเมื่อเมธตอดนั้นถูกเรียกจากเทรดใดๆก็ตาม จะทำให้ที่เวลาหนึ่งมีเพียงเทรดเดียวเท่านั้นที่ทำงานเมธตอดนั้นได้

สังเกตว่า เฉพาะสมาชิกที่เป็นเมธตอดเท่านั้น จึงจะมี keyword “synchronized” ได้ ไม่อาจใช้กับสมาชิกที่เป็น data members

เมื่อเทรดหนึ่งเรียกซิงน์โครไนซ์เมธตอดหากตอนนั้นยังไม่มีเทรดใดทำงานเมธตอดนั้น เทรดนั้นก็จะได้ทำงานหรือเรียกว่าครอบครอง แต่หากในระหว่างนั้นมีเทรดอื่นกำลังทำงานเมธตอด นั้นอยู่เทรดที่พยายามเรียกก็จะต้องรอจนกว่าเทรดที่ทำงานอยู่จะสิ้นสุดการทำงานในเมธตอดนั้น

2.2.3.10 **วิธีการใช้ฟังก์ชันเดดล็อค** (Deadlock)

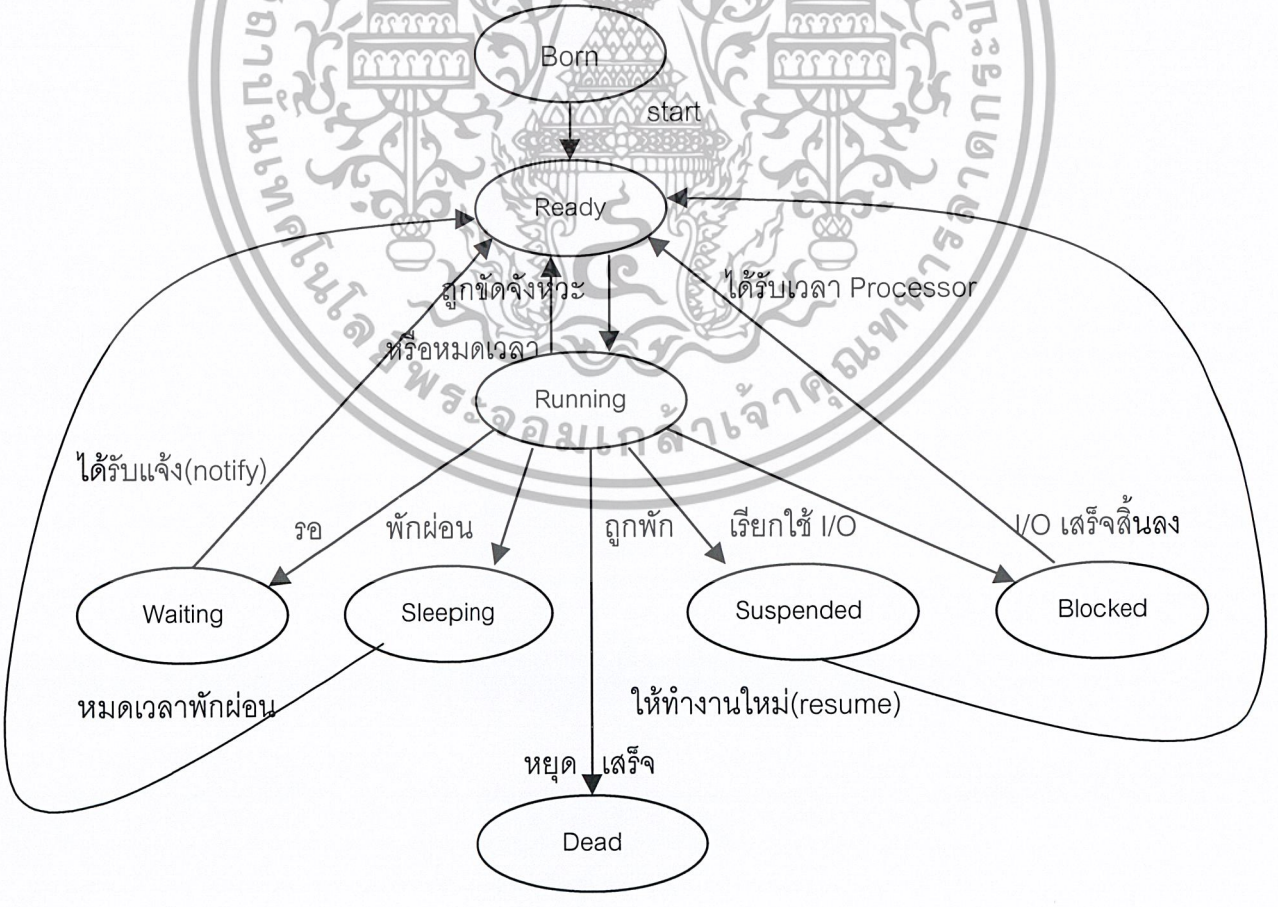
หากมีเทรดหนึ่งต้องรอให้อีกเทรดหนึ่งทำงานสิ้นสุดลงเสียก่อนจึงจะทำงานต่อไปได้ เราเรียกว่าเทรดนั้นขึ้นกับ (depend on) อีกเทรดหนึ่ง โปรแกรมมัลติเทรดดิ้งที่มีการขึ้นต่อกันระหว่างเทรดอาจมีปัญหาเกิดขึ้นในระหว่างทำงานได้ เช่นหากเทรดหนึ่งได้ครอบครองทรัพยากรที่เป็น

มูทัวลี่เอกซ์คลูซีฟ และถ้าเทรดนั้นไม่สามารถจบการทำงานลงได้เทรดอื่นๆที่รอใช้ทรัพยากรนั้นก็จะไม่สามารถทำงานได้ ปัญหาแบบนี้ถือเป็นความผิดพลาดของการสร้างโปรแกรมที่พอตรวจสอบและแก้ไขได้ แต่ก็มีปัญหาอีกแบบหนึ่งที่ตรวจสอบและแก้ไขได้ยากกว่าคือ ปัญหาเดดล็อคซึ่งเกิดขึ้นกับเทรดมากกว่าหนึ่งตัวขึ้นไปที่ไม่สามารถทำงานต่อไปได้ โดยจะอยู่ในสถานะบล็อคไปอย่างถาวร

เดดล็อคจะเกิดขึ้นได้เมื่อเทรดอย่างน้อยสองเทรดสมมติว่าเป็น Aและ B โดยที่ เทรด A ได้ครอบครองทรัพยากร X ที่เป็นมูทัวลี่เอกซ์คลูซีฟและ เทรด B ได้ครอบครองทรัพยากร Y ที่เป็นมูทัวลี่เอกซ์คลูซีฟ เช่นกัน แต่ใจขณะนั้น เทรด A ต้องการใช้ทรัพยากร Y และ เทรด B ก็ต้องการใช้ทรัพยากร X ทำให้เทรดทั้งสองต้องเข้าสู่สถานะบล็อคเพื่อรอให้ทรัพยากร ที่รอใช้งานถูกปล่อยออกมาจึงจะทำงานต่อไปได้ แต่ทั้ง เทรด A และ B จะไม่ปล่อยทรัพยากรที่ตัวครอบครองอยู่เนื่องจากยังทำงานไม่เสร็จ นั่นคือเทรดทั้งสองกำลังรอคอยสิ่งที่จะไม่เกิดขึ้น จึงต้องอยู่ในสถานะบล็อคตลอดไป

2.2.3.11 **วัฏจักรของเทรด**

วัฏจักร (lifetime) ของเทรดแสดงได้เป็นรูปดังรูปนี้



รูปที่ 2.79 แสดงวัฏจักรของเทรด

ในเวลาต่างๆเทรดจะอยู่ในสภาวะต่างๆ (Thread States) เริ่มจากสภาวะเริ่มต้น (Born State) เทรดจะอยู่ในสภาวะดังกล่าวจนกว่าเมธตอด Start ของเทรดดังกล่าวจะถูกเรียก เมื่อเรียกเมธตอด Start แล้วเทรดจะอยู่ในสภาวะพร้อม (Ready State) เทรดซึ่งมีความสำคัญสูงสุดจะเข้าไปอยู่ในสภาวะทำงาน (Running State) เมื่อได้รับเวลาจากหน่วยประมวลผล (Processor) เทรดจะหมดสภาพหรือตาย (Dead State) เมื่อมันทำงานเสร็จสมบูรณ์ หรือถูกเรียกด้วยเมธตอด stop เทรดที่ตายแล้วจะถูกกำจัดโดยระบบ

บางครั้งเทรดอาจอยู่ในสภาวะบล็อกเมื่อเทรดมีการเรียกใช้ Input/Output Block เทรดจะกลายสภาพเป็นสภาวะพร้อมเมื่อการทำงานของ I/O ที่รออยู่สิ้นสุดลงเทรดที่อยู่ในสภาวะบล็อกไม่สามารถใช้เวลาของหน่วยประมวลผล ถึงแม้หน่วยประมวลผลจะว่างอยู่ก็ตาม

เทรดจะอยู่ในสภาวะพักผ่อนเมื่อถูกเรียกด้วยเมธตอด sleep เทรดที่พักผ่อนจะกลับมาอยู่ในสภาวะพร้อมอีกครั้งเมื่อหมดเวลาพักผ่อนเทรดที่อยู่ในสภาวะพักผ่อนจะไม่สามารถใช้เวลาของหน่วยประมวลผลถึงแม้หน่วยประมวลผลจะว่างอยู่ก็ตาม

เมื่อเทรดที่ทำงานอยู่ถูกเรียกด้วยเมธตอด suspend จะกลายสภาพเป็นถูกพัก (Suspend State) เทรดที่ถูกพักจะกลายสภาพมาเป็นสภาวะพร้อมอีกครั้ง เมื่อถูกเรียกด้วยเมธตอด notify จากเทรดอื่นที่ทำงานกับวัตถุนั้นเทรดทุกอันจะเปลี่ยนสภาพเป็นพร้อมเมื่อถูกเรียกด้วยเมธตอด notifyAll

2.2.3.12 **ลำดับความสำคัญของเทรดและการจัดลำดับงาน** (Thread Priority and Thread Schedulling)

ทุกจาวาแอปพลิเคชั่นและจาวาแอปเพล็ตเป็นมัลติเทรด เทรดทุกตัวในจาวามีความสำคัญในช่วง Thread.MIN_PRIORITY และ Thread.MAX_PRIORITY ถ้าไม่มีการกำหนดค่าให้จาวาจะกำหนดให้มีความสำคัญเป็น Thread.NORM_PRIORITY

จาวาในบางระบบปฏิบัติการใช้แนวคิดไทม์สไลซิงค์ (Timeslicing) แต่บางระบบก็ไม่ใช้ กรณีที่ไม่ใช้ไทม์สไลซิงค์แต่ละเทรดในชุดของเทรดที่มีลำดับความสำคัญเท่ากันจะทำงานจนเสร็จสมบูรณ์เทรดอื่นจึงจะมีโอกาสทำงาน แต่ในไทม์สไลซิงค์แต่ละเทรดจะได้รับเวลาควอนตัม (Time Quantum) จำนวนหนึ่งเมื่อใช้เวลาดังกล่าวหมดแม้งานจะยังไม่เสร็จหน่วยประมวลผลจะถูกเรียกกลับไปทำงานให้กับ เทรดอื่นซึ่งมีความสำคัญเท่ากัน (ถ้ามี) ต่อไป

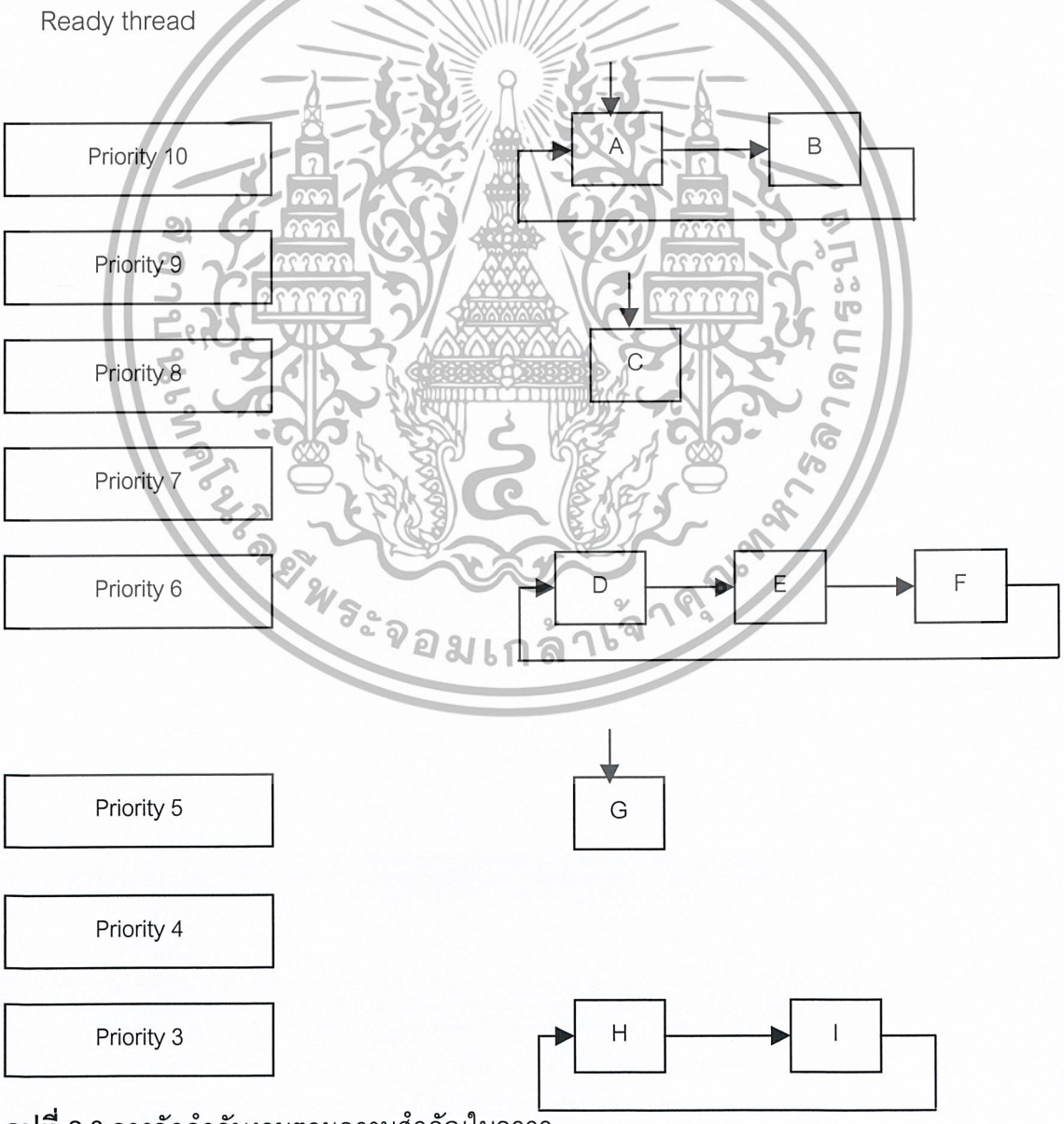
งานของตัวจัดลำดับงานของจาวา ก็คือ ให้เทรดที่มีความสำคัญทำงานก่อน และถ้าเทรดที่มีความสำคัญเท่ากันมีหลายตัว ก็จัดให้เทรดเหล่านั้นทำงานในลักษณะวนรอบ (Round-Robin)

ลำดับความสำคัญของเทรดสามารถเปลี่ยนแปลงได้ โดยใช่เมธตอด setPriority ซึ่งรับค่าจำนวนเต็มที่บอกถึงลำดับความสำคัญของเทรด ค่าจำนวนเต็มดังกล่าวต้องอยู่ในช่วง 1-10 ถ้าค่า

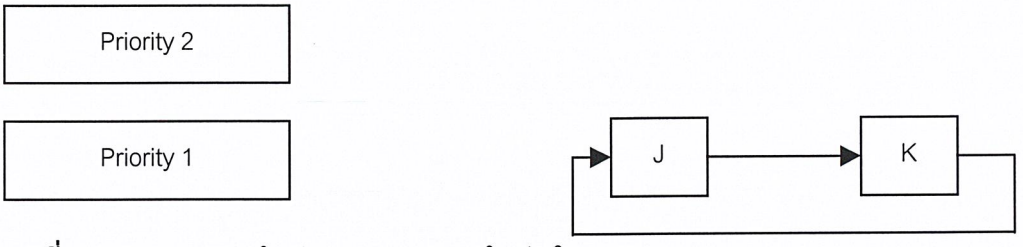
ดังกล่าวไม่อยู่ในช่วงเมธตอด setPriority จะ Throw IllegalArgumentException เมธตอด getPriority ส่งคืนค่าจำนวนเต็มแทนความสำคัญของเทรดนั้นๆ

เทรดสามารถเรียกเมธตอด yield เพื่อให้โอกาสเทรดอื่นได้ทำงาน ปกติเมื่อเทรดที่มีความสำคัญสูงกว่าพร้อมจะแย่งเวลาหน่วยประมวลผลไปจากเทรดปัจจุบันเมธตอด yield ให้โอกาสแก่เทรดที่มีความสำคัญเท่ากันในการทำงาน ในกรณีที่เป็นระบบไทม์สไลซิงค์เราไม่จำเป็นต้องใช้ เมธตอด yield เนื่องจากเทรดแต่ละเทรดมีความสำคัญเท่ากันเวียนกันทำงานด้วยเวลาควอนตัมของตนเอง

เทรดจะทำงานจนกระทั่งมันตาย ถูกบล็อกรอ I/O sleep, wait, yield หรือถูกแย่งจากเทรดที่มีความสำคัญสูงกว่า



รูปที่ 2.8 การจัดลำดับงานตามความสำคัญในจาวา



รูปที่ 2.8(ต่อ) การจัดลำดับงานตามความสำคัญในจาวา

2.3 โครงสร้างเว็บแอปพลิเคชัน (Web Application)

2.3.1 เว็บเซิร์ฟเวอร์ และเว็บบราวเซอร์

เว็บเซิร์ฟเวอร์ เป็น โปรแกรมที่อยู่ในเครื่องเซิร์ฟเวอร์ทำหน้าที่ในการรับคำร้องขอ และทำการประมวลผลแล้วส่งข้อมูลเว็บไปให้เครื่องฝั่งไคล์เอนท์ (Client) ถ้าจะให้เข้าใจได้ง่ายขึ้นกว่านี้อีก ก็คือโปรแกรมที่ใช้ในการให้บริการนั้นเอง

เว็บบราวเซอร์ เป็นโปรแกรมที่อยู่ในเครื่องฝั่งไคล์เอนท์มีหน้าที่ในการส่งข้อมูลร้องขอดูเว็บและนำเสนอข้อมูลเว็บ โดยตัวเว็บบราวเซอร์จะมีความเข้าใจในภาษามาตรฐานของเว็บก็คือภาษา HTML

2.3.2 ภาษา HTML (Hypertext Markup Language)

HTML เป็นรูปแบบมาตรฐานในการสร้างหน้าเว็บ โดยที่ภาษา HTML เป็นภาษาที่ง่ายต่อการเรียนรู้และเข้าใจ ในการสร้างหน้าเว็บ โดยใช้ HTML ได้ โดยมีพื้นฐานที่อยู่ในรูปแท็ก <...> โดยมีลักษณะการใช้แท็ก 2 แบบดังนี้

1) แท็กมีจุดเริ่มต้นและสิ้นสุด

2) แท็กเดี่ยวๆ

ตัวอย่างแท็ก HTML ที่จะต้องมีจุดเริ่มต้นและสิ้นสุด เช่น และ เป็นการทำให้ตัวอักษรเป็นตัวหนา ถ้าเราไม่ปิดด้วย ทั้งเอกสารจะกลายเป็นตัวหนาทั้งหมด เพราะไม่มี ปิด และโค้ด HTML ที่เป็นคำสั่งเดี่ยวๆก็คือ เขียนโค้ดไปแล้วไม่ต้องเขียนจุดสิ้นสุด เช่น <hr>

ในการเขียน HTML นั้นมีหลักการดังนี้

1) เริ่มต้นไฟล์จะต้องเขียนแท็ก <html> และท้ายไฟล์จะต้องปิดด้วย </html>

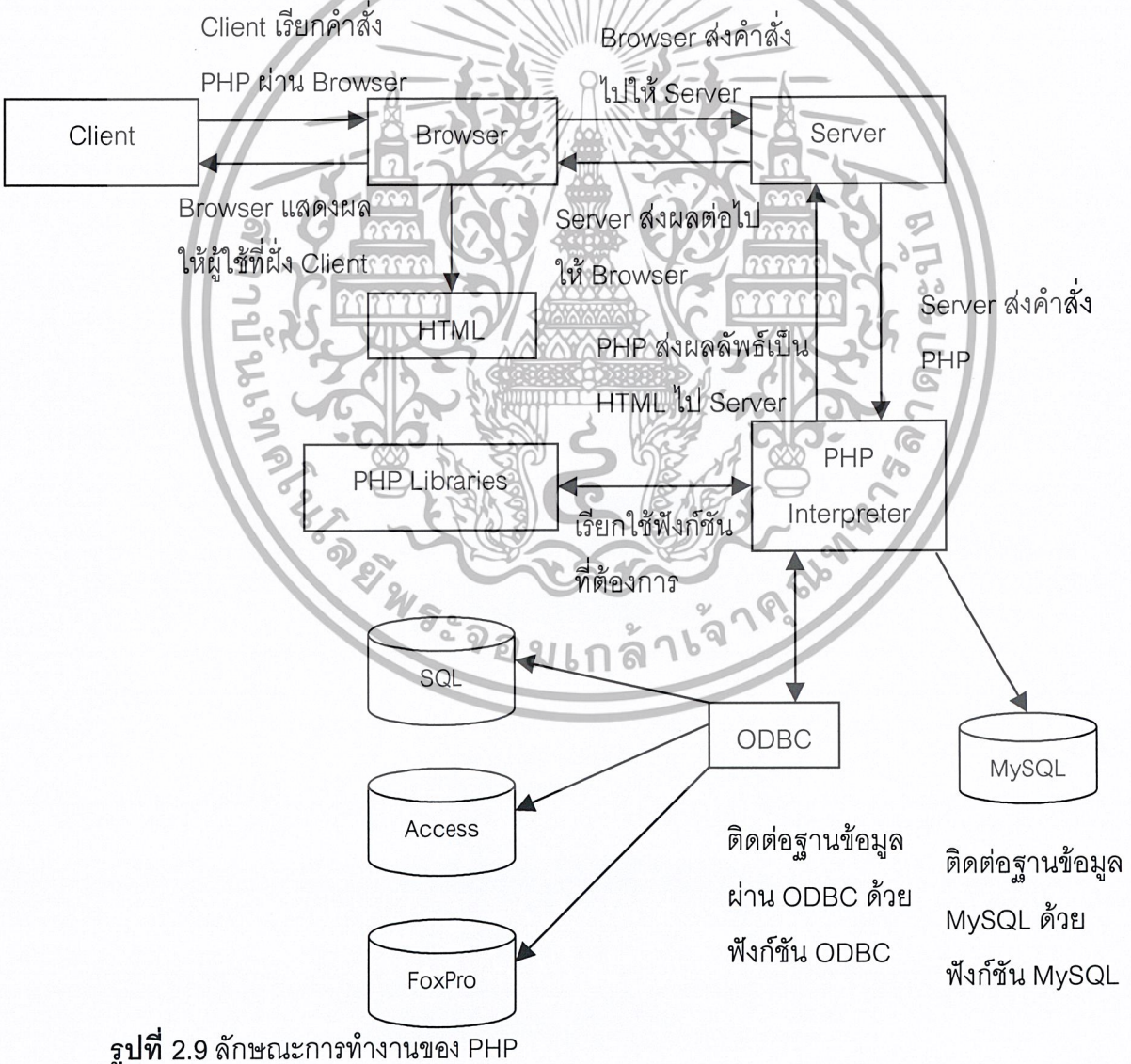
2) ต่อจาก <html> จะต้องเป็น <head>...</head> ซึ่งภายในแท็ก <head> จะใส่แท็กที่เป็นชื่อเรื่อง ซึ่งเป็นคำสั่งย่อยอีก 1 คำสั่ง คือ <title>...</title>

3) ต่อจาก <head>...</head> จะต้องเป็น <body>...</body> เราจะเขียนสิ่งที่ต้องการนำเสนอลงไประหว่างแท็ก <body>...</body> นี้

4) หากต้องการใส่หมายเหตุซึ่งจะไม่แสดงผลบนเว็บบราวเซอร์เขียนลงในระหว่างแท็ก <comment> ... <comment> หรือ <!-comment>

2.3.3 ภาษา PHP (PHP Hypertext Preprocessor)

เนื่องจาก PHP โดยมีตัวแปลและเอกซิคิวที่ฝั่งเซิร์ฟเวอร์อาจจะเรียกการทำงานว่าเป็น Server Side ส่วนการทำงานของบราวเซอร์ของผู้ใช้เรียกว่า Client Side โดยการทำงานจะเริ่มต้นที่ผู้ใช้ส่งความต้องการผ่านเว็บบราวเซอร์ทาง HTTP เมื่อเอกสาร PHP เข้ามาถึงเว็บเซิร์ฟเวอร์ก็จะถูกส่งไปให้ PHP เพื่อทำหน้าที่แปลคำสั่งและเอกซิคิวคำสั่งนั้น หลังจากนั้น PHP จะสร้างผลลัพธ์ในรูปแบบเอกสาร HTML ส่งกลับไปให้เว็บเซิร์ฟเวอร์เพื่อส่งต่อไปให้บราวเซอร์แสดงผลทางฝั่งผู้ใช้ต่อไป ซึ่งลักษณะการทำงานแบบนี้จะคล้ายกับการทำงานของ CGI (Commom Gateway Interface) หรืออาจจะกล่าวได้ว่า PHP ก็คือโปรแกรม CGI ประเภทหนึ่งก็ได้ ลักษณะการทำงานจะเป็นดังรูป



รูปที่ 2.9 ลักษณะการทำงานของ PHP

2.3.4 โปรแกรมฐานข้อมูล MySQL

MySQL เป็นฐานข้อมูลในลักษณะ DataBase Server แบบ open source ที่ได้รับความนิยมในการใช้งานสูงสุดโปรแกรมหนึ่งบนเครื่องให้บริการ มีความสามารถในการจัดการกับฐานข้อมูลด้วยภาษา SQL (Structures Query Language) อย่างมีประสิทธิภาพ มีความรวดเร็วในการทำงาน รองรับการทำงานจากผู้ใช้หลายๆ คนและหลายๆ งานได้ในขณะเดียวกัน น่าเชื่อถือและใช้งานได้ง่าย นอกจากนั้น MySQL ถูกออกแบบและพัฒนาขึ้นมาเพื่อทำหน้าเป็นเครื่องให้บริการรองรับการจัดการกับฐานข้อมูลขนาดใหญ่ ซึ่งการพัฒนายังคงดำเนินอยู่อย่างต่อเนื่อง ส่งผลให้มีฟังก์ชันการทำงานใหม่ๆ ที่อำนวยความสะดวกแก่ผู้ใช้งานเพิ่มขึ้นอยู่ตลอดเวลา รวมไปถึงการปรับปรุงด้านความต่อเนื่อง ความเร็วในการทำงาน และความปลอดภัย ทำให้ MySQL เหมาะสมต่อการนำไปใช้งานเพื่อเข้าถึงฐานข้อมูลบนเครือข่ายอินเทอร์เน็ต การใช้งานฐานข้อมูลทำได้ 2 วิธีคือ การเข้าใช้ฐานข้อมูลโดยตรงผ่านโปรแกรม MySQL และการใช้งานผ่านโปรแกรมที่เขียนขึ้นเพื่อใช้ติดต่อฐานข้อมูล เช่น โปรแกรมที่ถูกเขียนขึ้นด้วยภาษา PHP เป็นต้น ผู้ที่จะเข้าใช้งานฐานข้อมูลได้จะต้องได้รับการตรวจสอบสิทธิ์และพิสูจน์ตัวตนผู้ใช้ ซึ่งบัญชีรายชื่อผู้ใช้ของโปรแกรม MySQL นี้แยกจากบัญชีผู้ใช้งานของระบบโดยเด็ดขาด ไม่มีความเกี่ยวข้องกันแต่อย่างใด โดยจะถูกจัดเก็บและจัดการผ่านฐานข้อมูลของ MySQL ที่ใช้งาน

ในการใช้งานฐานข้อมูล MySQL จะต้องทำการสตาร์ทฐานข้อมูลก่อนทุกครั้งและเมื่อใช้งานเสร็จจะต้องทำการปิดฐานข้อมูล

2.3.4.1 ความสามารถและการทำงานของโปรแกรม MySQL

- MySQL ถือเป็นระบบจัดการฐานข้อมูล (DataBase Management System (DBMS)) ฐานข้อมูลมีลักษณะเป็นโครงสร้างของการเก็บรวบรวมข้อมูล การที่จะเพิ่มเติม เข้าถึงหรือประมวลผลข้อมูลที่เก็บในฐานข้อมูลจำเป็นจะต้องอาศัยระบบจัดการฐานข้อมูล ซึ่งจะทำหน้าที่เป็นตัวกลางในการจัดการกับข้อมูลในฐานข้อมูลทั้งสำหรับการใช้งานเฉพาะ และรองรับการทำงานของแอพลิเคชันอื่นๆ ที่ต้องการใช้งานข้อมูลในฐานข้อมูล เพื่อให้ได้รับความสะดวกในการจัดการกับข้อมูลจำนวนมาก MySQL ทำหน้าที่เป็นทั้งตัวฐานข้อมูลและระบบจัดการฐานข้อมูล

- MySQL เป็นระบบจัดการฐานข้อมูลแบบrelational ฐานข้อมูลแบบ relational จะทำการเก็บข้อมูลทั้งหมดในรูปแบบของตารางแทนการเก็บข้อมูลทั้งหมดลงในไฟล์เพียงไฟล์เดียว ทำให้ทำงานได้รวดเร็วและมีความยืดหยุ่น นอกจากนั้น แต่ละตารางที่เก็บข้อมูลสามารถเชื่อมโยงเข้าหากันทำให้สามารถรวมหรือจัดกลุ่มข้อมูลได้ตามต้องการ โดยอาศัยภาษา SQL ที่เป็นส่วนหนึ่งของโปรแกรม MySQL ซึ่งเป็นภาษามาตรฐานในการเข้าถึงฐานข้อมูล

- MySQL แจกจ่ายให้ใช้งานแบบ open source ผู้ใช้งาน MySQL ทุกคนสามารถใช้งานและปรับแต่งการทำงานได้ตามต้องการ สามารถดาวน์โหลดโปรแกรม MySQL ได้จากอินเทอร์เน็ตและนำมาใช้งานโดยไม่มีค่าใช้จ่ายใดๆ

2.3.4.2 **คำสั่งเบื้องต้นสำหรับ** MySQL

- help ดูระบบช่วยเหลือ
- status แสดงสถานะของ MySQL เช่น เวอร์ชัน,ฐานข้อมูลที่ใช้อยู่
- exit ออกจาก MySQL
- quit ออกจาก MySQL
- use [dbname] ใช้ฐานข้อมูล
- create database [dbname] สร้างฐานข้อมูลใหม่
- create table [tbname] (
  [field] [type] [option],
  ...
  [field] [type] [option],
  PRIMARY KEY (field)
  ); สร้างตารางใหม่
- show datadases แสดงฐานข้อมูลที่มีอยู่ใน MySQL
- show tables แสดงตารางที่มีอยู่ในฐานข้อมูลปัจจุบันที่ใช้อยู่
- select เลือกฟิลด์ที่จะแสดงผลข้อมูล
- insert into [tbname] เพิ่มข้อมูลเข้าสู่ตารางที่กำหนด
- delete form ลบข้อมูลออกจากตารางตามเงื่อนไข
- load data local infile โหลดข้อมูลจากเท็กซ์ไฟล์เข้าสู่ตาราง

2.3.4.3 **ฟังก์ชั่นที่ใช้ติดต่อกับฐานข้อมูล** MySQL

- mysql_connection() สำหรับเริ่มการติดต่อกับเซิร์ฟเวอร์ของ MySQL ซึ่งต้องใช้ชื่อโฮสต์ ชื่อผู้ใช้ และรหัสผ่าน
- mysql_select_db() สำหรับเลือกฐานข้อมูลบนเซิร์ฟเวอร์ของ MySQL
- mysql_query() สำหรับรันคิวรีจากคำสั่ง SQL ที่สร้างขึ้น
- mysql_fetch_array() สำหรับเก็บเรกคอร์ดผลลัพธ์จากคำสั่ง SQL ไว้ในอาร์เรย์
- mysql_free_result() สำหรับปล่อยให้รีซอร์สเป็นอิสระจากการติดต่อ
- mysql_close() สำหรับปิดการติดต่อที่กำลังดำเนินอยู่ปัจจุบัน

2.3.5 ภาษา XML (Extensive Markup Language)

XML (Extensive Markup Language) XML เป็นภาษา markup ที่เป็น text-based ที่ใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูลบนอินเตอร์เนทในปัจจุบัน โดยมีผู้ที่ทำหน้าที่รับผิดชอบและกำหนดมาตรฐานของ XMLคือ องค์การ W3C และต้องมีคุณสมบัติที่สำคัญดังต่อไปนี้

- จุดประสงค์ในการออกแบบXMLมาคือเพื่อเป็นมาตรฐานในการทำเอกสารบนเครือข่ายอินเตอร์เนทเพื่อการแลกเปลี่ยนข้อมูลที่เป็นลำดับขั้น เพราะฉะนั้น XMLต้องมีรูปแบบที่สามารถใช้ได้ทั่วไปบนinternet
- XMLมีคุณลักษณะ platform independence สามารถนำไปใช้กับคอมพิวเตอร์ระบบใด Platform ไหนก็ได้ เนื่องจากเอกสาร XML เป็น Text file ธรรมดา
- XML เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถนิยามความหมายของข้อมูลได้ จึงมีการจัดโครงสร้างข้อมูล แบ่งข้อมูลออกเป็นหมวดหมู่และส่วนประกอบย่อย
- ไม่มี tag ที่ถูกนิยามไว้ก่อน อนุญาตให้ผู้ใช้สร้างแท็กขึ้นมาเอง เพื่อใช้อธิบายข้อมูลได้ โดยที่ผู้ใช้กำหนดแท็กและใช้งานได้ทันที เนื่องจากแท็กเป็นอะไรก็ได้ที่ผู้ใช้กำหนดจึงทำให้ XML ขยายขีดความสามารถต่อไปได้ (Extensible) และข้อมูลสามารถอธิบายความหมายข้อมูลของตัวมันเองได้ (Self describe)
- ส่วนข้อมูลและส่วนการแสดงผลของเอกสาร XML ถูกแยกออกจากกันอย่างชัดเจน ในเอกสาร XML นั้น จะมีแต่ตัวเนื้อข้อมูล ส่วนการแสดงผลนั้น เราสามารถใช้ Style sheet ได้หลายประเภท เช่น CSS XSL เป็นต้น
- เนื่องจากเอกสาร XML เป็นเพียงข้อความง่ายๆที่ประกอบด้วยแท็กบางอย่างเท่านั้นจึงสามารถสร้างเอกสาร XML ด้วย text editor ทั่วๆไปได้
- การอ่านและแปรความหมายของเอกสาร XMLสามารถทำได้โดยใช้โปรแกรมที่เรียกว่า XML parser ได้ ตัวอย่างของโปรแกรมประเภท XML parser เช่น MSXML ซึ่งอยู่ในอินเตอร์เนท Explorer ของ Microsoft และ JAXP ของบริษัท Sun เป็นต้น
- XML มีวิธีกำหนดโครงสร้างเอกสาร 2วิธีคือ DTD และ XML Schema ซึ่งไม่ได้บังคับว่าจำเป็นต้องมี File กำหนดโครงสร้างเอกสาร แต่ถ้ามี และเอกสาร XML มีรูปแบบถูกต้องตาม DTD หรือ XML schema จะถือว่าเอกสาร XMLนั้นมีคุณสมบัติ Valid
- XML มีความกะทัดรัด เข้าใจง่ายและใช้ประโยชน์ได้กว้างขวาง
- XMLสามารถใช้ได้หลายภาษาผสมกัน เนื่องจาก XML สนับสนุน UNICODE
- XMLได้รับการสนับสนุนในโปรแกรมระบบฐานข้อมูลหลายๆค่าย สามารถดึงข้อมูลจากฐานข้อมูลมาอยู่ในรูปของ XML ได้

Richard Baldwin นิยามความหมายของ XML ไว้ดังนี้ "XML ทำให้ผู้ใช้สามารถสร้างและดูแลเอกสารที่มีโครงสร้าง (Structured documents) ที่บรรจุตัวอักษร (Plain text) โดยทำให้สามารถ Rendered หรือปรับเปลี่ยนการแสดงผลในรูปแบบที่หลากหลาย จุดประสงค์หลักของ XML คือการแยกส่วน ข้อมูลเพื่อประโยชน์ในการแสดงผล"

W3C ได้กล่าวถึงจุดมุ่งหมายหลักไว้ 10 หัวข้อ ดังนี้

1) XML มีการใช้งานโดยตรงบนเครือข่ายอินเตอร์เนท XML จะถูกออกแบบมาสำหรับจัดเก็บและจัดส่งข้อมูลบนเว็บ

2) XML มีการสนับสนุนโปรแกรมที่หลากหลาย ถึงแม้ว่าวัตถุประสงค์ที่สำคัญคือ การจัดส่งข้อมูลบนเว็บผ่านทางเซิร์ฟเวอร์และโปรแกรมบราวเซอร์ XML จะถูกออกแบบมาเพื่อใช้กับโปรแกรมที่มีรูปแบบต่าง ๆ ตัวอย่างเช่น การแลกเปลี่ยนข้อมูลระหว่างโปรแกรมทางด้านการเงิน การเผยแพร่และปรับปรุงโปรแกรมให้ทันสมัยและการเขียน Voice Script ให้สื่อสารได้ด้วยโทรศัพท์

3) XML จะต้องเข้ากันได้กับ SGML

4) XML จะต้องง่ายต่อการเขียนโปรแกรมเพื่อประมวลผลเอกสาร

5. จำนวนของทางเลือกเฉพาะของ XML ควรมีจำนวนน้อยที่สุดหรือไม่ควรมีเลย

6) เอกสาร XML จะต้องอ่านเข้าใจง่ายและมีความชัดเจน

7) XML ออกแบบมาเพื่อให้พัฒนาโปรแกรมได้อย่างรวดเร็ว

8) การออกแบบ XML ต้องมีรูปแบบที่เหมาะสมและกะทัดรัด

9) สามารถสร้างเอกสาร XML ได้ง่าย

10) Markup ของ XML ต้องไม่รวบรัดมากเกินไป

2.3.5.1 ตัวแปลเอกสาร XML (XML Parser)

XML Parser เป็นตัวแปลเอกสาร XML หน้าที่พื้นฐานก็คือ อ่าน สร้าง แก้ไข และเข้าถึง โดยสามารถจำแนกเป็น 2 ชนิดคือ

1) XML Parser แบบ Validate และแบบ Non-Validate

2) XML Parser แบบที่รองรับ DOM (Document Object Model) และ SAX (Simple API for XML)

เอกสาร XML มีคุณสมบัติเป็นเอกสารแบบ Well-formed XML คือมีการใช้แท็กและระบุแท็กตามรูปแบบที่ถูกต้อง นอกจากนี้ยังมีคุณสมบัติอีกอย่างหนึ่ง คือ Valid คือ เอกสาร XML ต้องมีโครงสร้างตรงตามข้อกำหนดใน DTD หรือ XML Schema

XML Parser แบบ Validate คือมีการตรวจสอบคุณสมบัติ Well-formed และ Valid ส่วน XML Parser แบบ Non-Validate จะตรวจสอบคุณสมบัติ Well-formed ไม่ตรวจสอบคุณสมบัติ Valid

2.3.5.2 **กระบวนการ** Parsing

โดยปกติทั่วไปแล้ว การที่เว็บเซิร์ฟเวอร์สามารถ load file XML แล้วนำมาแสดงเป็น HTML ได้นั้นจะเข้าสู่กระบวนการ parsing จะมีอยู่ด้วยกัน 2 วิธีหลัก ๆ คือ

1) DOM (Document Object Model) เป็นวิธีที่ประมวลโครงสร้างของเอกสาร XML ให้เป็นโครงสร้างแบบต้นไม้ดังนั้นการเข้าถึง เอกสารก็คือการเดิน (Tralvel) ไปตามโหนดต่างๆในโครงสร้างต้นไม้ จึงเรียกเป็น Tree-Based Parser เพื่อให้แอปพลิเคชั่นสามารถเข้าหาจุดต่าง ๆ ของโครงสร้างต้นไม้ได้ โดยที่ DOM จะโหลดไฟล์ XML ทั้งไฟล์ในหน่วยความจำซึ่งก็มีทั้งข้อดีและข้อเสีย คือ ข้อดี คือ การเรียกใช้งานจะช้า แต่หลังจากนั้นการเข้าถึงจุดต่าง ๆ ของไฟล์ จะเร็วเพราะถูกเก็บในหน่วยความจำแล้ว ข้อเสีย คือ ไม่เหมาะกับไฟล์ XML ที่มีขนาดใหญ่ โดยที่ขอบเขตจะถูกจำกัดด้วยหน่วยความจำที่เรียกใช้ในขณะนั้น

2) SAX (Simple API for XML) เป็นการทำงานแบบ event-based API คือจะรายงานข้อมูลตั้งแต่จุดเริ่มต้นและสิ้นสุดของอิลิเมนต์ต่าง ๆ ไปให้แอปพลิเคชั่นโดยไม่ต้องมีการสร้างโครงสร้างแบบต้นไม้ขึ้นมา ซึ่งจะเข้าถึงเอกสาร XML ทำงานได้ง่ายและไม่ซับซ้อน และที่สำคัญผู้ใช้สามารถทำ parsing เอกสารที่มีขนาดใหญ่กว่าปริมาณหน่วยความจำได้

2.3.5.3 **การประกาศเอกสาร** XML (XML Declararion)

XML Declararion เป็นการประกาศให้รู้ว่า เอกสารนี้เป็นเอกสาร XML ไม่ใช่ไฟล์ข้อความธรรมดา โดยมีรูปแบบดังนี้

<?xml version="1.0" encoding="ชุดอักษร" stanalone="yes|no"?>

ตัวอักษรทุกตัวของแต่ละแอตทริบิวต์ใน XML Declaration จะต้องเป็นตัวพิมพ์เล็ก ความหมายของแต่ละแอตทริบิวต์ เป็นดังนี้

- version เป็นแอตทริบิวต์ที่ต้องระบุไว้เสมอ เพื่อบ่งบอกเวอร์ชันของ XML
- encoding เป็นแอตทริบิวต์ที่ระบุเมื่อจำเป็นเท่านั้น โดยบ่งบอกชุดรหัสตัวอักษรที่ใช้ในเอกสารให้ XML Parser รู้
- stanalone เป็นแอตทริบิวต์ที่ระบุเมื่อจำเป็นเท่านั้น โดยบ่งบอกให้ทราบว่าเอกสาร XML นี้ขึ้นกับเอกสารอื่นหรือไม่ หากมีค่าเป็น “yes” แสดงว่าเป็นเอกสารที่มีข้อมูลของมันเองโดด หากเป็น “no” คือมีส่วนที่อ้างถึงอยู่ในเอกสารอื่น

2.3.5.5 **การประกาศชนิดของเอกสาร** XML (Document Type Declaration)

Document Type Declaration คือ การประกาศชนิดของเอกสาร XML รวมถึงไฟล์ DTD ที่กำหนดโครงสร้างของเอกสารด้วย

<?DOCTYPE rootelement ?>

การประกาศ Document Type Declaration มีทั้งประกาศแบบภายใน คือแทรก Document Type Declaration ในเอกสาร XML เรียกว่า “Internal DTD” และแบบประกาศภายนอก คือแยกเป็นไฟล์ต่างหาก เรียกว่า “External DTD”

2.3.5.6 กฎเกณฑ์เบื้องต้นของอิลิเมนต์

- เอกสาร XML จะมีรูทอิลิเมนต์ (Root Element) ได้เพียงหนึ่งเดียวเท่านั้น
- แท็กเปิดและแท็กปิดต้องเหมือนกัน
- ห้ามระบุแท็กเหลื่อมซ้อนกัน
- ชื่อแท็กมีคุณสมบัติ Case-sensitive คือ ตัวอักษรพิมพ์เล็ก พิมพ์ใหญ่ ถือว่าแตกต่างกัน
- แท็กที่ไม่มีข้อมูลข้างในมีวิธีเขียนได้ 2 แบบ คือ <title/> หรือ <title></title>
- ค่าของแอตทริบิวต์ต้องอยู่ในเครื่องหมายคำพูดแบบ double quote (“) หรือ single quote (‘)
- มีอักขระซึ่งสงวนไว้ 5 ตัว คือ
  - < เขียนแทนด้วย <
  - & เขียนแทนด้วย &
  - > เขียนแทนด้วย >
  - “ เขียนแทนด้วย "
  - ‘ เขียนแทนด้วย &apos
- การตั้งชื่อแท็กต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย Underscore (_) เท่านั้น และอักษร 3 ตัวแรกห้ามเป็นคำว่า XML ไม่ว่าตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็ก

โดยที่เนื้อหาภายในอิลิเมนต์มีโอกาสเป็นไปได้ 3 ลักษณะ คือ เป็นอิลิเมนต์อื่น เป็นข้อความปกติ ไม่มีอะไรอยู่เลย เป็นอะไรก็ได้ หรือบางครั้งอาจจะมีอิลิเมนต์ปนอยู่กับข้อความปกติก็ได้

บทที่3
การออกแบบและพัฒนาโปรแกรม

3.1 ขอบเขตการทำงาน

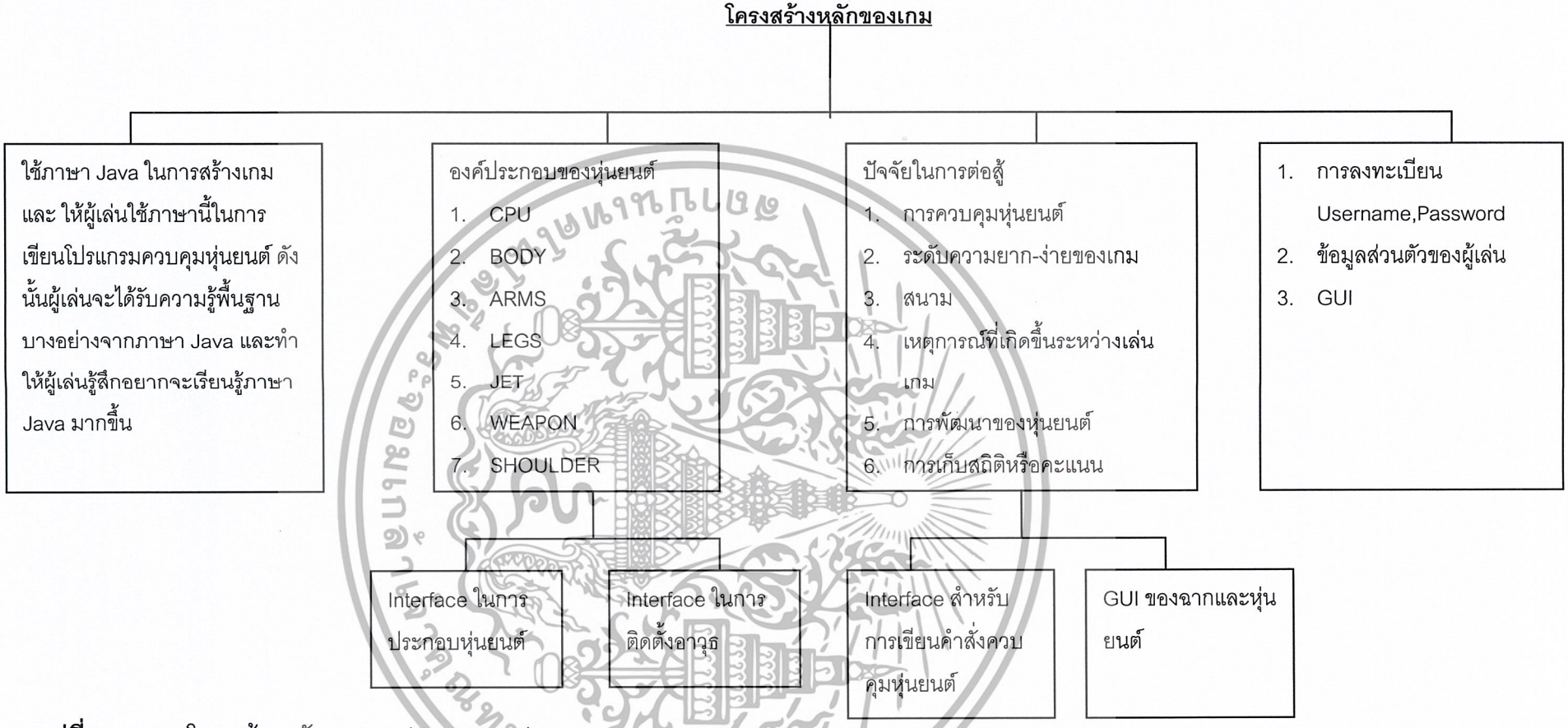
3.1.1 การออกแบบบทละคร (Scenario) ของเกม

เป็นเกมการแข่งขันของหุ่นยนต์ 2 ฝ่าย ผู้เล่นสามารถสร้างหุ่นยนต์เป็นของตนเองโดยสามารถนำเอาชิ้นส่วนต่างๆของหุ่นยนต์มาประกอบกัน นำเอาอาวุธมาติดตั้งให้กับหุ่นยนต์และป้อนโปรแกรมที่ใช้ควบคุมการทำงานของหุ่นยนต์ โดยหุ่นยนต์ที่ได้รับการโปรแกรมจะทำงานตามโปรแกรมนั้นๆ เกมแบ่งโหมดการเล่นออกเป็น 2 โหมด

1) สตอรี่โหมด (Story mode) จะมีการแข่งขันโดยแบ่งเป็นระดับ เมื่อสามารถชนะหุ่นยนต์ในแต่ละระดับครบตามที่ตั้งไว้ ก็สามารถนำหุ่นยนต์ไปแข่งขันในระดับที่สูงขึ้นได้ โดยหุ่นยนต์ในระดับที่สูงขึ้นจะมีประสิทธิภาพที่สูงขึ้นกว่าเดิม และเมื่อผู้เล่นเลื่อนระดับสูงขึ้น ผู้เล่นสามารถนำหุ่นยนต์ตัวเก่ามาทำการ Upgrade เพื่อให้หุ่นยนต์มีความสามารถที่สูงขึ้นได้ ผู้เล่นที่ชนะทุกระดับชั้นและสามารถชนะหัวหน้าหุ่นยนต์ได้ก็จะเป็นผู้ชนะ
2) ไฟต์ติ้งโหมด (Fighting mode) สามารถนำหุ่นยนต์มาแข่งขันกับเพื่อนๆหรือกับหุ่นที่มีอยู่ได้โดยไม่จำกัดว่าหุ่นยนต์ตัวนั้นจะอยู่ในระดับชั้นใด

3.1.2 โครงสร้างหลักของเกม (Core Value)

โครงสร้างหลักของเกม ดังรูปที่ 3.1



รูปที่ 3.1 แสดงโครงสร้างหลักของเกม (Core Value)

โครงสร้างหลักของเกมประกอบด้วย ภาษาจาวาเป็นภาษาที่ใช้ในการพัฒนาโปรแกรม การสร้างหุ่นยนต์ ปัจจัยที่ใช้ในการต่อสู้ของหุ่นยนต์ การเก็บข้อมูลต่างๆของผู้เล่น

3.1.3 การออกแบบส่วนประกอบของเกม

3.1.3.1 ลักษณะของเกม

1) การควบคุมหุ่นยนต์ ผู้เล่นสามารถป้อนคำสั่งต่างๆให้กับหุ่นยนต์เพื่อให้หุ่นยนต์ทำงานตามที่สั่ง การเขียนคำสั่งควบคุมใช้พื้นฐานของภาษาจาวา และเมธตอดที่เราสร้างขึ้นให้กับผู้เล่นนำไปใช้ ดังนั้นผู้เล่นจะได้รับความรู้และความเพลิดเพลิน โดยที่หุ่นยนต์ที่มีการทำงานเหมาะสมกับฉากหรือเหมาะสมกับฝ่ายตรงข้ามก็มีโอกาสที่จะสู้ชนะสูง

2) ระดับความยาก-ง่าย เกมนี้ได้แบ่งออกเป็นระดับต่างๆ 5 ระดับโดยเมื่อผู้เล่นมีระดับชั้นที่เปลี่ยนไปจะได้รับเงินเพิ่มมากขึ้นและมีส่วนประกอบให้เลือกซื้อได้มากขึ้น

ตารางที่ 3.1 แสดงรายการของส่วนประกอบที่มีให้เลือก (จำนวนประเภท) และจำนวนเงินที่ได้รับในระดับชั้นต่างๆ

Level	Body	Arm	Leg	Jet	CPU	Shoulder	HW	HA	LW	LA	Money
1	3	3	3	2	2	3	3	3	3	3	12,000
2	4	4	5	3	3	4	5	4	4	4	+5,000
3	5	5	6	4	4	5	6	6	5	6	+7,000
4	6	6	7	5	5	6	7	7	6	7	+10,000
5	7	7	8	5	5	7	7	7	7	7	+15,000

HW : Heavy Weapon

HA : Heavy Ammo

LW : Light Weapon

LA : Light Ammo

เริ่มจากง่ายสุดและยากขึ้นเรื่อยๆตามลำดับเพื่อให้เหมาะสมกับระดับของผู้เล่น ดังนั้นผู้เล่นจะได้รับพื้นฐานการวางแผนในการเล่นตั้งแต่ระดับเบื้องต้น และจะพัฒนาเทคนิค วิธีการเล่นให้ดีขึ้นเมื่อต้องเจอกับคู่ต่อสู้ที่เก่งกว่า

3) เหตุการณ์ที่เกิดขึ้นระหว่างการต่อสู้เหตุการณ์ต่างๆที่มีให้กับผู้เล่นเพื่อให้ผู้เล่นสามารถเขียนคำสั่งควบคุมหุ่นยนต์ ในเหตุการณ์ต่างๆที่เกิดขึ้นได้ ดังนั้นเมื่อเกิดเหตุการณ์ใดๆขณะต่อสู้หุ่นยนต์ที่สามารถจัดการกับเหตุการณ์ที่เกิดขึ้นได้ดีกว่า ก็จะมีโอกาสที่จะชนะได้สูงกว่า

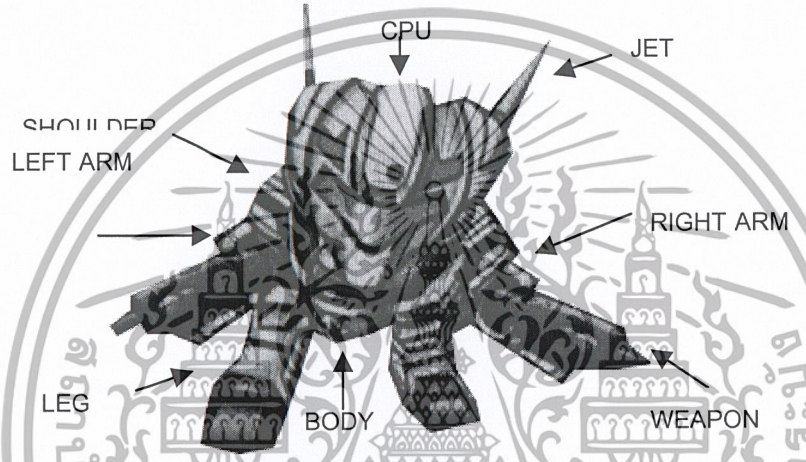
4) การพัฒนาของหุ่นยนต์ เมื่อผู้เล่นสามารถชนะหุ่นยนต์ของฝ่ายคอมพิวเตอร์ครบตามแต่ละระดับที่ตั้งไว้ ผู้เล่นจะสามารถนำหุ่นยนต์ไปแก้ไขเพิ่มเติม (Upgrade) ได้เช่น การเปลี่ยนอาวุธให้มีอานุภาพการทำลายสูงขึ้น หรือมีน้ำหนักเบา ก็จะทำให้หุ่นยนต์มีศักยภาพมากขึ้น และผู้

เล่นสามารถแก้ไขคำสั่งควบคุมใหม่เพื่อให้เหมาะสมกับหุ่นยนต์นั้นๆได้ เมื่อหุ่นยนต์ของผู้เล่นได้รับการพัฒนาแล้วสามารถจะส่งไปแข่งขันในระดับที่มีความยากมากขึ้นต่อๆไป

5) การเก็บสถิตของผู้เล่น เป็นตัวบ่งชี้ถึงศักยภาพของหุ่นยนต์นั้น เช่น การเก็บสถิติการแพ้ชนะของหุ่นยนต์ ทำ ให้ผู้เล่นอยากจะสร้างหุ่นยนต์ที่เก่งกาจไม่เคยแพ้ใคร และอาจจะนำสถิติของหุ่นยนต์ไปจัดอันดับได้

3.1.3.2 ลักษณะและรายละเอียดของหุ่นยนต์

ส่วนประกอบของหุ่นยนต์ ดังรูปที่ 3.2

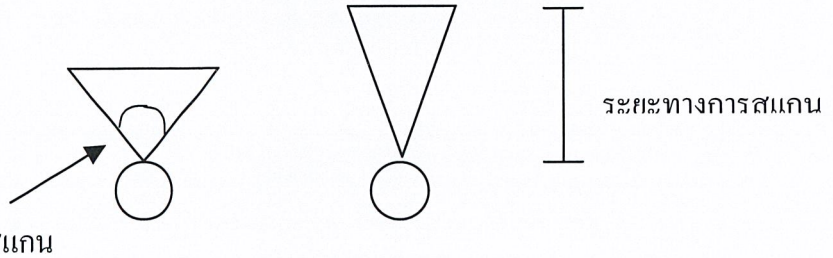


รูปที่ 3.2 แสดงส่วนประกอบต่างๆของหุ่นยนต์

โดยส่วนของ CPU และ Body นั้นจำเป็นจะต้องมีในการสร้างขึ้นเป็นหุ่นยนต์ ถ้าส่วนใดไม่มีจะไม่สามารถใช้งานและความสามารถในส่วนนั้นๆได้ แต่ละส่วนมีรายละเอียดดังนี้

CPU เปรียบเสมือนมันสมองของหุ่นยนต์ซึ่งคอยสั่งงานให้ทุกส่วนของหุ่นยนต์ทำงานตามโปรแกรมที่ได้รับและเป็นส่วนที่ควบคุมในเรื่องการ Scan

- การสแกน (Scan) จะแบ่งออกเป็น 2 ส่วน คือ ระยะทางการสแกนกับองศาการสแกน ดังรูป 3.3



รูปที่ 3.3 แสดงลักษณะการ scan ของหุ่นยนต์

BODY ใช้สำหรับเป็นส่วนกลางในการนำส่วนต่างๆ มาประกอบเป็นหุ่นยนต์ ถ้าไม่มีลำตัวจะไม่สามารถติดส่วนประกอบอื่นได้ มีค่าประจำ (Attribute) คือ พลังงาน (Energy),ขนาด (Size), น้ำหนัก (Weight)

LEFT ARM & RIGHT ARM เป็นส่วนที่ใช้สำหรับติดตั้งอาวุธเบา มีค่าประจำคือ พลังงาน, น้ำหนัก, ความแข็งแกร่ง (Strength)

LEG เป็นส่วนที่ใช้สำหรับการเคลื่อนที่ของหุ่นยนต์ มีค่าประจำ คือ พลังงงาน, น้ำหนัก, ความแข็งแกร่ง, ความเร็ว (Speed)

JET เป็นส่วนที่ใช้สำหรับการเคลื่อนที่ของหุ่นยนต์ มีค่าประจำ คือ พลังงาน, ความเร็ว, เชื้อเพลิง (Oil)

SHOULDER เป็นส่วนที่ใช้สำหรับติดตั้งอาวุธหนัก มีค่าประจำ คือ พลังงาน, น้ำหนัก, ความแข็งแกร่ง

WEAPON จะแบ่งออกเป็น อาวุธเบา และอาวุธหนัก โดย

- อาวุธเบา ระยะยิงสั้น และกระสุนมีพลังทำลายต่ำ แต่ความเร็วในการยิงจะมาก และบรรจุกระสุนได้มาก
- อาวุธหนัก ระยะยิงไกล และกระสุนมีพลังทำลายสูง แต่ความเร็วในการยิงจะน้อย และบรรจุกระสุนได้น้อย
- AMMO ขึ้นอยู่กับ WEAPON โดยจะมีค่าประจำคือ ความเร็ว,พลังทำลาย(Attack), จำนวนกระสุน (Amount)

3.1.3.3 **เหตุการณ์ระหว่างการต่อสู้ของหุ่นยนต์**

เหตุการณ์ที่สามารถกิดขึ้นได้ในระหว่างการต่อสู้ของหุ่นยนต์ประกอบด้วย 6 เหตุการณ์

1. เหตุการณ์ปกติ (Common)
2. เหตุการณ์พบศัตรู (Found Enermy)
3. เหตุการณ์โดนโจมตี (Attracked)
4. เหตุการณ์พบขอบสนาม (Found Border)
5. เหตุการณ์ชนศัตรู (Cash Enemy)
6. เหตุการณ์ชนขอบสนาม (Crash Border)

ในแต่ละเหตุการณ์นั้นสามารถที่จะเกิดขึ้นพร้อมๆ กันได้แล้วแต่ว่าจะเกิดเหตุการณ์ใดขึ้นบ้างในเวลานั้น คำสั่งที่ถูกเขียนขึ้นในแต่ละเหตุการณ์นั้นจะถูกประมวลผลทีละคำสั่งแบบลำดับ โดยคำสั่งแรกจะถูกทำงานจนเสร็จสิ้นก่อน คำสั่งต่อไปจึงจะสามารถทำงานได้ โดยมีข้อกำหนดของการทำงานของเหตุการณ์ต่างๆดังนี้

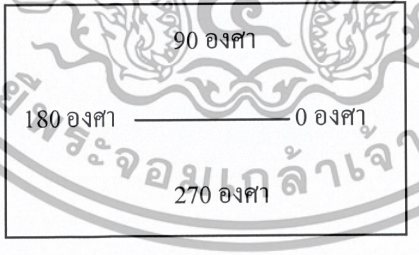
- ในตอนเริ่มต้นของเกมนั้นเหตุการณ์แรกที่จะถูกประมวลผลหรือถูกเรียกให้ทำงานก่อนคือเหตุการณ์ปกติ และเมื่อสิ้นสุดคำสั่งการทำงานของเหตุการณ์ปกติทั้งหมดแล้ว จะมีการทำงานซ้ำอีกครั้งตั้งแต่คำสั่งแรกจนถึงคำสั่งสุดท้ายอีกครั้งโดยอัตโนมัติ การทำงานจะวนอย่างนี้ไปเรื่อย ๆ จนกว่าจะจบเกมซึ่งในระหว่างนั้นก็สามารถเกิดเหตุการณ์อื่นพร้อม ๆ กันได้

- เมื่อเกิดเหตุการณ์พบศัตรูแล้วจะทำการประมวลผลคำสั่งที่ได้เขียนไว้จนกว่าจะหลุดจากเหตุการณ์พบศัตรู เมื่อหลุดจากเหตุการณ์นี้แล้ว คำสั่งของเหตุการณ์นี้จะถูกหยุดการประมวลผลทันที เมื่อเกิดเหตุการณ์นี้ใหม่อีกครั้งจะเริ่มทำการประมวลผลคำสั่งใหม่ตั้งแต่ต้นอีกครั้งหนึ่ง หากการทำงานสิ้นสุดลงหรือจบชุดคำสั่งทั้งหมดแล้ว ยังคงเกิดเหตุการณ์พบศัตรูอยู่จะมีการประมวลผลตั้งแต่คำสั่งแรกใหม่ตั้งแต่ต้นอีกครั้งหนึ่ง

- เหตุการณ์อื่นที่เหลือทั้งหมด คือ เหตุการณ์ โดนโจมตี เหตุการณ์พบขอบสนาม เหตุการณ์ชนศัตรู และเหตุการณ์ชนขอบสนาม เมื่อเกิดเหตุการณ์เหล่านี้ขึ้นแล้ว คำสั่งต่างๆที่ถูกเขียนไว้จะถูกประมวลผลทีละคำสั่งจนครบถึงคำสั่งสุดท้าย แม้ว่าในระหว่างที่ทำการประมวลผลคำสั่งต่างๆอยู่นั้น จะเกิดเหตุการณ์เหล่านี้ซ้ำอีกก็ตามจะถือว่าไม่เกิดผลใด ๆ คือจะยังทำการประมวลผลจนถึงคำสั่งสุดท้าย ต่อเมื่อประมวลผลคำสั่งจนครบหมดแล้ว เกิดเหตุการณ์นั้นอีกจึงจะเริ่มทำการประมวลผลคำสั่งใหม่อีกครั้งตั้งแต่คำสั่งแรก

3.1.3.4 **ลักษณะของสนามรบ**

สนามรบเป็นพื้นที่ที่ใช้ในการต่อสู้ของหุ่นยนต์และแสดงผลทั้งหมดในการต่อสู้ มีพื้นที่เป็นสี่เหลี่ยมและมีทิศทางเป็นองศา ดังรูปที่ 3.4



รูปที่ 3.4 แสดงลักษณะสนามรบ

การทำงานที่จะเกิดในสนามรบซึ่งสนามต้องทำการควบคุม แบ่งได้เป็น 5 ส่วนใหญ่ๆ ดังนี้

1) การเคลื่อนที่ สิ่งที่สัมพันธ์กันคือน้ำหนักของหุ่นยนต์และความเร็ว ซึ่งจะมีสมการในการคิดสำหรับการเคลื่อนที่ หุ่นยนต์จะส่งค่าพารามิเตอร์ของพิกัดที่จะเคลื่อนที่หรือค่าองศาและระยะทางจากจุดปัจจุบันไปยังสนามรบเพื่อให้สนามรบเป็นตัววาดกราฟิก

การชนขอบของสนามรบหรือชนกันเองระหว่างหุ่นยนต์จะทำให้หุ่นยนต์หยุดและเสียพลังงานในทุกส่วนโดยถ้ามีไอพ่นเป็นส่วนประกอบด้วยก็จะเสียปริมาณน้ำมันของไอพ่นไปเรื่อยๆ

2) การสแกนมีการกำหนดพื้นที่สามเหลี่ยมของการสแกนและมุมมองในการมองเห็นของหุ่นยนต์ซึ่งทุกตัวจะมีค่าเท่ากัน เมื่อมีการสแกนพบศัตรู จะสามารถรับค่าต่างๆได้ เช่น ตำแหน่งของศัตรู ความเร็วของศัตรู พลังของแต่ละส่วนของศัตรู และยังสามารถ บอกประเภทของสิ่งที่สแกนได้ เช่น ขอบของสนาม หุ่นยนต์ที่พบเป็นพันธมิตรหรือศัตรู

3) การถูกโจมตี หุ่นยนต์จะถูกโจมตีเมื่อจากฝ่ายตรงข้ามมีการสั่งยิงและในการยิงนั้นกระสุนมาโดนหุ่นยนต์ทำให้หุ่นยนต์เสียพลังงานของส่วนที่โดนซึ่งจะเป็นส่วนใดก็ได้

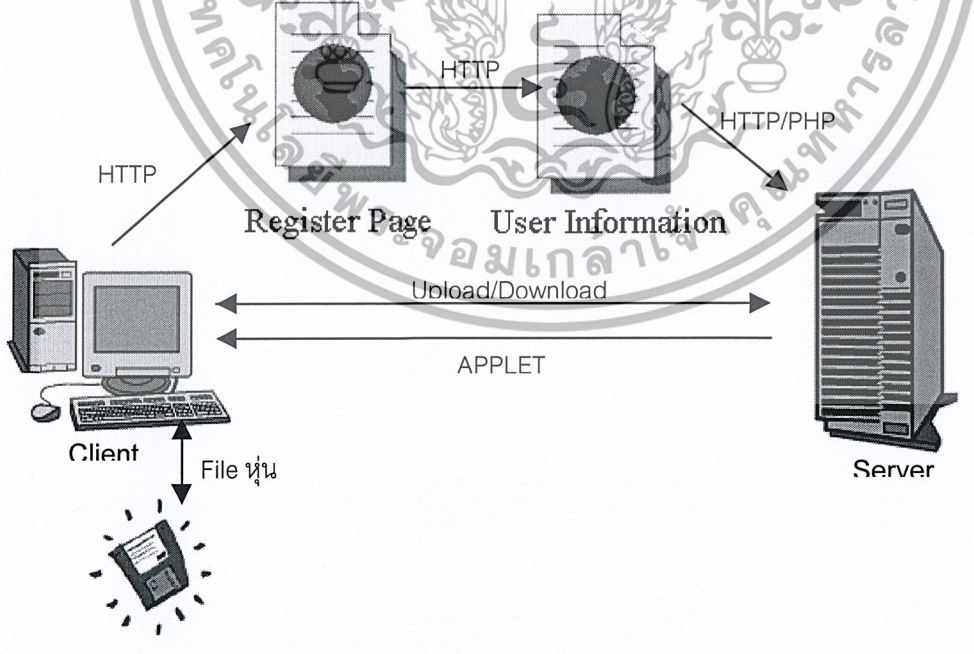
4) การโจมตี หุ่นยนต์สามารถทำการโจมตีได้ทุกเมื่อ โดยจะต้องระบุเป้าหมายของการโจมตีด้วย

5) การจัดการเกี่ยวกับเหตุการณ์ต่างๆที่อาจจะเกิดขึ้นได้ในขณะที่มีการต่อสู้กัน โดยในแต่ละเหตุการณ์นั้นผู้เขียนจะต้องเขียนคำสั่งเพื่อสั่งให้หุ่นยนต์ทำงานตามที่ต้องการและสนามจะจัดการทำตามคำสั่งนั้นๆ

3.2 การออกแบบสถาปัตยกรรมที่ใช้ในเกม

3.2.1 โครงสร้างสถาปัตยกรรมของเกม

โครงสร้างสถาปัตยกรรมของเกม ดังรูปที่ 3.5

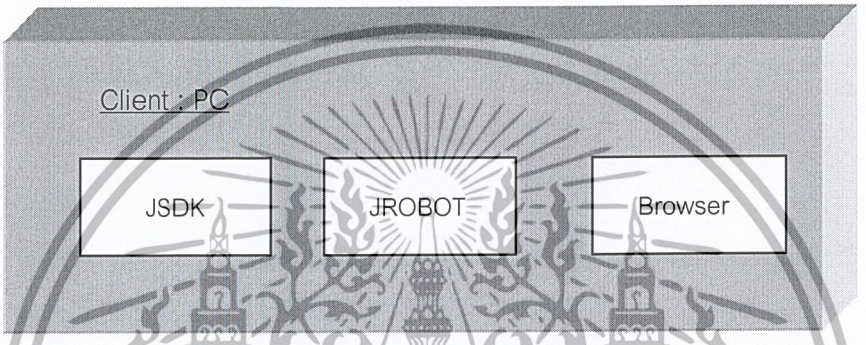


รูปที่ 3.5 แสดงโครงสร้างสถาปัตยกรรมของเกม

สถาปัตยกรรมที่ใช้ในเกมเป็นลักษณะ 2-Tier Client/Server โดยทางฝั่งไคล์เอนท์จะติดต่อกับ เซิร์ฟเวอร์ ผ่าน protocol HTTP โดยผู้เล่นจะต้องทำการลงทะเบียนผ่านหน้าเว็บเพื่อเข้าสู่หน้าต่างข้อมูลของผู้ใช้ สำหรับการติดต่อกับเซิร์ฟเวอร์ใช้ PHP เป็น Server Side Script ที่จะใช้โต้ตอบกับ เซิร์ฟเวอร์และผู้เล่นสามารถจะชมการแข่งขันโดยเรียกแอปเพล็ตผ่านเว็บบราวเซอร์รวมถึงสามารถจะบันทึกข้อมูลหุ่นยนต์ลงสื่อต่างๆ เช่น Floppy disk หรือ CD ได้

3.2.2 สถาปัตยกรรมที่ใช้ทางฝั่งผู้เล่น (Client)

Block Diagram : เครื่องคอมพิวเตอร์ทางฝั่งผู้เล่น ดังรูปที่ 3.6

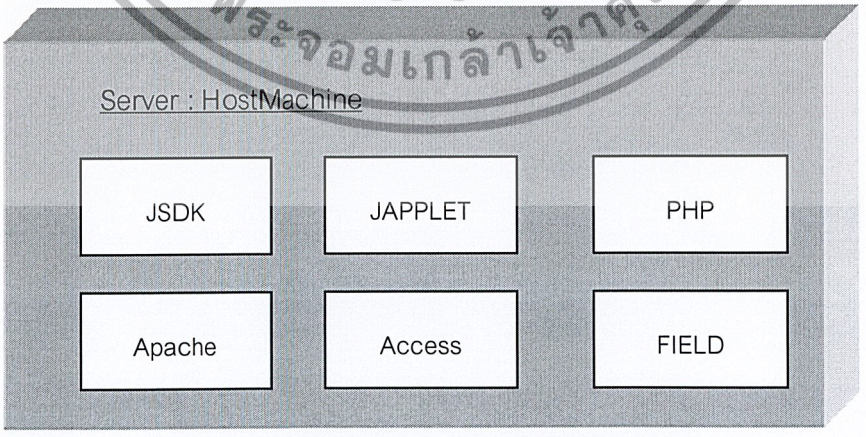


รูปที่ 3.6 แสดงบล็อกไดอะแกรม (Block Diagram) ของเครื่องคอมพิวเตอร์ทางฝั่งผู้เล่น

- JSDK และ JROBOT เป็นคอมโพเนนท์ที่จำเป็นสำหรับการสร้างหุ่นยนต์
- Browser จำเป็นในการแข่งขันของหุ่นยนต์

3.2.3 สถาปัตยกรรมที่ใช้ทางฝั่งเซิร์ฟเวอร์

Block Diagram : เครื่องคอมพิวเตอร์ทางฝั่งเซิร์ฟเวอร์ ดังรูปที่ 3.7

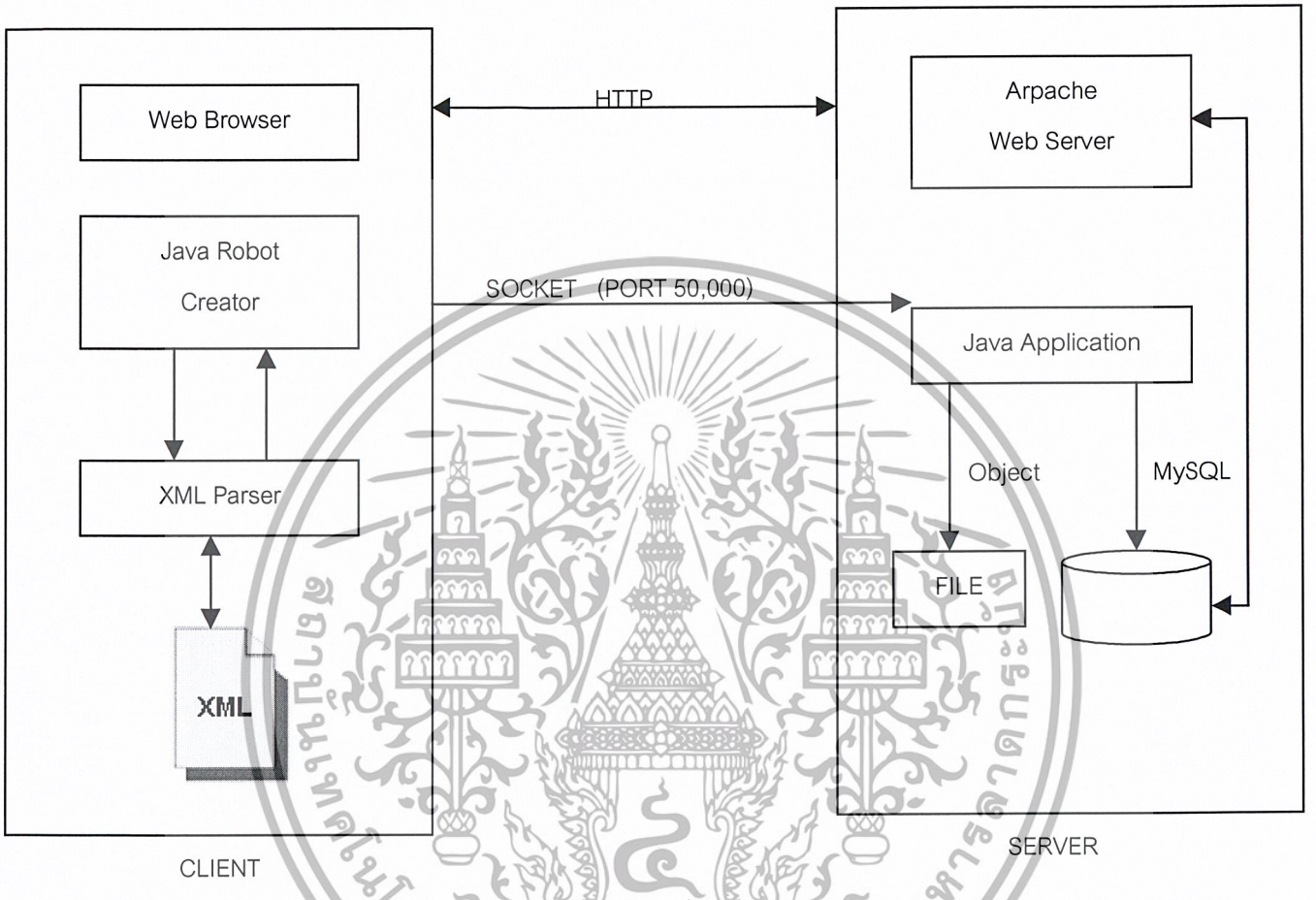


รูปที่ 3.7 แสดงบล็อกไดอะแกรม ของเครื่องคอมพิวเตอร์ทางฝั่งเซิร์ฟเวอร์

คอมโพเนนท์ต่างๆ ภายในเซิร์ฟเวอร์เป็นส่วนที่จำเป็นในการประมวลผลของการแข่งขัน และแสดงการแข่งขันให้ผู้เล่น

3.2.4 การทำงานร่วมกันระหว่างฝั่งผู้เล่นและฝั่งเซิร์ฟเวอร์

การทำงานร่วมกันระหว่างฝั่งผู้เล่นและฝั่งเซิร์ฟเวอร์ เป็นดังรูปที่ 3.8



รูปที่ 3.8 แสดงการทำงานร่วมกันระหว่างฝั่งผู้เล่นและฝั่งเซิร์ฟเวอร์

1.) Server จะทำการเปิด Socket และรอรับข้อมูลจาก Client ตลอดเวลา

2.) ผู้เล่นจะทำการสร้างหุ่นยนต์ด้วยโปรแกรม Java Robot Creator โดยในการสร้างหุ่นยนต์จะมีการนำเอาข้อมูลออกมาจากเอกสาร XML โดยผ่าน XML Parser โดยจะอาศัยการเข้าถึงข้อมูลแบบ SAX

3.) ผู้เล่นจะต้องทำการอัพโหลดหุ่นยนต์ที่สร้างเสร็จแล้วผ่านทางเว็บ โดยจะสมารถส่งข้อมูลผ่านโปรโตคอล HTTP

4.) การทำงานต่างๆบนเว็บจะประมวลที่ Arpache Web Server ซึ่งใช้ PHP เป็นภาษาที่ใช้ในการประมวลผล

5.) เมื่อผู้เล่นเลือกโหมดที่จะเล่น และเลือกหุ่นยนต์ที่จะทำการนำไปแข่งขัน Server จะส่งข้อมูลผ่านโปรโตคอล HTTP ให้มาแสดงผลที่การต่อสู้บนเครื่อง Client โดยใช้ JApplet ในการแสดงผล

6.) เมื่อแข่งขันเสร็จ Client จะทำการส่ง Object ที่เก็บข้อมูลต่างของหุ่นยนต์ เช่น พลังงาน โดยจะทำการเปิด Socket แล้วจึงส่งข้อมูลผ่านทาง Port 50,000 เพื่อนำไปให้กับ Java Application ทางฝั่ง Server เพื่อทำการบันทึกข้อมูลให้อยู่ในรูป file และข้อมูลที่เกี่ยวข้องกับสถิติการแข่งขันจะถูกเก็บลงในฐานข้อมูล MySQL

7.) ข้อมูลสถิติการแข่งขันจะนำมาแสดงในเว็บโดยจะใช้ PHP ในการเชื่อมต่อกับ MySQL

3.3 การออกแบบและพัฒนาโปรแกรม

ในการออกแบบโปรแกรมที่จำเป็นสำหรับเกมจะแบ่งออกเป็น 2 ส่วน คือ

1) จาวาแอปพลิเคชันเป็นส่วนสำหรับโปรแกรมในการสร้างหุ่นยนต์ โดยตัวโปรแกรมจะให้เราทำการสร้างหุ่นยนต์ด้วยการเลือกส่วนประกอบต่างๆของหุ่นยนต์และเขียนโปรแกรมที่จะใช้ในการควบคุมการทำงานของหุ่นยนต์ โดยจะใช้ภาษาจาวาและ Component Swing ในการพัฒนาโปรแกรมในส่วนนี้

2) เว็บแอปพลิเคชันเป็นส่วนที่เกี่ยวข้องกับการต่อสู้ของหุ่นยนต์, ส่วนเก็บข้อมูลหุ่นยนต์และสถิติการแข่งขันของผู้เล่น, ส่วนลงทะเบียนสมัครสมาชิก และส่วนสนามที่ใช้ต่อสู้

- ส่วนที่ใช้ในการสมัครสมาชิกจะเป็น Form HTMLร่วมกับ PHP และใช้เว็บเซิร์ฟเวอร์ คือ Apache Web Server ในการประมวลผล PHP

- ส่วนของการเก็บข้อมูลของผู้เล่น,หุ่นยนต์, User Name, Password และสถิติการแข่งขันจะเก็บข้อมูลผ่าน Form HTML ไปยัง MySQL และติดต่อ MySQL ด้วย PHP

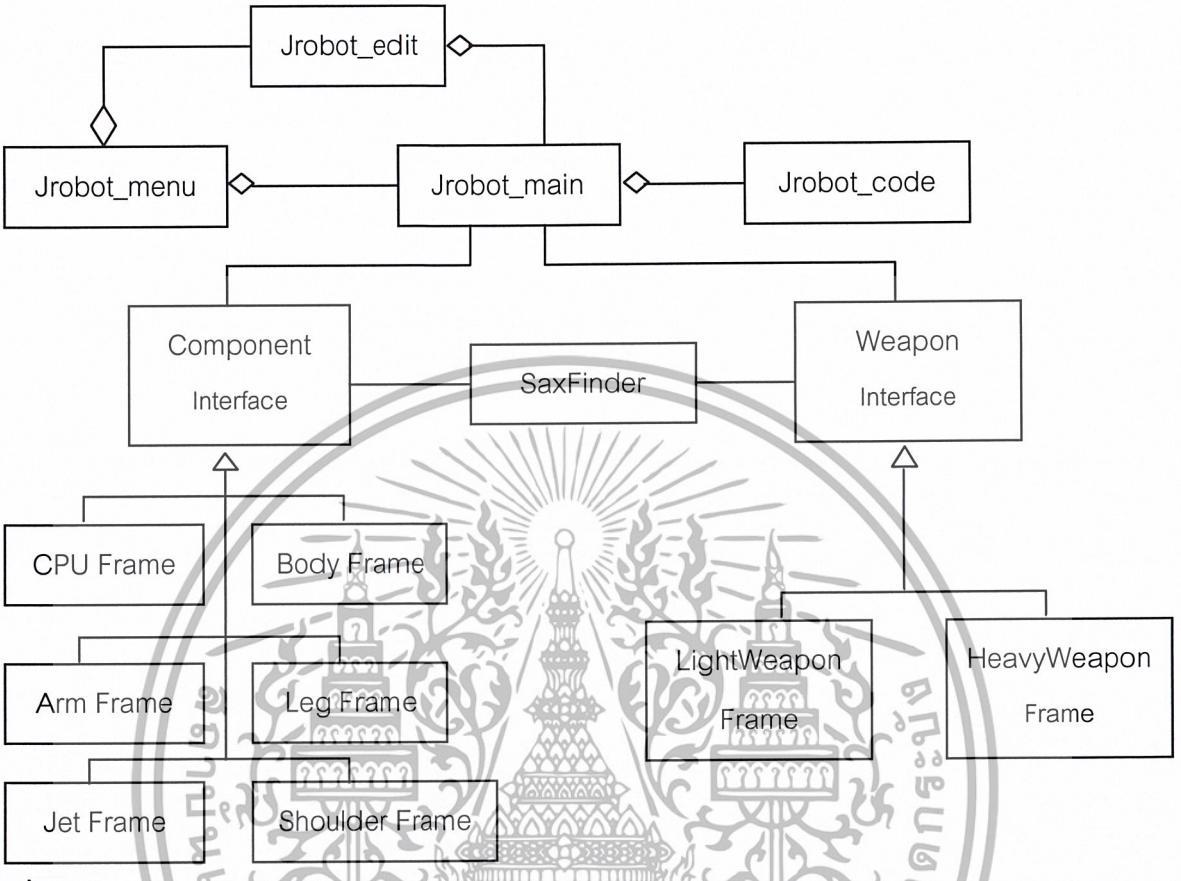
- ส่วนของการแข่งขันจะใช้แอปเพล็ตในการแสดงผลผ่านเว็บบราวเซอร์บนหน้าจอของผู้เล่น

- ส่วนติดต่อกับผู้เล่น เป็นหน้าเว็บที่จะให้ความเข้าใจเกี่ยวกับเกมนี้คืออะไร วิธีการเล่นเกม รู้ประวัติของผู้จัดทำและความเป็นมาของเกม Patch ต่างๆ ที่จะใช้ปรับปรุง (Update) เกมให้มีประสิทธิภาพสูงขึ้นการดาวน์โหลดแอปพลิเคชันที่จำเป็นในการเล่นเกม เช่น โปรแกรมสำหรับสร้างหุ่นยนต์ โปรแกรม JSDK 1.4.1 ที่ใช้ในการคอมไพล์หุ่นยนต์ เป็นต้น

การทำงานในส่วนของจาวาแอปพลิเคชั่นจะเป็นแบบ off-line คือไม่มีการเชื่อมต่อกับเซิร์ฟเวอร์ แต่ในส่วนของการแข่งขันจะต้องติดต่อกับเซิร์ฟเวอร์ เพราะจะเป็นการติดต่อผ่านทางหน้าเว็บ

3.3.1 คลาสไดอะแกรม (Class Diagram) ของโปรแกรมสร้างหุ่นยนต์

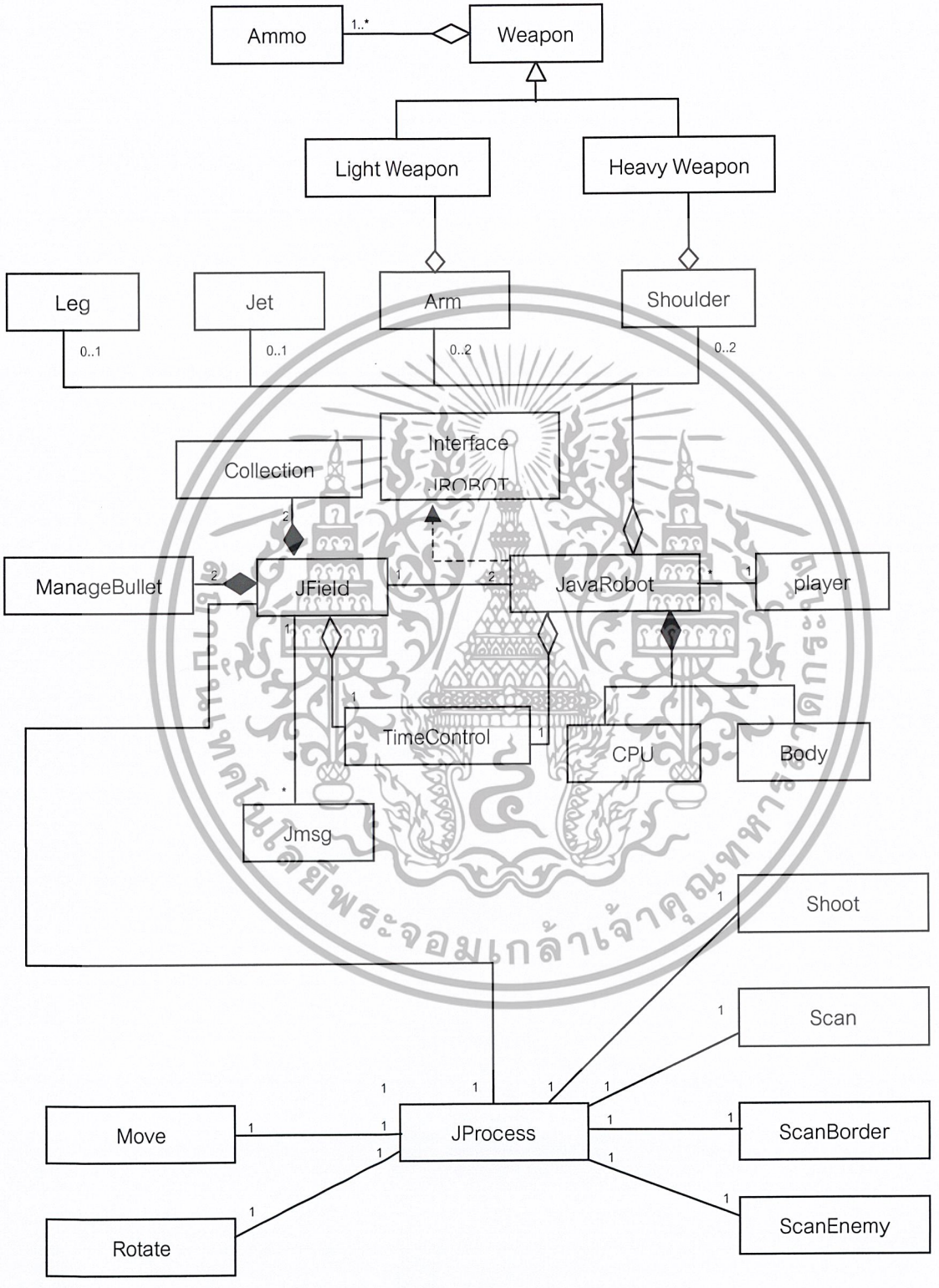
คลาสไดอะแกรมของโปรแกรมสร้างหุ่นยนต์ ดังรูปที่ 3.9



รูปที่ 3.9 แสดงคลาสไดอะแกรมของโปรแกรมสร้างหุ่นยนต์

3.3.2 คลาสไดอะแกรมของสนามรบ

คลาสไดอะแกรมของสนามรบ ดังรูปที่ 3.10



รูปที่ 3.10 แสดงคลาสไดอะแกรมของสนามรบ

จากคลาสไดอะแกรมสามารถอธิบายได้ดังนี้

Class Interface JROBOT เป็นต้นแบบของหุ่นยนต์ Class JavaRobot ถ่ายทอดคุณสมบัติจาก Class InterfaceJROBOT ซึ่งเป็น Abstract Class ที่ให้ ผู้เล่น extend ซึ่ง Class JavaRobot มีความสัมพันธ์กับ Class JField หรือสนามรบโดยใน 1 JField จะมี Robot ที่ต่อสู้กันได้เพียง 2 ตัว ซึ่งการต่อสู้จะทำให้เกิดผลการแพ้-ชนะ และมีความสัมพันธ์กับ Class Player หรือผู้เล่นโดยที่ผู้เล่น 1 คน สามารถมีหุ่นยนต์เป็นของตนเองได้หลายตัว โดย Class JavaRobot มีองค์ประกอบดังนี้

- องค์ประกอบที่จำเป็นต้องมี (composition) คือ Class CPU, Class Body โดย JRobot จะมี CPU และ Body ได้อย่างละ 1 ชิ้นเท่านั้น

- องค์ประกอบที่ไม่จำเป็นต้องมี (aggregration) คือ Class Leg, Class Jet, Class Shoulder, Class Arm โดย JRobot จะมี ขา 1คู่ หรือไม่มีก็ได้, Jet 1 ตัว หรือไม่มีก็ได้, ไหล่ จะมีหรือไม่มีก็ได้ ถ้ามีมีได้อย่างมากไม่เกิน 2 ข้าง, แขน จะมีหรือไม่มีก็ได้ ถ้ามีมีได้อย่างมากไม่เกิน 2 ข้าง

Class Shoulder จะมีองค์ประกอบที่ไม่จำเป็น (aggregration) คือ Class Heavy Weapon ซึ่ง Class Heavy Weapon ถ่ายทอดคุณสมบัติจาก Class Weapon โดยไหล่สามารถมีอาวุธหนักติดอยู่ที่ไหลหรือไม่มีก็ได้ ถ้ามีสามารถติดอาวุธได้ข้างละ 1 ชิ้น

Class Arm จะมีองค์ประกอบที่ไม่จำเป็น (aggregration) คือ Class Light Weapon ซึ่ง Class Light Weapon ถ่ายทอดคุณสมบัติจาก Class Weapon โดยแขนสามารถมีอาวุธเบาติดอยู่ที่แขนหรือไม่มีก็ได้ ถ้ามีสามารถติดอาวุธได้ข้างละ 1 ชิ้น

Class Weapon จะมีองค์ประกอบที่จำเป็นต้องมีคือ Class Ammo ซึ่งอาวุธจำเป็นต้องมีกระสุน โดยอาวุธ 1 ชิ้นจะมีกระสุนตั้งแต่ 1 กระสุนขึ้นไป

Class Player มีความสัมพันธ์กับ Class JavaRobot ซึ่งผู้เล่น 1 คน สามารถมีหุ่นยนต์เป็นของตัวเองอย่างน้อง 1 ตัวและจะมีเท่าไรก็ได้

Class JavaRobot และ Class JField มีองค์ประกอบที่ไม่จำเป็นคือ Class TimeControl ซึ่งเป็น Abstract Class ที่

Class JField มีองค์ประกอบที่จำเป็นต้องมีคือ Class Collection, Class ManageBullet และมีความสัมพันธ์กับ Class Jmsg

Class Jmsg ทำหน้าที่ส่งคำสั่งที่ทำให้หุ่นยนต์ทำงานออกไป

Class Collection ทำหน้าที่เก็บคำสั่งที่ถูกส่งมาจาก Class Jmsg เพื่อให้ Class JField นำคำสั่งไปประมวลผล

Class ManageBullet ทำหน้าที่จัดการเกี่ยวกับการยิงของหุ่นยนต์ ใช้เก็บจำนวนกระสุนที่ยิงออกไปแล้ว

Class JProcess มีความสัมพันธ์กับ Class Scan, Class ScanEnemy, Class ScanBorder, Class Shoot, Class ManageBullet, Class Move, Class Rotate ทำหน้าที่ประมวลผลคำสั่งว่าคำสั่งที่ Class JField สั่งให้ทำเป็นคำสั่งประเภทใดเพื่อให้แสดงผลได้ถูกต้อง

Class Scan ทำหน้าที่จัดการคำนวณตำแหน่งของจุด scan ของหุ่นยนต์ เมื่อมีการ scan การ scan จะขึ้นอยู่กับชนิด CPU ด้วย

Class ScanEnemy ทำหน้าที่จัดการคำนวณเกี่ยวกับการค้นหาศัตรู การ scan enemy จะขึ้นอยู่กับชนิด CPU ด้วย

Class ScanBorder ทำหน้าที่จัดการคำนวณตำแหน่งเกี่ยวกับการค้นหาขอบเขตของสนาม การ scan border จะขึ้นอยู่กับชนิด CPU ด้วย

Class Shoot ทำหน้าที่จัดการเกี่ยวกับการยิงซึ่งจะขึ้นอยู่กับชนิดของอาวุธด้วย

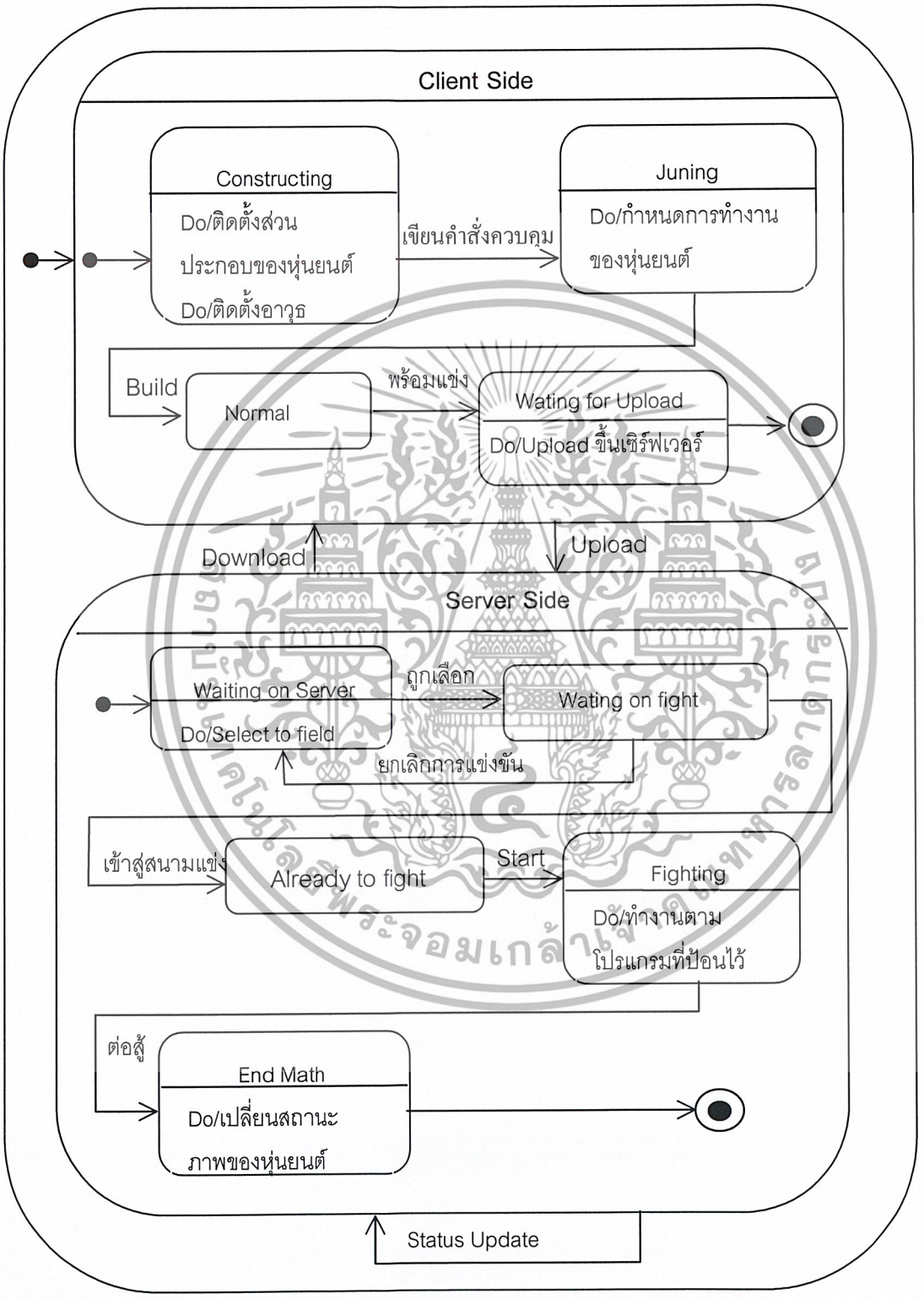
Class Move ทำหน้าที่จัดการเกี่ยวกับการเคลื่อนที่ของหุ่นยนต์ซึ่งจะขึ้นอยู่กับชนิดขององค์ประกอบที่ใช้ในการเคลื่อนที่ด้วย

Class Rotate ทำหน้าที่จัดการเกี่ยวกับการหมุนของหุ่นยนต์ซึ่งจะขึ้นอยู่กับชนิดขององค์ประกอบที่ใช้ในการเคลื่อนที่ด้วย

ข้อมูลรายละเอียดของแอตทริบิวต์และเมธอดของคลาสที่ใช้จะแสดงที่ภาคผนวก ค.

3.3.3 สเตทไดอะแกรม (State Diagram) ของหุ่นยนต์

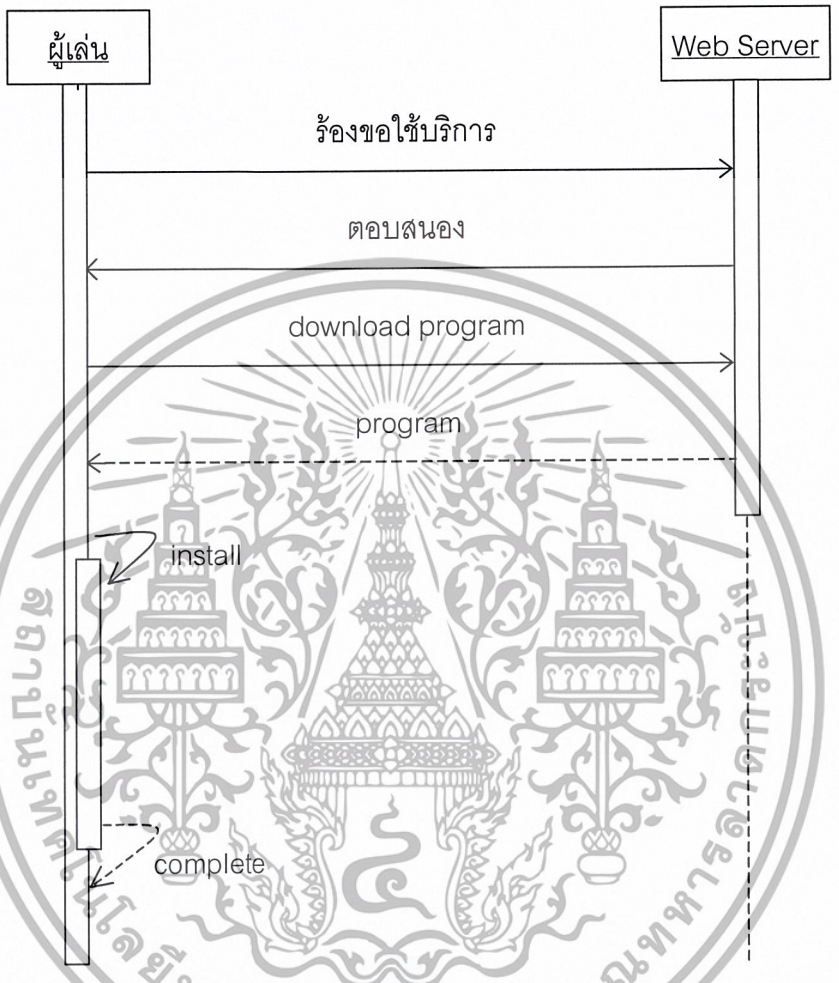
สเตทไดอะแกรมของหุ่นยนต์ ดังรูปที่ 3.11



รูปที่ 3.11 แสดงสเตทไดอะแกรมของหุ่นยนต์

3.3.4 ซีเควนซ์ไดอะแกรม (Sequence Diagram) ของการติดตั้งโปรแกรมสร้างหุ่นยนต์

ซีเควนซ์ไดอะแกรมของการติดตั้งโปรแกรมสร้างหุ่นยนต์ ดังรูปที่ 3.12



รูปที่ 3.12 แสดงซีเควนซ์ไดอะแกรมของการติดตั้งโปรแกรมสร้างหุ่นยนต์

ผู้เล่นทำการติดต่อกับ Web Server เพื่อทำการดาวน์โหลดโปรแกรมสร้างหุ่นยนต์โดยเซิร์ฟเวอร์จะทำการส่งโปรแกรมสร้างหุ่นยนต์มายังผู้เล่นเมื่อผู้เล่นได้รับทำการ Inatall เพื่อทำการติดตั้งโปรแกรมสร้างหุ่นยนต์

3.3.5 ซีเควนซ์ไดอะแกรมของการสร้างหุ่นยนต์

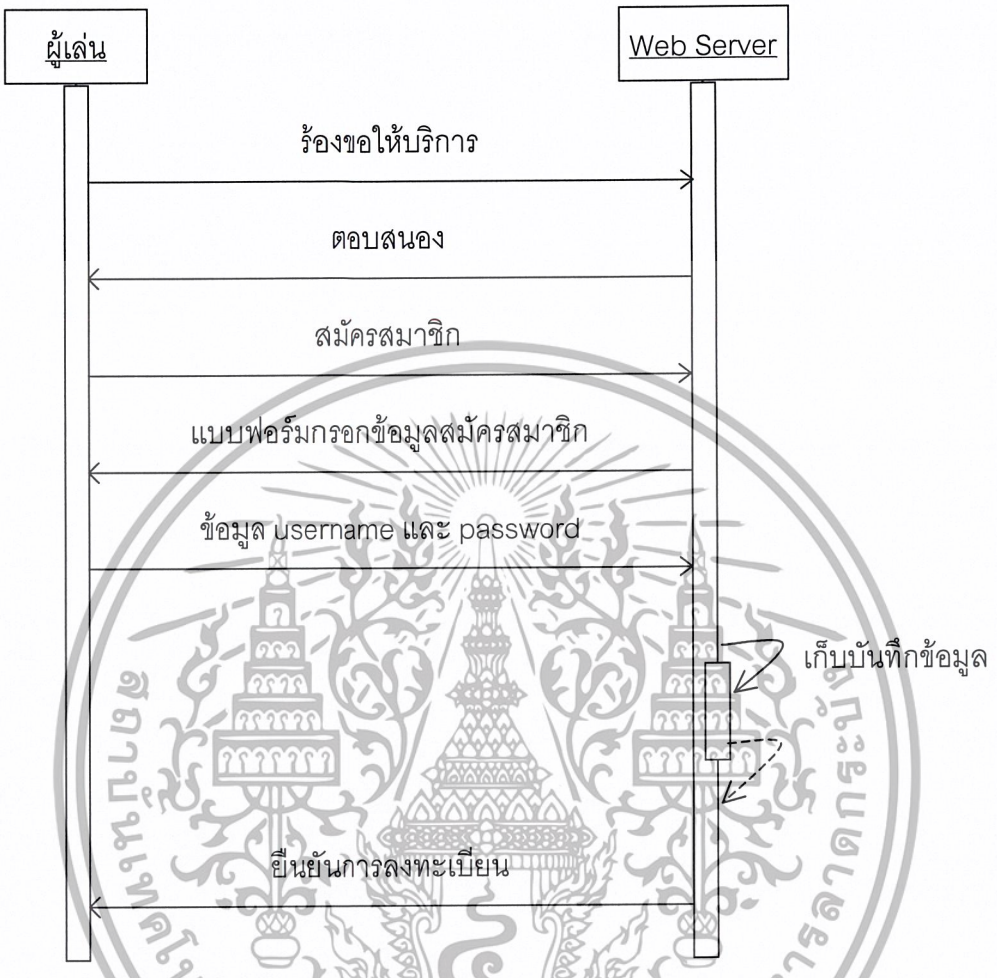
ซีเควนซ์ไดอะแกรมของการสร้างหุ่นยนต์ ดังรูปที่ 3.13

รูปที่ 3.13 แสดงซีเควนซ์ไดอะแกรมของการสร้างหุ่นยนต์

ผู้เล่นทำการเปิดโปรแกรมสร้างหุ่นยนต์จากนั้นเลือกคำสั่งสร้างหุ่นยนต์ ทำการตั้งชื่อให้กับหุ่นยนต์ เลือกส่วนประกอบให้กับหุ่นยนต์ และเขียนโปรแกรมบังคับ จากนั้นทำการกดปุ่ม BUILD เพื่อทำการสร้างหุ่นยนต์โดยโปรแกรมสร้างหุ่นยนต์จะทำการคอมไพล์โปรแกรมที่ผู้เล่นเขียนขึ้นและจะได้ file.class ให้กับผู้เล่น

3.3.6 ซีเควนซ์ไดอะแกรมของการสมัครสมาชิก

ซีเควนซ์ไดอะแกรมของการสมัครสมาชิก ดังรูปที่ 3.14

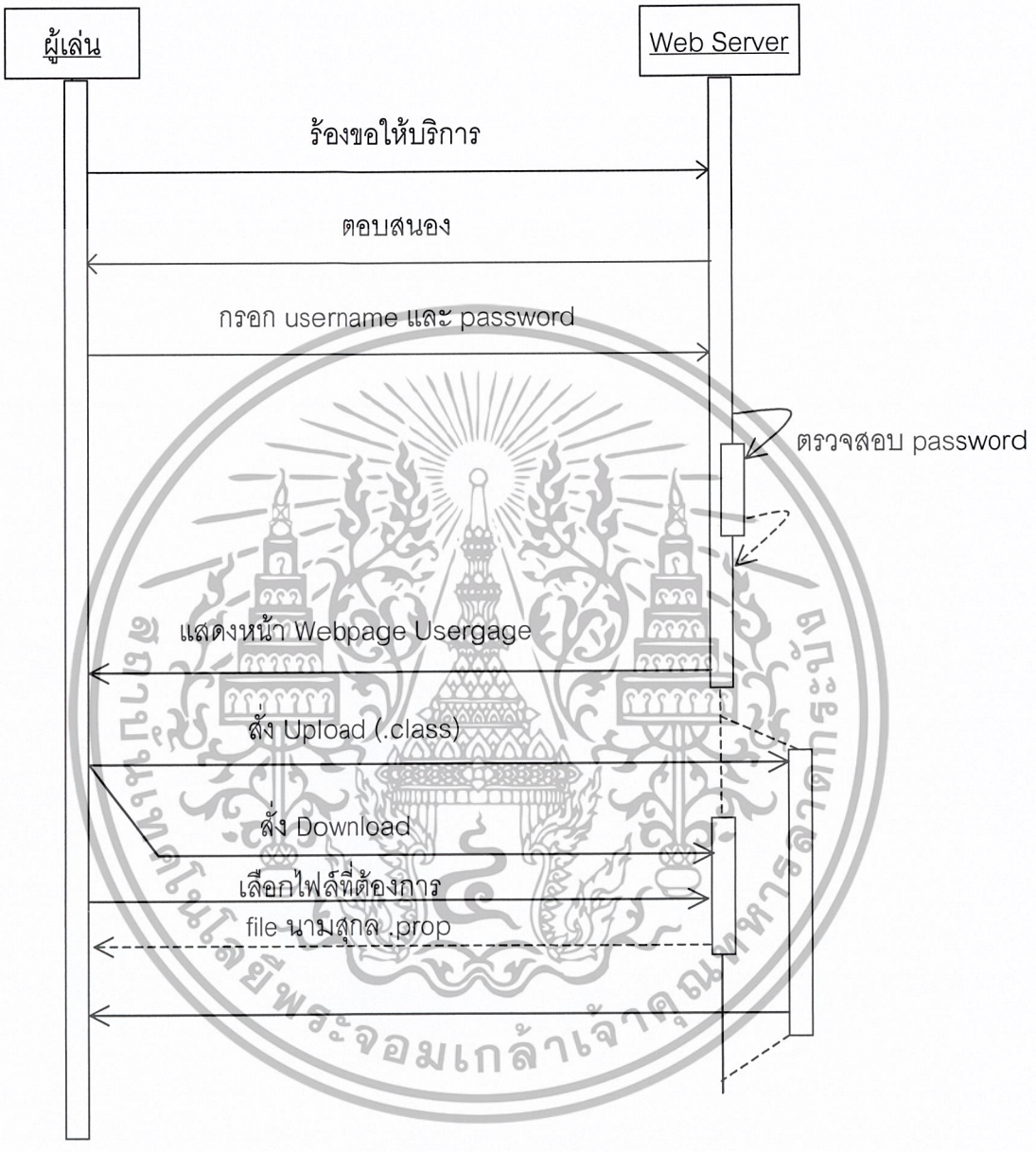


รูปที่ 3.14 แสดงซีเควนซ์ไดอะแกรมของการสมัครสมาชิก

ผู้เล่นทำการติดต่อกับ Web Server เพื่อทำการสมัครสมาชิกโดยกดที่ Sign Up เซิร์ฟเวอร์จะทำการส่งแบบฟอร์มมาให้ผู้เล่นกรอก เมื่อกรอกเสร็จกด Submit เพื่อยืนยันข้อมูล จากนั้นเซิร์ฟเวอร์จะทำการเก็บข้อมูลของผู้เล่นไว้และจะแสดงยืนยันแก่ผู้เล่นว่าลงทะเบียนเรียบร้อยแล้ว

3.3.7 ซีเควนซ์ไดอะแกรมของการ Upload/Download หุ่นยนต์

ซีเควนซ์ไดอะแกรมของการ Unload/Download หุ่นยนต์ ดังรูปที่ 3.15

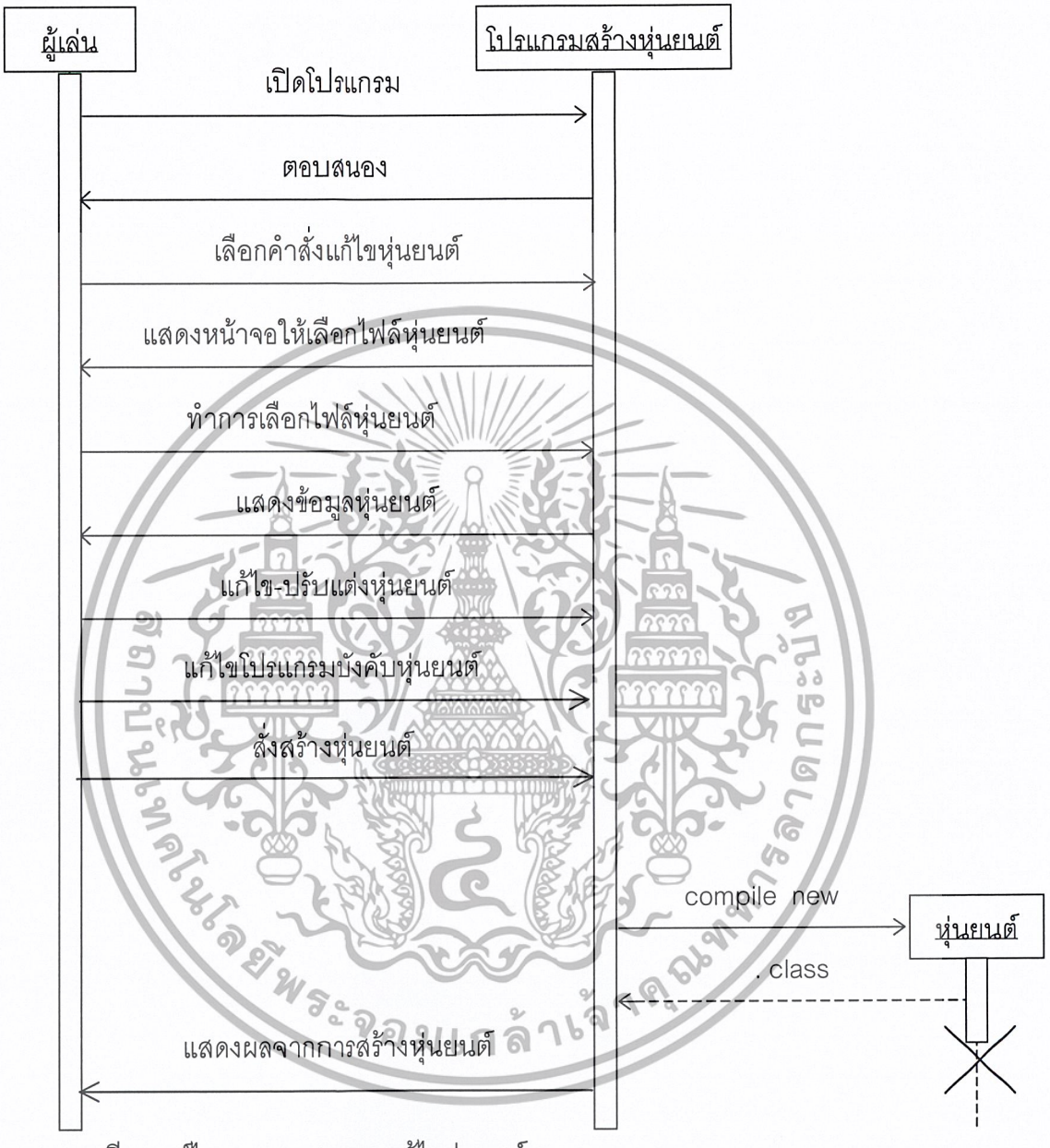


รูปที่ 3.15 แสดงซีเควนซ์ไดอะแกรมของการ Upload/Download หุ่นยนต์

ผู้เล่นทำการติดต่อกับ Web Server เข้าสู่ระบบด้วยการกรอก Username และ Password เซิร์ฟเวอร์ทำการตรวจสอบเมื่อตรวจสอบผ่านทำการแสดงหน้า User Page ซึ่งจะแสดงข้อมูลของผู้เล่น จากนั้นเลือก Upload เพื่อทำการส่งหุ่นยนต์ไปยังเซิร์ฟเวอร์เพื่อเก็บไว้สำหรับใช้งาน หรือเลือก Download เพื่อทำการนำหุ่นยนต์ที่เก็บที่เซิร์ฟเวอร์มาทำการปรับปรุงแก้ไขโดยเลือกไฟล์หุ่นยนต์ที่ต้องการเซิร์ฟเวอร์จะทำการส่ง file.prop ให้กับผู้เล่น

3.3.8 ซีเควนซ์ไดอะแกรมของการแก้ไขหุ่นยนต์

ซีเควนซ์ไดอะแกรมของการแก้ไขหุ่นยนต์ ดังรูปที่ 3.16

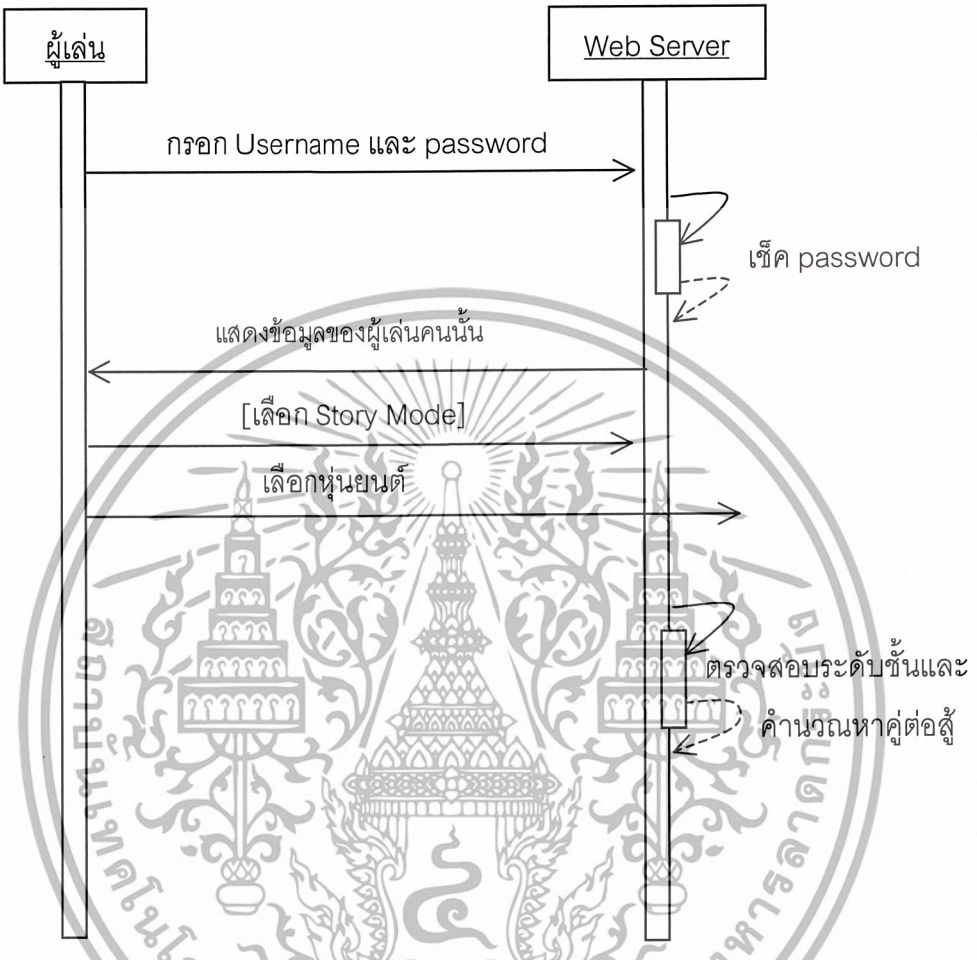


รูปที่ 3.16 แสดงซีเควนซ์ไดอะแกรมของการแก้ไขหุ่นยนต์

ผู้เล่นทำการเปิดโปรแกรมสร้างหุ่นยนต์เลือกคำสั่งแก้ไขหุ่นยนต์ จากนั้นทำการเลือกไฟล์หุ่นยนต์ที่ต้องการแก้ไข โปรแกรมจะแสดงข้อมูลส่วนประกอบเก่าของหุ่นยนต์ ทำการแก้ไขส่วนประกอบ จากนั้นจะแสดงข้อมูลเก่าของโปรแกรมบังคับหุ่นยนต์ ทำการแก้ไขโปรแกรมบังคับหุ่นยนต์ เมื่อแก้ไขเสร็จแล้วทำการสร้างหุ่นยนต์ โปรแกรมจะทำการคอมไพล์จะได้ file.class ให้กับผู้เล่น

3.3.9 ซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบสตอรี่โหมด (Story Mode)

ซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบสตอรี่โหมด ดังรูปที่ 3.17

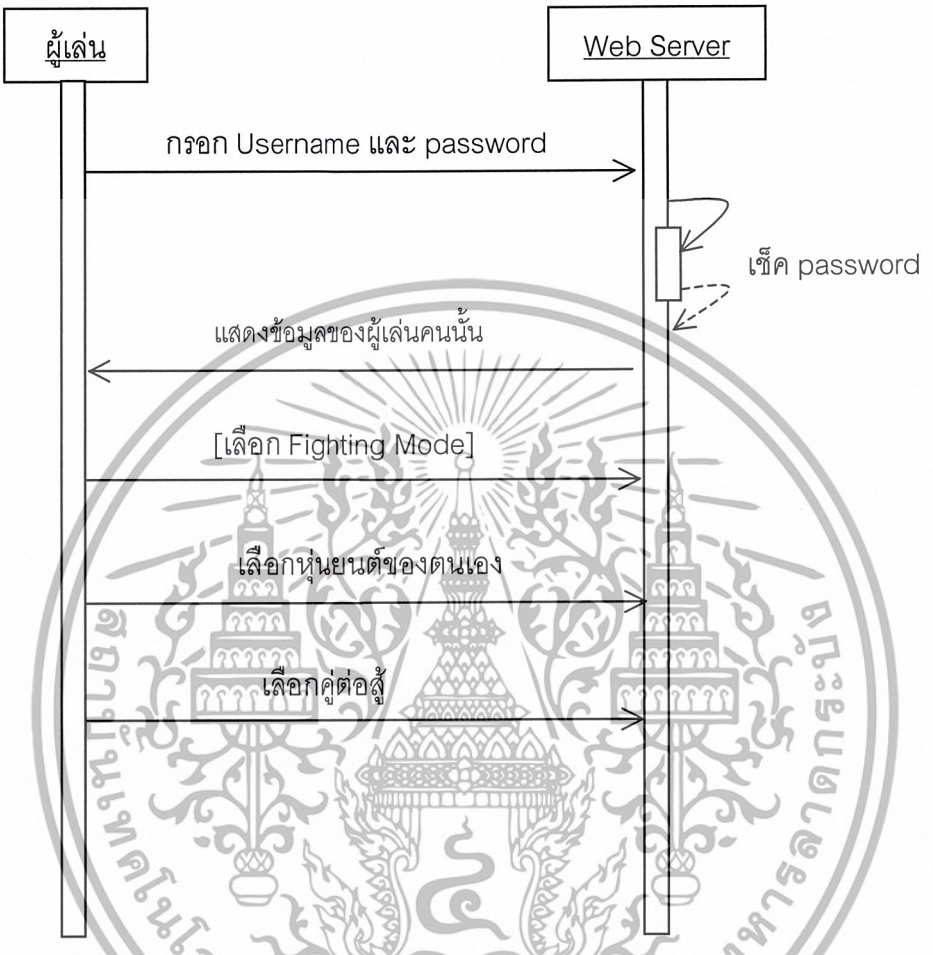


รูปที่ 3.17 แสดงซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบสตอรี่โหมด

ผู้เล่นเข้าสู่ระบบด้วยการกรอก Username และ password ให้เซิร์ฟเวอร์ทำการตรวจสอบเมื่อตรวจสอบพบเซิร์ฟเวอร์จะทำการส่งข้อมูลของผู้เล่นคนนั้นกลับมา ผู้เล่นทำการเลือกแบบสตอรี่หมด เซิร์ฟเวอร์จะทำการคำนวณระดับชั้นของหุ่นยนต์ที่จะต้องสู้ด้วย จากนั้นหุ่นยนต์จะถูกส่งไปรอทำการแข่งขัน

3.3.10 ซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบไฟต์ติ้งโหมด (Fighting Mode)

ซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบไฟต์ติ้งโหมด ดังรูปที่ 3.18

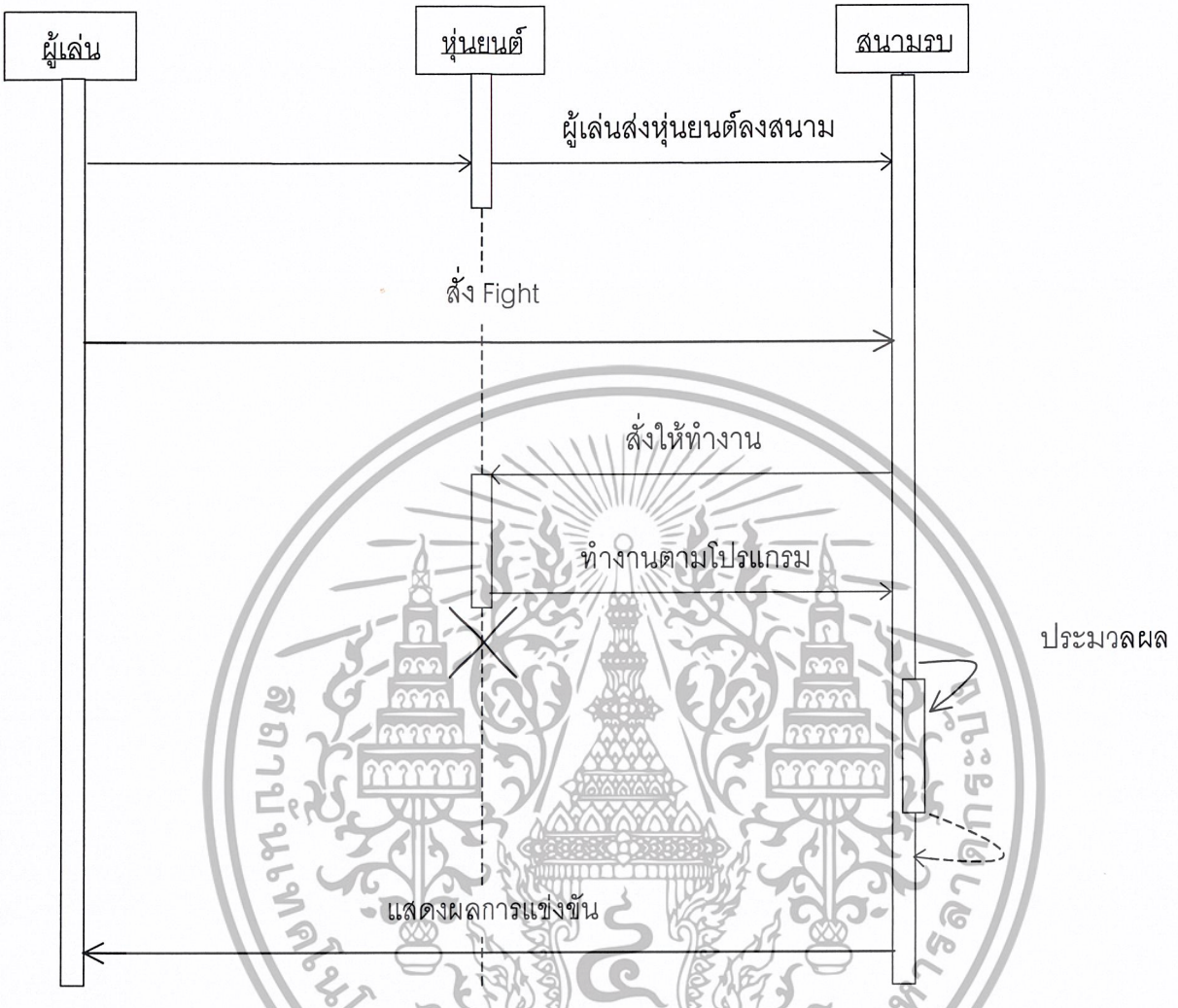


รูปที่ 3.18 แสดงซีเควนซ์ไดอะแกรมของการส่งหุ่นยนต์เข้าแข่งขันแบบไฟต์ติ้งโหมด

ผู้เล่นเข้าสู่ระบบด้วยการกรอก Username และ password ให้เซิร์ฟเวอร์ทำการตรวจสอบ เมื่อตรวจสอบพบเซิร์ฟเวอร์จะทำการส่งข้อมูลของผู้เล่นคนนั้นกลับมา ผู้เล่นทำการเลือกแบบไฟต์ติ้งโหมดผู้เล่นเลือกหุ่นยนต์ของตนเอง หุ่นยนต์ที่ต้องการสู้ด้วย จากนั้นหุ่นยนต์จะถูกส่งไปทำการแข่งขัน

3.3.10 ซีเควนซ์ไดอะแกรมของการรบ

ซีเควนซ์ไดอะแกรมของการรบ ดังรูปที่ 3.19



รูปที่ 3.19 แสดงซีเควนซ์ไดอะแกรมของการรบ

ผู้เล่นส่งหุ่นยนต์เข้าแข่งขันโดยส่งไปที่สนามรบ สนามรบจะทำการแสดงหน้าจอของสนามรบ ผู้เล่นเป็นผู้สั่ง Fight ให้เริ่มทำการต่อสู้ สนามรบจะเป็นตัวที่สั่งให้หุ่นยนต์ทำงาน โดยหุ่นยนต์จะทำงานตามโปรแกรมที่ผู้เล่นเขียนไว้ โดยสนามรบจะเป็นตัวประมวลผลคำสั่งว่าหุ่นยนต์จะต้องทำอะไร และเมื่อสิ้นสุดการต่อสู้สนามรบจะทำการแสดงผลการแข่งขันให้กับผู้เล่น

3.4 การประมวลผลหลักของเกม (Game Engine)

จะแบ่งการทำงานออกเป็นส่วนต่างๆ ดังนี้

3.4.1 การทำงานของสนาม

สนามสามารถแบ่งการทำงานหลักๆออกเป็น 2 ส่วน

- Thread
- Non-Thread

Thread จะทำหน้าที่ดังต่อไปนี้

1) นำเมสเซจ (Message) จากคอลเลคชัน (Collection) มาประมวลผลโดยคอลเลคชันเป็นที่ๆ เก็บรวบรวมเมสเซจที่หุ่นยนต์จะทำงานโดยคอลเลคชันจะมีลักษณะเป็นคิว (Queue)

2) วาดภาพลงบนแอปเพล็ต

3) ตรวจจับเหตุการณ์ที่เกิดขึ้นกับหุ่นยนต์แล้วแจ้งให้หุ่นยนต์รับรู้

Non-Thread จะทำหน้าที่ดังต่อไปนี้

ให้หุ่นยนต์สามารถส่งเมสเซจมาเก็บสะสมลงคอลเลคชัน เมื่อหุ่นยนต์อยู่ในเหตุการณ์ต่างๆ

3.4.2 การติดต่อระหว่างหุ่นยนต์กับสนาม

- ใช้ลักษณะการทำงานแบบ Messaging & Queue
- ในสนามจะมีคอลเลคชันที่ใช้รวบรวมเมสเซจของหุ่นยนต์ โดยจะแยกระหว่างหุ่นยนต์แต่ละตัวโดยคอลเลคชันจะทำงานในลักษณะคิว
- หุ่นยนต์จะส่งเมสเซจมาให้กับสนามเพื่อบอกว่าต้องการทำอะไร และสนามจะเก็บเมสเซจของหุ่นยนต์ที่ยังไม่ได้รับการประมวลผลไว้ในคอลเลคชัน

3.4.3 การตรวจจับเหตุการณ์

ในการประมวลผลเมสเซจคำสั่งของหุ่นยนต์ สนามจะคอยตรวจสอบว่ามีเหตุการณ์ใดที่เกิดขึ้นกับหุ่นยนต์บ้าง แล้จึงแจ้งให้หุ่นยนต์ทราบว่าเกิดเหตุการณ์ใดขึ้น

เหตุการณ์ที่สนามจะสามารถตรวจพบมีดังนี้

- Attack จะตรวจพบเมื่อหุ่นยนต์ถูกกระสุนของฝ่ายตรงข้าม
- FoundBorder จะตรวจพบเมื่อหุ่นยนต์สแกนพบขอบสนาม
- FoundEnermy จะตรวจพบเมื่อหุ่นยนต์สแกนพบหุ่นยนต์ฝ่ายตรงข้าม
- CrashBorder จะตรวจพบเมื่อหุ่นยนต์ชนถูกขอบสนาม
- CrashEnermy จะตรวจพบเมื่อหุ่นยนต์ชนถูกหุ่นยนต์ฝ่ายตรงข้าม

เมื่อหุ่นยนต์รับทราบว่าเหตุการณ์ใดเกิดขึ้น ก็จะส่งเมสเซจที่จัดการกับเหตุการณ์นั้นๆให้กับสนาม และสนามจะทำการตัดสินใจเองว่าอยู่ในสถานะที่สามารถเก็บคำสั่งของหุ่นยนต์ได้หรือไม่

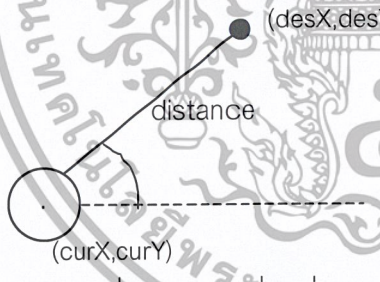
3.4.4 การควบคุมเหตุการณ์ที่เกิดขึ้น

การจัดการกับเหตุการณ์ต่างๆจะมีคำสั่งดังนี้

- การเคลื่อนที่ของหุ่นยนต์ (Move)
- การหมุนของหุ่นยนต์ (Rotate)
- การยิงของหุ่นยนต์ (Shoot)
- ควบคุมให้หุ่นยนต์หยุดการส่งเมสเซจในการจัดการกับเหตุการณ์นั้นไปยังคอลเลคชั่น (eventStop)
- ควบคุมให้หุ่นยนต์สามารถส่งเมสเซจในการจัดการกับเหตุการณ์นั้นไปยังคอลเลคชั่น (eventStart)
- ควบคุมให้สนามทำการลบคำสั่งเก่าที่อยู่ในคอลเลคชันตามลำดับของเหตุการณ์ (eventClear)

3.4.5 การประมวลผลคำสั่งต่าง ๆของหุ่นยนต์

3.4.5.1 การเคลื่อนที่ของหุ่นยนต์



รูปที่ 3.20 แสดงตำแหน่งที่ใช้ในการเคลื่อนที่

ทำการคำนวณตำแหน่ง (desX,desY) ของหุ่นยนต์ที่ที่จะทำการเดินไปใหม่ได้โดยพิจารณาจากสูตร

$desX = curX + Distance*Cos(\alpha)$

$desY = curY - Distance*Sin(\alpha)$

โดยสาเหตุที่แกน Y ใช้เครื่องหมายเป็น – เพราะแกน Y ของคอมพิวเตอร์จะตรงกันข้ามกับแกน Y ทางคณิตศาสตร์

จากนั้นทำการเปลี่ยนตำแหน่งของหุ่นยนต์ทีละช่องโดยพิจารณาจากสูตร

y = mx + c

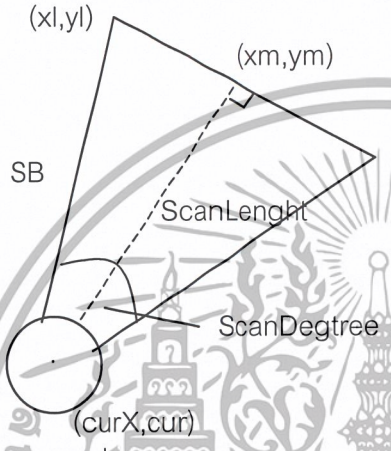
สมมติว่าที่จุด curX,curY เป็นจุด Origin ดังนั้นจะได้เป็น

y = mx

จะสามารถหา m ได้จาก

m = (desY-curY) / (desX-curX)

3.4.5.2 การสแกน



รูปที่ 3.21 แสดงตำแหน่งที่ใช้ในการสแกน

การคำนวณจุด Xm,Ym จะใช้วิธีการเดียวกับการหาตำแหน่งการเคลื่อนที่ โดยหาจากสูตร

Xm = curX + Distance * Cos(ScanDegree/2)

Ym = curY + Distance * Sin(ScanDegree/2)

คำนวณหา SB จากสูตร

SB = ScanLength/Cos(ScanDegree/2)

จากนั้นนำ SB ที่ได้ไปหาจุด Xl,Yl,Xr,Yr

Xl = curX + (SB*Cos((ScanDegree/2)+FaceDegree))

Yl = curY - (SB*Sin((ScanDegree/2)+FaceDegree))

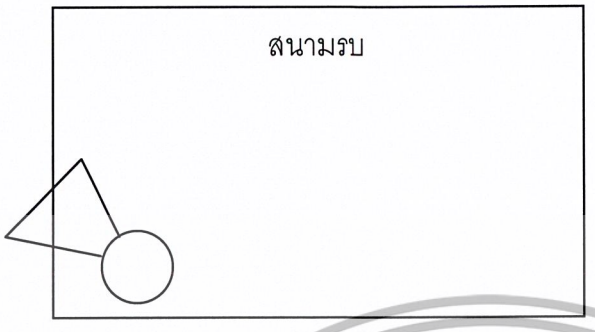
Xr = curX + (SB*Cos(FaceDegree-(ScanDegree/2)))

Yr = curY - (SB*Sin(FaceDegree-(ScanDegree/2)))

เมื่อทำการหาจุด Xm,Ym,Xl,Yl,Xr,Yr ซึ่งเป็นจุดยอดของการสแกนครบทั้งหมดแล้ว ก็ทำการลากเส้นตรงเชื่อมระหว่างจุดทั้ง 3 ก็จะได้เป็นขอบเขตของการสแกน

3.4.5.3 **การสแกนขอบสนาม**

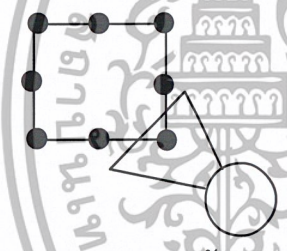
ในการสแกนถ้าพบว่าจุดใดจุดหนึ่งของขอบสนามอยู่ในขอบเขตของการสแกนก็จะถือว่าเจอขอบสนาม



รูปที่ 3.22 แสดงการสแกนพบขอบสนาม

3.4.5.4 **การสแกนพบศัตรู**

กำหนดจุดของหุ่นยนต์เป็น 8 จุด ดังนี้



รูปที่ 3.23 แสดงการสแกนพบศัตรู

ถ้าพบว่าจุดใดจุดหนึ่งใน 8 จุดซึ่งเป็นของหุ่นยนต์ฝั่งตรงข้ามอยู่ในขอบเขตของการสแกนจะถือว่าสแกนพบศัตรู

3.4.5.5 **การยิงกระสุน**

หลักการเคลื่อนที่ของกระสุนจะใช้การทำงานเช่นเดียวกับการเคลื่อนที่ของหุ่นยนต์และสนามจะเป็นตัวที่ใช้ควบคุมกระสุนที่ถูกยิงออกมา โดยจะมีคลาส ManageBullet เป็นตัวเก็บกระสุนที่ถูกยิงออกมา และเมื่อกระสุนเคลื่อนที่ไปจนถึงจุดๆหนึ่งก็จะถูกทำลายโดยการลบกระสุนออกจาก ManageBullet

3.4.5.6 **การคำนวณความแม่นยำในการยิงกระสุน**

ค่าความแม่นยำของอาวุธและกระสุนจะส่งผลต่ออัตราการเบี่ยงเบนของกระสุนขณะที่ถูกยิงออกมาจากอาวุธโดยหาจากสูตร

θ = constant – (WeaponAccuracy+AmmoAccuracy)

เมื่อหาองศาที่เบี่ยงเบนของกระสุนได้แล้ว ต้องนำมาคำนวณว่ากระสุนจะเบนขึ้นหรือเบนลง โดยจะทำการสุ่มตัวเลขดังนี้

สุ่มตัวเลขระหว่าง 0 – 1

ถ้า x <= 0.5

กระสุนจะถูกยิงออกไปด้วยมุมที่เท่ากับ FaceDegree + θ

ถ้าไม่อย่างนั้นแล้ว

กระสุนจะถูกยิงออกไปด้วยมุมที่เท่ากับ FaceDegree - θ

เมื่อกระสุนกระทบกับหุ่นยนต์จะมีการคำนวณว่าถูกส่วนใดของหุ่นยนต์ โดยจะมีวิธีการคำนวณดังนี้

1) คำนวณหาขนาดรวมของหุ่นยนต์ โดยหาจากผลรวมของส่วนประกอบต่างๆของหุ่นยนต์

2) ความน่าจะเป็นที่จะยิงโดนส่วนต่างๆยิงหุ่นยนต์

= ขนาดของส่วนประกอบ / ขนาดรวมของหุ่นยนต์

ความสัมพันธ์ระหว่างขนาดกับน้ำหนักคือ

ขนาด = น้ำหนัก/10

3) กำหนดความน่าจะเป็นของส่วนประกอบของหุ่นยนต์โดยใช้ความน่าจะเป็นแบบสะสม

ตารางที่ 3.2 แสดงความน่าจะเป็นของส่วนประกอบของหุ่นยนต์

ส่วนประกอบ	ความน่าจะเป็น	ความน่าจะเป็นสะสม	ช่วงความน่าจะเป็น
Body	0.4	0.4	0-0.40
ArmLeft	0.1	0.5	0.41-0.60
ArmRight	0.1	0.6	0.51-0.60
ShoulderLeft	0.1	0.7	0.61-0.70
ShoulderRight	0.1	0.8	0.71-0.80
Leg	0.2	1.0	0.81-1.00

ทำการสุ่มตัวเลขขึ้นมาโดยมีค่าระหว่าง 0 – 1.0 โดยที่ถ้าเลขที่สุ่มออกมาอยู่ในช่วงของส่วนประกอบใดก็จะยิงถูกส่วนประกอบชิ้นนั้น

บทที่ 4
การใช้งานโปรแกรม

จากการทดลองได้สร้าง 2 ส่วน คือ ส่วนของแอปพลิเคชันและ ส่วนของเว็บไซท์ในส่วนของแอปพลิเคชันจะเป็นส่วนที่ใช้สำหรับการสร้างหุ่นยนต์โดยมีการเลือกส่วนประกอบต่างๆ ประกอบขึ้นเป็นหุ่นยนต์ และเขียนโปรแกรมเพื่อสั่งให้หุ่นยนต์ทำงานในเหตุการณ์ต่างๆ ซึ่งจะสามารถดูรายละเอียดเพิ่มเติมในแต่ละส่วนสามารถทำอะไรได้บ้างที่ภาคผนวก และในส่วนของเว็บไซท์ใช้ในการแสดงรายละเอียดต่างๆของการเล่นเกม การสมัครสมาชิกก่อนที่จะเล่นเกม JavaRobot การดาวน์โหลดโปรแกรมที่จำเป็นที่จะต้องใช้ในการเล่นเกม เช่น โปรแกรม JavaRobot Creater การ Upload Robot เพื่อให้ผู้เล่นส่งหุ่นยนต์ของตัวเองที่ต้องการใช้ในการเล่น การ Download Robot เพื่อให้ผู้เล่นดาวน์โหลดหุ่นยนต์ไปทำการเปลี่ยนแปลงแก้ไข การให้ผู้เล่น Login เพื่อทำการเลือกโหมดการเล่นว่าจะเล่นแบบสตอรี่โหมดหรือ ไฟต์ติ้งโหมด โดยที่การแสดงการต่อสู้ของหุ่นยนต์ 2 ตัว เป็นรูปภาพ 2 มิติ

4.1 การทำงานของโปรแกรม

การทำงานทั้งหมดแบ่งเป็น 2 ส่วนดังนี้

4.1.1.ส่วนของโปรแกรมสำหรับสร้างหุ่นยนต์ประกอบด้วย

ตารางที่ 4.1 แสดงรายละเอียดส่วนของโปรแกรมสำหรับสร้างหุ่นยนต์

ส่วนประกอบ	รายละเอียด
หน้าสำหรับเลือกการทำงาน	เป็นหน้าแรกที่ใช้สำหรับเลือกว่าจะทำการสร้างหุ่นยนต์ใหม่ ทำการปรับปรุงแก้ไขหุ่นยนต์ตัวเดิมที่มีอยู่ หรือทำการออกจากโปรแกรม
หน้าสำหรับเลือกส่วนประกอบ	เป็นหน้าที่ใช้สำหรับเลือกชิ้นส่วนเพื่อมาประกอบเป็นหุ่นยนต์
หน้าสำหรับใส่คำสั่ง	เป็นหน้าที่ใช้สำหรับให้ผู้เล่นเขียนโปรแกรมบังคับหุ่นยนต์
หน้าสำหรับเลือกหุ่น	เป็นหน้าที่ใช้สำหรับเลือกไฟล์หุ่นยนต์ที่ต้องการนำมาปรับปรุงแก้ไข
หน้าส่วนประกอบ	เป็นหน้าที่แสดงค่าคุณสมบัติของแต่ละชิ้นส่วนโดยที่ชิ้นส่วนแต่ละชนิดจะมีหลายชิ้นเพื่อให้ผู้เล่นทำการเลือกตามความต้องการ

4.1.2 ส่วนของเว็บไซท์ประกอบด้วย

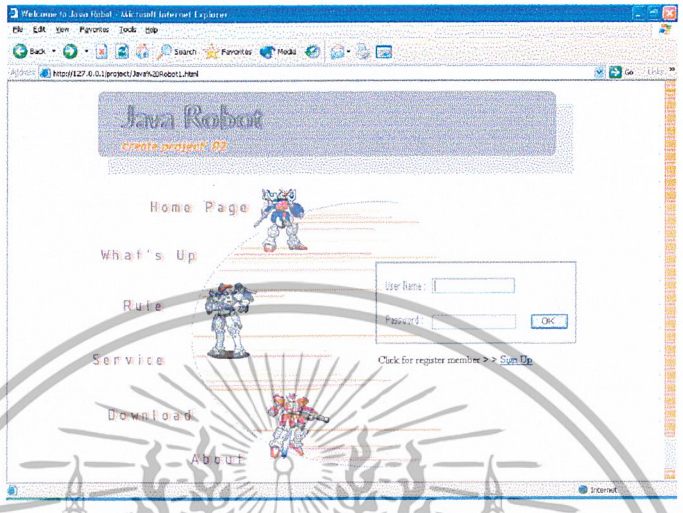
ตารางที่ 4.2 แสดงรายละเอียดส่วนของเว็บไซท์

ส่วนประกอบ	รายละเอียด
Homepage	เป็นหน้าสำหรับทำการ Login
What's Up	เป็นหน้าสำหรับอธิบายเกม JavaRobot ว่าเป็นเกมประเภทไหน
Rule	เป็นหน้าสำหรับอธิบายขั้นตอนในการเล่นและกฎของเกม Java Robot
Service	เป็นหน้าสำหรับอธิบายว่าเว็บไซท์นี้ให้บริการดาวน์โหลดโปรแกรมอะไรบ้าง โดยมีความจำเป็นในการเล่นเกม JavaRobot อย่างไร
Download	เป็นหน้าสำหรับดาวน์โหลดโปรแกรมที่จำเป็นในการเล่นเกม Java Robot
About	เป็นหน้าสำหรับบอกรายละเอียดของผู้สร้าง
Profile Information	เป็นหน้าสำหรับให้ผู้เล่นทำการสมัครสมาชิก
User Page	เป็นหน้าสำหรับบอกรายละเอียดเกี่ยวกับผู้เล่นว่ามีหุ่นยนต์ทั้งหมดกี่ตัว สถิติการแข่งขันเป็นอย่างไร เป็นต้น
Story Mode	เป็นหน้าสำหรับให้ผู้เล่นเลือกหุ่นยนต์ของตัวเองเข้าทำการแข่งขันตามแต่ระดับของหุ่นยนต์แต่ละตัว
Fighting Mode	เป็นหน้าสำหรับให้ผู้เล่นเลือกหุ่นยนต์ของตัวเองและหุ่นยนต์ที่ผู้เล่นต้องการสู้ด้วย
Upload Robot	เป็นหน้าสำหรับให้ผู้เล่นทำการ Upload หุ่นยนต์ของตัวเองไปเก็บที่เซิร์ฟเวอร์และยังให้ผู้เล่นกำหนดว่าหุ่นยนต์ตัวนี้ยอมอนุญาตให้ผู้เล่นคนอื่นเลือกทำการแข่งขันด้วยหรือไม่
Download Robot	เป็นหน้าสำหรับให้ผู้เล่นดาวน์โหลดหุ่นยนต์ของตัวเอง
หน้าแสดงผล	เป็นหน้าสำหรับแสดงผลการต่อสู้ของหุ่นยนต์ จากเหตุการณ์ต่างๆ 6 เหตุการณ์ที่สามารถเกิดขึ้นได้ คือ เหตุการณ์ปกติ เหตุการณ์พบศัตรู เหตุการณ์โดนถูกตี เหตุการณ์พบขอบสนาม เหตุการณ์ชนศัตรู เหตุการณ์ชนขอบสนาม

4.2 การเล่นเกม

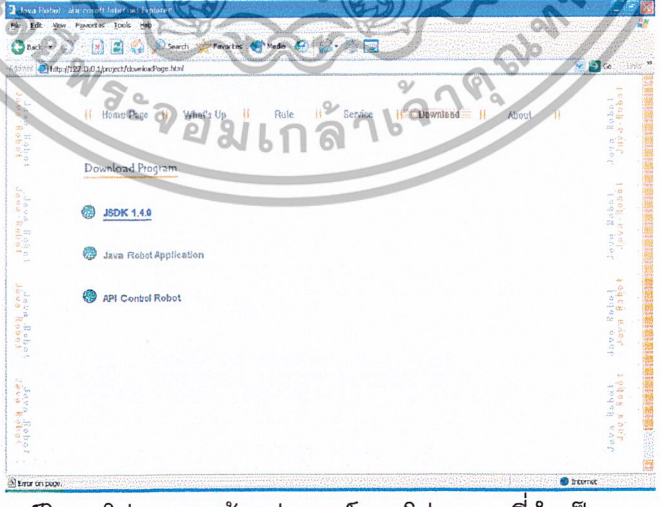
4.2.1 การดาวน์โหลดโปรแกรมและการสร้างหุ่นยนต์

เริ่มต้นจากการ Online เข้าสู่เว็บไซท์ของ JavaRobot ดังรูปที่ 4.1



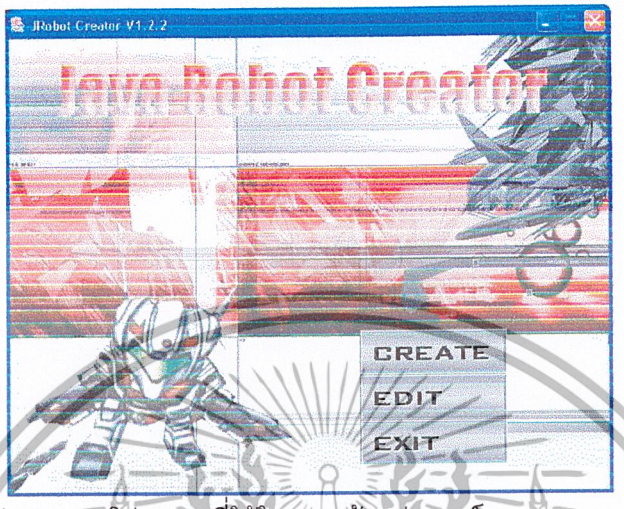
รูปที่ 4.1 แสดงเว็บไซท์หน้าแรกของเว็บไซท์ JavaRobot

จากนั้นคลิก Download เพื่อเข้าสู่หน้าดาวน์โหลด ดังรูปที่ 4.2 ในหน้านี้จะมีโปรแกรมให้ผู้เล่นทำการดาวน์โหลดโปรแกรมที่จำเป็นสำหรับการเล่นเกมดังนี้ โปรแกรม JDK เป็นคอมไพล์เลอร์ที่ใช้ในการสร้างหุ่นยนต์ โปรแกรม XML Parser ใช้ในการติดต่อกับไฟล์ XML โปรแกรมในการสร้างหุ่นยนต์ และ API Control Robot เป็นชุดคำสั่งที่ใช้ในการควบคุมหุ่นยนต์



รูปที่ 4.2 แสดงหน้าดาวน์โหลดโปรแกรมสร้างหุ่นยนต์และโปรแกรมที่จำเป็น

เมื่อทำการติดตั้งโปรแกรมที่ดาวน์โหลดมาเรียบร้อยแล้ว เริ่มต้นผู้เล่นต้องทำการสร้างหุ่นยนต์ของตัวเองก่อน โดยเปิดโปรแกรม JavaRobot Creater ที่ใช้ในสร้างหุ่นยนต์ เมื่อเปิดขึ้นมาจะเป็นดังรูปที่ 4.3



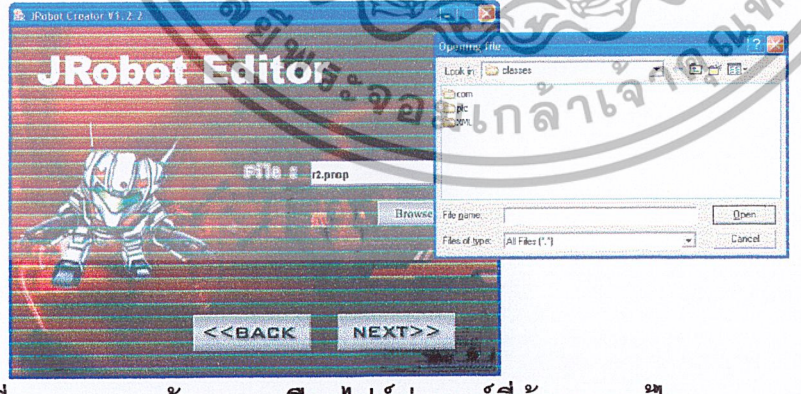
รูปที่ 4.3 แสดงหน้าแรกของโปรแกรมที่ใช้ในการสร้างหุ่นยนต์

- **ผู้เล่นที่ทำการเล่นเป็นครั้งแรกหรือต้องการสร้างหุ่นยนต์ตัวใหม่**

ทำการคลิกปุ่ม CREATE เพื่อทำการสร้างหุ่นยนต์

- **ผู้เล่นที่เคยสร้างหุ่นยนต์ไว้แล้วต้องการปรับปรุงแก้ไข**

ทำการคลิกปุ่ม EDIT เพื่อที่จะทำการปรับปรุงแก้ไขหุ่นยนต์ที่ผู้เล่นได้เคยสร้างไว้แล้ว โดยทำการเลือกไฟล์หุ่นยนต์ที่ต้องการแก้ไข ดังรูปที่ 4.4



รูปที่ 4.4 แสดงหน้าจอการเลือกไฟล์หุ่นยนต์ที่ต้องการแก้ไข

- EXIT เป็นการออกจากโปรแกรม

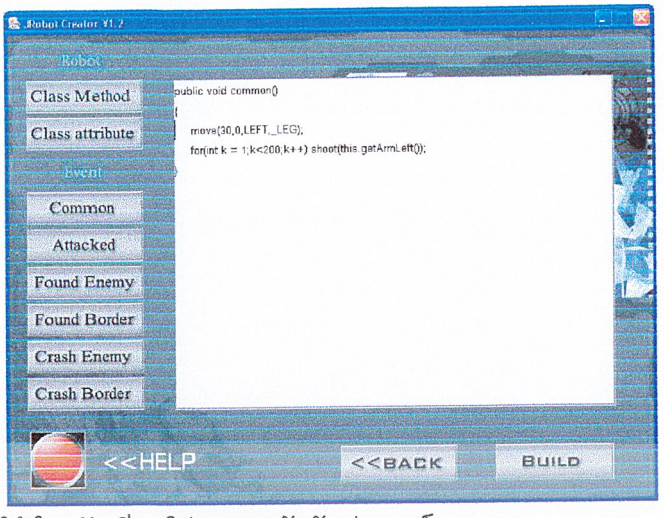
จากนั้นทำการตั้งชื่อหุ่นยนต์และเลือกส่วนประกอบดังรูปที่ 4.5

รูปที่ 4.5 แสดงหน้าที่ใช้เลือกส่วนประกอบ

โดยเมื่อทำการเลือกส่วนประกอบจะแสดงหน้าจอที่แสดงถึงส่วนประกอบประเภทนั้นๆ เช่น เมื่อเลือกขาจะแสดงหน้าจอที่ใช้ทำการเลือกขา

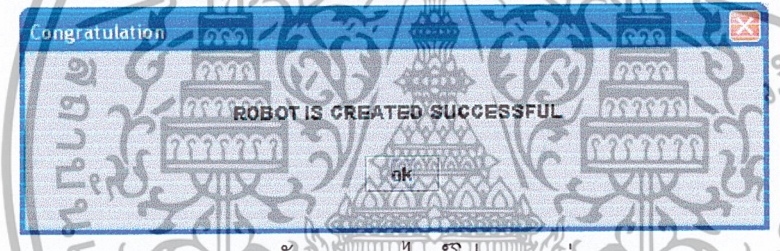
รูปที่ 4.6 แสดงหน้าจอที่ทำการเลือกขา

เมื่อเราทำการเลือกส่วนประกอบ(กด OK) เงินที่เรามีอยู่จะทำการลดลงโดยอัตโนมัติ แต่ถ้าเราไม่ต้องการเลือก(กด CANALE) จะไม่ทำการเลือกส่วนประกอบนั้น เมื่อเลือกส่วนประกอบเสร็จ แล้วคลิกปุ่ม NEXT เพื่อเข้าสู่หน้าที่ใช้เขียนโปรแกรมบังคับหุ่นยนต์ดังรูปที่ 4.7



รูปที่ 4.7 แสดงหน้าที่ใช้สำหรับเขียนโปรแกรมบังคับหุ่นยนต์

เมื่อเขียนโปรแกรมในการบังคับหุ่นเรียบร้อยแล้วคลิกปุ่ม Generate เพื่อทำการคอมไพล์โปรแกรมที่ผู้เล่นเขียนขึ้น เมื่อคอมไพล์ผ่านแล้วจะขึ้น message ดังรูปที่ 4.8



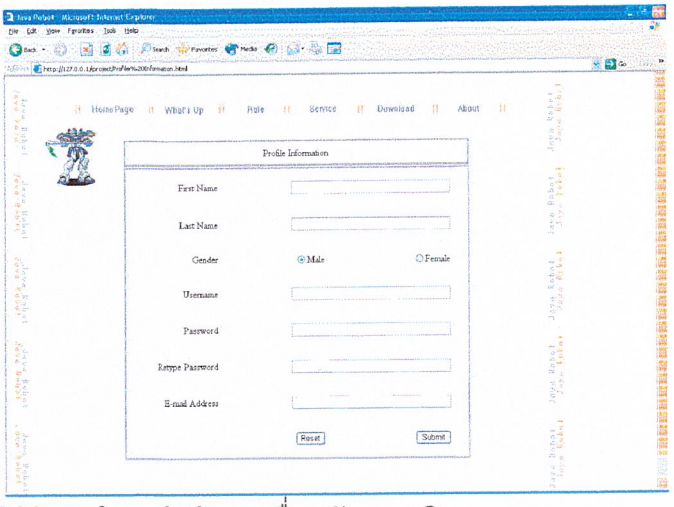
รูปที่ 4.8 แสดง message box หลังจากคอมไพล์โปรแกรมผ่าน

4.2.2 การเข้าสู่ระบบและการนำหุ่นยนต์ที่สร้างไปใช้งาน

เมื่อสร้างหุ่นยนต์เสร็จแล้วทำการ Online เข้าสู่เว็บไซท์ JavaRobot

- **ผู้เล่นที่ทำการเล่นเป็นครั้งแรก**

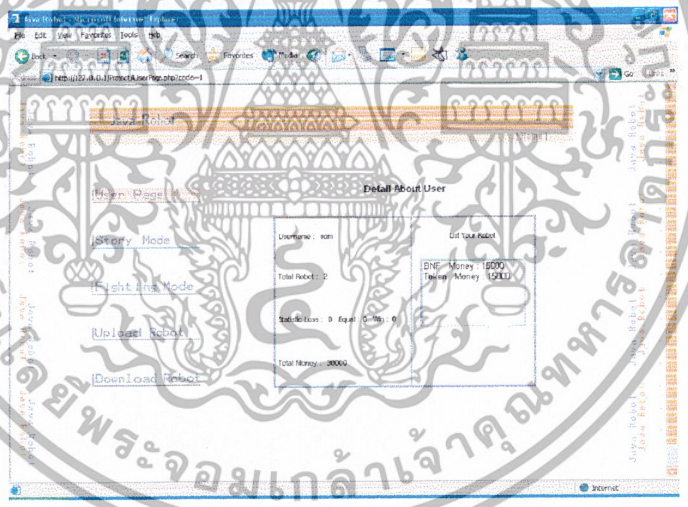
ทำการสมัครสมาชิกโดยคลิกที่ Sign Up จะเชื่อมต่อไปยังหน้าที่ให้ใส่ข้อมูล ดังรูปที่ 4.9 โดยผู้เล่นจะต้องทำการกรอกข้อมูลลงในแบบฟอร์มที่กำหนดไว้ให้ครบถ้วน เพื่อนำ Username และ Password ไปใช้ในการเข้าไปใช้บริการ



รูปที่ 4.9 แสดงหน้าที่ให้ผู้เล่นทำการใส่ข้อมูลเพื่อสมัครสมาชิก

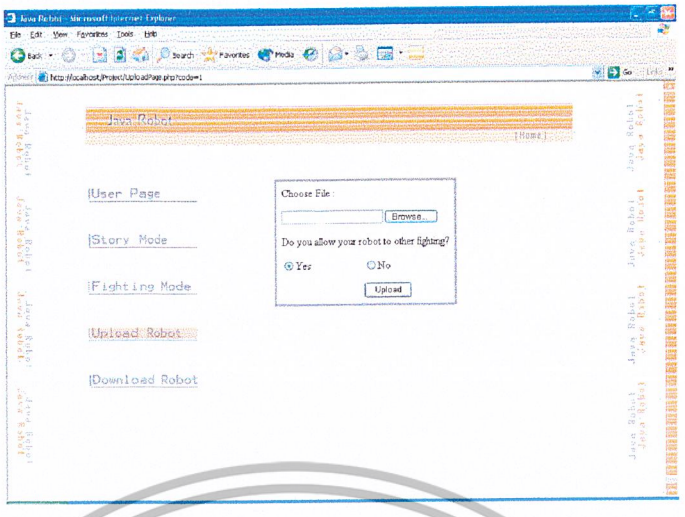
- **ผู้เล่นทำการสมัครสมาชิกแล้ว**

ทำการ Log in เข้าสู่ระบบโดยใส่ Username และ Password ในหน้าแรก จะเกิดหน้า User Page ซึ่งแสดงข้อมูลของสมาชิกดังรูปที่ 4.10



รูปที่ 4.10 แสดงหน้าข้อมูลของสมาชิก

ทำการ Upload หุ่นยนต์ที่สร้างขึ้นไปเก็บยังเซิร์ฟเวอร์โดยคลิกที่ Upload จะเกิดหน้าที่ให้ผู้เล่นเลือกไฟล์ที่ต้องการนำไปเก็บดังรูปที่ 4.11



รูปที่ 4.11 แสดงหน้า Upload

เมื่อทำการ Upload หุ่นยนต์เสร็จเรียบร้อยแล้วทำการเลือกการเล่นในแบบที่ต้องการ

4.2.3 การเลือกวิธีการเล่น

การเล่นเกมจะแบ่งออกเป็น 2 ลักษณะคือ การเล่นแบบสตอรี่โหมดและ แบบไฟต์ติ้งโหมด โดยมีรายละเอียดดังนี้

4.2.3.1 การเล่นแบบสตอรี่โหมด (Story Mode)

ในการเลือกเล่นแบบสตอรี่โหมดจะเป็นการต่อสู้กับหุ่นยนต์ที่ถูกกำหนดไว้โดยจะมีการกำหนดระดับของหุ่นยนต์ไว้ โดยที่ผู้เล่นจะต้องเลือกหุ่นยนต์เพื่อมาทำการต่อสู้กับหุ่นยนต์ในแต่ละระดับเพื่อให้หุ่นยนต์ของตนเองมีระดับที่สูงขึ้นและจะมีการเพิ่มจำนวนเงินและประเภทของชิ้นส่วนให้กับผู้เล่นเมื่อทำการต่อสู้ชนะ เพื่อให้ผู้เล่นคนนั้นๆสามารถนำเงินนั้นมาซื้อชิ้นส่วนใหม่ๆที่มีความสามารถเพิ่มมากขึ้น โดยถ้าชิ้นส่วนของหุ่นยนต์ถูกทำลายในการต่อสู้ชิ้นส่วนนั้นก็จะไม่สามารถใช้งานได้อีกต่อไปจะต้องทำการซื้อมาประกอบใหม่ โดยที่มีการแบ่งระดับของหุ่นยนต์ที่ต้องต่อสู้ไว้ 5 ระดับ

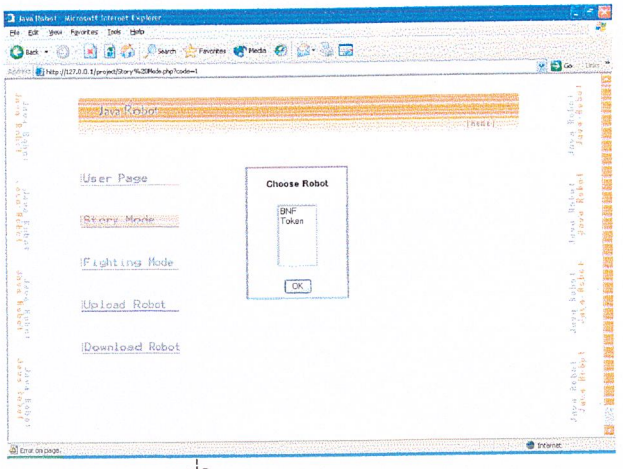
- **ผู้เล่นที่ทำการเล่นเป็นครั้งแรก**

เริ่มต้นเราจะอยู่ในระดับชั้นที่ 1 เมื่อชนะจะขยับขึ้นทีละ 1 ระดับ และในแต่ละระดับจะเพิ่มประเภทของชิ้นส่วนที่แตกต่างกันไป

- **ผู้เล่นที่เคยทำการเล่นแล้ว**

ระบบจะทำการคำนวณว่าหุ่นยนต์ที่เราเลือกมานั้นอยู่ในระดับชั้นใดจะต้องสู้กับหุ่นยนต์ในระดับชั้นใด

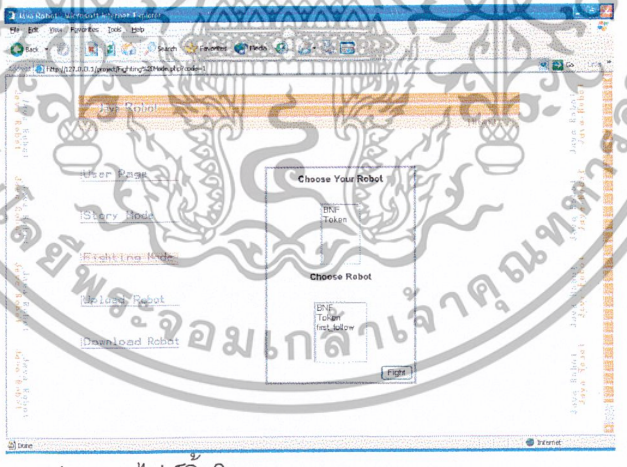
โดยวิธีการเล่นเป็นดังรูปที่ 4.12



รูปที่ 4.12 แสดงหน้าการเล่นแบบสตอรี่โหมด

4.2.3.2 การเล่นแบบไฟต์ติ้งโหมด (Fighting Mode)

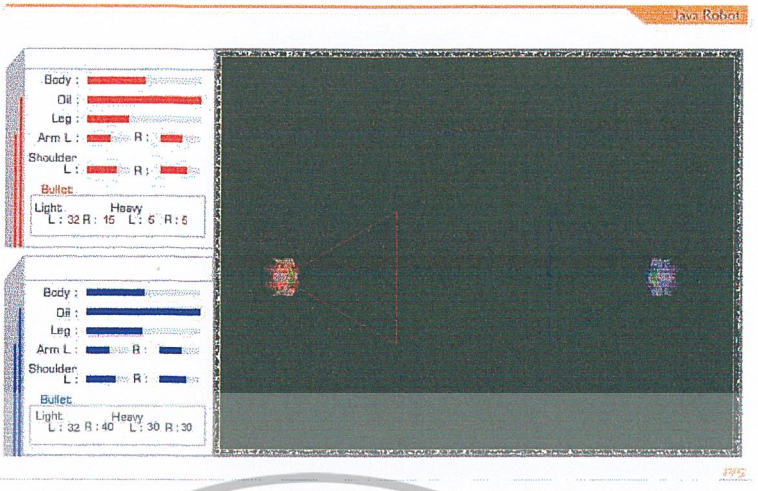
ในการเลือกเล่นแบบไฟต์ติ้งโหมดจะเป็นการต่อสู้กับหุ่นยนต์ที่ผู้เล่นคนอื่นสร้างขึ้น เพื่อให้ผู้เล่นสามารถทดสอบหุ่นยนต์ที่ตนเองสร้างขึ้นกับหุ่นยนต์ของคนอื่นเพื่อทำการเปรียบเทียบการเขียนโปรแกรมบังคับหุ่นยนต์ในแบบต่างๆเพื่อให้เหมาะสมกับหุ่นยนต์ของตนเองมากที่สุดในการต่อสู้กับหุ่นยนต์ในแบบต่างๆ โดยที่การเล่นแบบนี้จะไม่มีเงินหรือส่วนประกอบเพิ่มเติมให้กับผู้เล่น โดยวิธีการเล่นเป็นดังรูปที่ 4.13



รูปที่ 4.13 แสดงหน้าการเล่นแบบไฟต์ติ้งโหมด

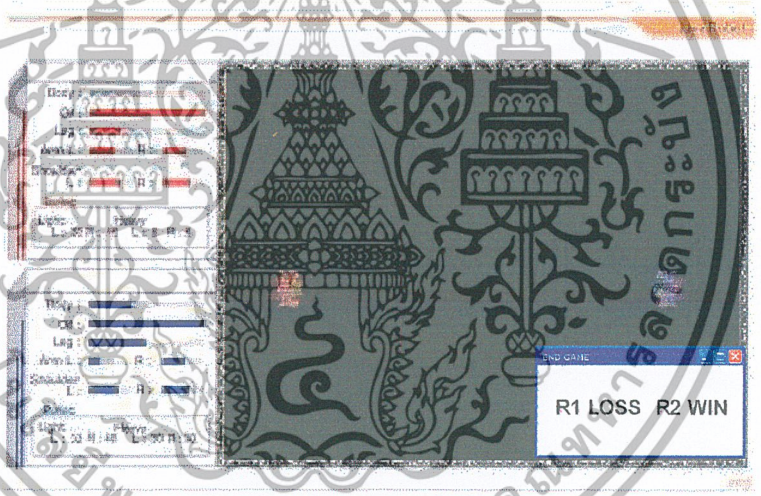
4.2.4 การแสดงการต่อสู้และการแสดงผลการต่อสู้

เมื่อเลือกหุ่นยนต์ของทั้ง 2 ฝ่ายแล้วจะเกิดหน้า JField หรือสนามรบ เพื่อส่งหุ่นยนต์ทั้ง 2 ตัว ลงไปทำการต่อสู้ ดังรูปที่ 4.14



รูปที่ 4.14 แสดงการต่อสู้ของหุ่นยนต์

เมื่อหุ่นยนต์ต่อสู้กันเสร็จจะเกิดผลแพ้-ชนะขึ้นมาแสดง ดังรูปที่ 4.15



รูปที่ 4.15 แสดงผลการต่อสู้เมื่อต่อสู้เสร็จ

บทที่ 5
สรุปผลการศึกษา ข้อเสนอแนะ และแนวทางการพัฒนาต่อ

5.1 สรุปผลปัญหาพิเศษ

1) เกมจะแบ่งการเล่นออกเป็น 2 แบบ คือ

- สตอรี่โหมดจะต้องสู้กับหุ่นยนต์ที่กำหนดไว้โดยจะแบ่งระดับชั้นของหุ่นยนต์ออกเป็น 5 ระดับ โดยหุ่นยนต์ที่ถูกสร้างขึ้นใหม่จะอยู่ในระดับชั้นที่ 1 จะสามารถทำการยกระดับชั้นของตนเองได้โดยทำการแข่งขันกับหุ่นยนต์ที่กำหนดไว้ในแบบสตอรี่โหมด เมื่อชนะจะได้รับการยกระดับ เงินรางวัล และประเภทของส่วนประกอบ โดยเงินรางวัลที่ได้รับสามารถนำไปซื้อส่วนประกอบเปลี่ยนใหม่เพื่อเพิ่มความสามารถให้แก่หุ่นยนต์ได้
- ไฟต์ติ้งโหมดจะเป็นการเลือกหุ่นยนต์ที่ต้องการแข่งขันได้เองแต่จะไม่มีรางวัลให้

2) โปรแกรม JavaRobot Creater แบ่งการทำงานเป็น

- สร้างหุ่นยนต์สามารถใช้สร้างหุ่นยนต์ขึ้นมาใหม่เพื่อนำมาใช้ในการเล่นเกม โดยที่การสร้างหุ่นยนต์จะต้องทำการเลือกส่วนประกอบของหุ่นยนต์โดยมีจำนวนเงินเป็นตัวกำหนดการเลือกของส่วนประกอบ ถ้ามีเงินไม่พอที่จะซื้อส่วนประกอบประเภทนั้นๆจะไม่สามารถเลือกซื้อส่วนประกอบนั้นได้ โดยจะมีการแสดงรายเอียดของส่วนประกอบแต่ละประเภทว่ามีลักษณะแตกต่างกันอย่างไร โดยที่มีการแสดงค่าเฉพาะของส่วนประกอบแต่ละประเภทเพื่อให้เลือกใช้ได้อย่างสะดวก และทำการเขียนโปรแกรมบังคับหุ่นยนต์ ได้มีการเขียนโปรแกรมต้นแบบไว้ เพื่อให้กับผู้เล่นที่เริ่มเล่นได้ใช้งาน
- การปรับปรุงแก้ไขหุ่นยนต์ที่ได้ทำการสร้างไว้แล้วเพื่อปรับปรุงให้มีความเหมาะสมในการเล่นมากที่สุด

3) ส่วนเว็บไซท์แบ่งการทำงานเป็น

- การลงทะเบียนของผู้เล่นก่อนที่จะเข้ามาใช้งาน
- การให้บริการการดาวน์โหลดโปรแกรมสร้างหุ่นยนต์ โปรแกรมที่ใช้คอมไพล์ และเอกสารที่จำเป็นในการเล่น

5.2 ปัญหา

1) การเล่นเกมหุ่นยนต์รบที่ควบคุมโดยภาษาจาวา ควรจะศึกษาถึงความพร้อมของอุปกรณ์ เนื่องจากการเล่นจะต้องมีระบบเครือข่ายที่ดี เนื่องจากต้องส่งผ่านข้อมูลผ่านระบบเครือข่ายเป็นจำนวนมาก รวมไปถึงประสิทธิภาพของคอมพิวเตอร์ที่นำมาใช้เล่นด้วย ถ้าคอมพิวเตอร์มีการประมวลผลช้า เกมจะมีการทำงานได้ช้าและขาดความราบรื่นในการเล่นเกม

2) ผู้เล่นที่เริ่มต้นเล่นเกมนี้ ผู้เล่นจะต้องเสียเวลาในการศึกษาวิธีการเล่นและเมธตอดที่ใช้ในการทำงานของหุ่นยนต์เป็นเวลานาน

3) ผู้เล่นจะต้องมีความอดทนในการเล่น เนื่องจากเมื่อเริ่มแรกอาจยังมองไม่เห็นเทคนิคและกลยุทธ์ต่างๆในการเล่นเกม

4) หุ่นยนต์ที่เป็นคู่ต่อสู้ในแบบสตอรี่โหมดในแต่ละระดับชั้นมีความสามารถไม่แตกต่างกันมากนัก เพราะไม่มีมาตรฐานที่ใช้วัดความสามารถ

5.3 ข้อเสนอแนะและแนวทางพัฒนาต่อ

1) ในการพัฒนารุ่นต่อไปของโปรแกรมเกมหุ่นยนต์รบด้วยภาษาจาวา จะทำได้ง่ายโดยไม่ค่อยก่อให้เกิดปัญหากับผู้พัฒนาโปรแกรม เนื่องจากคุณสมบัติของภาษาเชิงวัตถุ แต่ละส่วนของโปรแกรมจะถูกแบ่งให้เห็นได้อย่างชัดเจน เหมาะกับการทำงานเป็นที

2) กราฟฟิกจะถูกแสดงในแบบ 2 มิติ ซึ่งเหมาะสมในปัจจุบันเนื่องจากประสิทธิภาพการเชื่อมโยงระหว่างเครือข่ายในเมืองไทยยังมีไม่มากนัก แต่ในอนาคตถ้าความเร็วของเครือข่ายมีมากขึ้น การแสดงผลควรจะออกมาในรูปแบบ 3 มิติ

3) เหตุการณ์ที่นำมาให้ผู้เล่นกำหนดวิธีการต่อสู้ของหุ่นยนต์ ปัจจุบันมี 6 เหตุการณ์ คือ ปกติ โจมตี ถูกโจมตี ชนขอบ ชนศัตรู พบศัตรู ซึ่งสามารถที่จะขยายเพิ่มอีกหลายๆเหตุการณ์ได้ จะทำให้เกมมีความซับซ้อนมากขึ้นและมีความสนุกสนานมากขึ้น

4) ส่วนประกอบที่นำมาให้ผู้เล่นเลือกเพื่อจะนำไปใช้ในประกอบขึ้นเป็นหุ่นยนต์ปัจจุบันมี 12 ส่วน ซึ่งสามารถที่จะขยายเพิ่มอีกหลายๆส่วนได้ จะทำให้เกมมีความซับซ้อนมากขึ้นและมีความสนุกสนานมากขึ้น เช่น ส่วนของอาวุธโจมตี ซึ่งปัจจุบันมี 2 ประเภทคืออาวุธเบาและอาวุธหนัก

5) ในการต่อสู้แบบสตอรี่โหมดมีให้เล่นเพียง 5 ระดับชั้นเท่านั้น โดยในแต่ละระดับชั้นมีหุ่นยนต์เพียงระดับชั้นละตัวเท่านั้น โดยอาจทำการเพิ่มให้มีหุ่นยนต์ระดับชั้นละ 2-3 ตัว หรือเพิ่มระดับชั้นให้มีมากขึ้น เพื่อเพิ่มความหลากหลายในการเล่นกับหุ่นยนต์ในแบบต่างๆ

6) ปัจจุบันในสนามสามารถนำหุ่นยนต์เข้าต่อสู้ได้ครั้งละ 2 ตัวเท่านั้น การพัฒนาสามารถเพิ่มในส่วนที่ทำให้หุ่นยนต์สามารถต่อสู้กันเป็นทีมได้ อาจเป็นทีมละ 2 ตัว หรือ ทีมละ 3 ตัว หรือว่าจะสามารถต่อสู้ทีละหลายๆตัวได้พร้อมๆกัน

7) ปัจจุบันในสนามต่อสู้จะเป็นอวกาศ ซึ่งไม่มีสิ่งกีดขวางใดใดเลย การพัฒนาต่อสามารถเพิ่มในส่วนของสิ่งกีดขวาง เช่น มีบริเวณที่จะป้องกันการ สแกนของศัตรู มีดาวที่สามารถการป้องกันอาวุธของศัตรูได้ มีไฟฟ้าที่ถ้าเข้าไปในบริเวณนั้นๆ จะเสียพลัง มีสิ่งของให้เก็บเพื่อเพิ่มพลังชีวิตหรือเพิ่มพลังงาน เป็นต้น

บรรณานุกรม

กิตติ ภักดีวัฒนะกุล, Java **ฉบับโปรแกรมเมอร์**, 2001

ดร.วีระศักดิ์ ซึงถาวร, Java Programming Volume I, 2000

ดร.วีระศักดิ์ ซึงถาวร, Fundamental of JAVA Programming Volume 2, 2000

ฮันเตอร์ เดวิด, **คัมภีร์การใช้ XML ฉบับสมบูรณ์**, 2545

H.M.Deitel P.J. Deitel, Java How to Program Fourth Edition, 2000

H.M.Deitel P.J. Deitel S.E.Santry, Advanced Java 2 platform How to Program, 2000

Jesus Castagnetto Harish Rawat, Professional PHP Programming, 2000

ภาคผนวก ก.
การติดตั้งโปรแกรม

- JDK

โดยในที่นี้เป็นการขั้นตอนการติดตั้ง jdk 1.4.1

1) ดับเบิ้ลคลิกที่ตัวโปรแกรม jdk-1_4_1.exe โปรแกรม JDK จะถูก Install

2) Set Class path เพื่อให้สามารถเรียกใช้ Javac.exe และ Java.exe ได้จากในไดเรกทอรีใดๆ โดยพิมพ์ค่าเหล่านี้ลง ไปในไฟล์ autoexec.bat "SET PATH= C:\JDK1.4.1\BIN"

3) เมื่อเราต้องการที่จะ Compile โปรแกรมที่อ้างถึงคลาสต่างๆ จะทำได้โดยเพิ่มประโยค "SET CLASSPATH=.;C:\JDK1.4.1\LIB\CLASSES.ZIP" ที่ ไฟล์ autoexec.bat

4) ปกติเราจะใช้ default class path โดยจะ กำหนด ดังนี้ คือ

".;$JAVA\CLASSES;$JAVA\LIB\CLASSES.ZIP" ไฟล์ autoexec.bat

JavaRobot Creater

เป็นโปรแกรมที่ใช้ในการสร้างหุ่นยนต์

1) ดาวน์โหลดโปรแกรมจากเว็บไซท์ JavaRobot

2) ดับเบิลคลิกที่ ไฟล์ Robot.exe ที่ดาวน์โหลดมาได้

3) โปรแกรมจะถูกเปิดขึ้น ซึ่งจะสามารถสร้างหุ่นยนต์ได้

- XML Parser

1) ทำการสร้างไดเรกทอรีชื่อ XML จากนั้นทำการแตกไฟล์ SAX 1.0 ไว้ภายในไดเรกทอรี XML

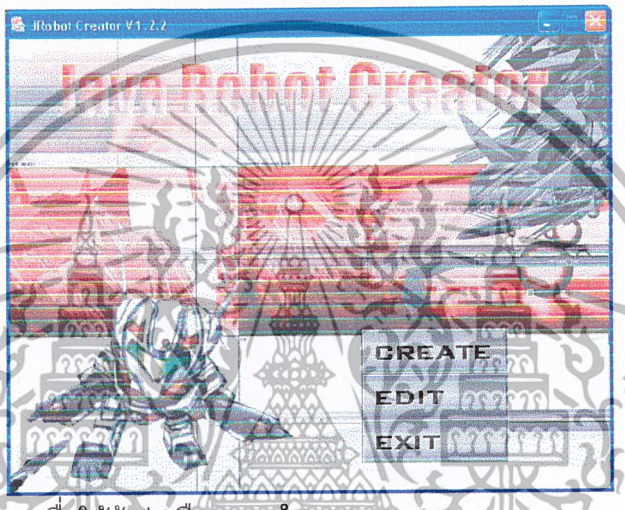
2) ทำการสร้างไดเรกทอรี xp ไว้ภายในไดเรกทอรี XML

3) ทำการ Set Class Path โดยที่เราเข้าไปที่ System ในคอนโทรลพาเนล โดยทำการแก้ไขในส่วนของ User Variable ถ้าไดเรกทอรี XML อยู่ใน Drive C: ทำการเพิ่มประโยค .;C:\XML\SAX;C:XML\xp\xp.jar

ภาคผนวก ข.
คู่มือการใช้โปรแกรม

คู่มือการใช้ส่วนของแอปพลิเคชัน (Application)

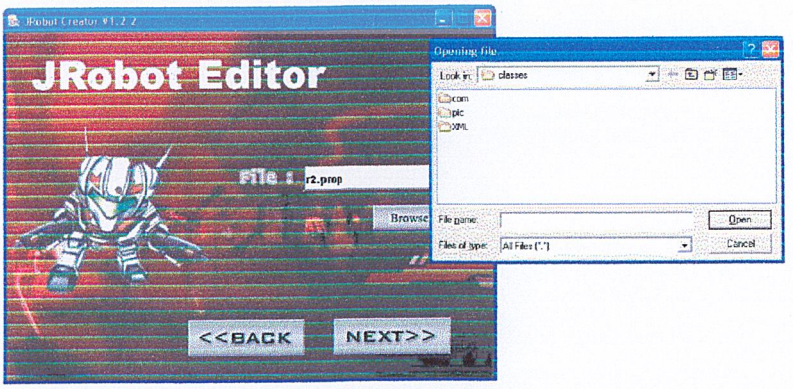
ส่วนของแอปพลิเคชันเป็นส่วนโปรแกรมที่ใช้สำหรับสร้างหุ่นยนต์ โดยสามารถดาวน์โหลดโปรแกรมนี้ได้จากเว็บไซท์ JavaRobot เมื่อทำการ Install เสร็จเรียบร้อยแล้ว ทำการเปิดโปรแกรมขึ้นมาจะเป็นดังรูปที่ ข-1 ให้ผู้เล่นทำการเลือกการทำงาน



รูปที่ ข-1 แสดงหน้าแรกเพื่อให้ผู้เล่นเลือกการทำงาน

- CREATE เป็นการเลือกเพื่อทำการสร้างหุ่นยนต์ตัวใหม่หรือสำหรับผู้ที่ไม่เคยมีหุ่นยนต์ของตัวเองมาก่อนโดยจะเชื่อมต่อกับหน้าเลือกส่วนประกอบ

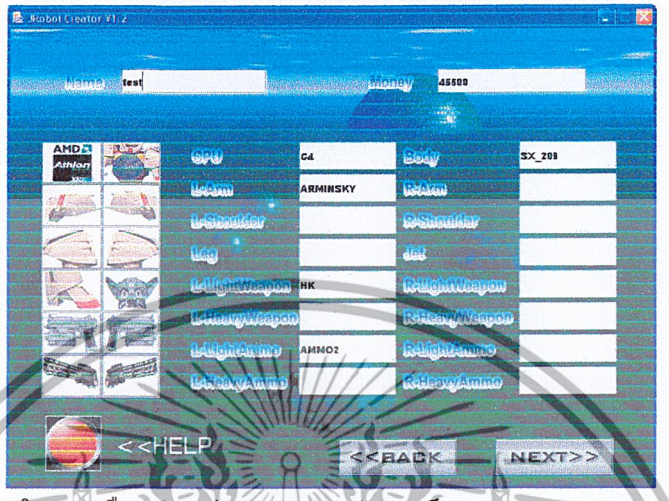
- EDIT เป็นการเลือกเพื่อให้ผู้เล่นที่มีหุ่นยนต์แล้วทำการปรับปรุงแก้ไขเพิ่มเติม โดยเมื่อกดแล้วจะเป็นดังรูปที่ ข-2 เพื่อให้ผู้เล่นเลือกไฟล์หุ่นยนต์ที่ต้องการแก้ไขโดยกดปุ่ม Browse



รูปที่ ข-2 แสดงหน้าจอในการเลือกไฟล์หุ่นยนต์ที่ต้องการนำมาแก้ไข

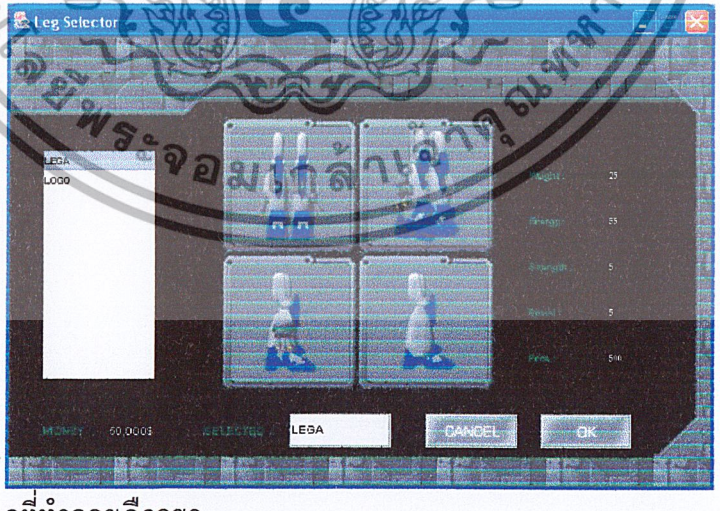
- EXIT เป็นการเลือกเพื่อออกจากโปรแกรม

เมื่อทำการเลือกการสร้างหุ่นยนต์ตัวใหม่หรือเลือกไฟล์หุ่นยนต์ที่ต้องการแก้ไขแล้วแล้วจะเกิดหน้าจอดังรูปที่ ข-3 เพื่อให้ทำการตั้งชื่อหุ่นยนต์และเลือกส่วนประกอบ



รูปที่ ข-3 แสดงหน้าจอในการเลือกส่วนประกอบของหุ่นยนต์

ส่วนประกอบของหุ่นยนต์ในแต่ละชิ้นส่วนจะมีให้เลือกหลายชนิดแตกต่างกันไปตามระดับชั้นของหุ่นยนต์ โดยถ้ายิ่งระดับชั้นสูงก็จะมีให้เลือกมากและมีความสามารถสูง โดยเมื่อทำการกดที่ชิ้นส่วนใดๆก็ตามจะเกิดหน้าจอเพื่ออุปกรณ์ชิ้นนั้นๆ เช่น รูปที่ ข-4 แสดงหน้าจอที่ทำการเลือกขา



รูปที่ ข-4 แสดงหน้าจอที่ทำการเลือกขา

โดยแต่ละชิ้นส่วนจะมีค่าที่เป็นคุณลักษณะเฉพาะตัวซึ่งเป็นดังนี้

CPU

- EyeSight บอกถึงระยะทางในการสแกนซึ่งถ้ามีระยะทางในการสแกนมากก็จะสามารถมองเห็นศัตรูได้ในระยะไกล

- Degree บอกถึงองศาในการสแกนซึ่งถ้ามีองศาในการสแกนมากก็จะสามารถเห็นศัตรูในด้านข้างได้ดี

Body, Arm, Shoulder, Leg

- Weight จะบอกน้ำหนักของส่วนนั้นนั้น ซึ่งถ้าน้ำหนักรวมของทุกส่วนยิ่งมากก็จะทำให้หุ่นยนต์เคลื่อนที่ได้ช้าลงไปด้วย

- Strength จะบอกถึงความแข็งแรงทนทานของชิ้นส่วนชิ้นนั้นๆว่ามีความแข็งแรงเพียงใด

- Energy (พลังงาน) จะบอกพลังของส่วนนั้นว่าจะสามารถทนการโจมตีได้แค่ไหน ถ้าพลังงานของ body หมดจะทำให้หุ่นยนต์ตัวนั้นถูกทำลายทันที แต่ถ้าพลังงานของ arm ข้างใดหมดข้างนั้นก็จะไม่สามารถใช้ข้างนั้นได้ และ ถ้าพลังงานของ leg หมดจะทำให้หุ่นยนต์ตัวนั้นไม่สามารถเคลื่อนที่ได้ นอกจากจะใช้ jet

- Speed (จะมีในส่วน leg ส่วนเดียว) ซึ่งใช้ในการเคลื่อนที่ โดย speed จะขึ้นกับน้ำหนักรวมของหุ่นยนต์ ด้วย เช่น ถ้า speed เท่ากันแต่น้ำหนักรวมของหุ่นยนต์ต่างกันก็จะทำให้ความเร็วในการเคลื่อนที่ต่างกัน

Jet

- Oil จำนวนน้ำมันที่ใช้ได้ จะลดลงไปถ้ามีการเคลื่อนที่ถ้าหมดจะไม่สามารถเคลื่อนที่ได้

- Speed ความเร็วของการใช้ jet น้ำหนักของหุ่นยนต์จะไม่ผลต่อความเร็วของ jet

Weapon

- Capacity จำนวนกระสุนที่สามารถบรรจุไว้ใช้งานเพื่อใช้โจมตีศัตรูถ้าหมดจะไม่สามารถโจมตีได้

- Range ระยะทางที่อาวุธสามารถยิงกระสุนออกไปโจมตีศัตรู ถ้าศัตรูอยู่ไกลกว่าระยะที่สามารถโจมตีได้กระสุนจะไม่สามารถทำอันตรายศัตรูได้

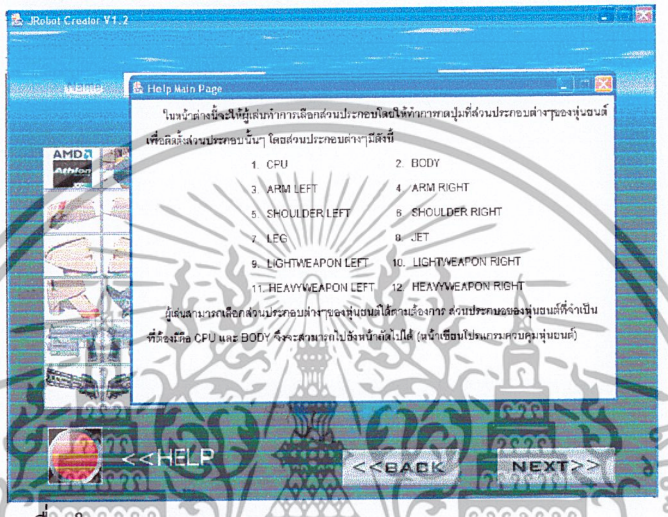
- Weight น้ำหนักของอาวุธ ซึ่งถ้าน้ำหนักรวมของทุกส่วนยิ่งมากก็จะทำให้หุ่นยนต์เคลื่อนที่ได้ช้าลงไปด้วย

Ammo

- Damage พลังทำลายของกระสุน ถ้ามีค่ามากๆจะสามารถทำอันตรายศัตรูให้เกิดความเสียหายได้มาก

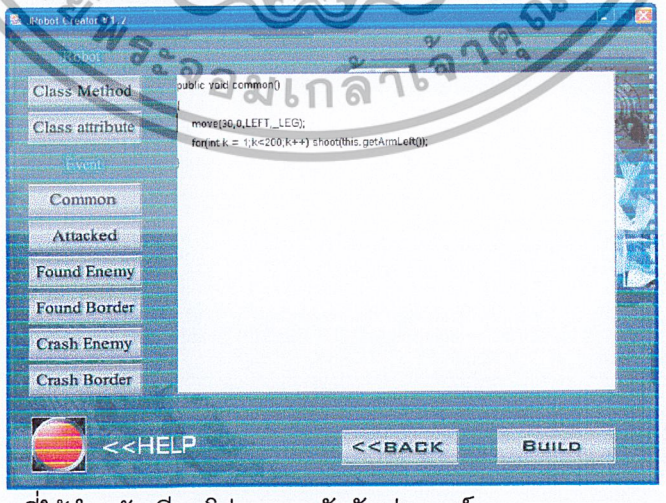
- Speed ความเร็วในการเคลื่อนที่ของกระสุนถ้ามีความเร็วมากๆจะทำให้มีโอกาสในการโดนมากเพราะศัตรูจะหลบไม่ทัน

ถ้าไม่เข้าใจการทำงานสามารถกด HELP เพื่อขอคำอธิบายได้ดังรูปที่ ข-5



รูปที่ ข-5 แสดงหน้าจอเมื่อทำการกด HELP

เมื่อเลือกชิ้นส่วนที่ต้องการครบแล้วทำการกด NEXT เพื่อไปในการทำงานยังส่วนถัดไป ถ้าต้องการกลับไปยังหน้าก่อนหน้าให้ทำการกดปุ่ม BACK รูปที่ ข-6 เป็นหน้าที่ใช้สำหรับเขียนโปรแกรมบังคับหุ่นยนต์ให้ทำงานในแต่ละเหตุการณ์ว่าเหตุการณ์ใดต้องทำงานอะไร



รูปที่ ข-6 แสดงหน้าจอที่ใช้สำหรับเขียนโปรแกรมบังคับหุ่นยนต์

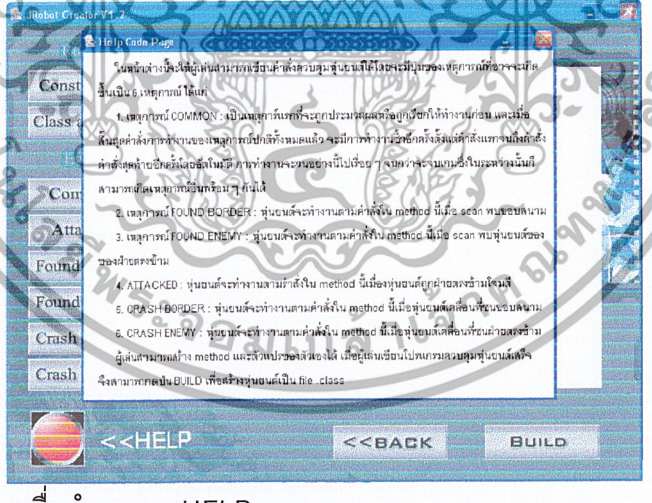
- ปุ่ม Class Method ให้ผู้เล่นสามารถสร้างเมธตอดขึ้นมาใช้งานเองได้
- ปุ่ม Class attribute ให้ผู้เล่นสามารถสร้างแอตทริบิวต์ขึ้นมาใช้งานเองได้

ผู้เล่นจะสามารถ Override Event Code ของเหตุการณ์ต่างได้ เมื่อทำการคลิกเลือกเหตุการณ์ที่ต้องการที่จะ Override ก็จะมีเมธตอดของเหตุการณ์นั้นให้ทำการเขียนโปรแกรมเป็นภาษาจาวาโดยสามารถใช้เมธตอดของส่วนประกอบที่เลือกมาได้ ซึ่งมีเหตุการณ์ที่สามารถเลือกได้ดังต่อไปนี้

- Common เหตุการณ์ปกติ
- Found Enemy เหตุการณ์พบศัตรู
- Attacked เหตุการณ์โดนโจมตี
- Found Border เหตุการณ์พบขอบสนาม
- Crash Enemy เหตุการณ์ชนศัตรู
- Crash Border เหตุการณ์ชนขอบสนาม

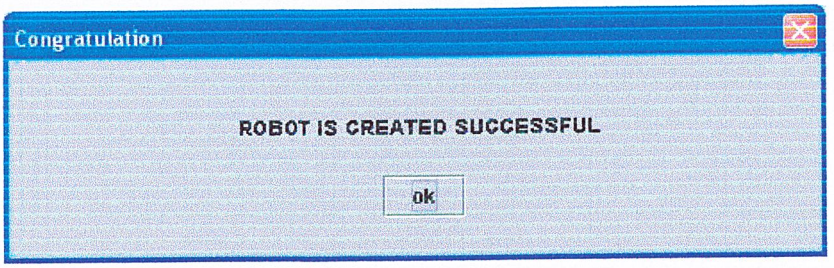
ถ้าต้องการกลับไปยังหน้าเลือกส่วนประกอบทำการกดปุ่ม BACK

ถ้าไม่เข้าใจการทำงานสามารถกด HELP เพื่อขอคำอธิบายได้ดังรูปที่ ข-7



รูปที่ ข-7 แสดงหน้าจอเมื่อทำการกด HELP

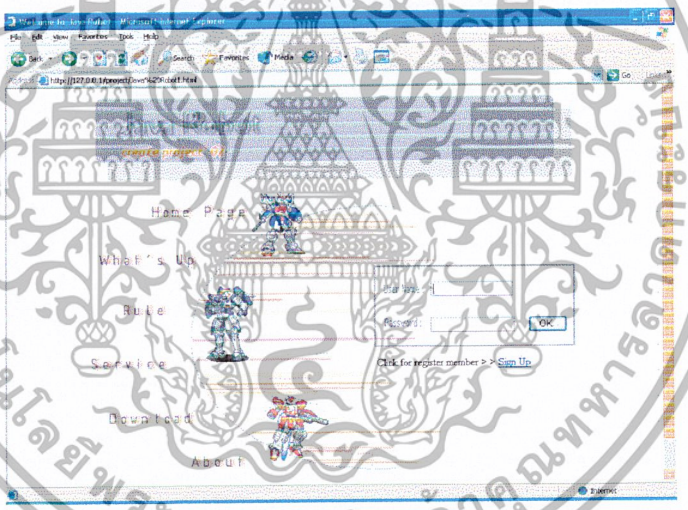
เมื่อเขียนโปรแกรมบังคับหุ่นยนต์เสร็จแล้วทำการกดปุ่ม BUILD เพื่อทำการคอมไพล์โปรแกรมที่ผู้เล่นเขียนขึ้นว่าถูกต้องหรือไม่ ถ้าถูกต้องจะเป็นดังรูปที่ ข-8



รูปที่ ข-8 แสดงหน้าจอเมื่อคอมไพล์โปรแกรมที่ผู้เล่นเขียนขึ้นผ่าน.

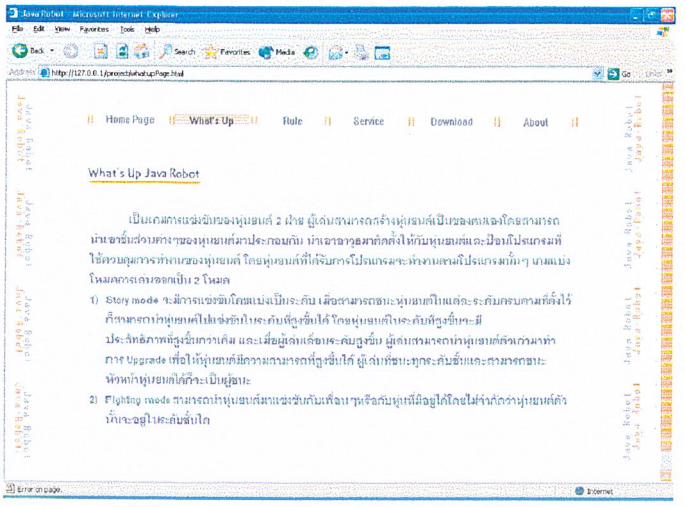
คู่มือการใช้ส่วนของเว็บ

หน้าแรกของเว็บ JavaRobot จะให้ผู้เล่นทำการสมัครสมาชิก หรือถ้าเป็นสมาชิกแล้วให้ทำการ Log in เพื่อเข้าไปใช้งาน หรือต้องการดูว่าเป็นเกมอะไร วิธีการเล่น บริการที่มีในเว็บ โปรแกรมที่จำเป็นสำหรับการสร้างหุ่นยนต์และคอมไพล์ และเกี่ยวกับผู้สร้าง



รูปที่ ข-9 แสดงหน้าแรกของเว็บ JavaRobot

หน้า What's Up เป็นหน้าที่แสดงให้ผู้เล่นรู้ว่าเป็นเกมเกี่ยวกับอะไร การเล่นเป็นอย่างไร สามารถเข้าไปดูได้ ดังรูปที่ ข-10

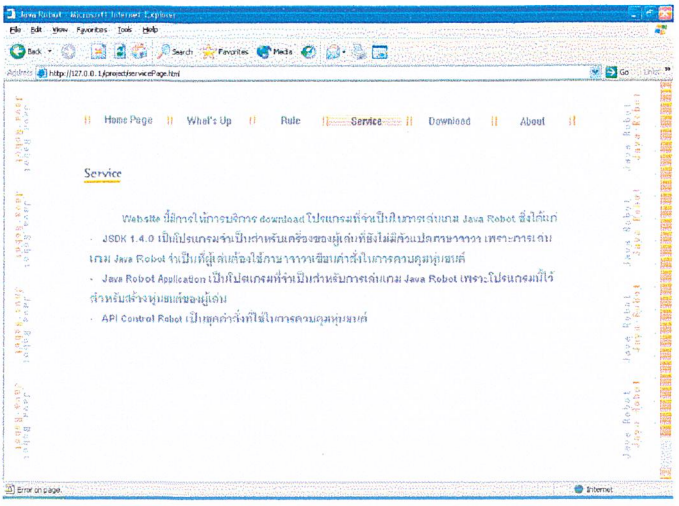


รูปที่ ข-10 แสดงหน้า What's Up

หน้า Rule เป็นหน้าที่แสดงให้ผู้เล่นรู้เกี่ยวกับวิธีการเล่นว่าขั้นตอนในการเล่นต้องทำอย่างไรบ้าง ตั้งแต่การสร้างหุ่นยนต์ การนำไปใช้งาน และการเล่นมีกี่แบบ แต่ละแบบเป็นอย่างไร สามารถเข้าไปดูได้ ดังรูปที่ ข-11

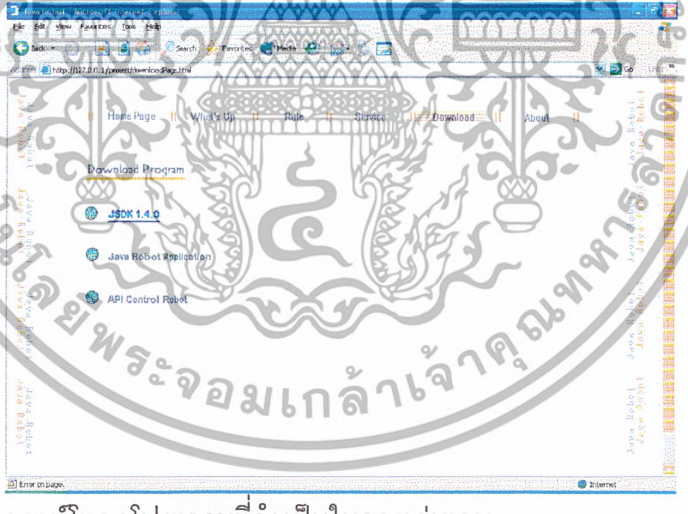
รูปที่ ข-11 แสดงหน้า Rule

หน้า Service เป็นหน้าที่แสดงให้ผู้เล่นทราบเกี่ยวกับบริการที่มีให้ภายในเว็บนี้ เช่น อธิบายว่าโปรแกรมที่ให้ดาวน์โหลดเอาไว้ใช้งานอะไร ดังรูปที่ ข-12



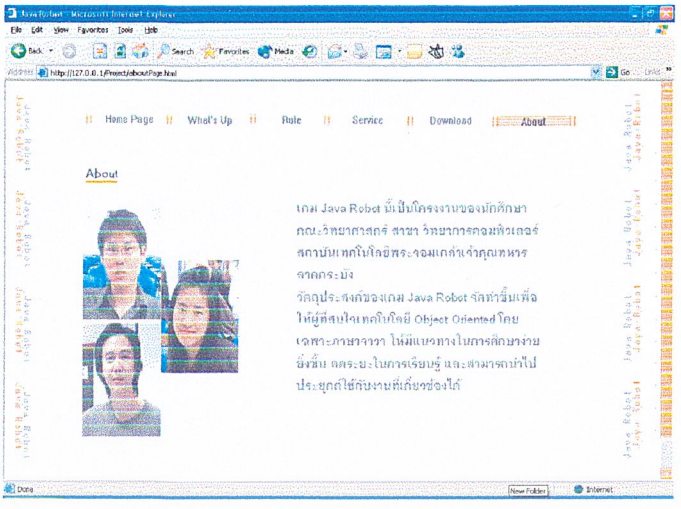
รูปที่ ข-12 แสดงหน้า Service

หน้า Download เป็นหน้าที่แสดงโปรแกรมที่จำเป็นในการเล่นเกม โดยผู้เล่นที่สนใจเกี่ยวกับเกมนี้สามารถดาวน์โหลดโปรแกรมที่ใช้สำหรับสร้างหุ่นยนต์ โปรแกรมที่ใช้สำหรับคอมไพล์ โปรแกรมที่ใช้อ่านไฟล์ XML และเอกสารประกอบเพื่อนำไปสร้างหุ่นยนต์และนำมาใช้เล่นในเว็บนี้ได้ ดังรูปที่ ข-13



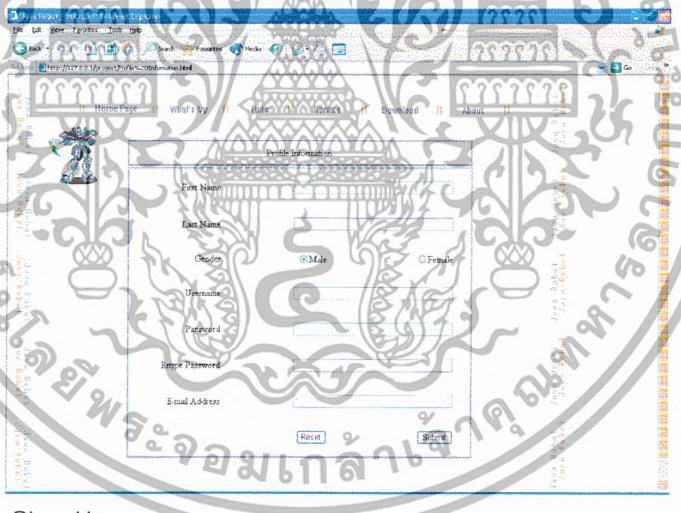
รูปที่ ข-13 แสดงหน้าดาวน์โหลดโปรแกรมที่จำเป็นในการเล่นเกม

หน้า About เป็นหน้าที่แสดงถึงรายละเอียดเกี่ยวกับผู้ที่ทำการพัฒนาโปรแกรม JavaRobot ว่าเป็นใคร ทำขึ้นเพื่ออะไร ดังรูปที่ ข-14



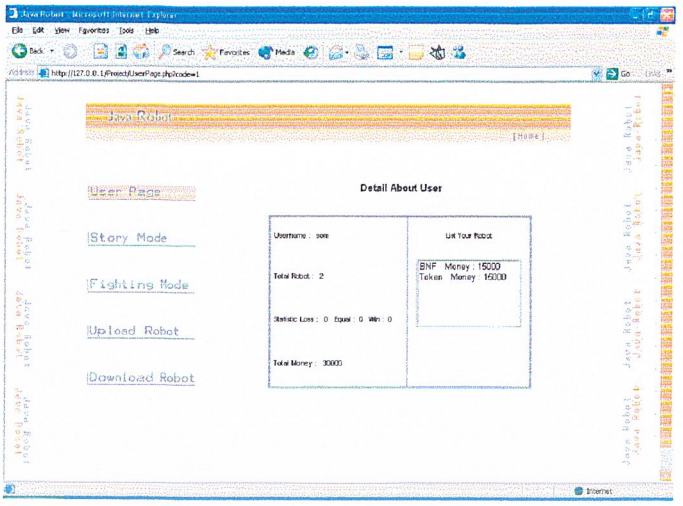
รูปที่ ข-14 แสดงหน้า About

หน้า Information เป็นหน้าที่ให้ผู้เล่นทำการลงทะเบียน เมื่อผู้เล่นต้องการเข้ามาใช้บริการจะต้องทำการลงทะเบียนสมาชิกก่อนโดยกด Sign Up โดยการลงทะเบียนเป็นสมาชิกโดยจะต้องกรอกรายละเอียดต่างๆในแบบฟอร์มให้ครบถ้วนสมบูรณ์ ดังรูปที่ ข-15



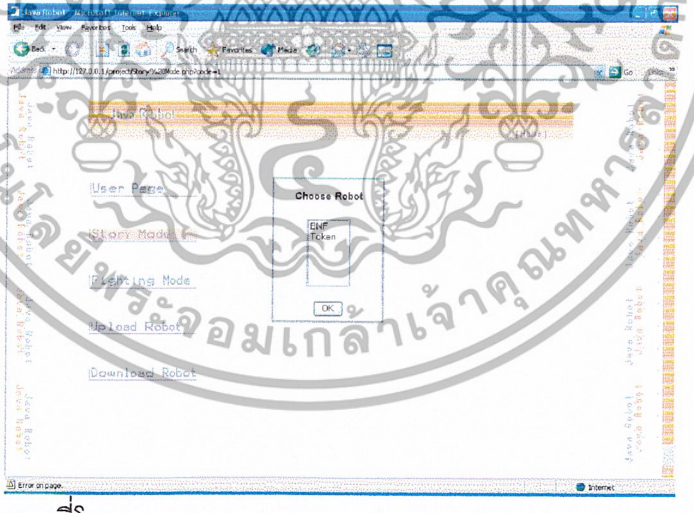
รูปที่ ข-15 แสดงหน้า Sign Up

หน้า User Page เป็นหน้าที่แสดงถึงข้อมูลของผู้เล่นคนนั้นๆ โดยที่เมื่อทำการลงทะเบียนเป็นสมาชิกแล้วทำการ Log in โดยใส่ Username และ Password ที่ได้ทำการกรอกรายละเอียดไว้เพื่อไปเพื่อเข้าไปใช้งาน และสามารถเลือกการทำงานต่างๆที่มีอยู่ได้ ดังรูปที่ ข-16



รูปที่ ข-16 แสดงหน้า User Page

หน้า Story Mode เป็นหน้าที่แสดงข้อมูลหุ่นยนต์ของเราว่ามีกี่ตัว ชื่ออะไรบ้าง โดยถ้าผู้เล่นต้องการเล่นแบบสตอรี่โหมด ดังรูปที่ ข-17 ก็ทำการเลือกหุ่นยนต์ที่ต้องการจะนำไปแข่งขันเซิร์ฟเวอร์จะทำการคำนวณหาคู่ต่อสู้ให้เองโดยอัตโนมัติ จะเป็นการต่อสู้กับหุ่นยนต์ที่ผู้สร้างสร้างขึ้น โดยเมื่อชนะหุ่นที่ผู้สร้างสร้างขึ้นจะเป็นการยกระดับให้กับหุ่นยนต์ของผู้เล่นคนนั้นๆ ซึ่งจะทำให้ผู้เล่นสามารถนำหุ่นยนต์ตัวนั้นไปทำการปรับเปลี่ยนชิ้นส่วนที่มีความสามารถมากขึ้นได้



รูปที่ ข-17 แสดงหน้าสตอรี่โหมด

เมื่อผู้เล่นมีระดับชั้นที่เปลี่ยนไปจะได้รับเงินเพิ่มมากขึ้นและมีส่วนประกอบให้เลือกซื้อได้มากขึ้นดังตาราง ข –1

ตาราง ข-1 แสดงรายการของส่วนประกอบที่มีให้เลือก(จำนวนประเภท)และจำนวนเงินที่ได้รับในระดับชั้นต่างๆ

Level	Body	Arm	Leg	Jet	CPU	Shoulder	HW	HA	LW	LA	Money
1	3	3	3	2	2	3	3	3	3	3	12,000
2	4	4	5	3	3	4	5	4	4	4	+5,000
3	5	5	6	4	4	5	6	6	5	6	+7,000
4	6	6	7	5	5	6	7	7	6	7	+10,000
5	7	7	8	5	5	7	7	7	7	7	+15,000

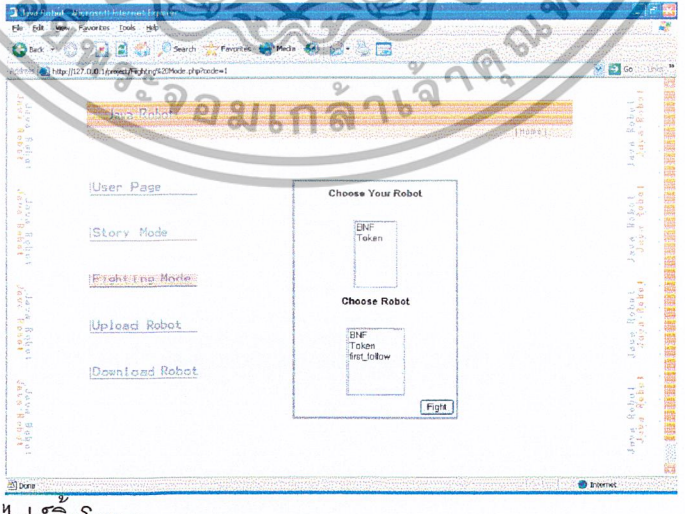
HW : Heavy Weapon

HA : Heavy Ammo

LW : Light Weapon

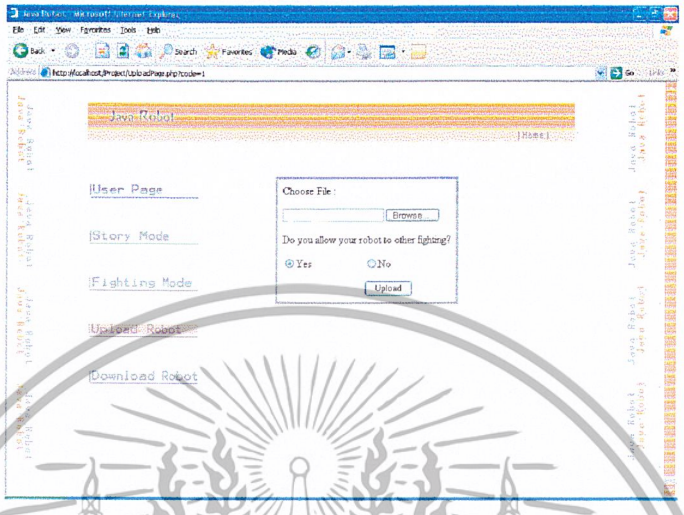
LA : Light Ammo

หน้า Fighting Mode เป็นหน้าที่แสดงถึงรายละเอียดหุ่นยนต์ของผู้เล่นว่ามีกี่ตัว ชื่ออะไรบ้าง และรายละเอียดหุ่นยนต์ของผู้เล่นคนอื่น โดยที่ถ้าผู้เล่นต้องการเล่นแบบไฟต์ติ้ง ดังรูปที่ ข-18 จะเป็นการต่อสู้กับหุ่นยนต์ที่ผู้เล่นคนอื่นๆสร้างขึ้น เพื่อให้ผู้เล่นสามารถทดสอบหุ่นยนต์ที่ตนเองสร้างขึ้นกับหุ่นยนต์ของผู้เล่นคนอื่นเพื่อทำการเปรียบเทียบการเขียนโปรแกรมบังคับหุ่นยนต์ในแบบต่างๆเพื่อให้เหมาะสมกับหุ่นยนต์ของตนเองมากที่สุดในการต่อสู้กับหุ่นยนต์ในแบบต่างๆ โดยไม่มีเงินหรือส่วนประกอบใดๆเพิ่มให้



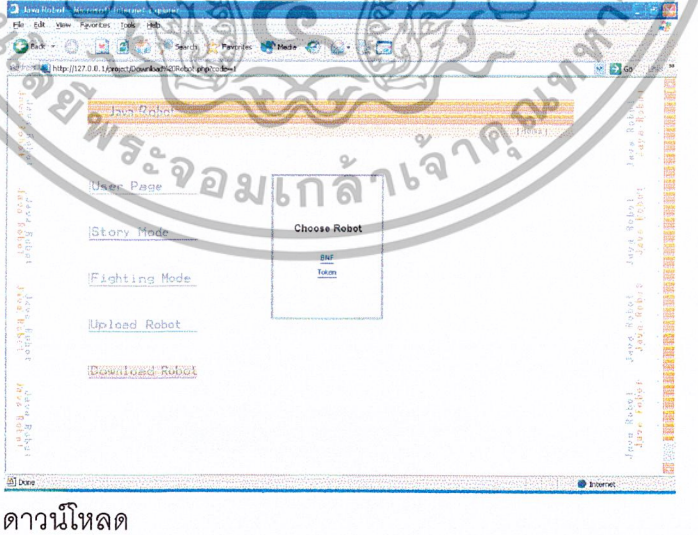
รูปที่ ข-18 แสดงหน้าไฟต์ติ้งโหมด

หน้า Upload เป็นหน้าที่ให้ผู้เล่นทำการ Upload หุ่นยนต์ที่ผู้เล่นสร้างขึ้นไปเก็บที่เซิร์ฟเวอร์ เนื่องจากเมื่อผู้เล่นสร้างหุ่นยนต์ขึ้นมาผู้เล่นยังไม่สามารถนำหุ่นยนต์ตัวนั้นไปใช้งานได้ ถ้าต้องการใช้งานจะต้องทำการ Upload หุ่นยนต์ไปเก็บยังเซิร์ฟเวอร์เพื่อใช้งาน ดังรูปที่ ข-19



รูปที่ ข-19 แสดงหน้า Upload

หน้า Download เป็นหน้าที่ให้ผู้เล่นทำการดาวน์โหลดหุ่นยนต์ของตนเองที่ถูกเก็บไว้ที่เซิร์ฟเวอร์มาทำการปรับปรุงแก้ไขทั้งในส่วนของส่วนประกอบและส่วนของการเขียนโปรแกรมบังคับหุ่นยนต์ โดยจะต้องทำการดาวน์โหลดหุ่นยนต์ตัวนั้นมาไว้ที่เครื่องของผู้เล่นก่อนแล้วจึงใช้โปรแกรมในการสร้างหุ่นยนต์ในการปรับปรุงแก้ไข ดังรูปที่ ข-20



รูปที่ ข-20 แสดงหน้าดาวน์โหลด

ภาคผนวก ค.
แอตทริบิวต์เมธตอดที่ใช้ในส่วนของสนามรบ

ตารางที่ ค-1 แสดงแอตทริบิวต์และเมธตอดของ Class Body

Body	
Energy : int	// พลังงานของเกราะหุ่นยนต์
Weight : int	// น้ำหนักของเกราะหุ่นยนต์
Strength : int	// ความแข็งแรงของเกราะหุ่นยนต์
name : String	// ชื่อของเกราะหุ่นยนต์
percentage : int	// ความน่าจะเป็นที่จะโดนยิงของเกราะหุ่นยนต์
size : int	// ขนาดของเกราะหุ่นยนต์
Body(int energy,int weight, int strength,String name)	// Constructor ของเกราะหุ่นยนต์
getEnergy() : int	// คืนค่าพลังงานของเกราะหุ่นยนต์
getWeight() : int	// คืนค่าน้ำหนักของเกราะหุ่นยนต์
getStrength() : int	// คืนค่าความแข็งแรงของเกราะหุ่นยนต์
getName() : String	// คืนค่าชื่อของเกราะหุ่นยนต์
getPercentage() : int	// คืนค่าความน่าจะเป็นที่จะโดนยิงของเกราะหุ่นยนต์
getSize() : int	// คืนค่าขนาดของเกราะหุ่นยนต์
setEnergy(int enegy)	// กำหนดค่าพลังงานของเกราะหุ่นยนต์
setPercentage(int p)	// กำหนดค่าความน่าจะเป็นที่จะโดนยิงของเกราะหุ่นยนต์
setSize(int s)	// กำหนดค่าขนาดของเกราะหุ่นยนต์

ตารางที่ ค-2 แสดงแอตทริบิวต์และเมธตอดของ Class Arm

Arm	
Energy : int	// พลังงานของแขนหุ่นยนต์
Strength : int	// ความแข็งแกร่งของแขนหุ่นยนต์
Weight : int	// น้ำหนักของแขนหุ่นยนต์
name : String	// ชื่อของแขนหุ่นยนต์
percentage : int	// ความน่าจะเป็นที่จะโดนยิงของแขนหุ่นยนต์
size : int	// ขนาดของแขนหุ่นยนต์
weapon : Weapon	// อาวุธที่ติดแขนหุ่นยนต์
Arm(int energy, int weight, int strength, String name)	// Constructor ของแขนหุ่นยนต์
Arm(int energy, int weight, int strength, Weapon Lightweapon, String name)	// Constructor ของแขนหุ่นยนต์
getEnergy() : int	// คืนค่าพลังงานของแขนหุ่นยนต์
getStrength() : int	// คืนค่าความแข็งแกร่งของแขนหุ่นยนต์
getWeight() : int	// คืนค่าน้ำหนักของแขนหุ่นยนต์
getName() : String	// คืนค่าชื่อของแขนหุ่นยนต์
getPercentage() : int	// คืนค่าความน่าจะเป็นที่จะโดนยิงของแขนหุ่นยนต์
getSize() : int	// คืนค่าขนาดของแขนหุ่นยนต์
getWeapon() : Weapon	// คืนค่าอาวุธที่ติดอยู่กับแขนหุ่นยนต์
setEnergy(int energy)	// กำหนดค่าพลังงานของแขนหุ่นยนต์
setPercentage(int p)	// กำหนดค่าความน่าจะเป็นที่จะโดนยิงของแขนหุ่นยนต์
setSize(int s)	// กำหนดค่าขนาดของแขนหุ่นยนต์

ตารางที่ ค-3 แสดงแอตทริบิวต์และเมธตอดของ Class Leg

Leg	
Energy : int	// พลังงานของขาหุ่นยนต์
Speed : int	// ความเร็วของขาหุ่นยนต์
Strength : int	// ความแข็งแกร่งของขาหุ่นยนต์
Weight : int	// น้ำหนักของขาหุ่นยนต์
name : String	// ชื่อของขาหุ่นยนต์
percentage : int	// ความน่าจะเป็นที่จะโดนยิงของขาหุ่นยนต์
size : int	// ขนาดของขาหุ่นยนต์
Leg(int energy, int weight, int strength, int speed, int name)	// Constructor ของขาหุ่นยนต์
getEnergy() : int	// คืนค่าพลังงานของขาหุ่นยนต์
getSpeed() : int	// คืนค่าความเร็วของขาหุ่นยนต์
getStrength() : int	// คืนค่าความแข็งแกร่งของขาหุ่นยนต์
getWeight() : int	// คืนค่าน้ำหนักของขาหุ่นยนต์
getName() : String	// คืนค่าชื่อของขาหุ่นยนต์
getPercentage() : int	// คืนค่าความน่าจะเป็นที่จะโดนยิงของขาหุ่นยนต์
getSize() : int	// คืนค่าขนาดของขาหุ่นยนต์
setEnergy(int energy)	// กำหนดค่าพลังงานของขา
setPercentage(int p)	// กำหนดค่าความน่าจะเป็นที่จะโดนยิงของขาหุ่นยนต์
setSize(int s)	// กำหนดค่าขนาดของขาหุ่นยนต์

ตารางที่ ค-4 แสดงแอตทริบิวต์และเมธตอดของ Class Shoulder

Shoulder	
Energy : int	// พลังงานของไหล่หุ่นยนต์
Strength : int	// ความแข็งแกร่งของไหล่หุ่นยนต์
Weight : int	// น้ำหนักของไหล่หุ่นยนต์
Name : String	// ชื่อของไหล่หุ่นยนต์
Percentage : int	// ความน่าจะเป็นที่จะโดนยิงของไหล่หุ่นยนต์
Size : int	// ขนาดของไหล่หุ่นยนต์
Shoulder(int energy, int weight, int strength, String name)	// Constructor ของไหล่
Shoulder(int energy, int weight, int strength, Weapon Heavyweapon, String name)	// Constructor ของไหล่
getEnergy() : int	// คืนค่าพลังงานของไหล่
getStrength() : int	// คืนค่าความแข็งแกร่งของไหล่
getWeight() : int	// คืนค่าน้ำหนักของไหล่
getName() : String	// คืนค่าชื่อไหล่หุ่นยนต์
getPercentage() : int	// คืนค่าความน่าจะเป็นที่จะโดนยิงของไหล่หุ่นยนต์
getSize() : int	// คืนค่าขนาดของไหล่หุ่นยนต์
getWeapon() : Weapon	// คืนค่าอาวุธที่ติดอยู่กับไหล่
setPercentage(int p)	// กำหนดค่าความน่าจะเป็นที่จะโดนยิงของไหล่หุ่นยนต์
setSize(int s)	// กำหนดขนาดของไหล่หุ่นยนต์

ตารางที่ ค-5 แสดงแอตทริบิวต์และเมธตอดของ Class Jet

Jet	
Oil : int	// เชื้อเพลิงของไอพ่นของหุ่นยนต์
Speed : int	// ค่าความเร็วของไอพ่นของหุ่นยนต์
name : String	// ชื่อของไอพ่นของหุ่นยนต์
Jet(int oil , int speed, String name)	// Constructor ของไอพ่นของหุ่นยนต์
getOil() : int	// คืนค่าเชื้อเพลิงของไอพ่นของหุ่นยนต์
getSpeed() : int	// คืนค่าความเร็วของไอพ่นของหุ่นยนต์
setOil(int oil)	// กำหนดค่าเชื้อเพลิงของไอพ่นของหุ่นยนต์
getName() : String	// คืนค่าชื่อของไอพ่นของหุ่นยนต์

ตารางที่ ค-6 แสดงแอตทริบิวต์และเมธตอดของ Class Weapon

Weapon	
accuracy : int	// ความแม่นยำของอาวุธของหุ่นยนต์
range : int	// รัศมีการยิงของอาวุธของหุ่นยนต์
weight : int	// น้ำหนักของอาวุธของหุ่นยนต์
ammo : Ammo	// ชนิดของกระสุนที่ใช้
amount : int	// จำนวนกระสุนที่มีอยู่ในอาวุธของหุ่นยนต์
name : String	// ชื่ออาวุธของหุ่นยนต์
Weapon(int weight,int rang, int accuracy, Ammo ammo, int amount, String name)	// Constructor ของอาวุธ
getAccuracy() : int	// คืนค่าความแม่นยำของอาวุธของหุ่นยนต์
getAmount(): int	// คืนค่าจำนวนกระสุนของหุ่นยนต์
getRange(): int	// คืนค่ารัศมีการยิงของอาวุธของหุ่นยนต์
getWeight(): int	// คืนค่าค่าน้ำหนักของอาวุธของหุ่นยนต์
getAmmo() : Ammo	// คืนค่ากระสุนที่ใช้
getName() : String	// คืนค่าชื่ออาวุธของหุ่นยนต์
setAmount(int a)	// กำหนดค่าจำนวนกระสุน

ตารางที่ ค-7 แสดงแอตทริบิวต์และเมธตอดของ Class Ammo

Ammo	
Accuracy : int	// ค่าความแม่นยำของกระสุน
Attack : int	// พลังทำลายของกระสุน
Speed : int	// ค่าความเร็วของกระสุน
CurX : int	// ตำแหน่ง x ปัจจุบันของกระสุน
CurY : int	// ตำแหน่ง y ปัจจุบันของกระสุน
Degree : int	// องศาการเคลื่อนที่ของกระสุน
DesX : int	// ตำแหน่ง x ปลายทางของกระสุน
DesY : int	// ตำแหน่ง y ปลายทางของกระสุน
Range : int	// ระยะทางสูงสุดที่กระสุนเคลื่อนที่ได้
Target : Jrobot	// หุ่นยนต์เป้าหมาย
Name : String	// ชื่อกระสุน
Ammo(int attack, int speed, int accuracy, String name)	// Constructor ของกระสุน
Ammo(Ammo ammo, int curX, int curY, int range, int degree)	// Constructor ของกระสุน
Ammo()	// Constructor ของกระสุน
getAccuracy() : int	// คืนค่าความแม่นยำของกระสุน
getCurrentX() : int	// คืนค่าตำแหน่ง x ปัจจุบันของกระสุน
getCurrentY() : int	// คืนค่าตำแหน่ง y ปัจจุบันของกระสุน
getDegree() : int	// คืนค่าองศาการเคลื่อนที่ของกระสุน
getDesX() : int	// คืนค่าตำแหน่ง x ปลายทางของกระสุน
getDesY() : int	// คืนค่าตำแหน่ง y ปลายทางของกระสุน
getRange() : int	// คืนค่าระยะทางที่เคลื่อนที่ได้ของกระสุน
getAttack() : int	// คืนค่าพลังทำลายของกระสุน
getSpeed() : int	// คืนค่าความเร็วของกระสุน
getTarget() : Jrobot	// คืนค่าเป้าหมายของกระสุน
getName() : String	// คืนค่าชื่อของกระสุน
isShootFinish() : boolean	// กระสุนถูกทำลายหรือไม่
setAccuracy(int a)	// กำหนดค่าความแม่นยำของกระสุน

ตารางที่ ค-7(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class Ammo

Ammo	
setCurrentX(int x)	// กำหนดค่าตำแหน่ง x ปัจจุบันของกระสุน
setCurrentY(int y)	// กำหนดค่าตำแหน่ง y ปัจจุบันของกระสุน
setDesX(int x)	// กำหนดค่าตำแหน่ง x ปลายทางของกระสุน
setDesY(int y)	// กำหนดค่าตำแหน่ง y ปลายทางของกระสุน
setRange(int r)	// กำหนดค่าระยะทางที่เคลื่อนที่ได้ของกระสุน
setAttack(int a)	// กำหนดค่าพลังทำลายของกระสุน
setSpeed(int s)	// กำหนดค่าความเร็วของกระสุน
setDegree(int d)	// กำหนดค่าองศาการเคลื่อนที่ของกระสุน
setTarget(Jrobot j)	// กำหนดค่าเป้าหมายของกระสุน

ตารางที่ ค-8 แสดงแอตทริบิวต์และเมธตอดของ Class CPU

CPU	
attacked : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์ถูกโจมตีอยู่หรือไม่
common : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์ปกติอยู่หรือไม่
name : String	// สถานะว่าขณะนั้นเกิดเหตุการณ์ถูกโจมตีอยู่หรือไม่
crashBorder : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์ชนขอบอยู่หรือไม่
crashEnemy : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์ชนศัตรูอยู่หรือไม่
foundBorder : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์พบขอบอยู่หรือไม่
foundEnemy : boolean	// สถานะว่าขณะนั้นเกิดเหตุการณ์พบศัตรูอยู่หรือไม่
scanXleft : int	// ตำแหน่ง x ด้านมุมซ้ายของการสแกน
scanXmiddle : int	// ตำแหน่ง x ตรงกลางของการสแกน
scanXright : int	// ตำแหน่ง x ด้านมุมขวาของการสแกน
scanYleft : int	// ตำแหน่ง y ด้านมุมซ้ายของการสแกน
scanYmiddle : int	// ตำแหน่ง y ตรงกลางของการสแกน
scanYright : int	// ตำแหน่ง y ด้านมุมขวาของการสแกน
faceDegree : int	// องศาใบหน้าของหุ่นยนต์

ตารางที่ ค-8(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class CPU

CPU	
- currentX : int	// ตำแหน่งปัจจุบันของหุ่นยนต์
- currentY : int	// ตำแหน่งปัจจุบันของหุ่นยนต์
- scanLength : int	// ระยะทางการสแกนของหุ่นยนต์
- scanDegree : int	// องศาการสแกนของหุ่นยนต์
CPU(int scanLength, int scanDegree, String name)	// Constructor ของ CPU
getAttacked() : boolean	// คืนค่าสถานะของเหตุการณ์ ถูกโจมตี
getCommon() : boolean	// คืนค่าสถานะของเหตุการณ์ common
getName : String	// คืนค่าชื่อของ CPU
getCrashBorder : boolean	// คืนค่าสถานะของเหตุการณ์ชนขอบ
getCrashEnemy : boolean	// คืนค่าสถานะของเหตุการณ์ชนศัตรู
getFoundBorder : boolean	// คืนค่าสถานะของเหตุการณ์พบขอบ
getFoundEnemy : boolean	// คืนค่าสถานะของเหตุการณ์พบศัตรู
getFaceDegree() : int	// คืนค่าองศาใบหน้า
getScanDegree() : int	// คืนค่าองศาการสแกน
getScanLength() : int	// คืนค่าความยาวการสแกน
getScanXleft() : int	// คืนค่าการสแกนจุด x ทางด้านซ้าย
getScanXmiddle() : int	// คืนค่าการสแกนจุด x ตรงกลาง
getScanXright() : int	// คืนค่าการสแกนจุด x ทางด้านขวา
getScanYleft() : int	// คืนค่าการสแกนจุด y ทางด้านซ้าย
getScanYmiddle() : int	// คืนค่าการสแกนจุด y ตรงกลาง
getScanYright() : int	// คืนค่าการสแกนจุด y ทางด้านขวา
getCurrentX (): int	// คืนค่า currentX
getCurrentY() : int	// คืนค่า currentY
setFaceDegree()	// กำหนดค่าองศาใบหน้า
setScanXleft()	// กำหนดค่าการสแกนจุด x ทางด้านซ้าย
setScanXmiddle()	// กำหนดค่าการสแกนจุด x ตรงกลาง
setScanXright()	// กำหนดค่าการสแกนจุด x ทางด้านขวา

ตารางที่ ค-8(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class CPU

CPU	
setAttacked(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์ถูกโจมตี
setCommon(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์ปกติ
setCrashBorder(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์ชนขอบ
setCrashEnemy(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์ชนศัตรู
setFoundBorder(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์พบขอบ
setFoundEnemy(boolean b)	// กำหนดสถานะของการเกิดเหตุการณ์พบศัตรู
setScanYleft()	// กำหนดค่าการสแกนจุด y ทางด้านซ้าย
setScanYmiddle()	// กำหนดค่าการสแกนจุด y ตรงกลาง
setScanYright()	// กำหนดค่าการสแกนจุด y ทางด้านขวา
setCurrentX()	// กำหนดค่า currentX
setCurrentY()	// กำหนดค่า currentY

ตารางที่ ค-9 แสดงแอตทริบิวต์และเมธตอดของ Class Scan

Scan	
xl : int	// ตำแหน่งของจุดมุมซ้ายของการสแกนในแกน x
yl : int	// ตำแหน่งของจุดมุมซ้ายของการสแกนในแกน y
xr : int	// ตำแหน่งของจุดมุมขวาของการสแกนในแกน x
yr : int	// ตำแหน่งของจุดมุมขวาของการสแกนในแกน y
xc : int	// ตำแหน่งของจุดมุมซ้ายของการสแกนในแกน x
yc : int	// ตำแหน่งของจุดกำเนิดของการสแกนในแกน y
scanBoundLength : int	// ระยะทางจากจุด (xc,yc) ถึงมุมใดมุมหนึ่งของการสแกน
scanSideLength : int	// ระยะทางจากจุด (xl,yl) ถึงจุด (xr,yr)
currentX : int	// ตำแหน่ง x ปัจจุบันของหุ่นยนต์
currentY : int	// ตำแหน่ง y ปัจจุบันของหุ่นยนต์
face: int	// องศาใบหน้าของหุ่นยนต์
xm : int	// ตำแหน่งของจุดตรงกลางขอบของการสแกนในแกน x
ym : int	// ตำแหน่งของจุดตรงกลางขอบของการสแกนในแกน y

ตารางที่ ค-9(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class Scan

Scan	
scanDegree : int	// องศาการสแกนของหุ่นยนต์
scanLength : int	// ระยะการสแกนของหุ่นยนต์
Scan(CPU cpu)	// Constructor ของการสแกน
FindSlope(int in_x, int in_y, int overx, int overy) : double	// คำนวณหาความชัน
getXL(int a) : int	// คืนค่าตำแหน่ง x ทางซ้าย
getYL(int a) : int	// คืนค่าตำแหน่ง y ทางซ้าย
getXR(int a) : int	// คืนค่าตำแหน่ง x ทางขวา
getYR(int a) : int	// คืนค่าตำแหน่ง y ทางขวา
getXM(int a) : int	// คืนค่าตำแหน่ง x ตรงกลาง
getYM(int a) : int	// คืนค่าตำแหน่ง y ตรงกลาง
getScanBoundLength()	// คืนค่าความยาวของขอบด้านข้างของการสแกน
getScanSideLength()	// คืนค่าความยาวของขอบด้านหน้าของการสแกน
findScanPoint(int faceDegree)	// ค้นหาจุดมุมของการสแกน
findBorderCutpoint(int x1,int x2)	// ค้นหาจุดของการสแกนที่ตัดกับขอบสนาม

ตารางที่ ค-10 แสดงแอตทริบิวต์และเมธตอดของ Class ScanBorder

ScanBorder	
BorDer : int	// พบขอบด้านใด
Direction : int	// ทิศทางที่พบขอบ
Distance : int	// ระยะห่างระหว่างหุ่นกับขอบ
FoundBorder : boolean	// สถานะว่าพบขอบหรือไม่
ScanDistance : int	// ระยะการสแกน
CheckSideBorder(int leftpointX, int leftpointY, int rightpointX, int rightpointY, int centerpointX, int centerpointY)	// คำนวณว่าพบขอบหรือไม่
getBorder() : int	// คืนค่าขอบที่พบ
getDirection() : int	// คืนค่าทิศทางที่พบ

ตารางที่ ค-10(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class ScanBorder

ScanBorder	
getDistance() : int	// คืนค่าระยะทางห่างจากขอบที่พบ
getFoundBorder() : boolean	// คืนค่าว่าพบขอบหรือไม่

ตารางที่ ค-11 แสดงแอตทริบิวต์และเมธตอดของ ScanEnemy

ScanEnermy	
FoundEnemy : boolean	// สถานะว่าพบศัตรูหรือไม่
EnemyXbc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXbl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXbr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXcl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXcr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXtc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXtemp : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXtl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyXtr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYbc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYbl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYbr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYcl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYcr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYtc : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYtemp : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYtl : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู
EnemyYtr : int	// ตำแหน่งขอบของตัวหุ่นยนต์ของศัตรู

ตารางที่ ค-11(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ ScanEnemy

ScanEnermy	
Myx1 : int	// ตำแหน่งจุดต่างๆของการสแกน
Myxl : int	// ตำแหน่งจุดต่างๆของการสแกน
Myxr : int	// ตำแหน่งจุดต่างๆของการสแกน
Myy1 : int	// ตำแหน่งจุดต่างๆของการสแกน
Myyl : int	// ตำแหน่งจุดต่างๆของการสแกน
Myyr : int	// ตำแหน่งจุดต่างๆของการสแกน
CheckFoundEnemy(int CurrentRX, Int CurrentRY, int ScanXmiddle, int ScanYmiddle, int ScanXleft, int ScanYleft, int ScanXRight, int ScanYRight)	// คำนวณว่าพบศัตรูหรือไม่
getFounEnemy() : boolean	// คืนค่าว่าพบขอบหรือไม่

ตารางที่ ค-12 แสดงแอตทริบิวต์และเมธตอดของ Class Rotate

Rotate	
degree : int	// องศาที่จะหมุน
direction : int	// การหมุนนั้นเสร็จสิ้นหรือยัง
destination : int	// องศาปลายทางที่จะหมุน
face : int	// องศาใบหน้าของหุ่นยนต์
Rotate(int degree, int direction)	// Constructor ของการหมุน
findDestination()	// คำนวณหามุมที่จะต้องหมุนไป
getDegree() : int	// คืนค่ามุมที่จะหมุน
isRotateFinish() : boolean	// การหมุนนั้นเสร็จสิ้นหรือยัง
setNewRobotFace(Jrobot robot, int rotateTime)	// กำหนดองศาใบหน้าของหุ่นใหม่

ตารางที่ ค-13 แสดงแอตทริบิวต์และเมธตอดของ Class Shoot

Shoot	
gun : Weapon	// อาวุธที่ใช้ยิง
bull : Ammo	// กระสุนที่ใช้ยิง
speed : int	// ความเร็วของการยิง
accuracy : int	// ความแม่นยำในการยิง
enemy : Jrobot	// หุ่นยนต์เป้าหมายของการยิง
moveTime : int	// ระยะทางที่กระสุนยังสามารถเคลื่อนที่ต่อไปได้
mb : ManageBullet	// ที่เก็บกระสุนที่ถูกยิงออกมาโดยทำงานในลักษณะคิว
robot : Jrobot	// หุ่นยนต์ที่ทำการยิง
Shoot(Weapon gun, Jrobot robot, ManageBullet mb)	// Constructor ของการยิง
getDerivation() : int	// คืนค่าความเบี่ยงเบนของกระสุน
getDegree() : int	// คืนค่าองศาที่เบี่ยงเบนไป
fire() : boolean	// สั่งยิง
findDestination(Ammo am)	// ค้นหาตำแหน่งสุดท้ายของกระสุน
setAmmoPosition(Ammo am)	// กำหนดตำแหน่งกระสุนใหม่

ตารางที่ ค-14 แสดงแอตทริบิวต์และเมธตอดของ Class Move

Move	
xLeft1 : int	// ตำแหน่งขอบซ้ายของหุ่นตัวที่1
xLeft2 : int	// ตำแหน่งขอบซ้ายของหุ่นตัวที่2
xRight1 : int	// ตำแหน่งขอบขวาของหุ่นตัวที่1
xRight2 : int	// ตำแหน่งขอบขวาของหุ่นตัวที่2
yDown1 : int	// ตำแหน่งขอบล่างของหุ่นตัวที่1
yDown2 : int	// ตำแหน่งขอบล่างของหุ่นตัวที่2
yUp1 : int	// ตำแหน่งขอบบนของหุ่นตัวที่1
yUp2 : int	// ตำแหน่งขอบบนของหุ่นตัวที่2

ตารางที่ ค-14(ต่อ) แสดงแอตทริบิวต์และเมธอดของ Class Move

Move	
distance : int	// ระยะทางการเคลื่อนที่ของหุ่น
degree : int	// องศาที่หุ่นเคลื่อนที่เทียบกับองศาใบหน้า
face : int	// องศาใบหน้าของหุ่น
overEnemy : boolean	// บอกให้ทราบว่าเกิดการชนหรือไม่
curX : int	// ตำแหน่ง x ปัจจุบันของหุ่น
curY : int	// ตำแหน่ง y ปัจจุบันของหุ่น
desX : int	// ตำแหน่ง x ปลายทางของหุ่น
desY : int	// ตำแหน่ง y ปลายทางของหุ่น
direction : int	// ทิศทางการเคลื่อนที่ของหุ่น
enemy : Jrobot	// หุ่นยนต์ของฝ่ายตรงข้าม
overBorder : boolean	// สถานะแสดงให้ทราบว่าหุ่นนั้นเคลื่อนที่ชนขอบหรือไม่
robot : Jrobot	// หุ่นยนต์ที่ทำการเคลื่อนที่
findDestination()	// คำนวณหาตำแหน่งปลายทาง
getCurX() : int	// คืนค่าตำแหน่ง x ปัจจุบันของหุ่น
getCurY() : int	// คืนค่าตำแหน่ง y ปัจจุบันของหุ่น
getDesX() : int	// คืนค่าตำแหน่ง x ปลายทางของหุ่น
getDesY() : int	// คืนค่าตำแหน่ง y ปลายทางของหุ่น
getDistance(): int	// คืนค่าระยะทางที่หุ่นจะต้องเคลื่อนที่
getOverEnemy()	// คืนค่าเพื่อบอกว่าหุ่นชนกันหรือไม่
isMoveFinish() : boolean	// เช็คว่าการเคลื่อนที่นั้นทำเสร็จเรียบร้อยหรือไม่
setDegree(int x)	// กำหนดองศาการเคลื่อนที่
setDirection(int x)	// กำหนดทิศทางการเคลื่อนที่
setDistance(int x)	// กำหนดระยะทางการเคลื่อนที่
setNewRobotPosition(int moveTime)	// คำนวณตำแหน่งของหุ่นยนต์ใหม่หลังจากการเคลื่อนที่
setNewRobotPosition(int moveTime, Jet jet)	// คำนวณตำแหน่งของหุ่นยนต์ใหม่หลังจากการเคลื่อนที่ด้วยไอพ่น
Move(int de,int di,int dis, JField jf,Jrobot enemy,)	// constructor ของการเคลื่อนที่
isCrashBorder() : boolean	// ตรวจสอบว่าการเคลื่อนที่ของหุ่นทำให้ชนขอบหรือไม่

ตารางที่ ค-15 แสดงแอตทริบิวต์และเมธอดของ Class JProcess

JProcess	
c1 : Collection	// เก็บเมสเสจที่จะประมวลผลของหุ่นยนต์ตัวที่ 1
c2 : Collection	// เก็บเมสเสจที่จะประมวลผลของหุ่นยนต์ตัวที่ 2
delayShoot1 : int	// เวลาหน่วงในการใส่กระสุนก่อนยิงของหุ่นตัวที่ 1
delayShoot2 : int	// เวลาหน่วงในการใส่กระสุนก่อนยิงของหุ่นตัวที่ 2
jm : Jmsg	// เมสเสจที่จะประมวลผล
mb1 : ManageBullet	// ที่เก็บกระสุนที่ถูกยิงของหุ่นตัวที่ 1
mb2 : ManageBullet	// ที่เก็บกระสุนที่ถูกยิงของหุ่นตัวที่ 2
move : Move	// คลาสที่ใช้ประมวลผลด้านการเคลื่อนที่
r1 : Jrobot	// หุ่นยนต์ตัวที่ 1
r2 : Jrobot	// หุ่นยนต์ตัวที่ 2
rotate : Rotate	// คลาสที่ใช้ประมวลผลด้านการหมุนของหุ่นยนต์
shoot : Shoot	// คลาสที่ใช้ประมวลผลด้านการยิงของหุ่นยนต์
JProcess(JField jf,Jrobot r1, Jrobot r2, Collection c1, Collection c2, ManageBullet mb1, ManageBullet mb2)	// Constructor ของ Jprocess
processJmsg(Jmsg jm)	// จัดการกับเมสเสจที่รับมาจาก Field
processMove(Jmsg jm)	// จัดการกับเมสเสจการเคลื่อนที่
processRotate(Jmsg jm)	// จัดการกับเมสเสจการหมุน
processShoot(Jmsg jm)	// จัดการกับเมสเสจการยิงกระสุน
processCrashBorder(Jrobot robot)	// จัดการกับการชนขอบของหุ่นยนต์
processCrashEnemy(Jrobot robot, Jrobot Erobot)	// จัดการกับการชนกันของหุ่นยนต์
processHitted(Jrobot robot)	// จัดการประมวลผลเมื่อหุ่นยนต์ถูกยิง
processJetMove(Jmsg jm)	// จัดการกับเคลื่อนที่ๆใช้ไอพ่น

ตารางที่ ค-16 แสดงแอตทริบิวต์และเมธตอดของ Class ManageBullet

ManageBullet	
number : int	// จำนวนกระสุนที่เหลืออยู่ใน ManageBullet
p : int	// พอยน์เตอร์ชี้ตำแหน่งของวัตถุขณะนั้น
v : Vector	// คลาสที่ใช้เก็บวัตถุที่มีลักษณะแตกต่างกัน
get(int index) : Ammo	// คืนค่ากระสุน
getNumber() : int	// คืนค่าจำนวนกระสุนที่อยู่ใน ManageBullet
getP() : int	// คืนค่าตำแหน่งพอยน์เตอร์
incP() : int	// เพิ่มค่าพอยเตอร์ขึ้นหนึ่ง
put(Ammo ammo)	// เก็บกระสุนลงในคลาส Vector
isEmpty() : boolean	// เช็คว่า ManageBullet ว่างอยู่หรือไม่
set(int index,Ammo ammo)	// แทนที่กระสุนลงในตำแหน่งดังกล่าว
setP(int pointer)	// กำหนดตำแหน่งพอยน์เตอร์
removeAmmo(int index)	// ลบกระสุนออกจาก ManageBullet

ตารางที่ ค-17 แสดงแอตทริบิวต์และเมธตอดของ Class InterfaceJRobot

InterfaceJRobot	
Attacked()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์ถูกโจมตี
Common()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์ปกติ
CrashBorder()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์ชนขอบ
CrashEnemy()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์ชนศัตรู
FoundBorder()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์เจอขอบ
FoundEnemy()	// ให้ผู้เล่น Implement เองภายใต้เหตุการณ์เจอศัตรู
getBody ()	// คืนค่าร่างกายของหุ่น
getCpu()	// คืนค่า Cpu ของหุ่น
getJet ()	// คืนค่า ไอพ่นของหุ่น
getLeftArm ()	// คืนค่าแขนซ้ายของหุ่น
getRightArm ()	// คืนค่าแขนขวาของหุ่น

ตารางที่ ค-17(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class InterfaceJRobot

InterfaceJRobot	
getLeftLightWeapon()	// คืนค่าอาวุธเบาข้างซ้ายของหุ่น
getRightLightWeapon ()	// คืนค่าอาวุธเบาข้างขวาของหุ่น
getLeftHeavyWeapon()	// คืนค่าอาวุธหนักข้างซ้ายของหุ่น
getRightHeavyWeapon()	// คืนค่าอาวุธหนักข้างขวาของหุ่น
getLegObject()	// คืนค่าขาของหุ่น
init(JField jf, Collection c, int id)	// กำหนดให้หุ่นยนต์อ้างถึง JField,Collection และ id

ตารางที่ ค-18 แสดงแอตทริบิวต์และเมธตอดของ Class JavaRobot

JavaRobot	
RobotNo : int	// หมายเลขของหุ่นยนต์
attackedStop : boolean	// สถานะว่าเหตุการณ์ถูกโจมตีหยุดอยู่หรือไม่
foundEnemyStop : boolean	// สถานะว่าเหตุการณ์พบศัตรูหยุดอยู่หรือไม่
foundBorderStop : boolean	// สถานะว่าเหตุการณ์พบขอบหยุดอยู่หรือไม่
crashEnemyStop : boolean	// สถานะว่าเหตุการณ์ชนศัตรูหยุดอยู่หรือไม่
crashBorderStop : boolean	// สถานะว่าเหตุการณ์ชนขอบหยุดอยู่หรือไม่
currentEvent : int	// เหตุการณ์ปัจจุบันของหุ่น
hitted : boolean	// สถานะที่บอกว่าหุ่นถูกยิงอยู่หรือไม่
c : Collection	// คลาสที่ใช้เก็บเมสเสจที่จะประมวลผลของหุ่นยนต์
tc : TimeControl	// จัดการเกี่ยวกับการควบคุมเวลาให้สัมพันธ์กัน
_BODY : Body	// เกราะของหุ่นยนต์
_CPU : CPU	// CPU ของหุ่นยนต์
_JET : Jet	// ไอพ่นของหุ่นยนต์
_LEG : Leg	// ขาของหุ่นยนต์
_ARM_L : Arm	// แขนซ้ายของหุ่นยนต์
_ARM_R: Arm	// แขนขวาของหุ่นยนต์

ตารางที่ ค-18(ต่อ) แสดงแอตทริบิวต์และเมธอดของ Class JavaRobot

JavaRobot	
_SHOULDER_L : Shoulder	// ไหล่ซ้ายของหุ่นยนต์
_SHOULDER_R: Shoulder	// ไหล่ขวาของหุ่นยนต์
damage : int	// ความเสียหายทั้งหมดที่จะเกิดขึ้นกับตัวหุ่น
_HEAVYAMMO_L : Ammo	// กระสุนของอาวุธประเภทหนักข้างซ้าย
_HEAVYAMMO_R: Ammo	// กระสุนของอาวุธประเภทหนักข้างขวา
_HEAVYWEAPON_L: Weapon	// อาวุธประเภทโจมตีหนักข้างซ้าย
_HEAVYWEAPON_R: Weapon	// อาวุธประเภทโจมตีหนักข้างขวา
_LIGHTAMMO_L: Ammo	// กระสุนของอาวุธประเภทโจมตีเบาข้างซ้าย
_LIGHTAMMO_R: Ammo	// กระสุนของอาวุธประเภทโจมตีเบาข้างขวา
_LIGHTWEAPON_L: Weapon	// อาวุธประเภทโจมตีเบาข้างซ้าย
_LIGHTWEAPON_R: Weapon	// อาวุธประเภทโจมตีเบาข้างขวา
allEventStart()	// สั่งให้ทุกเหตุการณ์สามารถทำงานได้ยกเว้น Common
allEventStop()	// สั่งให้ทุกเหตุการณ์สามารถหยุดทำงานได้ยกเว้น Common
common()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์ปกติที่เกิดขึ้น
attacked()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์ถูกโจมตีที่เกิดขึ้น
foundEnemy()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์พบศัตรูที่เกิดขึ้น
foundBorder()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์พบขอบที่เกิดขึ้น
crashEnemy()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์ชนศัตรูที่เกิดขึ้น
crashBorder()	// เป็นเมธอดให้ผู้เล่นสามารถ Implement ในการจัดการกับเหตุการณ์ชนขอบที่เกิดขึ้น

ตารางที่ ค-18(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class JavaRobot

JavaRobot	
attackedEnd()	// สร้างเมสเสจที่บอกตำแหน่งสิ้นสุดของเหตุการณ์ถูกโจมตี แล้วนำไปเก็บในคอลเลคชัน
foundEnemyEnd()	// สร้างเมสเสจที่บอกตำแหน่งสิ้นสุดของเหตุการณ์พบศัตรู แล้วนำไปเก็บในคอลเลคชัน
foundBorderEnd()	// สร้างเมสเสจที่บอกตำแหน่งสิ้นสุดของเหตุการณ์พบขอบ แล้วนำไปเก็บในคอลเลคชัน
crashEnemyEnd()	// สร้างเมสเสจที่บอกตำแหน่งสิ้นสุดของเหตุการณ์ชนศัตรู แล้วนำไปเก็บในคอลเลคชัน
crashBorderEnd()	// สร้างเมสเสจที่บอกตำแหน่งสิ้นสุดของเหตุการณ์ชนขอบ แล้วนำไปเก็บในคอลเลคชัน
attackedStart()	// สั่งให้เหตุการณ์ถูกโจมตีเริ่มทำงาน
foundEnemyStart ()	// สั่งให้เหตุการณ์พบศัตรูเริ่มทำงาน
foundBorderStart ()	// สั่งให้เหตุการณ์พบขอบเริ่มทำงาน
crashEnemyStart ()	// สั่งให้เหตุการณ์ชนศัตรูเริ่มทำงาน
crashBorderStart ()	// สั่งให้เหตุการณ์ชนขอบเริ่มทำงาน
attackedStop()	// สั่งให้เหตุการณ์ถูกโจมตีหยุดทำงาน
foundEnemyStop ()	// สั่งให้เหตุการณ์พบศัตรูหยุดทำงาน
foundBorderStop ()	// สั่งให้เหตุการณ์พบขอบหยุดทำงาน
crashEnemyStop ()	// สั่งให้เหตุการณ์ชนศัตรูหยุดทำงาน
crashBorderStop ()	// สั่งให้เหตุการณ์ชนขอบหยุดทำงาน
attackedClear()	// ลบเมสเสจของเหตุการณ์ถูกโจมตีในคอลเลคชัน
foundEnemyClear ()	// ลบเมสเสจของเหตุการณ์พบศัตรูในคอลเลคชัน
foundBorderClear ()	// ลบเมสเสจของเหตุการณ์พบขอบในคอลเลคชัน
crashEnemyClear ()	// ลบเมสเสจของเหตุการณ์ชนศัตรูในคอลเลคชัน
crashBorderClear ()	// ลบเมสเสจของเหตุการณ์ชนขอบในคอลเลคชัน
init(JField jf, Collection c, int id)	// กำหนดค่าเริ่มต้นของหุ่น
move(int distance, int degree, int direction, Leg leg)	// สั่งให้หุ่นยนต์เคลื่อนที่ด้วยขา

ตารางที่ ค-18(ต่อ) แสดงแอตทริบิวต์และเมธอดของ Class JavaRobot

JavaRobot	
move(int distance, int degree, int direction, Jet jet)	// สั่งให้หุ่นยนต์เคลื่อนที่ด้วยไอพ่น
rotate(int degree, int direction)	// สั่งให้หุ่นยนต์หมุน
shoot(Arm arm)	// สั่งให้หุ่นยนต์ยิง
getId()	// คืนค่าหมายเลขหุ่นยนต์
getHitted() : boolean	// คืนค่าสถานะว่าหุ่นยนต์ถูกยิงหรือไม่
getattackedStop()	// คืนค่าว่าขณะนั้นเหตุการณ์ถูกโจมตีนั้นทำงานได้หรือไม่
getfoundEnemyStop()	// คืนค่าว่าขณะนั้นเหตุการณ์พบศัตรูนั้นทำงานได้หรือไม่
getfoundBorderStop ()	// คืนค่าว่าขณะนั้นเหตุการณ์พบขอบนั้นทำงานได้หรือไม่
getcrashEnemyStop ()	// คืนค่าว่าขณะนั้นเหตุการณ์ชนศัตรูนั้นทำงานได้หรือไม่
getcrashBorderStop ()	// คืนค่าว่าขณะนั้นเหตุการณ์ชนขอบนั้นทำงานได้หรือไม่
getCurrentEvent() : int	// คืนค่าเหตุการณ์ปัจจุบันของหุ่นยนต์
getArmLeft()	// คืนค่าแขนซ้ายของหุ่นยนต์
getArmRight()	// คืนค่าแขนขวาของหุ่นยนต์
getBody()	// คืนค่าเกราะของหุ่นยนต์
getCPU()	// คืนค่า CPU ของหุ่นยนต์
getDamage()	// คืนค่าความเสียหายทั้งหมดที่เกิดกับหุ่นยนต์ขณะนั้น
getHeavyWeaponLeft()	// คืนค่าอาวุธประเภทโจมตีหนักด้านซ้ายของหุ่นยนต์
getHeavyWeaponRight()	// คืนค่าอาวุธประเภทโจมตีหนักด้านขวาของหุ่นยนต์
getJet()	// คืนค่าไอพ่นของหุ่นยนต์
getLeg()	// คืนค่าขาของหุ่นยนต์
getLightWeaponLeft()	// คืนค่าอาวุธประเภทโจมตีเบาด้านซ้ายของหุ่นยนต์
getLightWeaponRight()	// คืนค่าอาวุธประเภทโจมตีเบาด้านขวาของหุ่นยนต์
getShoulderLeft()	// คืนค่าไหล่ด้านซ้ายของหุ่นยนต์
getShoulderRight()	// คืนค่าไหล่ด้านขวาของหุ่นยนต์
setCurrentEvent(int event)	// กำหนดค่าเหตุการณ์ปัจจุบันของหุ่นยนต์
setHitted(boolean hit)	// กำหนดสถานะว่าหุ่นถูกยิงหรือไม่

ตารางที่ ค-18(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class JavaRobot

JavaRobot	
setDamage(int d)	// กำหนดค่าความเสียหายที่หุ่นยนต์ได้รับ
calPercentage()	// คำนวนความน่าจะเป็นในการถูกยิงของส่วนประกอบทั้งหมดของหุ่นยนต์
clearAllAction()	// ลบเมสเสจทั้งหมดในคลาส collection ทั้งหมด
commonClear()	// ลบเมสเสจของเหตุการณ์ปกติในคอลเลคชัน

ตารางที่ ค-19 แสดงแอตทริบิวต์และเมธตอดของ Class Collection

Collection	
p : int	// พอยน์เตอร์ที่ชี้ตำแหน่งปัจจุบันในคอลเลคชัน
v : Vector	// ใช้สำหรับเก็บเมสเสจ
get(int index) : Jmsg	// คืนค่าเมสเสจตามตำแหน่งที่ระบุ
getNumber() : int	// คืนค่าจำนวนเมสเสจภายในคอลเลคชัน
isEmpty() : boolean	// คืนคาว่าคอลเลคชันว่างอยู่หรือไม่
put(Jmsg msg)	// นำเมสเสจไปเก็บในคอลเลคชัน
removeMsg(int index)	// ลบเมสเสจตามตำแหน่งระบุในคอลเลคชันออก
set(int index, Jmsg jm)	// นำเมสเสจไปแทนที่ในคอลเลคชันตามตำแหน่งที่ระบุไว้
collection()	// constructor ของคลาส Collection
clearAll()	// ลบเมสเสจทั้งหมดในคอลเลคชัน
clearAttacked()	// ลบเมสเสจของเหตุการณ์ ถูกโจมตีทั้งหมดในคอลเลคชัน
clearCommon()	// ลบเมสเสจของเหตุการณ์ปกติทั้งหมดในคอลเลคชัน
clearCrashBorder()	// ลบเมสเสจของเหตุการณ์ชนขอบทั้งหมดในคอลเลคชัน
clearCrashEnemy()	// ลบเมสเสจของเหตุการณ์ชนศัตรูทั้งหมดในคอลเลคชัน
clearFoundBorder()	// ลบเมสเสจของเหตุการณ์พบขอบทั้งหมดในคอลเลคชัน
clearFoundEnemy()	// ลบเมสเสจของเหตุการณ์พบศัตรูทั้งหมดในคอลเลคชัน

ตารางที่ ค-20 แสดงแอตทริบิวต์และเมธตอดของ Class Jfield

JField	
R1 : String	// หุ่นยนต์ตัวที่ 1
R2 : String	// หุ่นยนต์ตัวที่ 2
c1 : Collection	// คิวเก็บ Msg ที่ยังไม่ถูกประมวลผลสำหรับหุ่นตัวที่ 1
c2 : Collection	// คิวเก็บ Msg ที่ยังไม่ถูกประมวลผลสำหรับหุ่นตัวที่ 2
mb1 : ManageBullet	// คิวเก็บกระสุนที่อยู่ในสนามของหุ่นยนต์ตัวที่ 1
mb2 : ManageBullet	// คิวเก็บกระสุนที่อยู่ในสนามของหุ่นยนต์ตัวที่ 2
cancallEvent1 : boolean	// สถานะที่บอกว่าสามารถนำเมสเสจมาเก็บได้หรือไม่
cancallEvent2 : boolean	// สถานะที่บอกว่าสามารถนำเมสเสจมาเก็บได้หรือไม่
crashBorder1 : boolean	// สถานะที่บอกว่าหุ่นยนต์ตัวที่ 1 ชนขอบหรือไม่
crashBorder2 : boolean	// สถานะที่บอกว่าหุ่นยนต์ตัวที่ 2 ชนขอบหรือไม่
crashEnemy1 : boolean	// สถานะที่บอกว่าหุ่นยนต์ตัวที่ 1 ชนหุ่นยนต์ตัวอื่นหรือไม่
crashEnemy2 : boolean	// สถานะที่บอกว่าหุ่นยนต์ตัวที่ 2 ชนหุ่นยนต์ตัวอื่นหรือไม่
delay : int	// ใช้หน่วงเวลาในการแสดงผล
endGame : boolean	// สถานะบอกให้ทราบว่าสิ้นสุดเกม
jp : JPprocess	// จัดการด้านการประมวลผลเมสเสจของหุ่นยนต์
r1 : R1	// หุ่นยนต์ตัวที่ 1
r2 : R2	// หุ่นยนต์ตัวที่ 1
r1Hitted : boolean	// สถานะว่าหุ่นยนต์ตัวที่ 1 ถูกยิงหรือไม่
r2Hitted : boolean	// สถานะว่าหุ่นยนต์ตัวที่ 2 ถูกยิงหรือไม่
scan1 : Scan	// จัดการด้านการสแกนของหุ่นยนต์ตัวที่ 1
scan2 : Scan	// จัดการด้านการสแกนของหุ่นยนต์ตัวที่ 2
scanb1 : ScanBorder	// จัดการด้านการสแกนขอบสนามของหุ่นยนต์ตัวที่ 1
scanb1 : ScanBorder	// จัดการด้านการสแกนขอบสนามของหุ่นยนต์ตัวที่ 2
scane1 : ScanEnemy	// จัดการด้านการสแกนศัตรูของหุ่นยนต์ตัวที่ 1
scane1 : ScanEnemy	// จัดการด้านการสแกนศัตรูของหุ่นยนต์ตัวที่ 2

ตารางที่ ค-20(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class Jfield

JField	
tc : TimeControl	// ควบคุมเวลาให้สัมพันธ์กันระหว่งหุ่นยนต์กับสนาม
th : Thread	// จัดการด้านการควบคุมด้านการรับเมสเสจมาประมวลผล
Timeout : boolean	// สถานะหมดเวลาในเกม
JField()	// Constructor ของ Jfield
drawRobotEnemy(Graphics2D g)	// แสดงผลเกี่ยวกับพลังงานของสนาม
paintBullet(Graphics2D g)	// แสดงผลเกี่ยวกับกระสุนในสนาม
sleep(int d)	// ใช้หน่วงเวลาในการแสดงผล
putMsg(Collection c, Jmsg j)	// นำ Msg ไปเก็บในคอลเลคชัน
callEvent()	// จัดการนำเมสเสจของเหตุการณ์ต่างไปเก็บลงคอลเลคชัน
workplace()	// จัดการด้านการทำงานหลักของสนามรบ
setCanCallAttacked(int robotID, boolean b)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์ถูกโจมตี
setCanCallCrashBorder(int robotID, boolean b)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์ชนขอบ
setCanCallCrashEnemy(int robotID, boolean b)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์ชนศัตรู
setCanCallFoundBorder(int robotID, boolean b)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์พบขอบ
setCanCallFoundEnemy(int robotID, boolean b)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์พบศัตรู
setCrashBorder(int robotNo, boolean crashBorder)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์ชนขอบ
setCrashEnemy(int robotID, boolean crashEnemy)	// กำหนดหุ่นยนต์ให้สามารถทำงานในเหตุการณ์ชนศัตรู
checkEndGame()	//ตรวจสอบว่าจบเกมหรือยัง
checkTimeOut()	//ตรวจสอบว่าหมดเวลาหรือยัง
eventHandler()	//ตรวจสอบและแจ้งเหตุการณ์ที่เกิดขึ้นให้กับหุ่นยนต์

ตารางที่ ค-21 แสดงแอตทริบิวต์และเมธอดของ Class TimeControl

TimeControl	
th : Thread	// ควบคุมการนับเวลา
t : TimeControl	// วัตถุ TimeControl ที่ให้หุ่นยนต์และสนามใช้ร่วมกัน
time : long	// เวลาของ TimeControl
TimeControl()	// Constructor ของ TimeControl
getTimeControl()	// คืนค่า TimeControl ปัจจุบัน
getTime() : long	// คืนค่าเวลา
run()	// เริ่มจับเวลา

ตารางที่ ค-22 แสดงแอตทริบิวต์และเมธอดของ Class Jmsg

Jmsg	
time : int	// เวลาขณะที่เมสเสจถูกสร้าง
whose : int	// หมายเลขหุ่นยนต์ที่สร้างเมสเสจนี้
distance : int	// ระยะทางที่เคลื่อนที่
degree : int	// องศาที่เคลื่อนที่หรือหมุน
direction : int	// ทิศทางการเคลื่อนที่หรือหมุน
event : int	// เหตุการณ์ขณะเมสเสจนี้ถูกสร้างขึ้น
finish : boolean	// สถานะการสิ้นสุดของเมสเสจ
side : int	// ด้านของอาวุธที่ใช้ยิง
type : int	// ประเภทของเมสเสจ
endEvent : int	// ประเภทการจบเหตุการณ์
arm : Arm	// แขนที่ใช้ยิงกระสุน
jet : Jet	// ไอพ่นที่ใช้เคลื่อนที่
leg : Leg	// ขาที่ใช้เคลื่อนที่
Jmsg (long time, int whose, int distance, int degree, int direction, int event, Leg leg)	// Constructor สำหรับสร้างเมสเสจของการเคลื่อนที่ด้วยขา

ตารางที่ ค-22(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class Jmsg

Jmsg	
Jmsg (long time, int whose, int distance, int degree, int direction, int event, Jet jet)	// Constructor สำหรับสร้างเมสเสจของการเคลื่อนที่ด้วยไอพ่น
Jmsg (long time, int whose, int degree, int direction, int event)	// Constructor สำหรับสร้างเมสเสจของการหมุน
Jmsg (long time, int whose, Arm arm, int event, int notUSE1, int notUSE2)	// Constructor สำหรับสร้างเมสเสจของการยิงกระสุน
Jmsg (int Event)	// Constructor สำหรับสร้างเมสเสจที่บอกว่าจบเหตุการณ์นั้นแล้ว
getTime() : long	// คืนค่าเวลาขณะที่เมสเสจนั้นถูกสร้าง
getwhose() : int	// คืนค่าหมายเลขของหุ่นที่สร้างเมสเสจนี้ขึ้นมา
getDistance() : int	// คืนค่าระยะทาง
getDegree() : int	// คืนค่าองศา
getDirection() : int	// คืนค่าทิศทาง
getEvent() : int	// คืนค่าเหตุการณ์ขณะที่เมสเสจนั้นถูกสร้าง
getFinish() : boolean	// คืนค่าว่าเมสเสจนี้ได้ถูกประมวลผลเสร็จแล้วหรือยัง
getArm()	// คืนค่าแขนที่ใช้ยิงกระสุน
getEndEvent() : int	// คืนค่าประเภทการจบเหตุการณ์
getJet() : Jet	// คืนค่าไอพ่นที่ใช้เคลื่อนที่
getLeg() : Leg	// คืนค่าขาที่ใช้เคลื่อนที่
isJetMove() : boolean	// ตรวจสอบว่าเป็นเมสเสจการเคลื่อนที่ด้วยไอพ่นหรือไม่
isMove() : boolean	// ตรวจสอบว่าเป็นเมสเสจการเคลื่อนที่ด้วยขาหรือไม่
isRotate() : boolean	// ตรวจสอบว่าเป็นเมสเสจการหมุนตัวหรือไม่
isShoot() : boolean	// ตรวจสอบว่าเป็นเมสเสจการยิงหรือไม่
setFinish(boolean b)	// กำหนดให้เมสเสจนั้นทำงานเสร็จสิ้น

ตารางที่ ค-22(ต่อ) แสดงแอตทริบิวต์และเมธตอดของ Class Jmsg

Jmsg	
setDegree(int d)	// กำหนดฟิลด์องศาของเมสเสจ
setDirection(int d)	// กำหนดฟิลด์ทิศทางของเมสเสจ
setDistance(int d)	// กำหนดฟิลด์ระยะทางของเมสเสจ
setTime(int t)	// กำหนดเวลาที่เมสเสจถูกสร้างขึ้น
setWhose(int w)	// กำหนดเจ้าของเมสเสจ