**บทที่ 2**

**ทฤษฎีและงานวิจัยที่เกี่ยวข้อง**

**2.1 Java**

ภาษาจาวา เป็นภาษาโปรแกรมเชิงวัตถุที่พัฒนาขึ้นโดย “เจมส์ กอสลิง” และทีมวิศวกรของเขา ซึ่งบริษัทซันไมโครซิสเต็ม ต้องการนำภาษาจาวามาใช้แทนภาษา C++ ชื่อของ “จาวา” มาจากชื่อกาแฟที่ทีมวิศวกรของซันดื่มตอนที่ร่วมกันพัฒนาภาษาจาวาขึ้นมา จาวาถูกคิดค้นและสร้างโดย บริษัท ซันไมโครซิสเต็ม (Sun Microsystems) ซึ่งเป็นบริษัทผู้ขายระบบ Unix ที่มีชื่อว่า Solaris ซึ่งจุดเด่นของภาษาจาวา อยู่ที่ผู้เขียนโปรแกรมสามารถใช้หลักการของ Object-Oriented Programming มาพัฒนาโปรแกรมของตนด้วย จาวาได้พัฒนาขึ้นโดยทีมวิจัยของบริษัท ซันไมโครซิสเต็ม (Sun Microsystems) พัฒนามาจากโครงการที่ต้องการพัฒนาระบบซอฟต์แวร์เพื่อควบคุมเครื่องใช้ไฟฟ้าขนาดเล็กภายในบ้านชื่อเดิมคือภาษา Oak ต่อมาเปลี่ยนชื่อเป็นภาษาจาวา ภาษาจาวาเริ่มเป็นที่นิยมแพร่หลายในปี ค.ศ. 1995 ภาษาจาวาเป็นภาษาที่ไม่ขึ้นกับแพลตฟอร์ม (platform independent) JDK 1.0 ประกาศใช้เมื่อปี1996 JDK เวอร์ชันปัจจุบันคือ Java 2

**วิวัฒนาการของภาษาจาวาจากรุ่นแรกถึงจาวา 1.5**

1. ค.ศ. 1996 ออกครั้งแรกสุด

2. ค.ศ. 1997 ปรับปรุงครั้งใหญ่ โดยเพิ่ม Inner Class

3. 4 ธันวาคม ค.ศ. 1998 รหัส Playground ด้านจาวาแพลตฟอร์มได้รับการเปลี่ยนแปลงครั้งใหญ่ใน API และ JVM (API สำคัญที่เพิ่มมาคือ Java Collections Framework และ Swing; ส่วนใน JVM เพิ่ม JIT Compiler) แต่ตัวภาษาจาวานั้น เปลี่ยนแปลงเพียงเล็กน้อย และทั้งหมดถูกเรียกชื่อใหม่ว่า "จาวา 2" แต่ระบบเลขรุ่นยังไม่เปลี่ยนแปลง

4. 8 พฤษภาคม ค.ศ. 2000 รหัส Kestrel แก้ไขเล็กน้อย

5. 13 กุมภาพันธ์ ค.ศ. 2002 รหัส Merlin เป็นรุ่นที่ถูกใช้งานมากที่สุดในปัจจุบัน (ขณะที่เขียน ค.ศ. 2005)

6. 29 กันยายน ค.ศ. 2004 รหัส Tiger (เดิมทีนับเป็น 1.5) เพิ่มคุณสมบัติใหม่ในภาษาจาวา เช่น Annotations ซึ่งเป็นที่ถกเถียงกันว่านำมาจากภาษาซีชาร์ป ของบริษัทไมโครซอฟท์, Enumerations, Varargs, Enhanced for loop, Autoboxing, และที่สำคัญคือ Generics

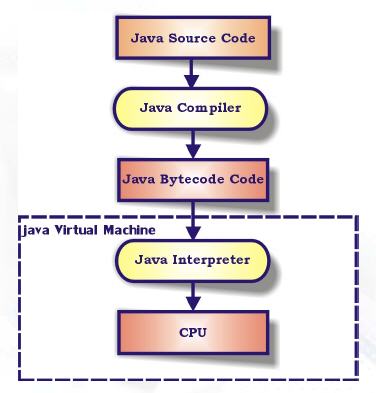
**การพัฒนาการในช่วงเวลาต่าง ๆ**

จาวาถูกพัฒนาตั้งแต่ปี 1991 โดยบริษัท Sun Microsystems ซึ่งเป็น ส่วนหนึ่งของ Green Project Write Once Run Anywhere โดยบริษัท ซันไมโครซิสเต็ม (Sun Microsystems) ได้ทำการวิจัยเพื่อพัฒนาซอฟต์แวร์ที่ใช้ควบคุมอุปกรณ์เล็กทรอนิคส์ขนาดเล็ก ซึ่งได้ผลลัพธ์ที่สำคัญคือ ภาษาโอ๊ค (Oak)

ค.ศ.1993 ภาษาโอ๊คได้ถูกปรับปรุงใหม่เพื่อใช้ในการสร้างเว็บแอพพลิเคชั่น (Web Application) พร้อมกับสร้างเว็บเบราว์เซอร์ (Web Browser) ที่รองรับ ชื่อว่าเว็บรันเนอร์ (Web Runner)

ค.ศ.1995 บริษัทซันได้เปิดตัวภาษาจาวา (Java) (ภาษาโอ๊คเดิม) พร้อมกับเว็บเบราว์เซอร์ ที่รองรับภาษานี้ ชื่อว่า ฮอตจาวา (HotJava) (WebRunner เดิม) ได้รับการสนับสนุนจากบริษัทใหญ่ทั้งเน็ตสเคบ (Netscape), ไมโครซอฟต์ (Microsoft), และ ไอบีเอ็ม (IBM) บริษัทซัน ได้เริ่มแจกจ่าย Java development Kit (JDK) ซึ่งเป็นชุดพัฒนาโปรแกรมภาษาจาวาในอินเทอร์เน็ต

ความหมายของภาษาจาวา (Java Language) คือ ภาษาคอมพิวเตอร์ที่ถูกพัฒนาขึ้นโดยบริษัท ซันไมโครซิสเต็มส์ เป็นภาษาสำหรับเขียนโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (OOP : Object-Oriented Programming) โปรแกรมที่เขียนขึ้นถูกสร้างภายในคลาส ดังนั้นคลาสคือที่เก็บเมทอด (Method) หรือพฤติกรรม (Behavior) ซึ่งมีสถานะ (State) และรูปพรรณ (Identity) ประจำพฤติกรรม (Behavior)



**ภาพที่ 2.1** ลำดับการทำงานของ Java

**2.2 Java Development Kit หรือ JDK**

Java Development Kit หรือ JDK คือชุดของเครื่องมือ (tools) ที่ใช้ในการพัฒนา

โปรแกรม JAVA ของบริษัทซันไมโครซิสเต็มส์ ซึ่งใครก็ตามที่ต้องการจะพัฒนาโปรแกรมโดยใช้ภาษา Java อย่างเช่น Java compiler, Java debugger, Java doc และ Java interpreter หรือ Java VM จะต้อง ลง JDK นี้ ไม่งั้นจะไม่สามารถ compile และ run java ได้ เวอร์ชันปัจจุบันของ JDK คือเวอร์ชั่น 7 ประกอบไปด้วยโปรแกรมต่างๆ อาทิเช่น

โปรแกรมคอมไพเลอร์ (javac.exe) ,โปรแกรมอินเตอร์พรีตเตอร์ (java.exe) ,โปรแกรมดีบักเกอร์ แต่จะไม่มีโปรแกรมอีดิเตอร์

ชุดพัฒนาโปรแกรม JDK ประกอบด้วย 3 รุ่นย่อยดังนี้

1. Java SE (Standard Edition)

ย่อมาจาก Java Standard Edition เป็นรูปแบบการเขียนโปรแกรมสำหรับการทำงานบนคอมพิวเตอร์ทั่วไป ถือว่าเป็นรูปแบบแรกของภาษา Java

2. Java ME (Micro Edition)

ย่อมาจาก Java Micro Edition เป็นรูปแบบการเขียนโปรแกรมกับอุปกรณ์ขนาดเล็ก(คือมีหน่วยความจำน้อย) เช่น โทรศัพท์มือถือ, เซ็ตท็อปบ็อกซ์ เป็นต้น

3. Java EE (Enterprise Edition)

ย่อมาจาก Java Enterprise Edition เป็นรูปแบบการเขียนโปรแกรมกับระบบงานขนาดใหญ่

**2.3 Java Runtime Environment หรือ JRE**

เป็นส่วนที่ใช้ในการรันโปรแกรมภาษาจาวาเป็นส่วนที่ใช้ในการรันโปรแกรม โดยจะทำงาน 3 ขั้นตอน ดังนี้

- โหลดไบต์โค้ดโดยใช้ Class loader คือการโหลดคลาสทุกคลาสที่เกี่ยวข้องในการรันโปรแกรม

- ตรวจสอบไบต์โค้ดโดยใช้ Bytecode Verifier คือการตรวจสอบว่าโปรแกรมจะต้องไม่มีคำสั่งที่ทำให้เกิดความผิดพลาดกับระบบ เช่น การแปลงข้อมูลที่ผิดพลาด หรือมีการแทรกแซงเข้าสู่ระบบภายในเป็นต้น

- รันไบต์โค้ด โดยใช้ Runtime Interpreter

****

**ภาพที่ 2.**3 ขั้นตอนการทำงานของ JRE

**2.4 SQL**

Structure Query Language หรือภาษาในการสอบถามข้อมูล เป็นภาษาใช้ปฏิบัติกับฐานข้อมูลแบบสัมพันธ์(Relational database) โดยเฉพาะ ภาษา SQL เริ่มพัฒนาครั้งแรกโดย Almaden research center ของบริษัท IBM โดยมีชื่อเริ่มแรกว่า “ซีเควล” (Seaquel) ต่อมาได้เปลี่ยนชื่อเป็น “เอสคิวแอล” (SQL) หลังจากนั้นภาษา SQL ได้ถูกนำมาพัฒนาโดยผู้ผลิตซอฟต์แวร์ด้านระบบจัดการฐานข้อมูลเชิงสัมพันธ์จนเป็นที่นิยมกันอย่างแพร่หลายในปัจจุบัน

**ประเภทของคำสั่งของภาษา SQL**

1. ภาษาสำหรับนิยามข้อมูล (Data Definition Language: DDL) ประกอบด้วยคำสั่งที่ใช้ในการกำหนดโครงสร้างข้อมูลว่ามีคอลัมน์อะไร แต่ละคอลัมน์เก็บข้อมูลประเภทใด รวมถึงการเพิ่มคอลัมน์ การกำหนดดัชนี การกำหนดวิวของผู้ใช้ เป็นต้น

2. ภาษาสำหรับการจัดการข้อมูล (Data Manipulation Language: DML) ประกอบด้วย คำสั่งที่ใช้ในการเรียกใช้ข้อมูล การเปลี่ยนแปลงข้อมูล การเพิ่มหรือลบข้อมูล เป็นต้น

3. ภาษาที่ใช้ในการควบคุมข้อมูล (Data Control Language: DCL) ประกอบด้วยคำสั่งที่ใช้ในการควบคุม หรือป้องกันการเกิดเหตุการณ์ที่ผู้ใช้หลายคนเรียกใช้ข้อมูลพร้อมกัน โดยที่ข้อมูลนั้น ๆ อยู่ในระหว่างการปรับปรุงแก้ไข ซึ่งเป็นเวลาเดียวกับที่ผู้ใช้อีกคนหนึ่งก็เรียกใช้ข้อมูลนี้ ทำให้ข้อมูลที่ผู้ใช้คนที่สองได้ไปเป็นค่าเก่าที่ไม่ถูกต้อง ทั้งนี้เพราะผู้ใช้คนแรกยังปรับปรุงแก้ไขข้อมูลไม่เสร็จ นอกจากนี้ ยังประกอบด้วยคำสั่งที่เกี่ยวข้องกับการควบคุมความปลอดภัยของข้อมูลด้วยการ ให้สิทธิ์ผู้ใช้ที่แตกต่างกัน

**2.5 JDBC**

JDBC เป็น JAVA API สำหรับการประมวลผลคำสั่ง SQL ซึ่งเป็น Class และ Interface ที่เป็นภาษา JAVA, JDBC ได้จัดเตรียม API มาตรฐานสำหรับเป็นเครื่องมือในการพัฒนา Database Application ในการเขียน Code โปรแกรมภาษา Java ซึ่ง JDBC สามารถติดต่อกับ Database ได้เกือบทุกประเภท เช่น Sysbase ,Oracle, MS SQL, Informix, Access, MySQL ที่สำคัญการพัฒนา Application ด้าน Database โดยใช้ Java สามารถนำไป Run ได้หลาย Platform ซึ่ง JDBC จะเข้ามาช่วยจัดการในด้านนี้ จะทำให้ทำการพัฒนา Application ได้ใช้คุณสมบัติของภาษาจาวาได้อย่างเต็มที่ JDBC ถูกประยุกต์ใช้งานในหลายด้าน เช่น ใน webpage ที่มี Applet ในการ Run Application แบบ Remote Database หรือมีการใช้ JDBC ในการติดต่อกับฐานข้อมูลซึ่ง JDBC เป็นคลาสคลาสหนึ่ง ฉะนั้นจึงสามารถใช้งานได้หลาย Platform เช่น Windows ,Linux, Unix, Solaris, MAC

**ข้อดีของ JDBC**

1. ใช้งานง่าย การตั้งค่าไม่ยุ่งยาก

2. สามารถทำงานร่วมกับภาษาจาวาได้เป็นอย่างดี เพราะ Driver ที่เขียนขึ้นนั้นส่วนมากจะพัฒนาจากภาษาจาวาและมีการ compile เป็น class file ทำให้ใช้งานง่าย

3. สามารถติดต่อฐานข้อมูลได้เกือบทุกประเภท ซึ่งส่วนใหญ่จะเป็นฐานข้อมูลแบบเชิงสัมพันธ์

4. ไฟล์ Driver มีขนาดเล็กกะทัดรัด

5. สามารถใช้งานได้ทั้ง สถาปัตยกรรม แบบ Two-Tier และ Three-Tier

6. สามารถใช้งานได้หลายระบบปฏิบัติการ

7. เป็น low-level Interface และสามารถใช้ High-level interface ด้วยถ้าต้องการ

8. การใช้งานจะเป็นเชิง Object-Oriented

9. Driver แต่ละตัวมีมาตรฐานใกล้เคียงกัน

**การทำงานที่สำคัญของ JDBC**

1. ทำการติดต่อกับ Data Base

2. สั่งคำสั่ง SQL ไปยัง Data Base

3. การับผลลัพธ์จากการใช้คำสั่ง SQL แล้วทำการประมวลผล

**ลักษณะที่สำคัญของ JDBC**

JDBC เป็น low-level Interface ซึ่งสามารถเรียกใช้คำสั่ง SQL ได้โดยตรงซึ่งสามารถใช้งานได้อย่างมีประสิทธิภาพ มีขนาดเล็กและง่ายต่อการใช้กว่าตัวติดต่อ Data Base ชนิดอื่น ๆ แต่ก็ได้มีออกแบบสำหรับใช้งานแบบ High-Level Interface ด้วย ซึ่ง High Level Interface คือ User Friendly ซึ่ง High-level API ของ JDBC ที่ถูกพัฒนาซึ่งมีดังนี้

1. JDBC ต้องการคำสั่ง SQL ที่จะทำการส่งเป็น String ไปยัง Method ของภาษา Java ทำให้ Programmer สามารถใช้ตัวแปรในการรับส่งคำสั่ง SQL ได้

2. มีการแปลง relation Database Table ไปเป็น Java Class ซึ่งเรียกว่า Object/Relational ซึ่งแต่ละแถวของ Table จะถูกแปลงไปเป็น Instant ของ Class และแต่ละ Column จะเป็นเหมือน attribute ของคลาสซึ่ง Programmer สามารถดำเนินการต่าง ๆ คล้ายด้วยว่า Table เป็นคลาสคลาสหนึ่งในจาวา

**ประเภทของ JDBC Driver**

1. JDBC-ODBC Bridge plus ODBC Driver เป็น Driver ของ JDBC ที่ใช้ในการติดต่อการทำงานผ่าน ODBC ซึ่งจะมีการใช้ ODBC ในการติดต่อกับฐานข้อมูลก่อนไม่ว่าจะเป็นในเครื่องเดียวกันหรือติดต่อผ่าน Server แล้วจึงใช้ JDBC ไปติดต่อกับ ODBC อีกครั้งหนึ่ง

2. Native-API partly-Java Driver สำหรับ Driver ประเภทนี้จะเปลี่ยน JDBC calls ไปเรียก Client API ของ Oracle , Sysbase หรือ DBMS ตัวอื่น ๆ ซึ่งจะคล้ายสะพานเชื่อมต่อไปยัง ฐานข้อมูลอีก

3. JDBC-Net pure Java Driver: Driver ประเภทนี้จะเปลี่ยน JDBC calls ไปยัง DBMS - independent net protocal ซึ่งหลังจากการเปลี่ยนไปเป็น DBMS Protocal โดย Server ซึ่ง Server ตัวนี้สามารถติดต่อเข้ากับ Java Client ไปยัง Database หลายประเภทได้ซึ่งจะสามารถใช้งานผ่านทาง Internet และ Intranet

4. Native-protocal pure Java Driver : Driver ประเภทนี้จะเปลี่ยน JDBC call ไปเป็น network protocal ซึ่งจะถูกใช้โดย DBMS ซึ่งจะมีการใช้งานโดยมีการเรียกใช้งานโดยตรงจาก client ไปยัง DBMS Server