จากจาวา 1.0 ถึงจาวา 7.0

From Java 1.0 to Java 7.0

นันทวัน นาคอร่าม
สาขาเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธนบุรี
Nantawan.nk@gmail.com

1. บทนำ

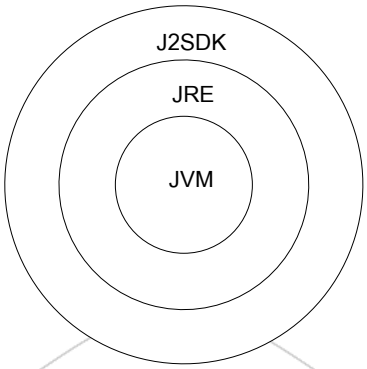
ภาษาจาวาเป็นภาษาโปรแกรมที่มีคุณลักษณะเด่นในเรื่องของการเป็นอิสระจากแพลตฟอร์ม (Platform) หมายถึงการที่ภาษาโปรแกรมนั้นเป็นอิสระจากฮาร์ดแวร์และซอฟต์แวร์ที่จะทำงานร่วมกับภาษาจาวา ทำให้ผู้ใช้สามารถนำโปรแกรมที่ผ่านการคอมไพล์แล้วไปทำงานบนแพลตฟอร์มใดๆ ที่มีการติดตั้งชุดคำสั่งภาษาจาวาไว้เรียบร้อยแล้ว มีความสะดวกต่อการนำโปรแกรมไปใช้งานและรองรับหลักการโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ภาษาจาวาถูกคิดค้นขึ้นมาโดยทีมวิศวกรของบริษัท Sun Microsystems ได้วิจัยและพัฒนาอุปกรณ์อิเล็กทรอนิกส์ที่ควบคุมโดยรีโมทคอนโทรล ซึ่งหัวหน้าทีมคือ James Gosling ทำหน้าที่ในการพัฒนาภาษาโปรแกรมที่จะติดตั้งในรีโมทคอนโทรลเพื่อควบคุมให้อุปกรณ์อิเล็กทรอนิกส์สามารถทำงานได้ โดยมีจุดมุ่งหมายว่า ภาษาที่ใช้จะต้องเป็นอิสระหรือไม่ยึดติดกับอุปกรณ์ฮาร์ดแวร์ชนิดใดชนิดหนึ่ง ซึ่งภาษาที่ได้รับการพัฒนา ก็คือ ภาษาจาวา และถูกนำมาใช้ในงานบนอินเตอร์เน็ตในปลายปี ค.ศ .1994

2. Java Platform

จาวาแพลตฟอร์ม คือ ชุดของโปรแกรมภาษาจาวาที่ทำงานแตกต่างกันไปตามฮาร์ดแวร์และระบบปฏิบัติการที่ใช้งาน ซึ่งแบ่งออกได้เป็น 3 ประเภท คือ

2.1 J2SE (Java 2, Standard Edition)

คือ ชุดคำสั่งภาษาจาวาที่เป็นชุดพื้นฐานและเป็นชุดหลักที่ขยายการทำงานไปเป็นชุดอื่นๆ โดย J2SE ใช้สำหรับทำงานบนคอมพิวเตอร์ที่เป็น Stand alone แบ่งส่วนประกอบภายใน J2SE ออกเป็น 2 ส่วนย่อย คือ



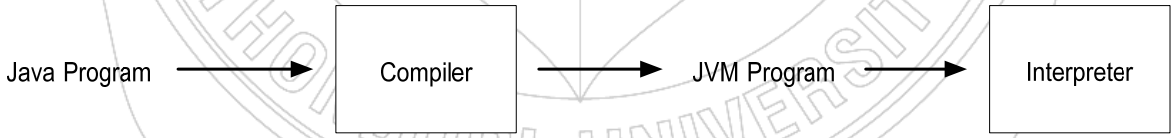
รูปที่ 1 : ส่วนประกอบของ J2SE

- **J2SDK** (Java 2, Software Development Kit)

ในยุคแรกจะถูกเรียกว่า Java Development Kit (JDK) ประกอบด้วย 2 ส่วน คือ 1) คอมไพเลอร์ ทำหน้าที่ในการแปลโปรแกรมภาษาจาวาไปเป็น Bytecode ที่เก็บอยู่ในไฟล์ที่มีนามสกุล .class และ 2) ดีบักเกอร์ ทำหน้าที่ช่วยค้นหาและแก้ไขข้อผิดพลาดที่เกิดขึ้นของโปรแกรม

- **J2RE** (Java 2, Runtime Environment)

ประกอบด้วยไลบรารีต่างๆ Java Virtual Machine : JVM และคอมโพเนนต์อื่นๆ ที่จำเป็นสำหรับการรันโปรแกรม ซึ่ง JVM จะทำหน้าที่ในการจำลองคอมพิวเตอร์เสมือน (Virtual Machine) บนคอมพิวเตอร์ที่ทำงานจริง โดยคอมพิวเตอร์เสมือนจะทำหน้าที่ในการแปล Bytecode ไปเป็นภาษาเครื่องที่เหมาะสมกับแพลตฟอร์มของคอมพิวเตอร์ที่ทำงานจริง จึงทำให้ไฟล์ที่มีนามสกุล .class สามารถทำงานบนแพลตฟอร์มชนิดใดๆ ได้ เนื่องจากการแปล Bytecode เป็นภาษาเครื่องถูกทำงานโดยคอมพิวเตอร์เสมือน ไม่เกี่ยวข้องกับแพลตฟอร์มจริงของเครื่องแต่อย่างใด ส่งผลให้เขียนโปรแกรมเพียงแค่ครั้งเดียวแต่สามารถนำไปทำงานได้บนทุกแพลตฟอร์ม



รูปที่ 2 : กระบวนการทำงานของโปรแกรมภาษาจาวา

2.2 J2EE (Java 2, Enterprise Edition)

สนับสนุนการทำงานที่โปรแกรมมีการประมวลผลทางฝั่งเซิร์ฟเวอร์แล้วส่งผลลัพธ์ไปแสดงที่ฝั่งไคลเอ็นต์ เรียกว่า Server-side Application เช่น Java Server Pages (JSP), Java Servlets, Enterprise Java Beans (EJB) และ Web Services เป็นต้น

2.3 J2ME (Java 2, Micro Edition)

เป็นเทคโนโลยีที่ทำงานกับอุปกรณ์ขนาดเล็ก เช่น โทรศัพท์มือถือ, PDAs, Smart Phone เป็นต้น

3. รูปแบบการเขียนโปรแกรมภาษาจาวา

การเขียนโปรแกรมภาษาจาวาแบ่งออกเป็น 2 รูปแบบ โดยแยกตามวัตถุประสงค์ในการใช้งาน คือ โปรแกรมที่ทำงานอย่างอิสระบนเครื่องคอมพิวเตอร์ประเภทต่างๆ และโปรแกรมที่ทำงานบนระบบเครือข่าย

3.1 Java Application

คือ โปรแกรมที่ทำงานแบบ Standalone ไม่ต้องการเว็บบราวเซอร์สำหรับการประมวลผล และสามารถรันได้บนเครื่องทุกเครื่องที่มี JRE ติดตั้งอยู่ การรัน Java application ทำได้โดยใช้ Java Interpreter คือ java.exe โดยอาจติดต่อกับผู้ใช้แบบ Console Applications หรืออาจติดต่อกับผู้ใช้แบบกราฟิก (Graphical User Interface)

3.2 Java Servlet/Java Applet

คือ โปรแกรมที่ทำงานบนเครือข่าย โดย Java Servlet เป็นโปรแกรมที่ทำงานบนเซิร์ฟเวอร์ ส่วน Java Applet ทำงานบนเว็บบราวเซอร์ ซึ่งจะต้องสร้างหน้าเว็บเพจขึ้นมาแล้วจึงทำการเรียกใช้งาน Java Applet โดยใช้ Default JVM ที่มีอยู่บนเว็บบราวเซอร์ในการรันโปรแกรม

4. คุณลักษณะเด่นของภาษาจาวา

1) ความง่าย (Simple) จากการที่ภาษาจาวามีพื้นฐานมาจากภาษาซี จึงตัดปัญหาที่เกิดจากการทำงานกับภาษาซีทิ้งไป เช่น เรื่องของพอยน์เตอร์ สตรัคเจอร์ การจองหน่วยความจำ (Memory Allocation) เป็นต้น ทำให้ง่ายต่อการใช้งาน

2) ความคงทน (Robust) เนื่องด้วยจาวามีตัวช่วยจัดการหน่วยความจำให้อัตโนมัติ (Garbage Collection) จึงช่วยลดข้อผิดพลาดที่อาจเกิดขึ้นจากการที่ผู้ใช้เขียนโปรแกรมทำการจองพื้นที่หน่วยความจำและไม่ได้คืนพื้นที่หน่วยความจำให้กับระบบจนทำให้เกิดปัญหาในการทำงานขึ้น หรือการที่คอมไพเลอร์ของภาษาจาวาจะช่วยตรวจสอบการทำงานทุกครั้งว่ามีข้อผิดพลาดใดที่อาจจะเกิดขึ้นในระหว่างการทำงานหรือไม่ ถ้ามีคอมไพเลอร์จะแจ้งให้ทราบ

3) ความปลอดภัย (Secure) จาวาถูกออกแบบมาให้มีระบบรักษาความปลอดภัยเป็นอย่างดี เช่น การทำงานของ Sandbox ที่ทำหน้าที่สิ่งแปลกปลอมในระหว่างที่มีการรันโปรแกรม หรือการที่จาวาตัดเรื่องของพอยน์เตอร์ทิ้งไป ทำให้ตัดปัญหาในเรื่องของการจองพื้นที่ในหน่วยความจำเพื่อทำงานกับโปรแกรมที่ไม่เกี่ยวข้องในระหว่างรันโปรแกรมได้

4) ความเป็นอิสระจากแพลตฟอร์ม (Platform Independent) ตัวแปลภาษาจาวาถูกออกแบบมาให้เป็นกลาง ไม่ยึดติดกับอุปกรณ์ที่เป็นฮาร์ดแวร์และซอฟต์แวร์ใดๆ จึงทำให้โปรแกรมที่เป็น Bytecode สามารถทำงานได้บนทุกแพลตฟอร์ม

5) Multithread จาวาสนับสนุนการทำงานหลายอย่างในเวลาเดียวกัน โดยใน 1 โปรแกรมภาษาจาวาสามารถมี Thread ได้หลายตัว ซึ่งจะช่วยให้โปรแกรมสามารถจัดการงานต่างๆ ไปพร้อมกันได้ในเวลาเดียวกัน

6) สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming)

5. วิวัฒนาการของภาษาจาวา

ภาษาจาวาถูกพัฒนาขึ้นตั้งแต่ปี 1991 โดยทีมวิศวกรของบริษัท Sun Microsystems โดยในปี 1995 ทางบริษัทได้ปล่อยชุดคำสั่งภาษาจาวา (Java Development Kit) หรือ JDK 1.0 แจกจ่ายให้ดาวน์โหลดฟรีบนอินเทอร์เน็ต ตั้งแต่นั้นมาได้มีการพัฒนาชุดคำสั่ง JDK มาอย่างต่อเนื่อง และในปี 2004 ทางบริษัทได้เปิดตัว JDK 1.5 หรือที่เรียกว่าเป็นเวอร์ชันที่ 2 ออกมาให้ใช้งานฟรี ต่อมาในปี 2008 ทางบริษัทได้เปิดตัว JDK 1.6 หรือ JDK 6.0 เป็นเวอร์ชั่นที่ปรับปรุง 5.0 ให้ดีขึ้นโดยไม่ได้เปลี่ยนsyntax ของภาษาจาวาและล่าสุดในปี 2011ทางบริษัท Oracleได้เปิดตัว Java 7 โดยสามารถสรุปรายละเอียดในแต่ละเวอร์ชันดังนี้

5.1 Java SE 5.0

ตัวอย่างคุณสมบัติใหม่ที่เพิ่มขึ้นในเวอร์ชัน 5.0 เช่น

- Generics คือ คุณสมบัติที่ช่วยตรวจสอบการทำงานกับชนิดข้อมูล โดยเพิ่มความสามารถให้คอมไพเลอร์สามารถตรวจสอบข้อผิดพลาดที่เกิดขึ้นจากการทำงานกับชนิดข้อมูลที่ผิดประเภท ซึ่งเดิมคอมไพเลอร์ไม่สามารถตรวจสอบข้อผิดพลาดเหล่านี้ได้ จึงแสดง Exception ขึ้น แต่ถ้ามีการใช้ Generic จะช่วยให้คอมไพเลอร์ตรวจสอบข้อผิดพลาดได้ตั้งแต่ช่วงคอมไพล์โปรแกรม

- Enhanced for Loop ช่วยให้สามารถเขียนคำสั่งทำซ้ำได้สั้นลง เพิ่มความสะดวกสบายในการเขียนโปรแกรมได้มากขึ้น

- Autoboxing/Unboxing คือ การแปลงจากค่าข้อมูลจากชนิดข้อมูลพื้นฐานเป็น Wrapper class ของชนิดข้อมูลนั้นๆ โดยอัตโนมัติ

5.2 Java SE 6.0

สรุปคุณสมบัติใหม่ที่เพิ่มขึ้นในเวอร์ชัน 6.0 ช่วยให้การพัฒนาซอฟต์แวร์ทำได้ง่าย เร็วและประหยัดยิ่งขึ้น นำเสนอฟังก์ชั่นเพิ่มเติมสำหรับเว็บเซอร์วิส และสนับสนุนภาษาแบบไดนามิค เช่น PHP, Python, Ruby เป็นต้น

- Scripting Language คือ การเขียนโค๊ดคำสั่งเพิ่มเข้าไป เช่น Java script เป็นต้น จะทำให้พัฒนาระบบงานได้รวดเร็วกว่าเดิม

- JavaKernel คือ การลดขนาดของไฟล์ดาวน์โหลดจาก 14.5 MB เหลือ 4.5 MB ทำให้การทำงานคล่องตัวมากขึ้น

- สนับสนุน metadata ของ web service

- JavaQuickStarter ทำให้ applet และแอพพลิเคชันรันตอนเริ่มต้นได้เร็วขึ้น

- สถาปัตยกรรมปลั๊กอินแบบใหม่ สามารถรัน applet แบบแยกโพรเซสแล้วใช้การวาดลงไปบน canvas ของบราวเซอร์แทน

- ช่วยให้การพัฒนาทำได้ง่ายขึ้นด้วยการอัปเดตหน้าจอใช้งานของทูลสำหรับ Java Virtual Machine (JVM (TM)) ตลอดจน Java Platform Debugger Architecture (JPDA)

- ปรับปรุง XML APIs สำหรับ parse, bind และติดต่อ web service

- JTabbedPane สามารถกำหนด tab ให้เป็นแบบ JComponent ได้ จากที่เดิมตั้งได้แต่เพียงข้อความ

5.3 Java SE 7.0

คุณสมบัติใหม่ที่เพิ่มขึ้นในเวอร์ชัน 7.0 ช่วยให้การพัฒนาโปรแกรมภาษาจาวาทำได้ง่ายและรวดเร็วขึ้น ตัดปัญหาที่น่าจะเกิดในระหว่างการทำงานทิ้งไป ตรงตามความต้องการของผู้เขียนโปรแกรมมากขึ้น ตัวอย่างเช่น

- รองรับการ switch ด้วย string เพิ่มเติมขึ้นจากเดิมที่คำสั่ง switch ทำงานได้กับเฉพาะข้อมูลที่เป็นเลขจำนวนเต็มหรือชนิดข้อมูลที่เป็นแบบ enum เท่านั้น

- รองรับข้อมูลที่เป็นเลขจำนวนเต็มฐานสอง ทำให้ผู้เขียนโปรแกรมสามารถประกาศตัวแปรเพื่อเก็บชนิดข้อมูลที่เป็นเลขฐานสองได้โดยอัตโนมัติ เช่น

byte myBits = 0b0110_1101;

- รองรับการทำงานกับข้อมูลตัวเลขที่มีเครื่องหมาย underscore (_) ซึ่งคอมไพเลอร์จะทำการตัดเครื่องหมาย underscore ทิ้งไปและสนใจเฉพาะตัวเลขเท่านั้น เช่น ตัวแปร myBits ที่เก็บค่าเลขฐานสอง 0b01101101111000111 แต่เป็นการยากต่อการอ่านค่านี้ จึงนำเครื่องหมาย underscore มาคั่นเพื่อให้ง่ายต่อการอ่านซึ่งคอมไพเลอร์จะไม่สนใจเครื่องหมาย underscore และทำการตัดทิ้งให้โดยอัตโนมัติ ซึ่งจะช่วยลดข้อผิดพลาดให้กับผู้เขียนโปรแกรมเมื่อจำเป็นต้องทำงานกับข้อมูลตัวเลขมากๆ

byte myBits = 0b01101101111000111;

byte myBits = 0b0110_1101_1100_0111;

- ความสะดวกในการใช้เครื่องหมาย diamond (<>) สำหรับการสร้างออบเจ็คที่เป็น generic จะทำได้ง่ายขึ้น

เดิมการสร้างออบเจ็คที่เป็น generic จะต้องระบุคำสั่งแบบเต็มทั้งด้านซ้ายและด้านขวาของเครื่องหมาย assignment (=) ในการสร้างออบเจ็คที่มีชนิดเป็น generic เช่น

```
Map<String, List<String>> myMap =
new HashMap<String, List<String>>();
```

สำหรับ java 7.0 ระบุการสร้างออบเจ็คที่มีชนิดเป็น generic เฉพาะด้านซ้ายของเครื่องหมาย assignment เท่านั้น และด้านขวาระบุเป็นพารามิเตอร์ว่างแทน เช่น

```
Map<String, List<String>> myMap =
new HashMap<>();
```

- ปรับแก้รูปแบบของคำสั่ง try-catch เป็นโครงสร้างแบบ try-with-resources statement สามารถทำงานกับทรัพยากรอื่นๆ ได้เพิ่มขึ้น ซึ่งเดิมการใช้คำสั่ง try-catch ทำงานกับทรัพยากรที่มากกว่า 1 อย่างจะต้องใช้คำสั่ง try-catch ซ้อนกันหลายชั้นเพื่อดักจับข้อผิดพลาดที่น่าจะเกิดขึ้นและตัดการเชื่อมต่อกับทรัพยากรเหล่านั้นเมื่อไม่มีการใช้งานต่อไป แต่โครงสร้างแบบ try-with-resources statement จะช่วยจัดการปัญหาต่างๆ ได้ และจะปิดการเชื่อมต่อกับทรัพยากรเหล่านั้นให้โดยอัตโนมัติ ลดการดักจับข้อผิดพลาดที่จะเกิดขึ้นเมื่อมีการทำงานกับทรัพยากรต่างๆ ตัวอย่างเช่น การทำงานกับไฟล์ เดิมเมื่อต้องการทำงานกับไฟล์จะต้องประกอบไปด้วย 3 ขั้นตอน คือ 1) การเปิดไฟล์ 2) การทำงานกับข้อมูลในไฟล์ และ 3) การปิดไฟล์ ดังรูปที่ 3

```
FileInputStream fIn = null;
try {
        fIn = new FileInputStream("somefilename");
        // Access the file ...
} catch(IOException e) {
        // ...
} finally {
        // Close file.
        try {
                if(fIn != null) fIn.close();
        } catch(IOException e) {
                // ...
        }
}
```

รูปที่ 3 : คำสั่ง try-catch ในเวอร์ชันก่อนหน้า

สำหรับขั้นตอนการทำงานของโครงสร้างแบบ try-with-resources จะเหลือเพียง 2 ขั้นตอน คือ 1) ประกาศตัวแปรและกำหนดทรัพยากรที่ต้องการทำงานด้วย และ 2) ทำงานกับทรัพยากร และปิดการเชื่อมต่อให้โดยอัตโนมัติ จากคำสั่งข้างต้นสามารถแก้ไขใหม่ดังรูปที่ 4

```
try(FileInputStream fIn = new FileInputStream("somefilename")) {
        // Access the file ...
} catch(IOException e) {
        // ...
}
```

รูปที่ 4 : โครงสร้างแบบ try-with-resources

หรือตัวอย่างการใช้คำสั่ง sql เพื่อทำงานกับฐานข้อมูล ซึ่งโครงสร้างแบบ try-with-resources จะปิดการเชื่อมต่อกับ sql ให้โดยอัตโนมัติ ดังรูปที่ 5

```
public static void viewTable(Connection con) throws SQLException {
    String query = "select COF_NAME, PRICE from COFFEES";
    try (Statement stmt = con.createStatement()) {
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            String coffeeName = rs.getString("COF_NAME");
            float price = rs.getFloat("PRICE");
            System.out.println(coffeeName + ": " + price);
        }
    }
}
```

รูปที่ 5 : การใช้โครงสร้างแบบ try-with-resources ทำงานกับคำสั่ง SQL

- เพิ่มฟีเจอร์ด้าน 3D ผ่านเอนจินกราฟิกตัวใหม่จากโครงการ Prism
- ปรับรูปแบบของ API ใหม่สำหรับการเรียกภาษาในกลุ่มภาษา Dynamic ให้ทำงานเร็วขึ้น
- โฟกัสที่ JavaFX โดยใช้ JavaFX เป็นตัวเชื่อมการทำงานระหว่างจาวา จาวาสคริปต์ และ HTML5

6. เอกสารอ้างอิง

[1] อรพิน ประวัติบริสุทธิ์, "คู่มือเขียนโปรแกรมด้วยภาษา Java" *บริษัทโปรวิชั่น จำกัด*, 2537.

[2] รศ. ธีรวัฒน์ ประกอบผล, "คู่มือการเขียนโปรแกรมภาษา Java", *สำนักพิมพ์ซิมพลิฟาย*, 2553.

[3] ยุทธนา ลีลาศวัฒนกุล, "เริ่มต้นการเขียนโปรแกรมด้วยภาษา Java", *สำนักพิมพ์ดวงกมลสมัย*, 2548.

[4] พนิดา พานิชกุล, "การเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นด้วยภาษา Java", *เคทีพี คอม แอนด์ คอนซัลท์*, 2548.

[5] วีระศักดิ์ ซึงถาวร, "Java Programming Vol. 1 (Javase 5.0)", *บริษัทจำกัดมหาชน ซีเอ็ดยูเคชัน*, 2549.

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

[7] http://www.itexchanges.com.

[8] http://www.ryt9.com.

[9] http://deans4j.wordpress.com.

[10] http://www.thaidev.org

[11] http://www.oraclejavamagazine-digital.com : Premiere issue 2011.

[12] http://oracle.in.th