การเขียนโปรแกรมคอมพิวเตอร์ 2

Computer Programming II

จากภาษาซีสู่ภาษาจาวา

From C to Java Language

ภิญโญ แท้ประสาทสิทธิ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

(taeprasartsit_p at silpakorn dot edu, pinyotae at gmail dot com)

Web site: ???

Facebook group: ComputerProgramming@CPSU

ปฏิบัติการสัปดาห์ที่ 1

ความแตกต่างของโค้ดภาษาซีกับจาวา

- เนื่องจากภาษาซีเป็นภาษาแบบลำดับขั้นตอน (procedural) แต่จาวาเป็นภาษาเชิงวัตถุ (object oriented)
  - ดังนั้นความแตกต่างในการเขียนภาษาทั้งสองจึงถือว่าสูงมาก
- อย่างไรก็ตาม หากเรายังไม่ได้ใช้ความสามารถทางด้านการเขียนโปรแกรมเชิงวัตถุในภาษาจาวาอย่างเต็มที่ ความแตกต่างถือว่าไม่มาก
- ดังนั้นเราจะเริ่มศึกษาจากจุดนี้ก่อน เพื่อสร้างความคุ้นเคยกับภาษาจาวา

โปรแกรมพิมพ์ข้อความ

- ตอนเขียนภาษาซี โปรแกรมพิมพ์ข้อความของเรามีหน้าตาง่าย ๆ ดังนี้

```
#include <stdio.h>

void main() {
    printf("Silpakorn");
}
```

- สำหรับจาวา โค้ดที่ให้ผลเหมือนกัน หน้าตาเป็นดังนี้

```
public class Print {
    public static void main(String[] args) {
        System.out.print("Silpakorn");
    }
}
```

การพิมพ์ผลลัพธ์ผ่าน System.out

- System เป็น*คลาส*พื้นฐานในจาวา
  - มี*วัตถุ*สำหรับแสดงผลลัพธ์ที่ชื่อว่า out
- ใน out นั้นก็มี*เมธอด*จำนวนมากที่อำนวยความสะดวกในการแสดงผล
- อันที่พื้นฐานที่สุดมีสองอันคือ print และ println
- ในตัวอย่างที่ผ่านมาเราเลือกใช้ print และเขียนโค้ดว่า

```
System.out.print("Silpakorn");
```

- แต่โดยปรกติแล้ว เราอยากให้แสดงผลเสร็จแล้วขึ้นบรรทัดใหม่ตามมา ถ้าเป็นเช่นนี้แล้ว เราสามารถใช้ println และเขียนโค้ดว่า

```
System.out.println("Silpakorn");
```

ความแตกต่างระหว่าง print และ println

- โค้ดทั้งสองให้ผลลัพธ์ที่ “เกือบ” จะเหมือนกัน
- ในกรณีที่เราอยากจะให้ผลลัพธ์จากการใช้ print มีการขึ้นบรรทัดใหม่ตามมา เราสามารถใช้ \n ได้ในลักษณะเดียวกับภาษาซี เช่น

```
System.out.print("Silpakorn\n");
```

- หากเราต้องการให้จบการพิมพ์ด้วยการขึ้นบรรทัดใหม่ เราก็ควรเลือกใช้เมธอด println ตั้งแต่แรก (ทำให้มีแนวโน้มว่า println เป็นที่นิยมกว่า)
- จุดแตกต่างอีกอย่างหนึ่งก็คือ เราสามารถใช้ println “เปล่า ๆ” ได้ ดังนี้

```
System.out.println();
```

- โค้ดข้างบนจะทำให้เกิดการเลื่อน*เคอร์เซอร์*ขึ้นบรรทัดใหม่
- ส่วน print นั้นไม่สามารถใช้แบบเปล่า ๆ ได้ (เพราะมันไร้ความหมาย)

```
System.out.print();
```

การรับข้อมูลเข้าจากผู้ใช้

- เราสามารถรับข้อมูลเข้าจากผู้ใช้ได้หลายลักษณะ ไม่ว่าจะเป็น จำนวนเต็ม เลขทศนิยม หรือ ข้อความ
- ในภาษาซีเราใช้คำสั่ง printf เช่น หากเราต้องการอ่านค่าจำนวนเต็มมาเก็บไว้ในตัวแปร x เราจะเขียนว่า

```
#include <stdio.h>

void main() {
    int x;
    scanf("%d", &x);
}
```

- ส่วนในภาษาจาวา เราจะมีความซับซ้อนในตอนต้น เพราะเราจะสร้างวัตถุ Scanner ขึ้นมาก่อน แต่หลังจากนั้นจาวาจะดูเข้าใจง่ายกว่า

พื้นฐานการใช้ Scanner

- แม้เราจะสร้างวัตถุ Scanner ได้หลายแบบ แต่แบบพื้นฐานที่ใช้งานได้สะดวกเป็นดังนี้

```
Scanner scan = new Scanner(System.in);
```

- เมื่อเราจะใช้วัตถุจากคลาส Scanner (ในที่นี้วัตถุอยู่ในตัวแปรชื่อ scan) เพื่ออ่านค่าจำนวนเต็มแบบ int เราใช้เมธอด nextInt() เช่น

```
int x = scan.nextInt();
```

- และโปรแกรมสมบูรณ์ที่เทียบเท่ากับโค้ดภาษาซีก่อนหน้านี้คือ

```
import java.util.Scanner;
public class Scan {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int x = scan.nextInt();
    }
}
```

คั่นรายการ: แพ็คเกจในภาษาจาวา

- ในภาษาจาวามีคลาสอยู่เป็นจำนวนมาก จึงมีการจัดหมวดหมู่ให้เรียบร้อยเป็น*แพ็คเกจ*
  - แพ็คเกจเป็นเหมือนกล่องหรือบ้านสำหรับใส่คลาสต่าง ๆ เอาไว้
  - เรามักรวมของที่เกี่ยวข้องกันเป็นอย่างสูงไว้ในแพ็คเกจด้วยกัน
  - แพ็คเกจอาจจะมีแพ็คเกจย่อยในได้
  - เปรียบเหมือนบ้านหลังหนึ่ง อาจจะมีห้องอยู่ภายในหลายห้อง
- การจะระบุชื่อคลาสที่จะใช้งาน เรามักจะต้องระบุชื่อแพ็คเกจด้วย
  - เนื่องจากคลาสในสองแพ็คเกจสามารถชื่อซ้ำกันได้
  - แต่ในแพ็คเกจ ชื่อคลาสจะไม่ซ้ำกันแน่นอน
  - ดังนั้นการระบุชื่อคลาสจึงต้องทำควบคู่กันไปกับชื่อแพ็คเกจ

แล้ว Scanner อยู่ในแพ็คเกจหรือไม่

- คลาส Scanner ถูกจัดไว้ในแพ็คเกจ java.util
  - เขียนแบบนี้แสดงว่า java คือแพ็กเกจหลัก
  - ส่วน util เป็นแพ็คเกจย่อยใน java อีกที
- ดังนั้นชื่อที่เราจะต้องใช้อ้างถึง Scanner จึงต้องเขียนเป็น

```
java.util.Scanner
```

- และจากโปรแกรมจาวาเดิม หากเราเขียนแบบชื่อเต็ม ตอนสร้างวัตถุจากคลาส Scanner เราจะเขียนว่า

```
java.util.Scanner scan = new java.util.Scanner(System.in);
```

- แต่ถ้าเขียนชื่อเต็มแบบนี้ไปเสียทุกครั้ง โค้ดจะรุ่มร่ามและอ่านยาก จาวาจึงมีกลไกให้สามารถเขียนชื่อแบบย่อได้ด้วยการใช้ import

การใช้ import แบบง่าย ๆ

- หากเราอยากเขียนชื่อคลาสโดยไม่ต้องใช้แพ็คเกจนำหน้า เราสามารถใช้คำสั่ง import ไว้ตรงต้นไฟล์เพื่อระบุชื่อคลาสที่เราอยากเขียนแบบย่อ เช่น

```
import java.util.Scanner;
```

- ในตัวอย่างข้างบน ต่อไปเราจะอ้างถึงคลาส java.util.Scanner ได้ด้วยการเขียนว่า Scanner ก็เพียงพอ ทำให้ได้โค้ดที่ดูง่ายกว่าขึ้นจาก

```
java.util.Scanner scan = new java.util.Scanner(System.in);
```

- ไปเป็น

```
Scanner scan = new Scanner(System.in);
```

ตัวอย่าง: โปรแกรมบวกเลข

เราจะเขียนโปรแกรมที่รับจำนวนเต็มสองค่าคือ x และ y มาบวกกัน และแสดงผลบวกออกมาเป็นเป็นคำตอบ

```
import java.util.Scanner;

public class Adder {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int x = scan.nextInt();
        int y = scan.nextInt();
        System.out.println(x + y);
    }
}
```

ความสะดวกของการใช้ Scanner และ println

- สำหรับ Scanner เราจะเห็นได้ว่า เมธอด `nextInt()` ส่งค่าตัวเลขที่อ่านได้มาเป็นผลลัพธ์ ทำให้เราเก็บค่านั้นไว้ ณ ตอนประกาศตัวแปรได้
  - เราไม่ต้องคอยจำเรื่องการใช้ %d ที่ดูเข้าใจยากอย่างในภาษาซี แต่ชื่อเมธอดอย่าง `nextInt()` มันสื่อความหมายชัดเจน จำง่าย
- ส่วน println นั้นพิมพ์ข้อมูลทุกประเภทออกมาได้ เราไม่ต้องคอยบอกว่ามันเป็น int, float หรือ ข้อความ
  - เพราะจาวาจะพิจารณาจากชนิดข้อมูล และตัดสินให้เราแบบอัตโนมัติ
  - เราจึงเขียนโค้ดได้เป็น `System.out.println(x + y);`
  - ในตัวอย่างนี้ผลของการบวกจะถูกแปลงออกมาเป็นข้อความที่ใช้พิมพ์ออกมาทาง `println` ได้แบบอัตโนมัติ สะดวกมาก

การแสดงผลตัวเลขปนกับข้อความ

- เราสามารถต่อข้อความกับตัวเลขเป็นข้อความใหม่ได้
  - เช่น ถ้าหากเรามีตัวเลข x จากผู้ใช้ int x = scan.nextInt(); และเราต้องการแสดงค่าออกมาเป็นข้อความในรูปแบบ “input = ...”
  - เราสามารถใช้ความสามารถในการต่อข้อความกับตัวเลขที่จาวาเตรียมไว้ ซึ่งทำได้โดยการใช้เครื่องหมายบวก ดังแสดงในตัวอย่างข้างล่างนี้

```
int x = scan.nextInt();
System.out.println("input = " + x);
```

- **หมายเหตุ** การต่อข้อความนี้ ไม่ใช่ความสามารถของ println แต่เป็นความสามารถที่ติดมากับการจัดการข้อความในจาวา

เรื่องแปลก ๆ เมื่อพยายามต่อข้อความกับตัวเลข

- วกกลับไปตัวอย่างเรื่องการแสดงผลการบวก x กับ y
- เราอาจจะคิดว่า ถ้าเราจะแสดงผลลัพธ์ในรูป “sum = ...” เราสามารถเขียนโค้ดได้เป็น

```
System.out.println("sum = " + x + y);
```

- แต่เอาเข้าจริง ๆ เขียนแบบข้างบนจะผิด เป็นต้นว่าถ้า x และ y มีค่าเป็น 5 และ 7 ผลลัพธ์จากโปรแกรมจะเป็น

```
sum = 57
```

- เรื่องชวนงงคือว่า ถ้าเราเปลี่ยนโค้ดไปเป็น

```
System.out.println(x + y + " = sum");
```

- ผลลัพธ์ที่ได้จะเป็น

```
12 = sum
```

ทำไมจึงเป็นเช่นนั้น

- เป็นเพราะการบวก ไม่ว่าจะเป็นการบวกจำนวนเต็มหรือการบวกเพื่อต่อข้อความจะคิดจากซ้ายไปขวาตามลำดับ
  - ในตอนที่เราเขียนว่า `"sum = " + x + y` จาวาจะพยายามตีความผลของ `"sum = " + x`
  - ซึ่งเป็นการบวกเพื่อต่อข้อความ ดังนั้นเราจะได้ข้อความใหม่เป็น `"sum = 5"`
  - เมื่อพบเครื่องหมายบวกตัวที่สองจึงเป็นการบวกเพื่อต่อข้อความอีกครั้ง และเป็นการนำ 7 ไปต่อท้ายข้อความและได้ผลเป็น `"sum = 57“`

แล้วผลลัพธ์อีกอันล่ะ มาได้อย่างไร

- ในกรณีของ `x + y + " = sum"`
  - การบวกครั้งแรกเป็นการพบกันระหว่างจำนวนเต็ม x และ y
  - จึงเป็นการบวกจำนวนเต็มธรรมดา และได้ผลลัพธ์เป็น 12 ในตัวอย่างนี้
  - ส่วนการบวกครั้งที่สอง เป็นการต่อข้อความ
  - โดยจาวาจะแปลงเลขที่มีค่าเป็น 12 ให้กลายเป็นข้อความ `"12"`
  - จากนั้นจึงนำข้อความที่แปลงจากตัวเลขไปต่อกับข้อความ `" = sum"`
  - ทำให้ได้ผลลัพธ์เป็น `12 = sum`

แต่เราต้องการเขียนว่า sum = 12 ไม่ใช่ 12 = sum

- ถ้าเป็นแบบนี้ เราสามารถใช้วงเล็บเข้ามาช่วยเพื่อบังคับให้การบวกของ x กับ y เกิดขึ้นก่อน

```
System.out.println("sum = " + (x + y));
```

- เพราะการบวกระหว่าง x กับ y เป็นการบวกของจำนวนเต็มสองตัว ผลลัพธ์ที่เกิดขึ้นย่อมเป็นการบวกเลขธรรมดา ไม่ใช่การต่อข้อความ
- ทางแก้อีกอันหนึ่งก็คือ การสร้างตัวแปรขึ้นมาเก็บค่าผลบวกไว้ก่อน เช่น

```
int result = x + y;
System.out.println("sum = " + result);
```

การแสดงผลในลักษณะเดียวกับ printf ด้วย format

- การใช้ `println` และการต่อข้อความด้วยเครื่องหมายบวก นับว่าสะดวกมากในภาษาจาวา
- แต่ก็ใช่ว่ามันจะสะดวกกว่าการใช้ `printf` แบบภาษาซีในทุกกรณี
- โดยเฉพาะตอนที่เราต้องการจัดรูปแบบการแสดงผลให้เข้มงวด อย่างเช่น การแสดงเลขทศนิยมให้ได้ 6 หลักพอดีเป็นต้น
- ในกรณีเช่นนี้ เราใช้เมธอด `format` ใน System.out มาจัดการได้ เช่น

```
System.out.format("%.6f", 0.123456789);
```

  - ซึ่งจะให้ผลลัพธ์เป็น

```
0.123457
```

  - สังเกตด้วยว่าผลลัพธ์มีการปัดทศนิยมแบบเดียวกับ printf(“%.6f”, ...);

การอ่านค่าแบบเลขทศนิยม

- เลขทศยมแบบพื้นฐานในภาษาจาวามีสองแบบคือ float และ double
- เมธอดใน Scanner ที่ทำหน้าที่อ่านค่าทั้งสองชนิดมีชื่อที่ตรงไปตรงมา
  - `nextFloat()` สำหรับการอ่าน float
  - `nextDouble()` สำหรับการอ่าน double
- ตัวอย่าง จงอ่านเลขแบบ double สองค่าและแสดงผลบวกออกมาเป็นเลขทศนิยม 6 หลัก

```
Scanner scan = new Scanner(System.in);
double x = scan.nextDouble();
double y = scan.nextDouble();
System.out.format("%.6f", x + y);
```