CHAPTER 08

เมท็อด (Methods)

1. ความรู้เบื้องต้นเกี่ยวกับเมท็อด (Introduction to Methods)

1. นิยามของเมท็อด

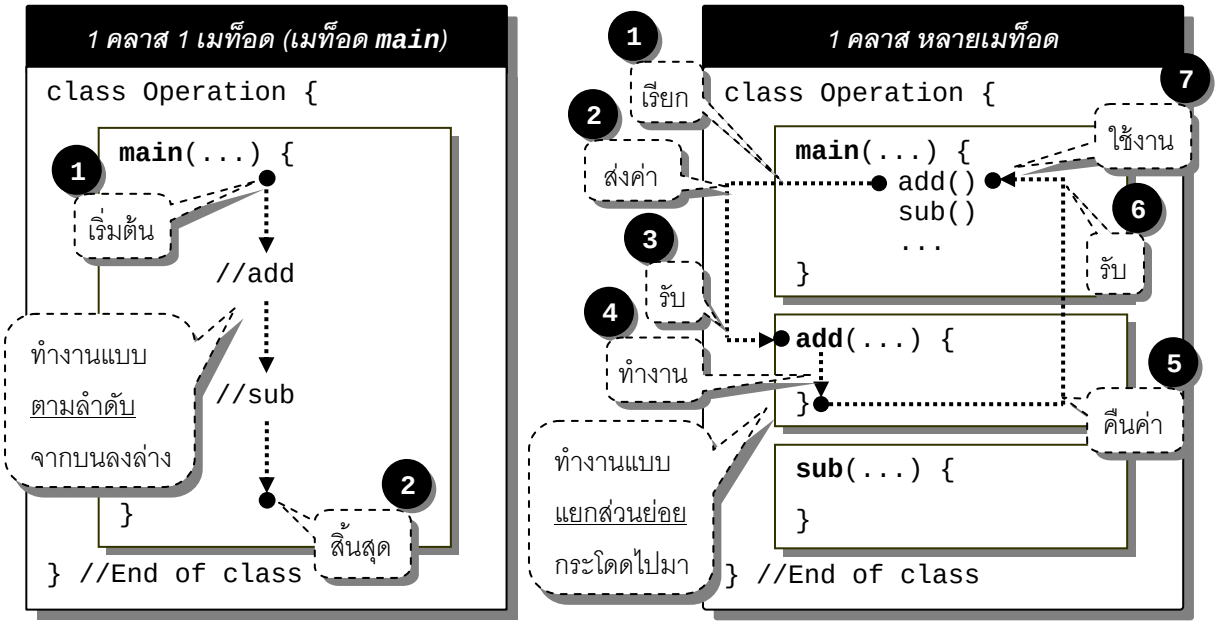
1) เมท็อด (Methods) คือ *โปรแกรมย่อย*ที่ทำหน้าที่เฉพาะ ซึ่งมีลักษณะเดียวกับ Subroutine, Sub-program และ Function โดยเมท็อดช่วยลดความซ้ำซ้อนของโปรแกรมที่ทำงานซ้ำๆ ได้

2) เมท็อดทุกเมท็อดจะปรากฏอยู่ในคลาส (Class) ซึ่งคลาสจะรวบรวมเมท็อดเอาไว้เป็นกลุ่มก้อน

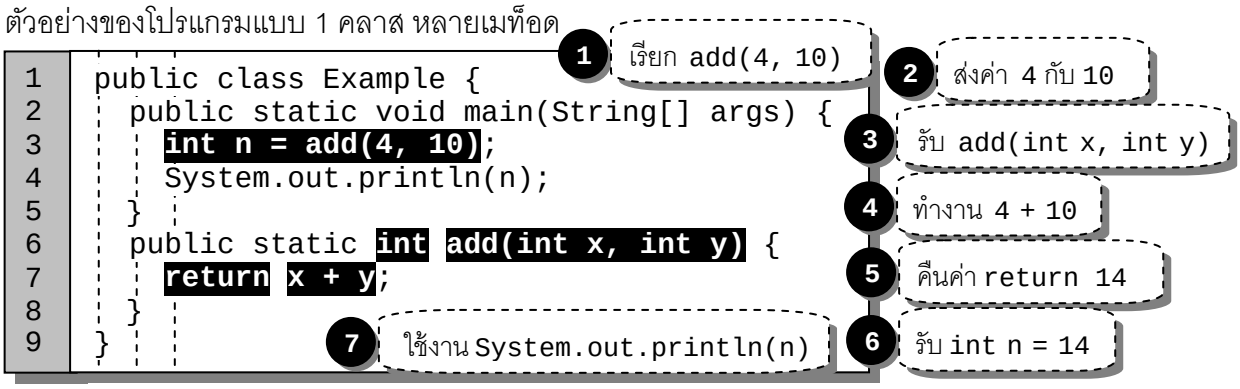
3) รูปแบบของโปรแกรมในช่วงก่อนหน้านี้เป็นโปรแกรมแบบ 1 คลาส 1 เมท็อด แต่รูปแบบของโปรแกรมในช่วงต่อจากนี้เป็นต้นไปจะเป็นโปรแกรมแบบ 1 คลาส หลายเมท็อด หรือ หลายคลาสหลายเมท็อด

(1) *โปรแกรมแบบ 1 คลาส 1 เมท็อด* จะรวมส่วนของโปรแกรมทุกส่วนไว้ในเมท็อด `main` และมีการประมวลผลแบบตามลำดับจากบนลงล่าง

(2) *โปรแกรมแบบ 1 คลาส หลายเมท็อด* จะแบ่งส่วนของโปรแกรมแต่ละส่วนออกเป็นเมท็อดแต่ละเมท็อด โดยมีเมท็อด `main` เป็นศูนย์กลาง และมีการประมวลผลแบบแยกส่วนย่อยกระโดดไปมาตามการเรียกใช้งานของเมท็อดต่างๆ (การประมวลผลขึ้นกับคำสั่งในเมท็อด `main` ถ้าไม่มี `main` จะไม่ประมวลผล)



4) ตัวอย่างของโปรแกรมแบบ 1 คลาส หลายเมท็อด



© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

<u>โจทย์ข้อที่ 1</u> <u>[ระดับง่าย]</u> จากตัวอย่างโปรแกรมต่อไปนี้ จงพิจารณาคำกล่าวแต่ละข้อว่ากล่าว<u>ผิด</u> (☒) หรือ<u>ถูก</u> (☑) *(20 คะแนน)*

```
1  import java.util.Scanner;
2  public class Test {
3    public static void main(String[] args) {
4      Scanner kb = new Scanner(System.in);
5      double a = kb.nextDouble();
6      int b = kb.nextInt();
7      double n = mul(a, b);
8      int m = mod(13);
9      System.out.println(n + m);
10   }
11   private static double mul(double x, int y) {
12     double n = x * y;
13     return n;
14   }
15   protected static int mod(int x) {
16     return x % 3;
17   }
18 }
```

ตั้งใจจะใส่ค่า 5.0

ตั้งใจจะใส่ค่า 2

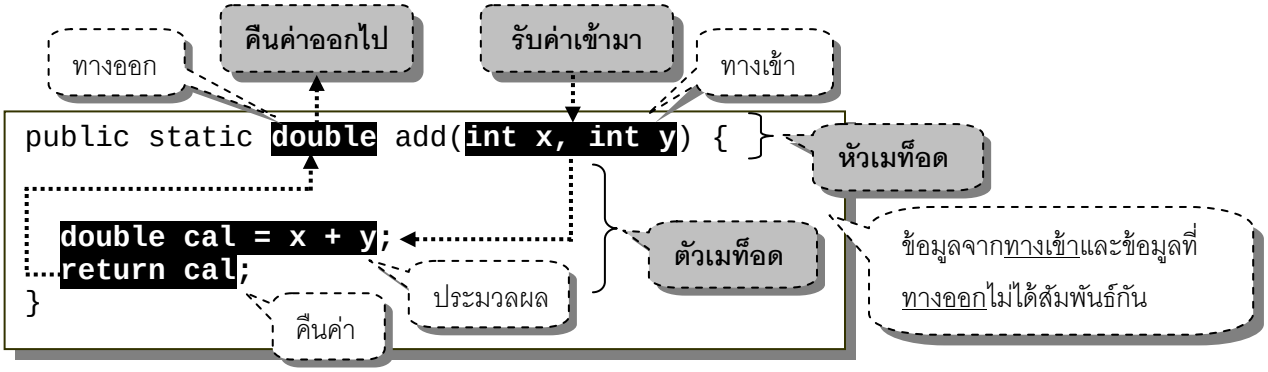
บางเมท็อดอาจเขียนอยู่ในรูปแบบอื่นที่ไม่ใช่ `public static` ก็ได้ เช่น `private static` เป็นต้น ซึ่งจะได้กล่าวรายละเอียดอีกครั้งในบทที่ 12

1) ☐ คลาส `Test` มีเมท็อดทั้งหมด 3 เมท็อด
2) ☐ ถ้าตัดเมท็อด `main` ออกจะทำให้แปลผล (Compile) <u>ไม่</u>ผ่าน และสั่งงาน (Run) <u>ไม่</u>ผ่าน
3) ☐ เมท็อด `main` สามารถเรียกใช้เมท็อด `mod` ก่อนเรียกใช้เมท็อด `mul` ได้
4) ☐ เมท็อด `mod` สามารถเรียกใช้เมท็อด `mul` ได้
5) ☐ เมท็อด `mod` สามารถเรียกใช้เมท็อด `main` ได้
6) ☐ เมท็อด `mod` สามารถเรียกใช้เมท็อด `mod` ได้
7) ☐ ถ้าต้องการเรียกใช้เมท็อด `mul` จะต้องส่งค่า 2 ค่าที่เป็นจำนวนจริงและจำนวนเต็มไปยังเมท็อด `mul`
8) ☐ ถ้าต้องการเรียกใช้เมท็อด `mod` จะต้องส่งค่า 1 ค่าที่เป็นจำนวนจริงไปยังเมท็อด `mod`
9) ☐ ค่า `n` ในบรรทัดที่ 7 กับค่า `n` ในบรรทัดที่ 12 มีค่าเท่ากันเพราะเป็นตัวแปรตัวเดียวกัน
10) ☐ ค่า `x` ในบรรทัดที่ 15 กับค่า `x` ในบรรทัดที่ 16 มีค่าเท่ากันเพราะเป็นตัวแปรตัวเดียวกัน
11) ☐ ค่า `x` ในเมท็อด `mod` จะมีค่าเท่ากับ 4 เมื่อเรียกใช้งานคำสั่ง `mod(4)`
12) ☐ ผลลัพธ์ที่ได้จากการประมวลผลในเมท็อด `mul` จะมีประเภทข้อมูลเป็น `int`
13) ☐ ผลลัพธ์ที่ได้จากการประมวลผลในเมท็อด `mod` จะมีประเภทข้อมูลเป็น `int`
14) ☐ ผลลัพธ์ที่ได้จากการประมวลผลบรรทัดที่ 7 และ 8 คือ 10.0 และ 1.0 ตามลำดับ
15) ☐ ผลลัพธ์ที่ได้จากการประมวลผลคำสั่ง `mul(2, 1)` คือ 2.0
16) ☐ ผลลัพธ์ที่ได้จากการประมวลผลคำสั่ง `mod(8, 3)` คือ 2.0
17) ☐ ผลลัพธ์ที่แสดงผลบนจอภาพที่ได้จากการประมวลผลของคลาส `Test` คือ 11.0
18) ☐ คำสั่ง `return` ในบรรทัดที่ 13 และ 16 เป็นคำสั่งที่ใช้คืนผลคำตอบที่ได้จากการประมวลผลของเมท็อด
19) ☐ คำสั่งภายในเมท็อด `mul` ตั้งแต่บรรทัดที่ 12-13 สามารถเขียนใหม่ได้เป็น `return x * y;`
20) ☐ เมท็อด `main` ไม่มีคำสั่ง `return` เพราะว่าเมท็อด `main` เป็นประเภท `void`

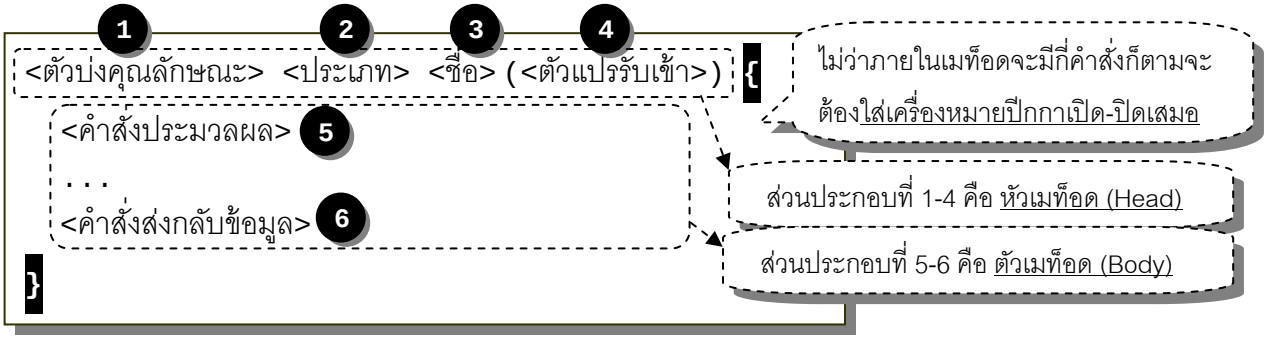
2. โครงสร้างของเมท็อดและการเขียนเมท็อด (Method Structure and Creating Method)

1. โครงสร้างโดยรวมของเมท็อด

สามารถแบ่งออกเป็น 2 ส่วนหลักได้แก่ หัวเมท็อดและตัวเมท็อด ดังตัวอย่าง



จากตัวอย่างสามารถแจกแจงส่วนประกอบย่อยของเมท็อดออกเป็น 6 ส่วนดังต่อไปนี้



2. หัวเมท็อด (Head)

เป็นส่วนที่แสดงถึงรูปแบบ โครงสร้างและข้อกำหนดของเมท็อด ซึ่งมีส่วนประกอบดังต่อไปนี้

1) **ตัวบ่งคุณลักษณะ** (Modifier) คือ ส่วนที่กำหนดคุณลักษณะและคุณสมบัติบางประการของเมท็อด ซึ่งได้แก่ `public`, `private`, `protected`, `static` ฯลฯ โดยจะอธิบายอีกครั้งในเรื่องคลาสและอ๊อบเจ็ค แต่ในบทนี้ให้เขียนตัวบ่งคุณลักษณะของทุกเมท็อดเป็นแบบ **`public static`** เท่านั้น ดังตัวอย่าง

```
public static double power(double n)
```

ส่วนนี้มีก็ได้ไม่มีก็ได้ ตัดออกได้

2) **ประเภทเมท็อด** (Method Type) **หรือประเภทข้อมูลที่ส่งกลับ** (Return Type) คือ ประเภทข้อมูลที่เป็นผลลัพธ์สุดท้ายของเมท็อดที่จะคืนค่ากลับ ซึ่งได้แก่ประเภทข้อมูล `byte`, `short`, `int`, `long`, `float`, `double` ฯลฯ หากเมท็อดไม่มีการคืนค่ากลับให้ระบุเป็นประเภท `void` ดังตัวอย่าง

```
protected void show(float y, String s)
```

ส่วนนี้ต้องมีเสมอ ตัดออกไม่ได้

ในกรณีที่ประเภทข้อมูลที่ส่งกลับมีโครงสร้างข้อมูลเป็นแบบอาเรย์ ให้ระบุเครื่องหมายวงเล็บเหลี่ยม (Brackets) *ต่อท้าย*ประเภทเมท็อดนั้นด้วย ดังตัวอย่าง

```
private static int[] sort(int x[])
```

ถ้าเป็นอาเรย์สองมิติให้ระบุเป็น **`int[][]`**

ในกรณีที่คืนค่ามากกว่า 1 ค่าจะต้องอยู่ในรูปอาเรย์

3) **ชื่อเมท็อด** (Method Name) คือ ชื่อที่อ้างถึงหรือชื่อเรียกเมท็อด ซึ่งควรขึ้นต้นด้วยอักษรตัวพิมพ์เล็ก (เป็นไปตามกฎการตั้งชื่อตัวแปร) และชื่อควรเป็นคำกริยาหรือคำที่แสดงการกระทำ เช่น `getID(…)`, `addNumber(…)` เป็นต้น และหลังชื่อเมท็อดต้องตามด้วยเครื่องหมายวงเล็บเปิด-ปิดเสมอ ดังตัวอย่าง

```
public static double add(int x, int y)
```

ส่วนนี้ต้องมีเสมอ ตัดออกไม่ได้

4) **ตัวแปรรับเข้า** (Parameters) คือ ค่าหรือตัวแปรที่รับเข้ามายังเมท็อดเพื่อใช้ประมวลผลภายในเมท็อด ซึ่งตัวแปรรับเข้าของเมท็อดหนึ่งๆ จะมีหรือไม่มีก็ได้ ถ้ามีจะมีกี่ตัวก็ได้ ถ้ามีหลายตัวแต่ละตัวต้องคั่นด้วยเครื่องหมายจุลภาค (Comma) และทุกตัวต้องมีประเภทข้อมูลกำกับเสมอ เช่น `set(int a, int b)` ห้ามเขียนเป็น `set(int a, b)` โดยเด็ดขาด ดังตัวอย่าง

คล้ายกับการประกาศตัวแปร

```
public double mul(int x, double y, int z)
```

ส่วนนี้มีก็ได้ไม่มีก็ได้ ตัดออกได้

ในกรณีที่พารามิเตอร์มีโครงสร้างข้อมูลเป็นแบบอาเรย์ ให้ระบุเครื่องหมายวงเล็บเหลี่ยมไว้*ข้างหน้า*หรือ*ต่อท้าย*ชื่อพารามิเตอร์นั้นด้วย ดังตัวอย่าง

```
public static int find(int x, int y[][])
```

มีพารามิเตอร์ที่เป็นอาเรย์สองมิติ

โจทย์ข้อที่ 2 [ระดับง่าย] จงเติมเต็มส่วน*หัวเมท็อด*ให้สมบูรณ์ตามรูปแบบของเมท็อดต่อไปนี้ ***(10 คะแนน)***

1) **เมท็อดชื่อ `square` มีข้อมูลรับเข้า 1 ค่าที่เป็นจำนวนจริง เพื่อใช้คำนวณหาค่ายกกำลังสองของจำนวนนั้น พร้อมทั้งส่งค่าคำตอบกลับ** ***(2 คะแนน)***

`[ ] double square( [ ] )`

2) **เมท็อดชื่อ `root` มีข้อมูลรับเข้า 1 ค่าที่เป็นจำนวนเต็ม เพื่อใช้คำนวณหาค่ารากที่สองของจำนวนนั้น พร้อมทั้งส่งค่าคำตอบกลับ** ***(2 คะแนน)***

`protected static [ ] [ ] ( [ ] )`

3) **เมท็อดชื่อ `showName` มีข้อมูลรับเข้า 1 ค่าที่เป็นชื่อของบุคคลใดๆ เพื่อใช้แสดงชื่อของบุคคลนั้นขึ้นบนจอภาพ โดยไม่ต้องส่งค่าคำตอบกลับ** ***(2 คะแนน)***

`private static [ ] showName( [ ] )`

4) **เมท็อดชื่อ `countX` มีข้อมูลรับเข้าที่เป็นอาเรย์ 1 มิติประเภทจำนวนเต็มและตัวเลขจำนวนเต็ม เพื่อใช้นับจำนวนสมาชิกในอาเรย์ที่มีค่าเท่ากับจำนวนเต็มที่รับมา พร้อมทั้งส่งค่าคำตอบกลับ** ***(2 คะแนน)***

`static [ ] [ ] [ ]`

5) **เมท็อดชื่อ `revArray` มีข้อมูลรับเข้า 1 ค่าที่เป็นอาเรย์ 1 มิติประเภทจำนวนจริง เพื่อใช้กลับค่า (Reverse) สมาชิกในอาเรย์นั้น พร้อมทั้งส่งค่าคำตอบกลับ** ***(2 คะแนน)***

`[ ] [ ] revArray [ ]`

3. **ตัวเมท็อด** (Body)

เป็นส่วนที่แสดงรายละเอียดของการประมวลผลและการส่งกลับข้อมูลของเมท็อด ซึ่งมีดังต่อไปนี้

1) **การประมวลผล** ซึ่งได้แก่การประกาศตัวแปรเพื่อใช้ภายในเมท็อด และการใช้คำสั่งต่างๆ เช่น `if-else`, `while`, `for`, `Scanner` ในการประมวลผลภายในเมท็อด ซึ่งหลักการเหล่านี้จะเหมือนกับเนื้อหาก่อนหน้านี้ตั้งแต่บทที่ 1-7 ที่ได้เรียนมาแล้วทุกประการ

2) **การส่งกลับข้อมูล** เป็นการส่งค่าหรือคืนค่าคำตอบที่ได้จากการประมวลผลในเมท็อดกลับไปยังเมท็อดที่เรียกใช้ ซึ่งค่าที่ส่งกลับ (Return Value) จะต้องตรงกับประเภทเมท็อด (Method Type) หรือ ประเภทข้อมูลที่ส่งกลับ (Return Type) ที่ประกาศไว้ที่หัวเมท็อดเสมอ หรือถ้าไม่ตรงกันจะต้องสามารถ*สอดใส่กันได้* ซึ่งการส่งกลับจะใช้คำสั่ง `return` และสามารถทำได้เพียงครั้งเดียวเท่านั้น ถ้ามีคำสั่ง `return` หลายกรณีจะทำเพียงกรณีแรกที่เป็นจริง แต่ถ้าประเภทเมท็อดเป็น `void` ไม่ต้องมีคำสั่ง `return` ดังตัวอย่าง

```
1  public static double div(int a, int b){
2    double x = (double) a / (double) b;
3    return x;
4  }
```

เมท็อดเป็นประเภท `double`

ค่าที่จะส่งกลับ (ค่าของตัวแปร `x`) จะต้องมีประเภทข้อมูลเป็น `double`

3) **การส่งกลับข้อมูลแบบหลายค่า** ถ้าเมท็อดประมวลผลแล้วได้ผลลัพธ์ที่เป็นคำตอบสุดท้ายมากกว่า 1 ค่า จะต้องส่งค่ากลับโดยใช้อาเรย์ (หนึ่งมิติ) โดยการเก็บค่าคำตอบทุกค่าไว้ในอาเรย์แล้วคืนค่าของอาเรย์นั้นเพียงค่าเดียวกลับไปยังเมท็อดที่เรียกใช้ ซึ่งจะทำให้ได้ผลคำตอบทุกค่าคืนกลับมาด้วย ดังตัวอย่าง

```
1  public static int[] findFirstLast(int a[]) {
2    int x[] = new int[2];
3    x[0] = a[0];
4    x[1] = a[a.length – 1];
5    return x;
6  }
```

ประเภทข้อมูลที่ส่งกลับที่หัวเมท็อดต้องเป็นอาเรย์

การส่งกลับข้อมูลแบบอาเรย์จะเขียนเฉพาะชื่อของอาเรย์ โดยไม่ต้องใส่เครื่องหมายวงเล็บเหลี่ยม

โจทย์ข้อที่ 3 [ระดับง่าย] จงพิจารณาเมท็อดต่อไปนี้เขียนผิด (☒) หรือถูก (☑) ตามหลักไวยากรณ์ของภาษาจาวาพร้อมทั้งบอกเหตุผลกำกับ *(10 คะแนน)*

1) ☐
```
public static printError(String msg) {
  System.err.println(msg);
}
```

2) ☐
```
protected static int flip(int n) {
  if (n == 1) n = 0;
}
```

3) ☐
```
public static float max(long x, y) {
  if (x > y) return x;
  else return y;
}
```

4) ☐
```
private static void showChar() {
  for(char i = '0'; i <= '9'; i++) {
    System.out.println(i);
  }
  return;
}
```

5) ☐
```
static String calGrade(int x) {
  if (x > 80) return "A";
  else if (x < 50) return "F";
  else System.out.println("C");
}
```

6) ☐
```
double getLen(double dx, double dy) {
  double dx = Math.abs(dx);
  return Math.sqrt(dx*dx + dy*dy);
}
```

7) ☐
```
public private boolean checkLen(int x[], int y[]) {
  return x.length == y.length;
}
```

8) ☐
```
public float getLocationPoint() {
  return 0.0;
}
```

9) ☐
```
public static int fac(int x) {
  if(x <= 1) return 1;
  else return fac(x – 1) * x;
}
```

10) ☐
```
int[] getThreeMember(int[] x) {
  int n[] = { x[0], x[1], x[2] };
  return n[];
}
```

โจทย์ข้อที่ 4 [ระดับง่าย] จงเขียนเมท็อดอย่างง่ายตามรูปแบบการทำงานที่ระบุไว้ต่อไปนี้ *(16 คะแนน)*

1) เมท็อดชื่อ **addRealNumber** มีข้อมูลรับเข้าเป็นตัวเลขจำนวนจริง 3 จำนวน ซึ่งใช้ในการหาผลบวกของตัวเลข 3 จำนวนนั้นแล้วมีการคืนค่ากลับ *(4 คะแนน)*

2) เมท็อดชื่อ **printX** มีข้อมูลรับเข้าเป็นตัวเลขจำนวนเต็ม 1 จำนวน เพื่อใช้ในการแสดงผลลัพธ์ออกทางจอภาพภายในเมท็อดนั้นโดยไม่คืนค่ากลับ *(4 คะแนน)*

3) เมท็อดชื่อ **divideByInt** ใช้ในการหาผลหารระหว่างตัวเลข 2 จำนวน ซึ่งเป็นจำนวนจริงและจำนวนเต็มอย่างละ 1 จำนวน แล้วมีการคืนค่ากลับ *(4 คะแนน)*

4) เมท็อดชื่อ **`fullName`** ใช้ในการรวมชื่อและนามสกุลที่รับเข้ามาให้เป็นชื่อเต็ม (ระหว่างชื่อและนามสกุลต้องมีการเว้นวรรคด้วย) แล้วคืนค่ากลับ *(4 คะแนน)*

โจทย์ข้อที่ 5 [ระดับง่าย] จงเขียนเมท็อด **`fac(…)`** ที่สมบูรณ์เพื่อใช้ในการคำนวณหาค่าแฟกทอเรียล (Factorial) ของตัวเลขจำนวนเต็มที่ระบุ เช่น **`fac(3) = 6`**, **`fac(5) = 120`** เป็นต้น *(10 คะแนน)*

โจทย์ข้อที่ 6 [ระดับง่าย] เมท็อดชื่อ **`findMax`** มีข้อมูลรับเข้าที่เป็นชุดของตัวเลขจำนวนจริง 50 จำนวน เพื่อนำมาหาค่าตัวเลขที่มากที่สุดใน 50 จำนวนนั้น แล้วมีการคืนค่ากลับ *(10 คะแนน)*

โจทย์ข้อที่ 7 [ระดับปานกลาง] จงเขียนเมท็อด **`underX(…)`** ที่สมบูรณ์เพื่อใช้คำนวณและนับว่ามีสมาชิกในอาเรย์หนึ่งมิติชื่อ **`a`** กี่จำนวนที่มีค่าน้อยกว่าจำนวนเต็ม **`x`** โดยให้รับอาเรย์ **`a`** และตัวแปร **`x`** ผ่านทางพารามิเตอร์ *(10 คะแนน)*

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

โจทย์ข้อที่ 8 [ระดับปานกลาง] จงเขียนเมท็อด **isPrime(…)**ที่สมบูรณ์เพื่อใช้ตรวจสอบตัวเลขจำนวนเต็มที่รับเข้ามาทางพารามิเตอร์ว่าเป็นจำนวนเฉพาะหรือไม่ *(10 คะแนน)*

โจทย์ข้อที่ 9 [ระดับปานกลาง] จงเขียนเมท็อด **memberOfArray(…)** ที่สมบูรณ์เพื่อค้นหาลำดับของสมาชิกตัวแรกในอาเรย์หนึ่งมิติชื่อ **a** ที่มีค่าเท่ากับจำนวนเต็ม **x** ถ้าค้นเจอมากกว่า 1 จำนวนให้คืนค่าเพียงสมาชิก*ตัวแรก*ที่ค้นเจอโดยเริ่มจากตัวซ้ายสุด แต่ถ้าค้นไม่เจอให้คืนค่า -1 *(10 คะแนน)*

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

โจทย์ข้อที่ 10 [ระดับยาก] จงเขียนเมท็อด `appendArray(…)` เพื่อรับพารามิเตอร์สองค่าที่เป็นอาเรย์ชนิดจำนวนเต็มทั้งคู่แล้วคืนค่าเป็นอาเรย์ชนิดจำนวนเต็มที่เกิดจากการเอาข้อมูลในอาเรย์ของพารามิเตอร์ตัวที่สองไปต่อท้ายตัวที่หนึ่ง เช่น `a[] = {1, 2, 3 }` และ `b[] = {5, 6}` จะได้ผลลัพธ์ดังนี้ `appendArray(a, b) = {1, 2, 3, 5, 6}` เป็นต้น *(10 คะแนน)*

3. การเรียกใช้เมท็อดอย่างง่าย (Simple Method Calling)

1. **บทบาทของเมท็อด** ในการเรียกใช้เมท็อดจะประกอบไปด้วยเมท็อด 2 ฝ่ายได้แก่
   1) เมท็อดผู้เรียก ทำหน้าที่เป็นผู้ไปเรียกใช้เมท็อดอื่น
   2) เมท็อดผู้ถูกเรียก ทำหน้าที่เป็นผู้ที่ให้เมท็อดอื่นมาเรียกใช้ตัวมัน

```
1  public static void main(String[] args) {
2    int n = add(5, 3);
3  }
4  public static int add(int x, int y) {
5    return x + y;
6  }
```

เมท็อด `main` เป็นผู้เรียก (เรียกใช้เมท็อด `add`)

เมท็อด `add` เป็นผู้ถูกเรียก (ถูกเรียกใช้โดยเมท็อด `main`)

2. **ขั้นตอนการเรียกใช้เมท็อดอย่างง่าย**
   1) ระบุชื่อเมท็อดที่ต้องการจะเรียก เช่น `add(…)` (ต้องมีเครื่องหมายวงเล็บเปิด-ปิดเสมอ)
   2) ระบุค่าพารามิเตอร์ทุกตัวของเมท็อดให้ถูกต้องตามประเภทข้อมูล เช่น `add(5, 3)` (ค่าของพารามิเตอร์สามารถระบุเป็นชื่อตัวแปรหรือนิพจน์ก็ได้ เช่น `add(x, y * 2)`)
   3) สร้างตัวแปรเพื่อมารับคำตอบที่ได้จากการประมวลผลของเมท็อด เช่น `int n = add(5, 3);`
      (1) ถ้าเมท็อดที่ต้องการจะเรียกคืนค่าข้อมูลประเภท `byte`, `short`, `int`, `long`, `float`, `double` ฯลฯ ให้สร้างตัวแปรเพื่อรับคำตอบตามประเภทข้อมูลนั้นๆ เช่น `String s = getName(53300121);`
      (2) ถ้าเมท็อดที่ต้องการจะเรียกไม่คืนค่าหรือมีประเภทข้อมูลเป็น **`void`** ไม่ต้องสร้างตัวแปรเพื่อรับคำตอบ (สามารถเรียกใช้เมท็อดโดยไม่ต้องมีตัวแปรมารับ) เช่น `showName(53300121);`

```
1  public static void main(String[] args) {
2    String s = getName(53300121);
3    System.out.println(s);
4    showName(53300121);
5  }
6  public static String getName(int id) {
7    return "Wongyos";
8  }
9  public static void showName(int id) {
10   System.out.println("Wongyos");
11 }
```

เมท็อด main เรียกใช้เมท็อด getName โดยส่งพารามิเตอร์ประเภท int ไปประมวลผลและสร้างตัวแปร s เพื่อรับคำตอบที่คืนกลับมา

เมท็อด main เรียกใช้เมท็อด showName แบบโดดๆ โดยส่งพารามิเตอร์ประเภท int ไปประมวลผล ซึ่งไม่มีการคืนค่ากลับ

4) การส่งค่าและรับค่าอาเรย์

(1) การส่งค่าอาเรย์ไปยังเมท็อดใดๆ ให้ระบุเพียงชื่ออาเรย์เท่านั้น โดยไม่ต้องใส่วงเล็บเหลี่ยม เช่น

```
int n[] = { 1, 2, 3, 4 };
int c = underX(n, 3);
```

(2) การรับค่าที่คืนกลับมาจากเมท็อดใดๆ ในรูปของอาเรย์ ให้ประกาศอาเรย์มารับ (ไม่ต้องใช้คำสั่ง new) เช่น

```
int x[] = { 1, 2, 3, 4 }, y[] = { 5, 7, 9 };
int z[] = appendArray(x, y);
```

3. **ข้อสังเกตในการเรียกใช้เมท็อด**

1) การเรียกใช้เมท็อด (ที่สร้างเสร็จสมบูรณ์แล้ว) จะระบุเพียงค่าพารามิเตอร์ของเมท็อดที่ต้องการจะเรียกใช้เท่านั้น โดยไม่ต้องระบุประเภทข้อมูลของพารามิเตอร์ (byte, short ฯลฯ) เช่น `int n = add(x, y);`

2) การประกาศและสร้างเมท็อด (ที่ยังไม่เคยสร้างแต่กำลังจะสร้าง) จะต้องระบุทั้งชื่อพารามิเตอร์และประเภทข้อมูลของพารามิเตอร์ทุกๆ ตัว ของเมท็อดที่จะสร้างขึ้นใหม่ เช่น `public ... add(int x, int y)`

```
1  public static void main(String[] args) {
2    int x = 5, y = 3;
3    int n = add(x, y);
4  }
5  public static int add(int x, int y) {
6    return x + y;
7  }
```

การเรียกใช้เมท็อด → ใส่ค่าพารามิเตอร์โดยไม่ต้องระบุประเภทข้อมูล

การประกาศและสร้างเมท็อด → ใส่ชื่อพารามิเตอร์และระบุประเภทข้อมูล

ตัวแปร x และ y ในเมท็อด main จะเป็นคนละตัว กับตัวแปร x และ y ในเมท็อด add

โจทย์ข้อที่ 11 [ระดับง่าย] จงเขียนคำสั่งการเรียกใช้งานเมท็อด จากหัวเมท็อดที่กำหนดให้ต่อไปนี้ โดยสามารถกำหนดค่าของพารามิเตอร์ของเมท็อดที่เรียกใช้ได้อย่างอิสระ *(10 คะแนน)*

ข้อ	หัวเมท็อด	เขียนคำสั่งเพื่อเรียกใช้งานเมท็อด
1.	`public static void set(int m, int n)`	
2.	`public static int cals(int m, float n)`	
3.	`public static String toString(int[] n)`	

ข้อ	หัวเมท็อด	เขียนคำสั่งเพื่อเรียกใช้งานเมท็อด
4.	`public static int[] get(int m, String s)`	
5.	`public static void sort(double d[])`	
6.	`public static boolean check(long id)`	
7.	`public static int flip(int m, boolean b)`	
8.	`public static double[] inputArray()`	
9.	`public static void showLogo()`	
10.	`public static byte f(int m, int n, int p)`	

โจทย์ข้อที่ 12 [ระดับง่าย – ระดับยาก] จงเขียนเมท็อดต่าง ๆ ในคลาส **ArrayUtility** ให้สมบูรณ์เพื่อใช้ประมวลผลกับอาเรย์ โดยมีรายละเอียดของเมท็อดดังต่อไปนี้ และกำหนดให้เมท็อดแต่ละเมท็อดสามารถเรียกใช้งานซึ่งกันและกันได้อย่างอิสระ *(75 คะแนน)*

```
import java.util.Scanner;
public class ArrayUtility {
```

//[ระดับง่าย] เมท็อด **main(…)** เพื่อเรียกใช้งานเมท็อดอื่น ๆ ให้ครบถ้วนพร้อมทั้งแสดงผลลัพธ์ขึ้นบนจอภาพให้สวยงาม โดยสามารถกำหนดค่าพารามิเตอร์ของแต่ละเมท็อดได้อย่างอิสระ *(5 คะแนน)*

```
  public static void main(String[] args) {
    int a[] = { 5, 1, 6, 1, 4, 1, 2, 1, 4, 3, 1, 5, 7, 2, 1, 1 };
    sort(a);
    System.out.println(max(a));
    System.out.println(min(a));
```

ตัวอย่างการเรียกใช้เมท็อดพร้อมทั้งการแสดงผล

ข้อ	เขียนคำสั่งเพื่อเรียกใช้งานเมท็อด	เมท็อดที่เรียก
1.		`count(…)`
2.		`mode(…)`
3.		`majority(…)`
4.		`median(…)`
5.		`range(…)`

```
  } //End of main
```

//[ระดับปานกลาง] เมท็อด **`swap(…)`** ใช้ในการสลับค่าสมาชิกในอาเรย์ตามตำแหน่ง **`i`** และ **`j`** ที่ระบุ เช่นคำสั่ง **`swap(a, i, j)`** คือการสลับค่าสมาชิกตำแหน่งที่ **`i`** และ **`j`** ของอาเรย์ **`a`** เป็นต้น โดยไม่ต้องคืนค่ากลับ *(5 คะแนน)*

//[ระดับปานกลาง] เมท็อด **`sort(…)`** ใช้จัดเรียงข้อมูลในอาเรย์จากน้อยไปหามากโดยใช้การจัดเรียงแบบฟอง (Bubble Sort) โดยไม่ต้องคืนค่ากลับ *(10 คะแนน)*

//[ระดับปานกลาง] เมท็อด **`count(…)`** ใช้นับจำนวนสมาชิกในอาเรย์ที่มีค่าเท่ากับจำนวนเต็ม **`k`** ที่ระบุ (นับความถี่) แล้วคืนค่าจำนวนที่นับได้ *(10 คะแนน)*

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

//[ระดับยาก] เมท็อด **mode(…)** ใช้หาค่าฐานนิยม โดยจะคืนค่าสมาชิกในอาเรย์ที่มีค่าความถี่สูงสุดหรือปรากฏจำนวนครั้งมากที่สุด ถ้ามีหลายค่าให้คืนค่าใดก็ได้เพียงค่าเดียว *(10 คะแนน)*

//[ระดับยาก] เมท็อด **majority(…)** ใช้หาค่าหมู่มาก โดยจะคืนค่าสมาชิกในอาเรย์ที่มีค่าความถี่มากกว่าครึ่งหนึ่งของจำนวนสมาชิกทั้งหมด ถ้าหาไม่ได้ให้คืนค่า -1 *(10 คะแนน)*

//[ระดับปานกลาง] เมท็อด **median(…)** ใช้หาค่ามัธยฐาน พร้อมทั้งคืนค่ากลับ *(10 คะแนน)*

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

//[ระดับง่าย] เมท็อด **max(…)** ใช้หาค่าสูงสุดของสมาชิกในอาเรย์ พร้อมทั้งคืนค่ากลับ *(5 คะแนน)*

```
```

//[ระดับง่าย] เมท็อด **min(…)** ใช้หาค่าต่ำสุดของสมาชิกในอาเรย์ พร้อมทั้งคืนค่ากลับ *(5 คะแนน)*

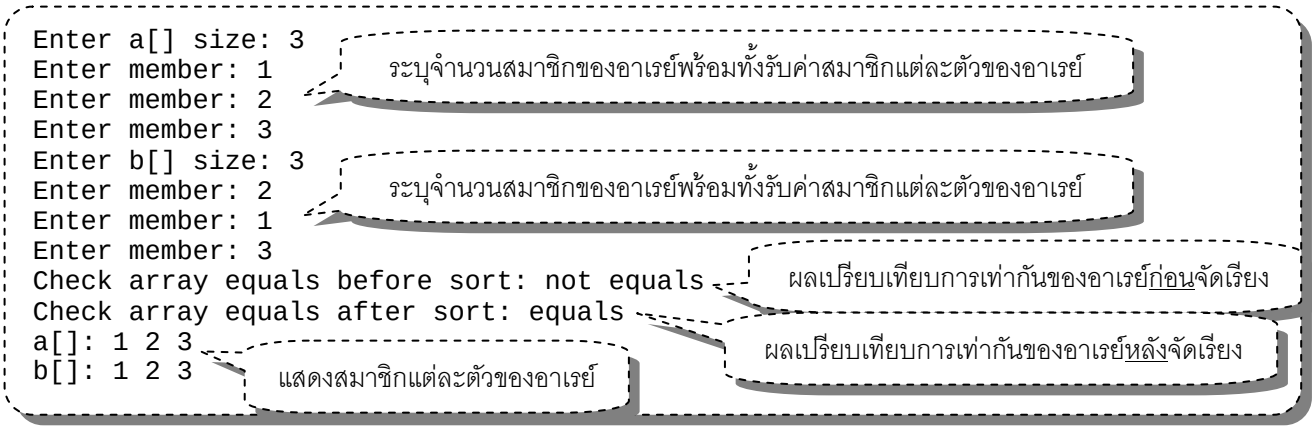
```
```

//[ระดับง่าย] เมท็อด **range(…)** ใช้หาค่าพิสัยของอาเรย์ พร้อมทั้งคืนค่ากลับ *(5 คะแนน)*

```
```

```
} //End of class
```

โจทย์ข้อที่ 13 [ระดับยาก] โปรแกรมต่อไปนี้เป็นโปรแกรมที่เขียนคำสั่งทุกคำสั่งไว้ในเมท็อด **main(…)** เพียงเมท็อดเดียว โดยโปรแกรมนี้ใช้สำหรับสร้างอาเรย์ 1 มิติประเภทจำนวนเต็ม 2 ตัว พร้อมทั้งรับค่าสมาชิกผ่านทางแป้นพิมพ์เก็บไว้ในอาเรย์แต่ละช่องจนครบสมบูรณ์ หลังจากนั้นทำการตรวจสอบการเท่ากันของอาเรย์ทั้งสองว่าเท่ากันหรือไม่พร้อมทั้งแสดงผลลัพธ์ขึ้นบนจอภาพ จากนั้นทำการจัดเรียงข้อมูลจากน้อยไปหามากของอาเรย์ทั้งสองแล้วตรวจสอบการเท่ากันของอาเรย์อีกครั้งหนึ่งว่าเท่ากันหรือไม่พร้อมทั้งแสดงผลลัพธ์ขึ้นบนจอภาพ สุดท้ายทำการแสดงสมาชิกแต่ละตัวของอาเรย์ทั้งสองขึ้นบนจอภาพอย่างสวยงามดังตัวอย่างข้างล่าง จากโปรแกรมที่กำหนดให้จงเขียนโปรแกรมใหม่โดยแบ่งส่วนของโปรแกรมต่างๆ ออกเป็นเมท็อดย่อยให้เหมาะสม (ประมาณ 5-6 เมท็อด) เพื่อให้การทำงานของโปรแกรมเป็นสัดส่วนยิ่งขึ้น แต่ให้คงไว้ซึ่งการแสดงผลลัพธ์ที่เหมือนเดิมทุกประการ *(40 คะแนน)*




```
1   import java.io.*;
2   import java.util.Scanner;
3   import jlab.graphics.DWindow;
4   public class OneMainOneMethod {
5     public static void main(String[] args) {
6       Scanner kb = new Scanner(System.in);
7
8       System.out.print("Enter a[] size: ");
9       int x = kb.nextInt();
10      int a[] = new int[x];
11      for (int i = 0; i < a.length; i++) {
12        System.out.print("Enter member: ");
13        a[i] = kb.nextInt();
14      }
15
16      System.out.print("Enter b[] size: ");
17      int y = kb.nextInt();
18      int b[] = new int[y];
19      for (int i = 0; i < b.length; i++) {
20        System.out.print("Enter member: ");
21        b[i] = kb.nextInt();
22      }
23
24      System.out.print("Check array equals before sort: ");
25      if (a.length == b.length) {
26        int count = 0;
27        for (int i = 0; i < a.length; i++) {
28          if (a[i] == b[i]) count++;
29        }
30        if (count == a.length) {
31          System.out.println("equals");
32        } else {
33          System.out.println("not equals");
34        }
35      } else {
36        System.out.println("not equals");
37      }
38
39      for (int i = a.length - 1; i >= 1; i--) {
40        for (int j = 0; j < i; j++) {
41          if (a[j] > a[j + 1]) {
42            int t = a[j];
43            a[j] = a[j + 1];
44            a[j + 1] = t;
45          }
46        }
47      }
```

```
48      for (int i = b.length - 1; i >= 1; i--) {
49        for (int j = 0; j < i; j++) {
50          if (b[j] > b[j + 1]) {
51            int t = b[j];
52            b[j] = b[j + 1];
53            b[j + 1] = t;
54          }
55        }
56      }
57
58      System.out.print("Check array equals after sort: ");
59      if (a.length == b.length) {
60        int count = 0;
61        for (int i = 0; i < a.length; i++) {
62          if (a[i] == b[i]) count++;
63        }
64        if (count == a.length) {
65          System.out.println("equals");
66        } else {
67          System.out.println("not equals");
68        }
69      } else {
70        System.out.println("not equals");
71      }
72
73      System.out.print("a[]: ");
74      for (int i = 0; i < a.length; i++) {
75        System.out.print(a[i] + " ");
76      }
77      System.out.println();
78
79      System.out.print("b[]: ");
80      for (int i = 0; i < b.length; i++) {
81        System.out.print(b[i] + " ");
82      }
83      System.out.println();
84    } //End of main
85  } //End of class
```

เขียนโปรแกรมใหม่โดยแบ่งออกเป็นเมท็อดย่อย (ประมาณ 5-6 เมท็อด รวมเท็อด **main(…)** แล้ว) โดยวิธีการแบ่งเมท็อดย่อยให้พิจารณาจากสัดส่วนการทำงานย่อยของโปรแกรมเดิม

```
import java.util.Scanner;
public class OneMainManyMethods {
```

คะแนนจะแปรผันไปตามโปรแกรมและเมท็อดที่ออกแบบ

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

```
} //End of class
```

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

4. เมท็อดหลายแบบในชื่อเดียวกัน (Overloading Methods)

1. นิยามของเมท็อดหลายแบบในชื่อเดียวกัน (Overloading Methods)

คือ เมท็อดตั้งแต่ 2 เมท็อดใดๆ ขึ้นไปที่มีชื่อเหมือนกันแต่มี*รายการ*ของพารามิเตอร์ต่างกัน ดังรายละเอียดต่อไปนี้

1) จำนวนพารามิเตอร์ต่างกัน เช่น

```
public static int test(int x)
public static int test(int x, int y)
```

เมท็อด test ตัวที่หนึ่งมีพารามิเตอร์ 1 ตัว
เมท็อด test ตัวที่สองมีพารามิเตอร์ 2 ตัว

2) ประเภทข้อมูลของพารามิเตอร์ต่างกัน

```
public static int test(int x)
public static int test(double x)
```

เมท็อด test ตัวที่หนึ่งมีพารามิเตอร์ประเภท int
เมท็อด test ตัวที่สองมีพารามิเตอร์ประเภท double

2. ข้อสังเกตเกี่ยวกับเมท็อดหลายแบบในชื่อเดียวกัน

1) เมท็อดแต่ละเมท็อดจะมีประเภทข้อมูลที่ส่งกลับ (Return Type) ที่ต่างกันหรือเหมือนกันก็ได้ (ไม่ต้องไปสนใจ และไม่ใช่เป็นประเด็นที่จะใช้พิจารณาเมท็อดประเภทนี้)
2) การเปรียบเทียบความเหมือนหรือไม่เหมือนของพารามิเตอร์ของแต่ละเมท็อด ให้พิจารณาที่ประเภทข้อมูลของพารามิเตอร์เท่านั้น ไม่ได้พิจารณาที่ชื่อของพารามิเตอร์แต่อย่างใด (ต่างเมท็อดกันชื่อเหมือนกันได้)
3) ระบบจะเลือกเมท็อดที่เหมาะสมที่สุดเพียงเมท็อดเดียวขึ้นมาใช้งาน ถ้าไม่มีเมท็อดใดเลยที่เหมาะสมจะแปลผลไม่ผ่าน (Compile-time Error)

โจทย์ข้อที่ 14 [ระดับง่าย] จงเลือกใช้เมท็อดหลายแบบในชื่อเดียวกัน (Overloading Method) ที่เหมาะสมที่สุดจากคำสั่งต่อไปนี้ พร้อมทั้งแสดงผลลัพธ์ที่เกิดจากคำสั่งนั้น *(15 คะแนน)*

หมายเลขเมท็อด	รายละเอียดของเมท็อด

1	`public static double m(int n) {`
	`  return n * n;`
	`}`

2	`public static int m(double n) {`
	`  return (int)(n * n);`
	`}`

3	`public static boolean m(int n, int x) {`
	`  return n == x;`
	`}`

4	`public static double m(int a, int b, int c) {`
	`  return (double)(a * b * c);`
	`}`

5	`public static int m(char c) {`
	`  return (int) c;`
	`}`

6	`public static String m(double n, int x) {`
	`  return n + "" + x;`
	`}`

7	`public static int m(double x, double n) {`
	`  return (int)(n * x);`
	`}`

ข้อ	คำสั่ง	หมายเลขเมท็อด	ผลลัพธ์
1.	`System.out.println(m(10));`		
2.	`System.out.println(m(1, 2.5));`		
3.	`System.out.println(m(11.0));`		
4.	`System.out.println(m(6.0f));`		
5.	`System.out.println(m(12, 2));`		
6.	`System.out.println(m('1'));`		
7.	`System.out.println(m(10.0, 2));`		
8.	`System.out.println(m(2, 2, 1));`		
9.	`System.out.println(m(m(5, 1, 2)));`		
10.	`System.out.println(m(m(5, 1.0f)));`		
11.	`System.out.println(m(1.0f, 2));`		
12.	`System.out.println(m(0L, 1, 1));`		
13.	`System.out.println(m(0.0f, (byte) 1));`		
14.	`System.out.println(m((byte) 1.3, 1));`		
15.	`System.out.println(m(1.1));`		

โจทย์ข้อที่ 15 [ระดับง่าย] จงเขียนเมท็อดหลายแบบในชื่อเดียวกัน (Overloading Method) ตามข้อกำหนดต่อไปนี้ พร้อมทั้งเรียกใช้งานเมท็อดเหล่านี้ให้ถูกต้องสมบูรณ์ *(15 คะแนน)*

```
import java.util.Scanner;
public class Overload {
    //เมท็อด main(…) เพื่อเรียกใช้เมท็อด showName(…) ทุกตัวให้ครบสมบูรณ์ และกำหนดค่าพารามิเตอร์ได้อย่างอิสระ (3 คะแนน)
    public static void main(String[] args) {
        
    } //End of main
```

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)

//เมท็อด **showName(…)** แบบที่ไม่มีพารามิเตอร์เพื่อใช้แสดงชื่อของนิสิตขึ้นบนจอภาพ *(3 คะแนน)*

//เมท็อด **showName(…)** แบบที่มีพารามิเตอร์ 1 ตัวที่รับชื่อนิสิตชื่อใดๆ เข้ามาเพื่อใช้แสดงชื่อของนิสิตคนนั้นขึ้นบนจอภาพ *(3 คะแนน)*

//เมท็อด **showName(…)** แบบที่มีพารามิเตอร์ 2 ตัวที่รับคำนำหน้าชื่อและชื่อนิสิตชื่อใดๆ เข้ามาเพื่อใช้แสดงคำนำหน้าชื่อตามด้วยชื่อของนิสิตคนนั้นขึ้นบนจอภาพ *(3 คะแนน)*

//เมท็อด **showName(…)** แบบที่มีพารามิเตอร์ 2 ตัวที่รับชื่อนิสิตชื่อใดๆ และจำนวนเต็ม x อีกหนึ่งค่าเพื่อใช้แสดงชื่อของนิสิตคนนั้นจำนวน x ครั้งขึ้นบนจอภาพ *(3 คะแนน)*

```
} //End of class
```

© สงวนลิขสิทธิ์ พฤศจิกายน 2553 (ปรับปรุงครั้งที่ 7 ฉบับใช้ติวภาค 2/2553) เรียบเรียงโดย วงศ์ยศ เกิดศรี (แบงค์)