public String toString( )

สมชาย ประสิทธิ์จูตระกูล

public String toString( )

- เป็นเมท็อดที่มีไว้เรียกกับออปเจกต์เพื่อคืนสตริงที่แทนข้อมูลและสภาวะต่างๆ ของออปเจกต์นั้น

```
Point p = new Point(2, 3);
System.out.println(p.toString());
```

```
java.awt.Point[x=2,y=3]
```

```
ArrayList x = new ArrayList();
x.add("kid");
x.add("spj");
x.add("nat");
System.out.println(x.toString());
```

```
[kid, spj, nat]
```

toString()

- ถ้าส่งออปเจกต์ให้ print( ) และ println( ), ตัว print และ println จะเรียก toString( ) ของออปเจกต์ที่ได้รับ
- ถ้าเรานำออปเจกต์มา + กับสตริง ระบบจะเรียก toString( ) ของออปเจกต์นั้น

```
Point p = new Point(2, 3);
System.out.println(p);
```

```
java.awt.Point[x=2,y=3]
```

```
Point p = new Point(2, 3);
String s = "" + p;
System.out.println(s);
```

```
java.awt.Point[x=2,y=3]
```

ตัวอย่าง : toString( ) ของ Point

```
public class Point {
  public int x;
  public int y;
  . . .
  public String toString() {
    return getClass().getName() +
           "[x=" + x + ",y=" + y + "]";
  }
  . . .
}
```

```
System.out.println(new Point(2, 3));
System.out.println(new Point(12, 52));
```

```
java.awt.Point[x=2,y=3]
java.awt.Point[x=12,y=52]
```

ถ้าคลาสไม่มี toString( )

```
public class Test {
  public int x;
  public int y;

  public static void main(String[] args) {
    System.out.println(new Test());
    System.out.println(new Test());
  }
}
```

```
Test@2bbd86
Test@1a7bf11
```

Test ไม่มี toString( ) แต่บรรพบุรุษ (คลาส Object) มี

public String toString( )

- เป็นเมท็อดหนึ่งของคลาส Object
  - คืนสตริง ชื่อคลาส + “@” + hashCode()

```
class Object {
  . . .
  public String toString() {
    return getClass().getName() + "@" +
           Integer.toHexString(hashCode());
  }
}
```

- toString( ) ของ Object ไม่สื่อความหมาย
- ดังนั้น คลาสทั่วไปควร override toString( ) เพื่อความสะดวกในการ debug โปรแกรม

ทำไมต้อง getClass().getName() ?

```
public class Point {
  private int x, y;
  public String toString() {
    return getClass().getName() +
           "[x=" + x + ",y=" + y + "]";
  }
  ...
}
```

```
public class Point {
  private int x, y;
  public String toString() {
    return "Point" +
           "[x=" + x + ",y=" + y + "]";
  }
  ...
}
```

ทำไมต้อง getClass().getName() ?

```
public class Point {
  private int x, y;
  public String toString() {
    return "Point" + "[x=" + x + ",y=" + y + "]";
  }
  ...
}
```

```
public class ColorPoint extends Point {
  private Color color;
  ...
}
```

```
ColorPoint p = new ColorPoint(3, 2, Color.RED);
System.out.println( p );
```

p.toString()

Point[x=3,y=2]

ColorPoint[x=3,y=2]

รูปแบบของผลลัพธ์ที่ได้จาก toString( )

- บางคนใช้ toString( ) เพื่อเข้าถึงข้อมูลของออปเจกต์
- ถ้าเราไม่ต้องการให้ทำเช่นนั้น
  - เขียนไว้ให้ชัดเจนใน javadoc ว่า “อย่าทำ”
  - ให้บริการ accessor methods

```
public class Point3D {
  private int x, y, z;
  /*
   * Returns a brief description of this 3d-point.
   * The format of the returned string is subject to change.
   */
  public String toString() {
    return "[x=" + x + ",y=" + y + ",z=" + z + "]";
  }
  public int getX() { return x; }
  public int getY() { return y; }
  public int getZ() { return z; }
  . . .
```

ถ้าผลของ toString( ) มีรูปแบบแน่นอน

```
public class Point3D {
  private int x, y, z;
  /**
   * Returns the string representation of this 3d-point.
   * The format is "(X,Y,Z)" where X, Y, and Z are comma
   * separated, and can be any number of digits.
   */
  public String toString() {
    return "(" + x + "," + y + "," + z + ")";
  }
  /**
   * Constructs a newly allocated Point3D object that
   * represents 3d-point indicated by the parameter whose
   * fo
   * @p                                          object.
   * @e
   *       if the String does not contain a parsable 3d-point.
   */
  public Point3D(String s)
         throws IllegalArgumentException {...}
```

ควรระบุรูปแบบนั้นไว้ใน javadoc

มี constructor ที่รับสตริงในรูปแบบดังกล่าว

```
Point3D p1 = new Point3D();
...
Point3D p2 = new Point3D( p1.toString() );
```

สรุป

- คลาส Object มี toString() ที่ไม่สื่อความหมาย
- ไม่มีข้อบังคับใดๆ ว่าคลาสใหม่ต้อง override toString
- แต่เราควร override toString() เมื่อเขียนคลาสใหม่
- toString() มีไว้ใช้แสดงข้อมูลที่สำคัญของออปเจกต์
  - เรียกให้ เมื่อมีการบวกสตริงกับออปเจกต์
  - เรียกให้ เมื่อส่งออปเจกต์ไป print
  - เรียกให้ เมื่อใช้ในคำสั่ง assert
- อย่าลืมใช้ getClass().getName()
- ต้องเขียนใน javadoc ให้ชัดเจนว่า ผลลัพธ์ที่ได้จาก toString() มีรูปแบบที่แน่นอนหรือไม่