Static Fields and Methods

zyBook Chap 9.12

Non-Static vs. Static Fields

- **Non-static** fields
  - A.k.a. **instance variables**
  - Attributes/Properties/Fields of an object

- **Static** fields
  - A.k.a. **class variables**
  - Information **shared** by **all instances** of this class

Recap: An object is an instance of a class

Static Fields

Static field → a field of the class instead of a field of each class object

- Declared and initialized in the class
- Shared by all instances of the class
- **Independent of any class object**
- **Public** static field can be accessed without creating a class object: `<ClassName>.<fieldName>`
  - E.g., Math.PI

```java
public class Robot {
    // Non-static field / instance variables
    private double posX;
    private double posY;
    private int id;

    // Static field / class variable
    public static int nextRobotID = 1;

    public Robot(double posX, double posY) {
        this.posX = posX;
        this.posY = posY;
        id = nextRobotID;

        ++nextRobotID;
    }

    public String toString() {
        return "r" + id + ": (" +
                posX + ", " + posY + ")";
    }
}
```

Example

```java
public class Robot {
    // Non-static field / instance variables
    private double posX;
    private double posY;
    private int id;

    // Static field / class variable
    public static int nextRobotID = 1;

    public Robot(double posX, double posY) {
        this.posX = posX;
        this.posY = posY;
        id = nextRobotID;

        ++nextRobotID;
    }

    public String toString() {
        return "r" + id + ": (" +
                posX + ", " + posY + ")";
    }
}
```

```java
import java.util.Arrays;

public class RobotClient {
    public static void main(String[] args) {
        // Array of Objects - Two-phase initialization
        Robot[] r = new Robot[5];
        for (int i = 0; i < r.length; ++i){
            r[i] = new Robot(i, i);
            System.out.println("Constructed robot " + r[i]);
            System.out.println("The ID of next robot is "
                                   + Robot.nextRobotID);
        }
    }
}
```

```
$ javac RobotClient.java
$ java RobotClient
Constructed robot r1: (0.0, 0.0)
The ID of next robot is 2
```

r[0] = new Robot (0, 0)

That is,
r[0].posX == 0.0
r[0].posY == 0.0
r[0].id == 1

Example

```java
public class Robot {
    // Non-static field / instance variables
    private double posX;
    private double posY;
    private int id;

    // Static field / class variable
    public static int nextRobotID = 1;

    public Robot(double posX, double posY) {
        this.posX = posX;
        this.posY = posY;
        id = nextRobotID;

        ++nextRobotID;
    }

    public String toString() {
        return "r" + id + ": (" +
                posX + ", " + posY + ")";
    }
}
```

```java
import java.util.Arrays;

public class RobotClient {
    public static void main(String[] args) {
        // Array of Objects – Two-phase initialization
        Robot[] r = new Robot[5];
        for (int i = 0; i < r.length; ++i){
            r[i] = new Robot(i, i);
            System.out.println("Constructed robot " + r[i]);
            System.out.println("The ID of next robot is "
                                   + Robot.nextRobotID);
        }
    }
}
```

```
$ javac RobotClient.java
$ java RobotClient
Constructed robot r1: (0.0, 0.0)
The ID of next robot is 2
Constructed robot r2: (1.0, 1.0)
The ID of next robot is 3
Constructed robot r3: (2.0, 2.0)
The ID of next robot is 4
Constructed robot r4: (3.0, 3.0)
The ID of next robot is 5
Constructed robot r5: (4.0, 4.0)
The ID of next robot is 6
```

Static Methods vs. Non-Static (Instance) Methods

Static member method → a class method that is **independent of class objects**.

- Typically used to and **can only access** and **mutate** the **private static fields** from outside the class.

```java
public class Robot {
    // Non-static field / instance variables
    private double posX;
    private double posY;
    private int id;

    // Static field / class variable
    private static int nextRobotID = 1;

    public Robot(double posX, double posY) {
        this.posX = posX;
        this.posY = posY;
        id = nextRobotID;

        ++nextRobotID;
    }

    // Non-static method / Instance method
    public int getID() {
        return id;
    }

    // Static method
    public static int getNextRobotID() {
        return nextRobotID;
    }

    public String toString() {
        return "r" + id + ": (" +
                posX + ", " + posY + ")";
    }
}
```

```java
import java.util.Arrays;

public class RobotClient {
    public static void main(String[] args) {
        // Array of Objects - Two-phase initialization
        Robot[] r = new Robot[5];
        for (int i = 0; i < r.length; ++i){
            r[i] = new Robot(i, i);
            System.out.println("Constructed robot #" + r[i].getID());
            System.out.println("The ID of next robot is "
                                + Robot.getNextRobotID());
        }
    }
}
```

```
$ javac RobotClient.java
$ java RobotClient
Constructed robot #1
The ID of next robot is 2
Constructed robot #2
The ID of next robot is 3
Constructed robot #3
The ID of next robot is 4
Constructed robot #4
The ID of next robot is 5
Constructed robot #5
The ID of next robot is 6
```

Example

```java
public class Robot {
    // Non-static field / instance variables
    private double posX;
    private double posY;
    private int id;

    // Static field / class variable
    private static int nextRobotID = 1;

    public Robot(double posX, double posY) {
        this.posX = posX;
        this.posY = posY;
        id = nextRobotID;

        ++nextRobotID;
    }

    // Non-static method / Instance method
    public int getID() {
        return id;
    }

    // Static method
    public static int getNextRobotID() {
        return nextRobotID;
    }

    public String toString() {
        return "r" + id + ": (" +
               posX + ", " + posY + ")";
    }
}
```