

Chapter 2 – Using Objects

- 1. Types**
- 2. Variables & Identifiers**
- 3. Objects, Classes and Method**
- 4. Parameters**
- 5. Constructing Objects**
- 6. Accessor and Mutator Methods**
- 7. Packages**
- 8. Example**

Types

- A **type** defines a set of values and the operations that can be carried out on the values
 - Examples:
 - *13 has type int*
 - *"Hello, World" has type String*
- A **type defines** a set of values and the **operations** that can be **carried** out on the values

Number types are primitive types

Numbers are not objects

Variables

- Use a **variable** to store a value that you want to use at a later time
- A variable has a **type**, a **name**, and a **value**:

```
String greeting = "Hello, World!"  
PrintStream printer = System.out;  
int width = 13;
```

Identifiers

- **name** of a variable, method, or class
 - ✓ *Cannot start with a digit*
 - ✓ *Cannot use other symbols such as ? or %*
 - ✓ *Spaces are not permitted inside identifiers*
 - ✓ *You cannot use reserved words such as public*
 - ✓ *They are case sensitive*

Objects, Classes and Method

- **Object**: entity that you can manipulate in your programs (by calling methods)
 - Each object **belongs** to a **class**
- **Class**: declares the methods that you can apply to its objects
 - Class determines legal methods
- **Method**: sequence of instructions that accesses the data of an object
 - You manipulate objects by calling its methods

Overloaded method: when a class declares two methods with the same name, but different parameters

Parameters

- **Parameter:** an input to a method

- **Implicit parameter:** the object on which a method is invoked:

```
greeting.length();
```

- **Explicit parameters:** all parameters except the implicit parameter:

```
System.out.println(greeting)
```

- **Not all methods have explicit parameters:**

```
greeting.length() // has no explicit parameter
```

Constructing Objects

```
new Rectangle(5, 10, 20, 30)
```

- Detail:

- 1.The *new* operator makes a *Rectangle* object

- 2.It *uses* the *parameters* (in this case, 5, 10, 20, and 30) to initialize the data of the object

- 3.It returns the object

- Usually the **output** of the **new** operator is **stored** in a **variable**:

```
Rectangle box = new Rectangle(5, 10, 20,  
30);
```

Accessor and Mutator Methods

- **Accessor method:** does **not change** the state of its **implicit parameter**:

```
double width = box.getWidth();
```

- **Mutator method:** **changes** the state of its **implicit parameter**:

```
box.translate(15, 25);
```

implicit parameter : “the object on which the method is invoked ”

Packages

- **Package**: a **collection** of **classes** with a related purpose
- **Import** library classes by **specifying** the package and class name:

```
import java.awt.Rectangle;
```

You don't need to import classes in the java.lang package such as String and System

Syntax `import packageName.ClassName;`

Example

Package name Class name

```
import java.awt.Rectangle;
```

Import statements must be at the top of the source file.

You can look up the package name in the API documentation.

Example

```
1  import java.awt.Rectangle;
2
3  public class MoveTester
4  {
5      public static void main(String[] args)
6      {
7          Rectangle box = new Rectangle(5, 10, 20, 30);
8
9          // Move the rectangle
10         box.translate(15, 25);
11
12         // Print information about the moved rectangle
13         System.out.print("x: ");
14         System.out.println(box.getX());
15         System.out.println("Expected: 20");
16
17         System.out.print("y: ");
18         System.out.println(box.getY());
19         System.out.println("Expected: 35");
20     }
21 }
```