Building Java Programs

Chapter 3:
Parameters, Return, and Interactive Programs

Lecture 3-1: Parameters
(reading: 3.1)

Lecture outline

- parameters
  - passing parameters to static methods
  - writing methods that accept parameters

Parameters

reading: 3.1

self-check: #1-6
exercises: #1-3

Redundant recipes

- Recipe for baking **20** cookies:
  - Mix the following ingredients in a bowl:
    - **4** cups flour
    - **1** cup butter
    - **1** cup sugar
    - **2** eggs
    - **1** bag chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

- Recipe for baking **40** cookies:
  - Mix the following ingredients in a bowl:
    - **8** cups flour
    - **2** cups butter
    - **2** cups sugar
    - **4** eggs
    - **2** bags chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

Parameterized recipe

- Recipe for baking **40** cookies:
  - Mix the following ingredients in a bowl:
    - **8** cups flour ...

- Recipe for baking **N** cookies:
  - Mix the following ingredients in a bowl:
    - **N/5** cups flour
    - **N/20** cups butter
    - **N/20** cups sugar
    - **N/10** eggs
    - **N/20** bags chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

- **parameter**: A value that distinguishes similar tasks.

Redundant figures

- Consider the task of printing the following lines/boxes:

```
************

******

********************************

*********
*        *
*********

****
*    *
*    *
****
```

A redundant solution

```
public class Stars1 {
    public static void main(String[] args) {
        drawLineOf13Stars();
        drawLineOf7Stars();
        drawLineOf35Stars();
        draw10x3Box();
        draw5x4Box();
    }

    public static void drawLineOf13Stars() {
        for (int i = 1; i <= 13; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void drawLineOf7Stars() {
        for (int i = 1; i <= 7; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void drawLineOf35Stars() {
        for (int i = 1; i <= 35; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
    ...
```

- This code is redundant.

- Would constants help?

- What is a better solution?

  - drawLine - A method to draw a line of any number of stars.
  - drawBox - A method to draw a box of any size.

7

Parameterization

- **parameterized method**: One that is given extra information (e.g. # of stars to draw) when it is called.

- **parameter**: A value passed to a method by its caller.

- Example:
  - Instead of `drawLineOf7Stars`, `drawLineOf13Stars`, ...
  - Write `drawLineOfStars` method to draw a line of any length.
    - When the method is run, we will specify how many stars to draw.

Parameterization

- Writing a parameterized method requires 2 steps:
  - *Declare* the method to accept the parameter
  - *Call* the method and pass the parameter value(s) desired

```
              7
  main  ─────────────►  drawLine  ─────────────►  *******

         13
            ─────────►  drawLine  ─────────────►  *************
```
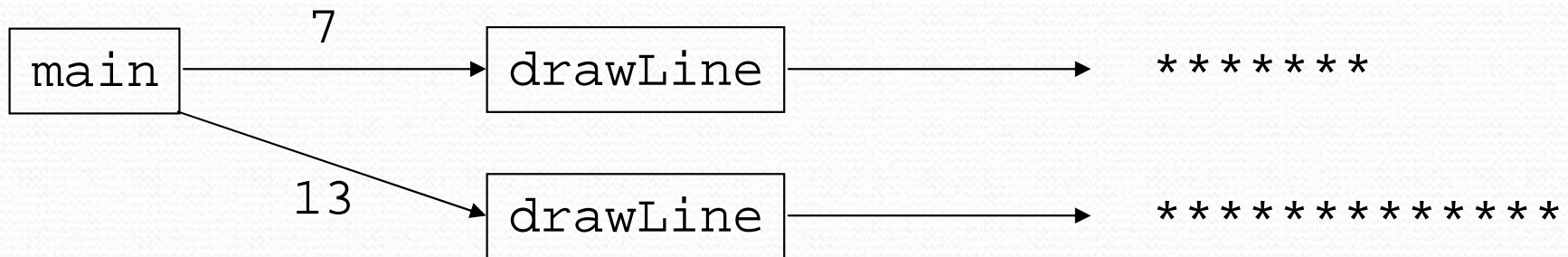
Declaring parameters

```
public static void <name> ( <type> <name> ) {
    <statement(s)> ;

}
```

- Example:
```
public static void printDots(int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(".");
    }
}
```

  - When `printDots` is called, the caller must specify how many spaces to print.

Passing parameters

<name> (**_<expression>_**);

- **passing a parameter**: Calling a parameterized method and specifying a value for its parameter(s).

- Example:

```
System.out.print("*");
printDots(7);
System.out.print("**");
int x = 3 * 5;
printDots(x + 2);
System.out.println("***");
```
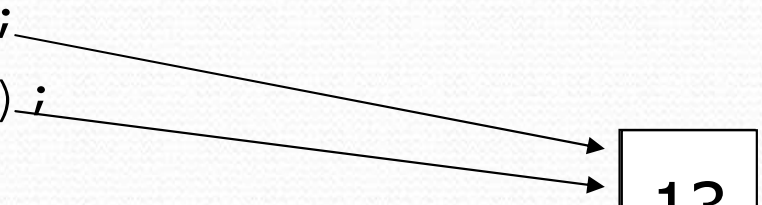
Output:

```
*.........**..................***
```

How parameters are passed

- When the parameterized method is called:
  - The value is stored into the parameter variable.
  - The method's code executes using that value.

```
public static void main(String[] args) {
    printDots(7);
    printDots(13);
}

public static void printDots(int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(".");
    }
}
```

13

Common errors

- If a method accepts a parameter, it is illegal to call it without passing any value for that parameter.

    ```
    printDots();    // ERROR: parameter value required
    ```

- The value passed to a method must be of the correct type, matching the type of its parameter variable.

    ```
    printDots(3.7);    // ERROR: must be of type int
    ```

- Exercise: Change the Stars program to use a parameterized static method for drawing lines of stars.

Stars solution

```java
// Prints several lines of stars.
// Uses a parameterized method to remove redundancy.
public class Stars2 {
    public static void main(String[] args) {
        drawLine(13);
        drawLine(7);
        drawLine(35);
    }

    // Prints the given number of stars plus a line break.
    public static void drawLine(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

Multiple parameters

- A method can accept multiple parameters.
  - When calling it, you must pass values for each parameter.

- Declaration syntax:

```
public static void <name> ( <type> <name> ,
          <type> <name> , ..., <type> <name> ) {
    <statement(s)> ;
}
```

- Call syntax:
```
<name> ( <value> , <value> , ..., <value> );
```

Multiple parameters example

```
public static void main(String[] args) {
    printNumber(4, 9);
    printNumber(17, 6);
    printNumber(8, 0);
    printNumber(0, 8);
}

public static void printNumber(int number, int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}
```

Output:
```
444444444
171717171717

00000000
```

- Modify the Stars program to draw boxes using parameters.

Stars solution

```java
// Prints several lines and boxes made of stars.
// Third version with multiple parameterized methods.
public class Stars3 {
    public static void main(String[] args) {
        drawLine(13);
        drawLine(7);
        drawLine(35);
        System.out.println();
        drawBox(10, 3);
        drawBox(5, 4);
        drawBox(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void drawLine(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
```

17

Stars solution, cont'd.

```
...
    // Prints a box of stars of the given size.
    public static void drawBox(int width, int height) {
        drawLine(width);

        for (int i = 1; i <= height - 2; i++) {
            System.out.print("*");
            printSpaces(width - 2);
            System.out.println("*");
        }

        drawLine(width);
    }

    // Prints the given number of spaces.
    public static void printSpaces(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print(" ");
        }
    }
}
```

Value semantics

- **value semantics**: When primitive variables (`int`, `double`) are passed as parameters, their values are copied.
  - Modifying the parameter will not affect the variable passed in.

```
public static void main(String[] args) {
    int x = 23;
    strange(x);
    System.out.println("2. x = " + x);    // unchanged
    ...
}
```

```
public static void strange(int x) {
    x = x + 1;
    System.out.println("1. x = " + x);
}
```

Output:

1. x = 24
2. x = 23

Parameter "mystery" problem

- What is the output of the following program?

```
public class Mystery {
    public static void main(String[] args) {
        int x = 5, y = 9, z = 2;
        mystery(z, y, x);
        System.out.println(x + " " + y + " " + z);
        mystery(y, x, z);
        System.out.println(x + " " + y + " " + z);
    }

    public static void mystery(int x, int z, int y) {
        x++;
        y = x - z * 2;
        x = z + 1;
        System.out.println(x + " " + y + " " + z);
    }
}
```

Strings

- **string**: A sequence of text characters.

  ```
  String <name> = "<text>";
  String <name> = <expression>;
  ```

  - Examples:

  ```
  String name = "Marla Singer";

  int x = 3;
  int y = 5;
  String point = "(" + x + ", " + y + ")";
  ```

Strings as parameters

```java
public class StringParameters {
    public static void main(String[] args) {
        String teacher = "Helene";
        sayHello(teacher);
        sayHello("Marty");
    }

    public static void sayHello(String name) {
        System.out.println("Welcome, " + name);
    }
}
```

Output:
```
Welcome, Helene
Welcome, Marty
```

- Modify the Stars program to use string parameters.

Stars solution

```java
// Prints several lines and boxes made of stars.
// Fourth version with String parameters.
public class Stars4 {
    public static void main(String[] args) {
        drawLine(13);
        drawLine(7);
        drawLine(35);
        System.out.println();
        drawBox(10, 3);
        drawBox(5, 4);
        drawBox(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void drawLine(int count) {
        printString("*", count);
        System.out.println();
    }
    ...
```

Stars solution, cont'd.

```java
...
    // Prints a box of stars of the given size.
    public static void drawBox(int width, int height) {
        drawLine(width);

        for (int i = 1; i <= height - 2; i++) {
            System.out.print("*");
            printString(" ", width - 2);
            System.out.println("*");
        }

        drawLine(width);
    }

    // Prints the given number of spaces.
    public static void printString(String s, int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print(s);
        }
    }
}
```