Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

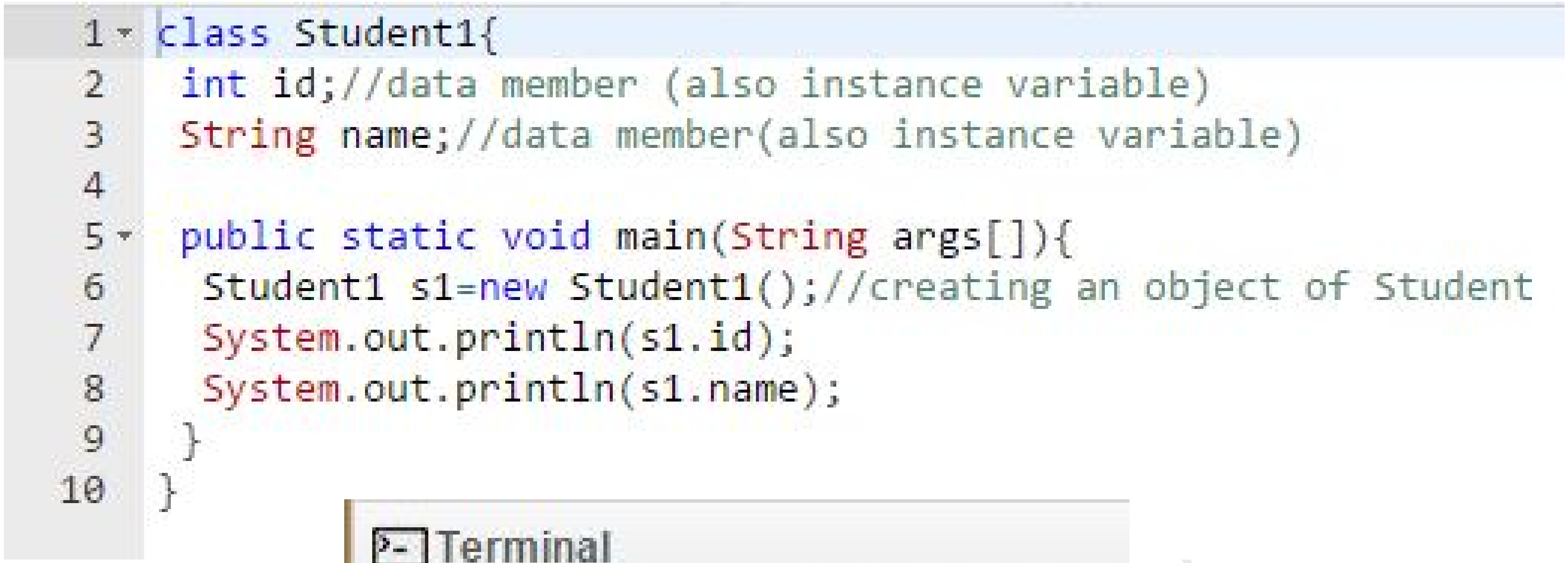
Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

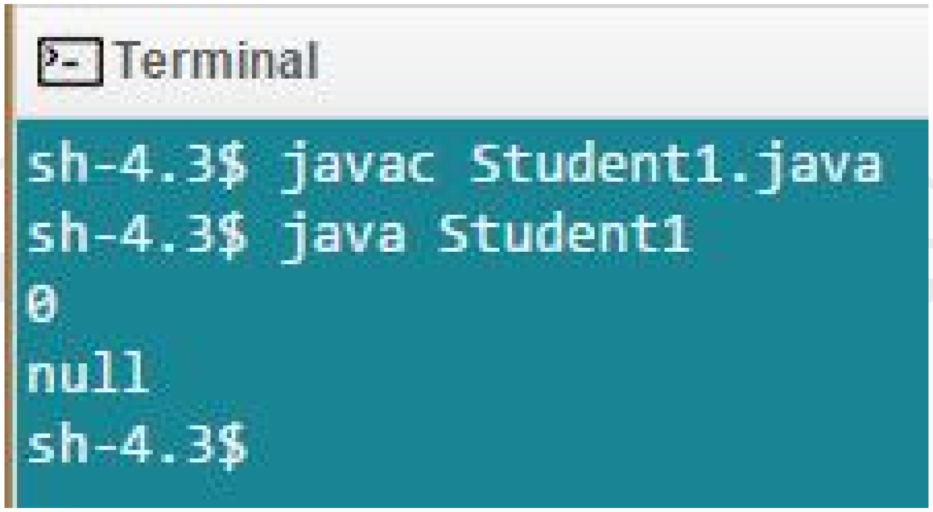
Roorkee 247 667

balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/



```java
class Student1{
 int id;//data member (also instance variable)
 String name;//data member(also instance variable)

 public static void main(String args[]){
  Student1 s1=new Student1();//creating an object of Student
  System.out.println(s1.id);
  System.out.println(s1.name);
 }
}
```



Terminal

```
sh-4.3$ javac Student1.java
sh-4.3$ java Student1
0
null
sh-4.3$
```

Methods Declaration

- A Java method is a collection of statements that are grouped together to perform an operation.
  - When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

- Now you will learn how to create your own methods with or without return values, invoke a method with or without parameters, and apply method abstraction in the program design.

Creating Method:

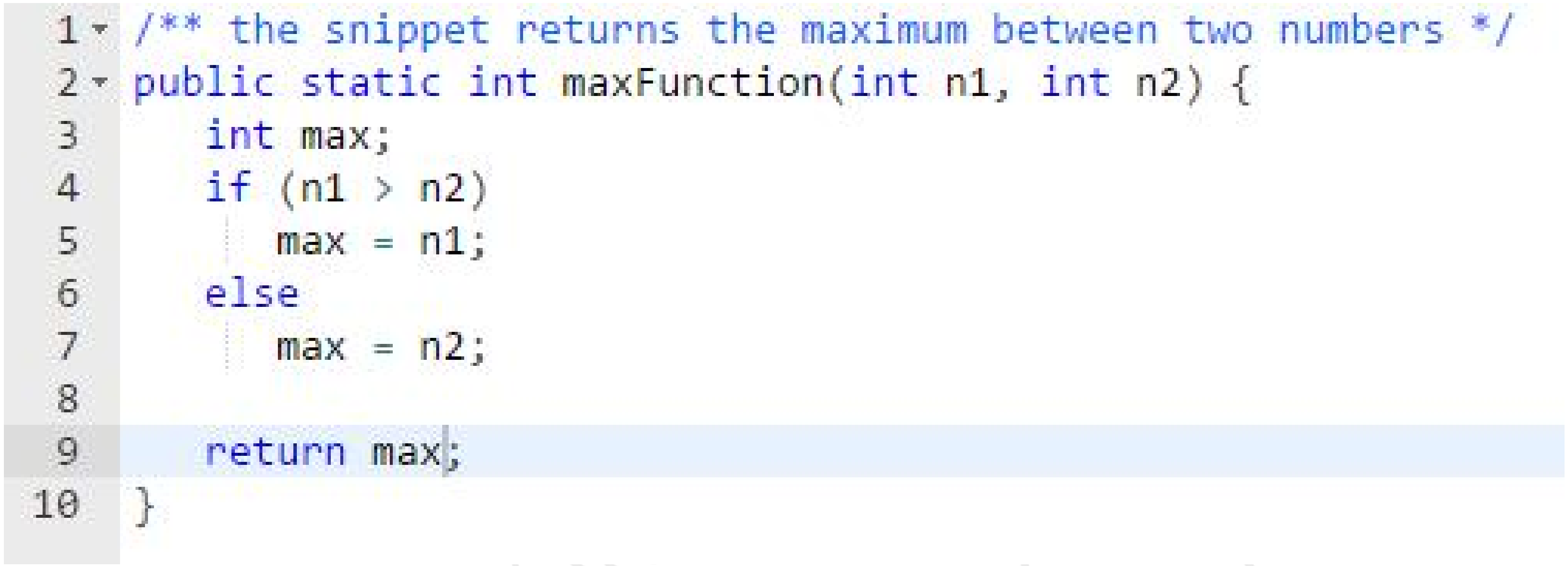
```
public static int Name_of_Method(int a, int b)
{ // body }
```

- Here,
  - **public static** : modifier.
  - **int**: return type
  - **Name_of_Method**: name of the method
  - **a, b**: formal parameters
  - **int a, int b**: list of parameters

Syntax

- `modifier returnType nameOfMethod (Parameter List) { // method body }`
- The syntax includes:
- **modifier:** It defines the access type of the method and it is optional to use.
- **returnType:** Method may return a value.
- **nameOfMethod:** This is the method name. The method signature consists of the method name and the parameter list.
- **Parameter List:** The list of parameters, it is the type, order, and number of parameters of a method. These are optional, method may contain zero parameters.
- **method body:** The method body defines what the method does with statements.

Example



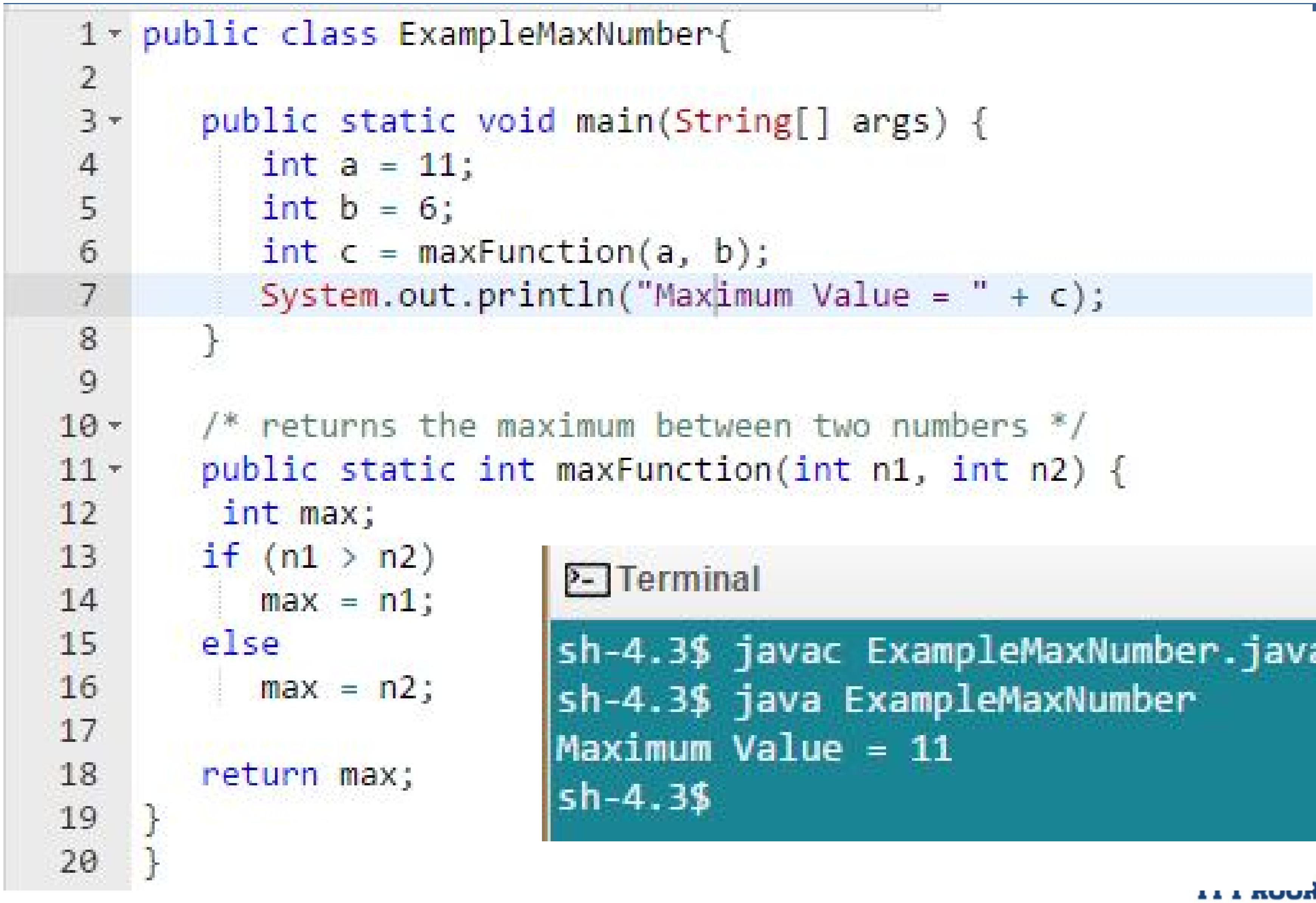
```
/** the snippet returns the maximum between two numbers */
public static int maxFunction(int n1, int n2) {
    int max;
    if (n1 > n2)
        max = n1;
    else
        max = n2;

    return max;
}
```

Method Calling

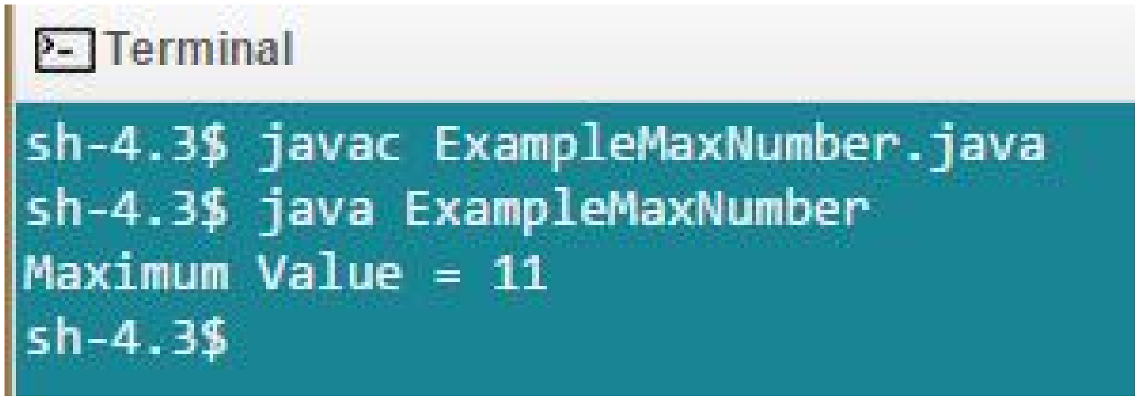
- For using a method, it should be called.
- There are two ways in which a method is called,

  i.e. method returns a value or returning nothing (no return value).

- The called method then returns control to the caller in two conditions, when:
  - return statement is executed.
  - reaches the method ending closing brace.

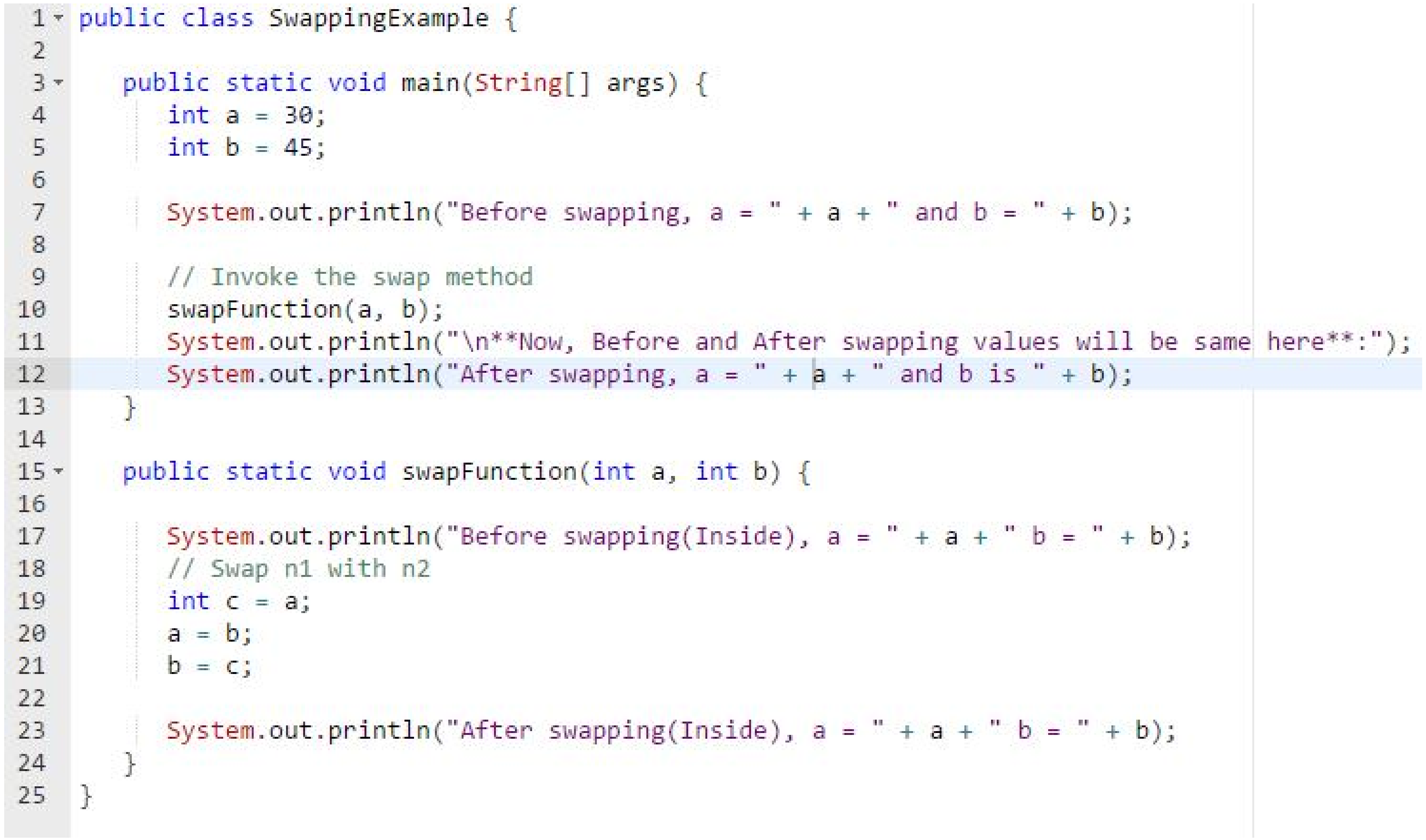


```java
public class ExampleMaxNumber{

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = maxFunction(a, b);
      System.out.println("Maximum Value = " + c);
   }

   /* returns the maximum between two numbers */
   public static int maxFunction(int n1, int n2) {
    int max;
   if (n1 > n2)
      max = n1;
   else
      max = n2;

   return max;
}
}
```



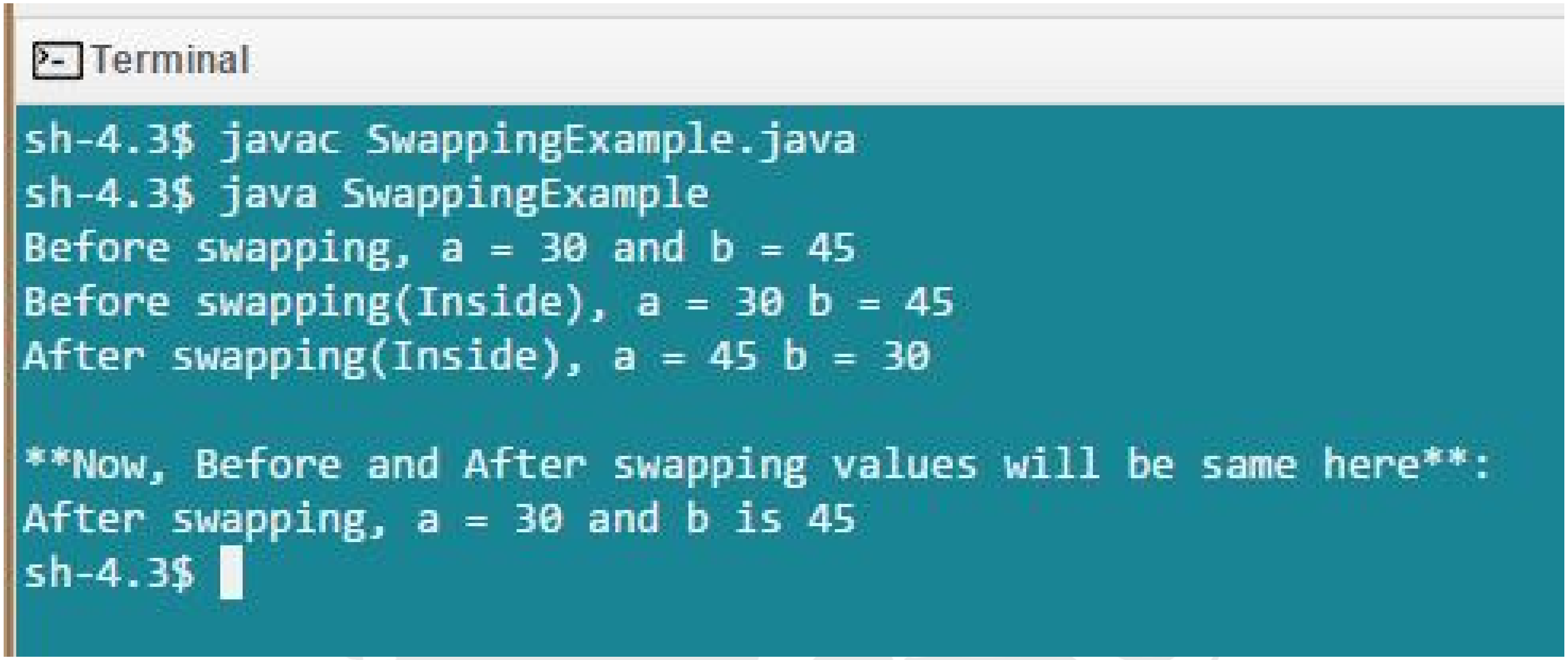
```
sh-4.3$ javac ExampleMaxNumber.java
sh-4.3$ java ExampleMaxNumber
Maximum Value = 11
sh-4.3$
```

The void Keyword and Call by Value



```java
public class SwappingExample {

    public static void main(String[] args) {
        int a = 30;
        int b = 45;

        System.out.println("Before swapping, a = " + a + " and b = " + b);

        // Invoke the swap method
        swapFunction(a, b);
        System.out.println("\n**Now, Before and After swapping values will be same here**:");
        System.out.println("After swapping, a = " + a + " and b is " + b);
    }

    public static void swapFunction(int a, int b) {

        System.out.println("Before swapping(Inside), a = " + a + " b = " + b);
        // Swap n1 with n2
        int c = a;
        a = b;
        b = c;

        System.out.println("After swapping(Inside), a = " + a + " b = " + b);
    }
}
```

Output

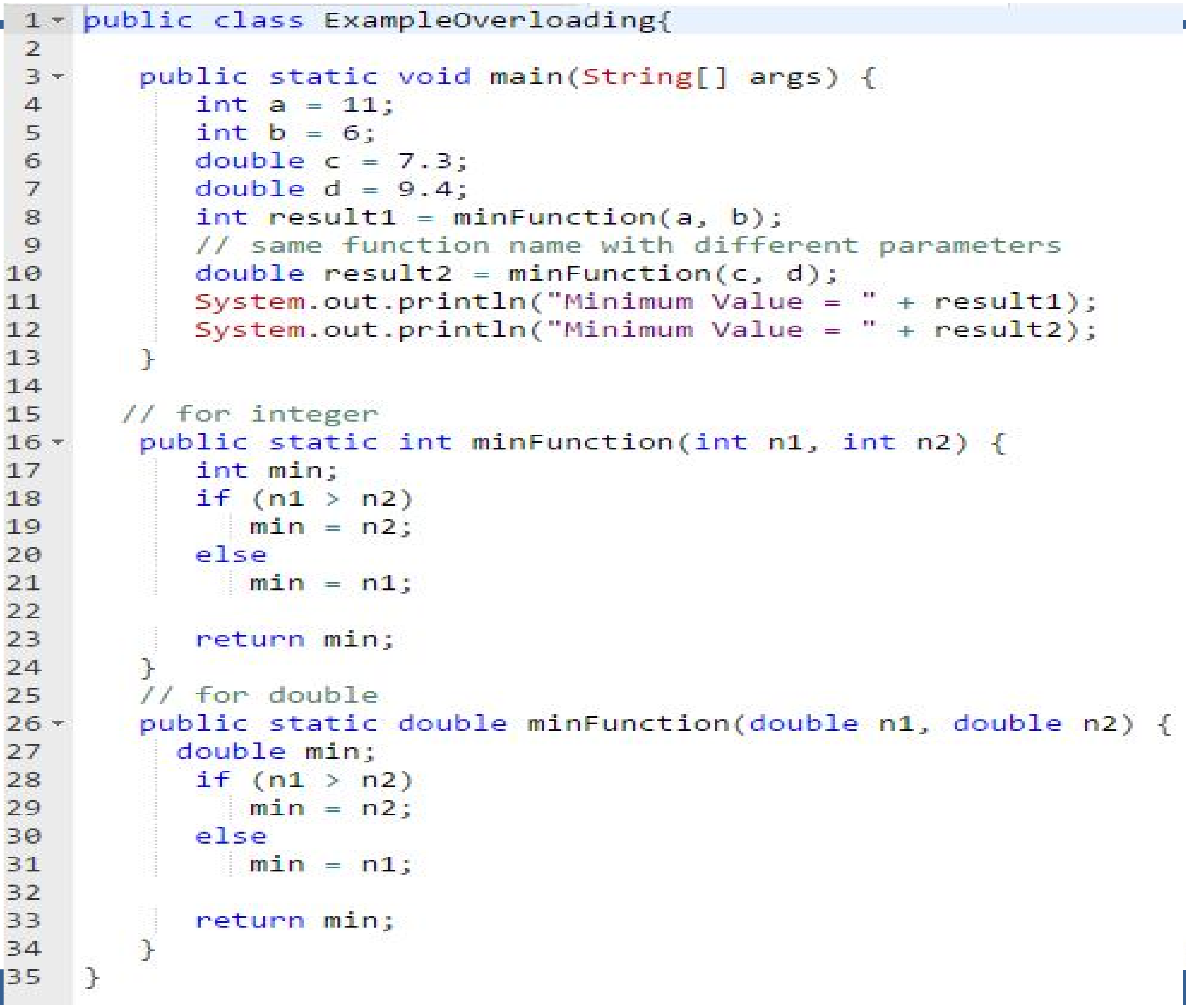


```
sh-4.3$ javac SwappingExample.java
sh-4.3$ java SwappingExample
Before swapping, a = 30 and b = 45
Before swapping(Inside), a = 30 b = 45
After swapping(Inside), a = 45 b = 30

**Now, Before and After swapping values will be same here**:
After swapping, a = 30 and b is 45
sh-4.3$ 
```

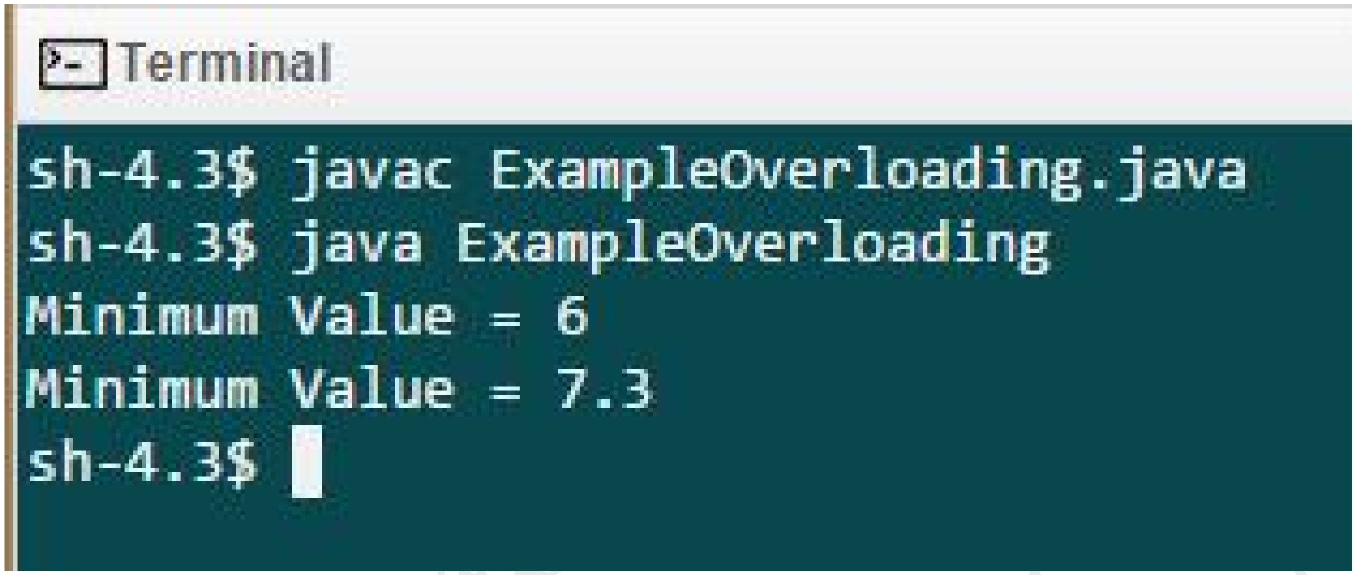
Call by Reference

- There is only call by value in JAVA, not call by reference.

Method Overloading



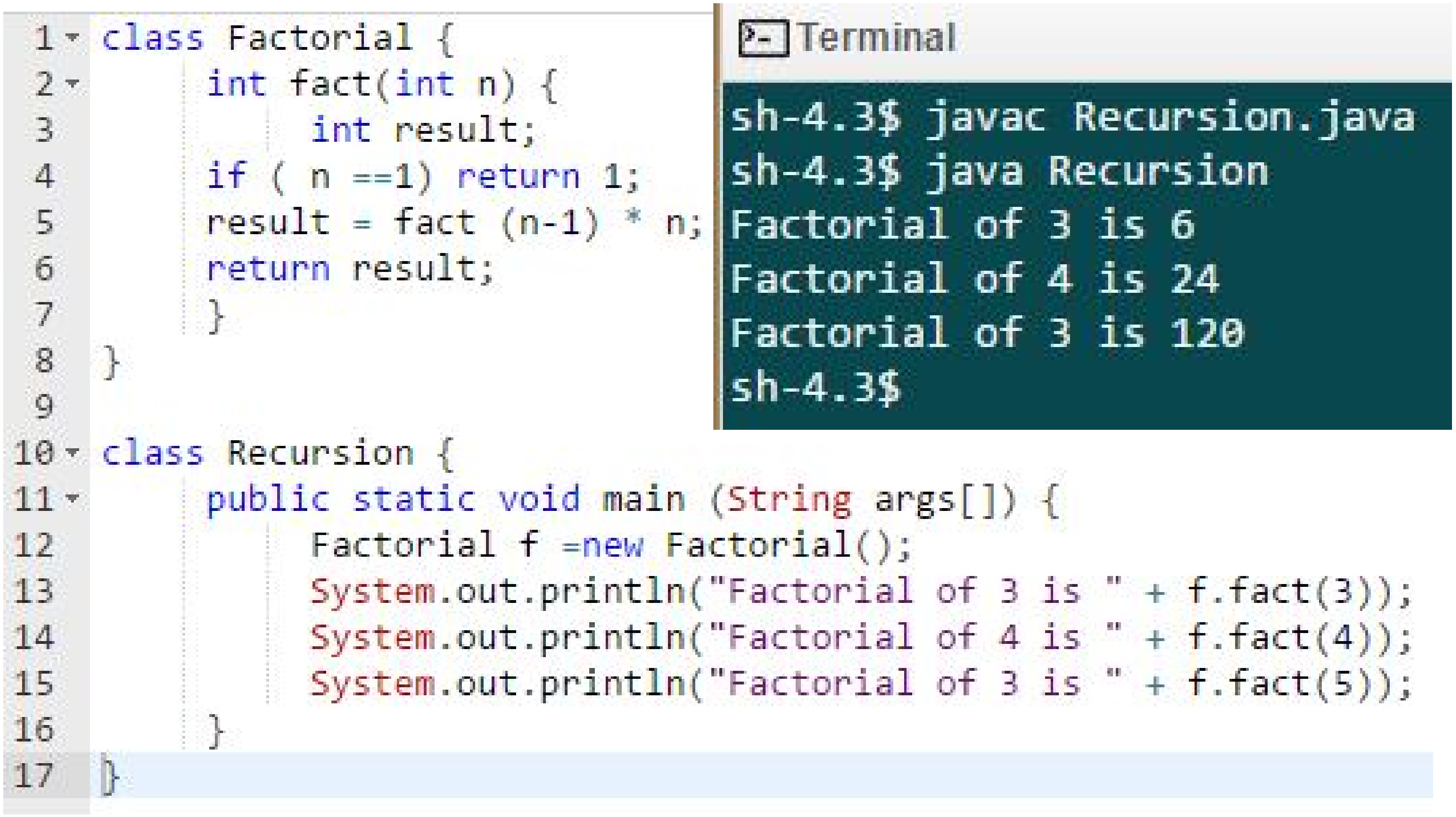
```java
public class ExampleOverloading{

    public static void main(String[] args) {
        int a = 11;
        int b = 6;
        double c = 7.3;
        double d = 9.4;
        int result1 = minFunction(a, b);
        // same function name with different parameters
        double result2 = minFunction(c, d);
        System.out.println("Minimum Value = " + result1);
        System.out.println("Minimum Value = " + result2);
    }

   // for integer
    public static int minFunction(int n1, int n2) {
        int min;
        if (n1 > n2)
           min = n2;
        else
           min = n1;

        return min;
    }
    // for double
    public static double minFunction(double n1, double n2) {
       double min;
        if (n1 > n2)
           min = n2;
        else
           min = n1;

        return min;
    }
}
```



```
sh-4.3$ javac ExampleOverloading.java
sh-4.3$ java ExampleOverloading
Minimum Value = 6
Minimum Value = 7.3
sh-4.3$
```

Recursion in JAVA

- Write a program to find a factorial of a given number using recursive method.



```java
class Factorial {
	int fact(int n) {
		int result;
	if ( n ==1) return 1;
	result = fact (n-1) * n;
	return result;
	}
}

class Recursion {
	public static void main (String args[]) {
		Factorial f =new Factorial();
		System.out.println("Factorial of 3 is " + f.fact(3));
		System.out.println("Factorial of 4 is " + f.fact(4));
		System.out.println("Factorial of 3 is " + f.fact(5));
	}
}
```

```
sh-4.3$ javac Recursion.java
sh-4.3$ java Recursion
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 3 is 120
sh-4.3$
```