Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

Department of Computer Science and Engineering

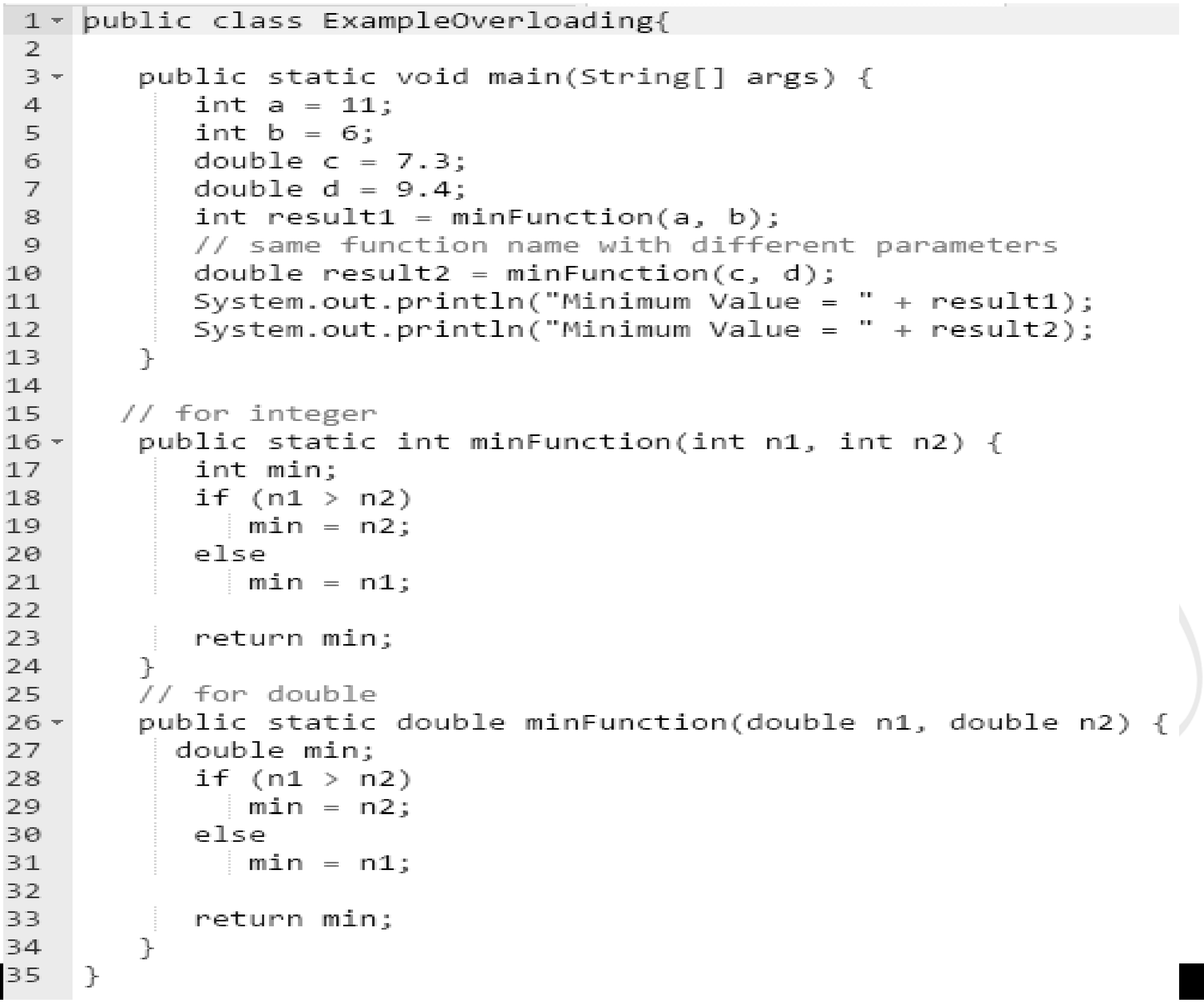
Indian Institute of Technology Roorkee

Roorkee 247 667

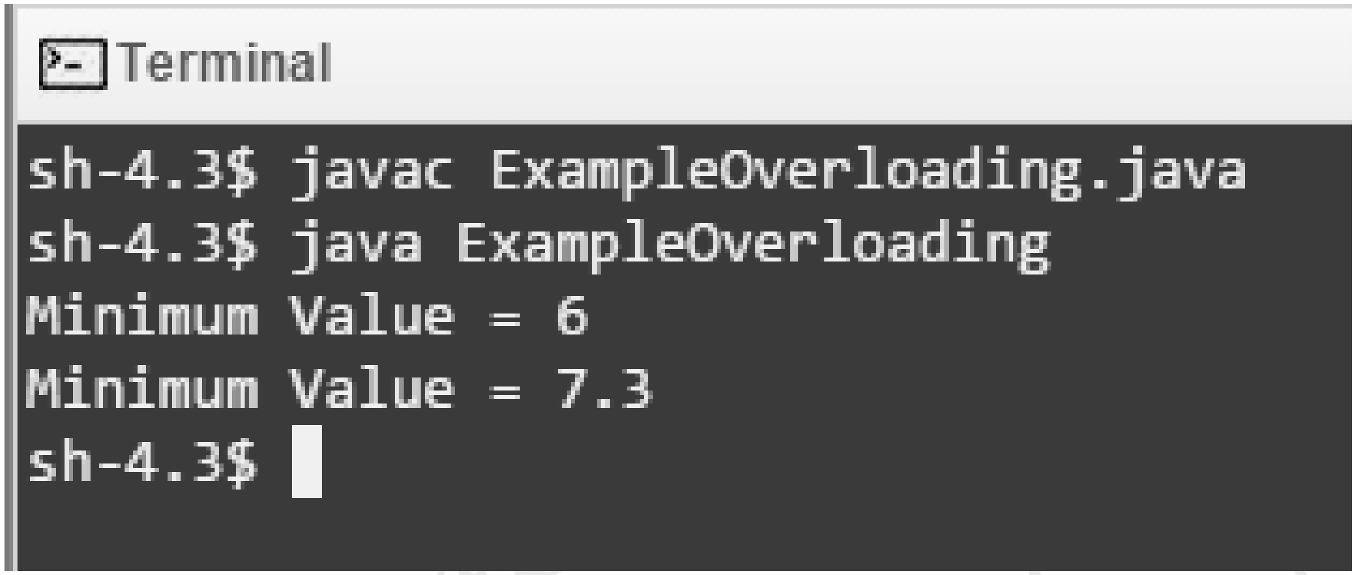
balarfcs@iitr.ac.in

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

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;
    }
}
```

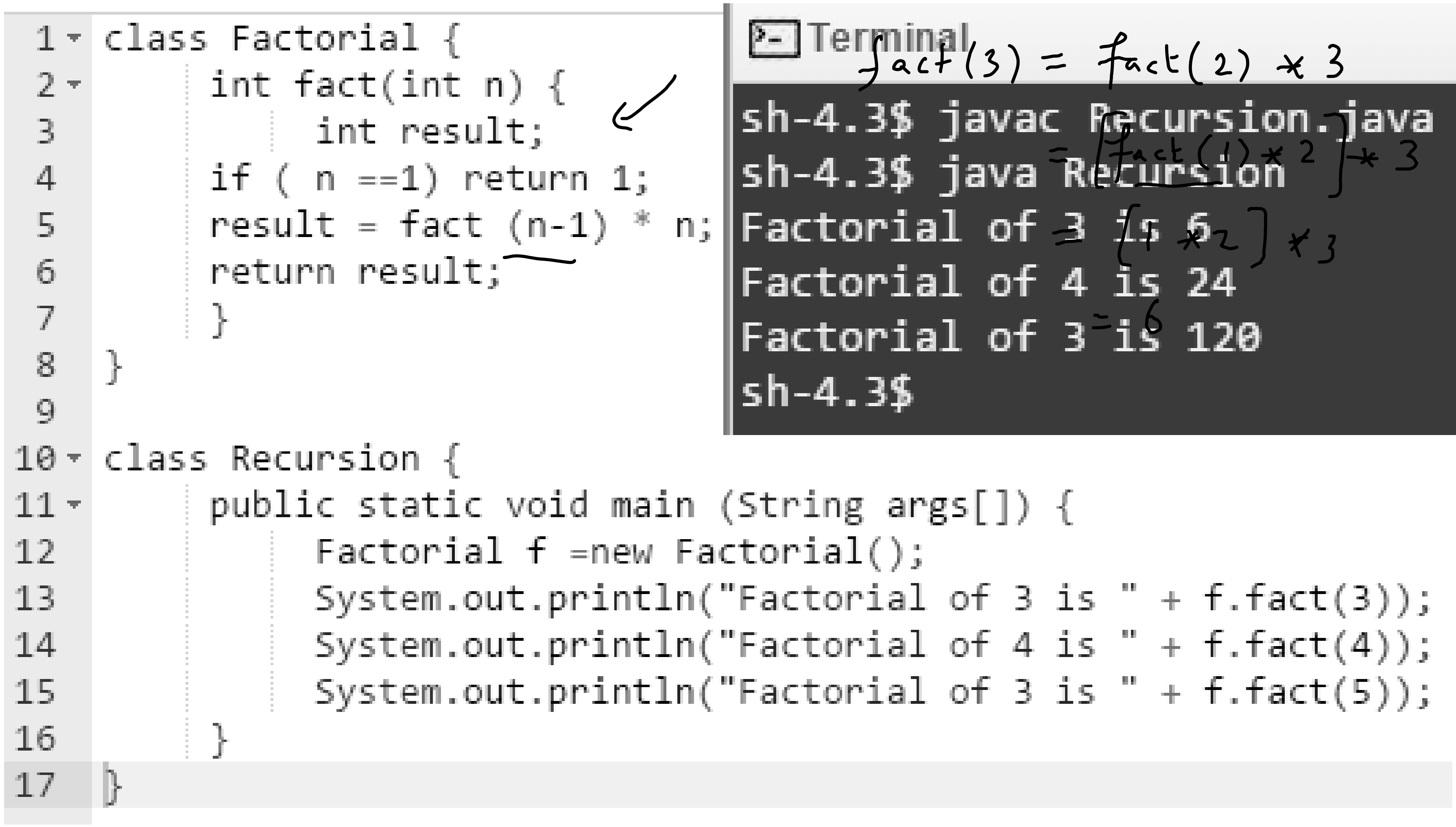


```
sh-4.3$ javac ExampleOverloading.java
sh-4.3$ java ExampleOverloading
Minimum Value = 6
Minimum Value = 7.3
sh-4.3$
```

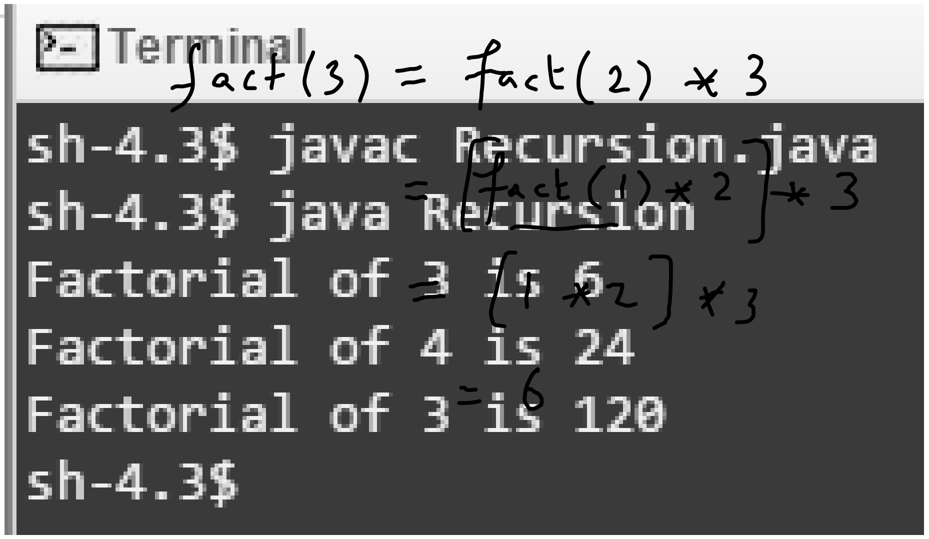
Recursion in JAVA

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

$n! = n * (n-1)!$



```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));
	}
}
```



Terminal

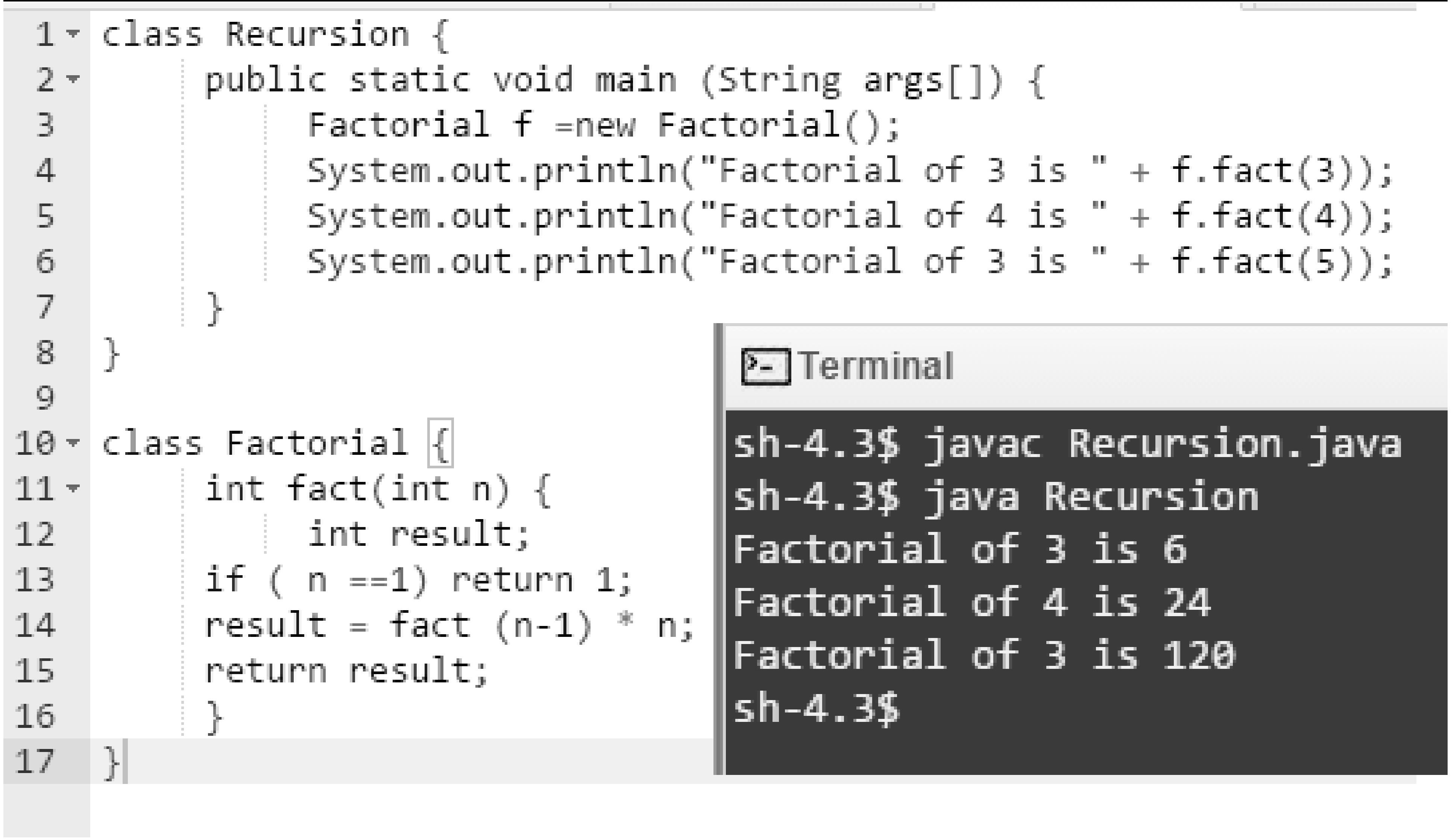
```
sh-4.3$ javac Recursion.java
sh-4.3$ java Recursion
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 3 is 120
sh-4.3$
```

$fact(3) = fact(2) * 3$

$= [fact(1) * 2] * 3$

$= [1 * 2] * 3$

$= 6$


```java
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));
	}
}

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

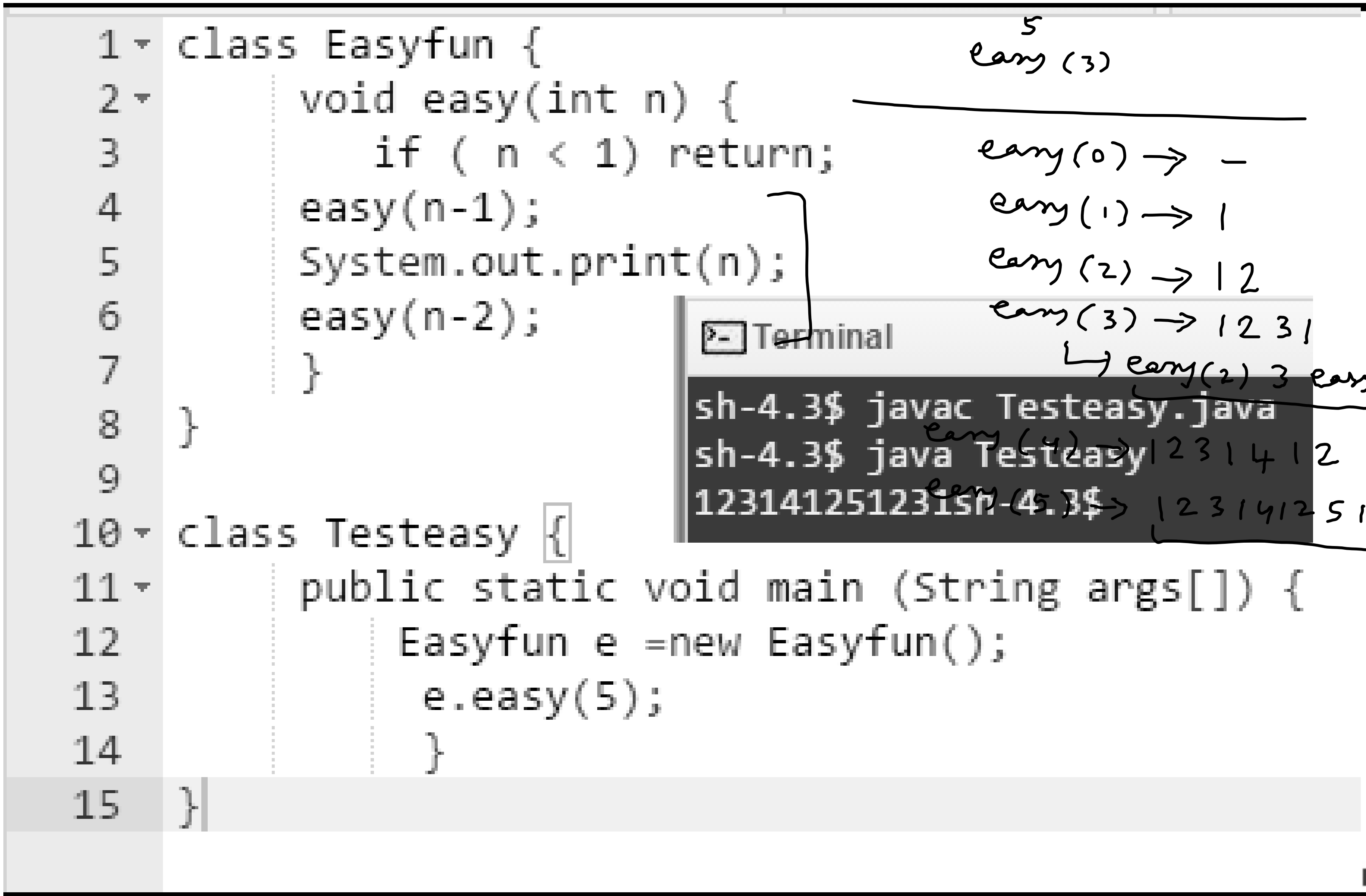
Terminal

```
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$
```

Recursion in JAVA

easy(5)
easy(4)
5
easy(3)

easy(0) → –
easy(1) → 1
easy(2) → 1 2
easy(3) → 1 2 3 1
↳ easy(2) 3 easy(1)
easy(4) → 1 2 3 1 4 1 2
easy(5) → 1 2 3 1 4 1 2 5 1 2 3 1

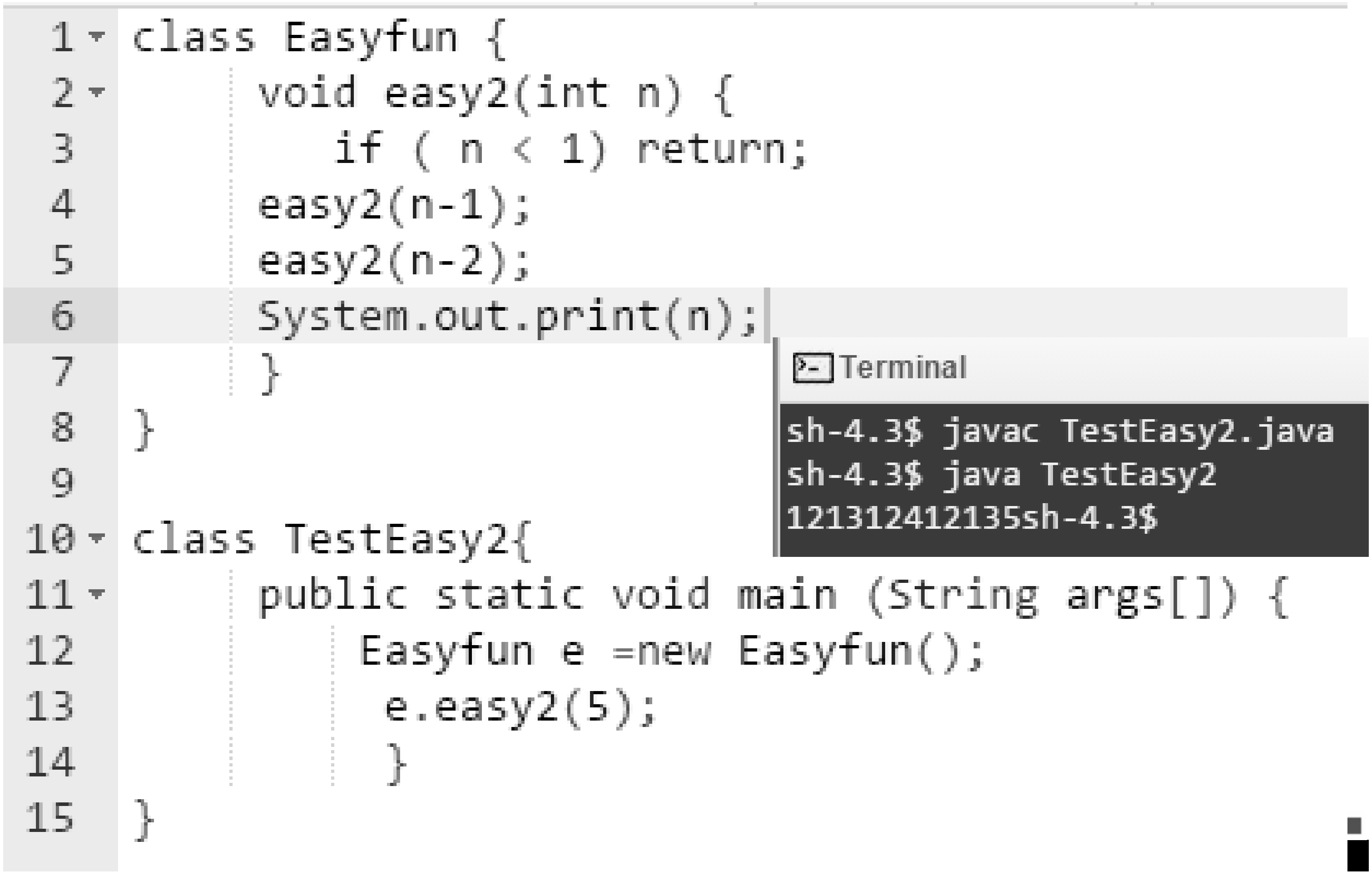


```
class Easyfun {
        void easy(int n) {
           if ( n < 1) return;
        easy(n-1);
        System.out.print(n);
        easy(n-2);
        }
}

class Testeasy {
        public static void main (String args[]) {
            Easyfun e =new Easyfun();
             e.easy(5);
             }
}
```

Terminal

```
sh-4.3$ javac Testeasy.java
sh-4.3$ java Testeasy
123141251231sh-4.3$
```

Recursion in JAVA

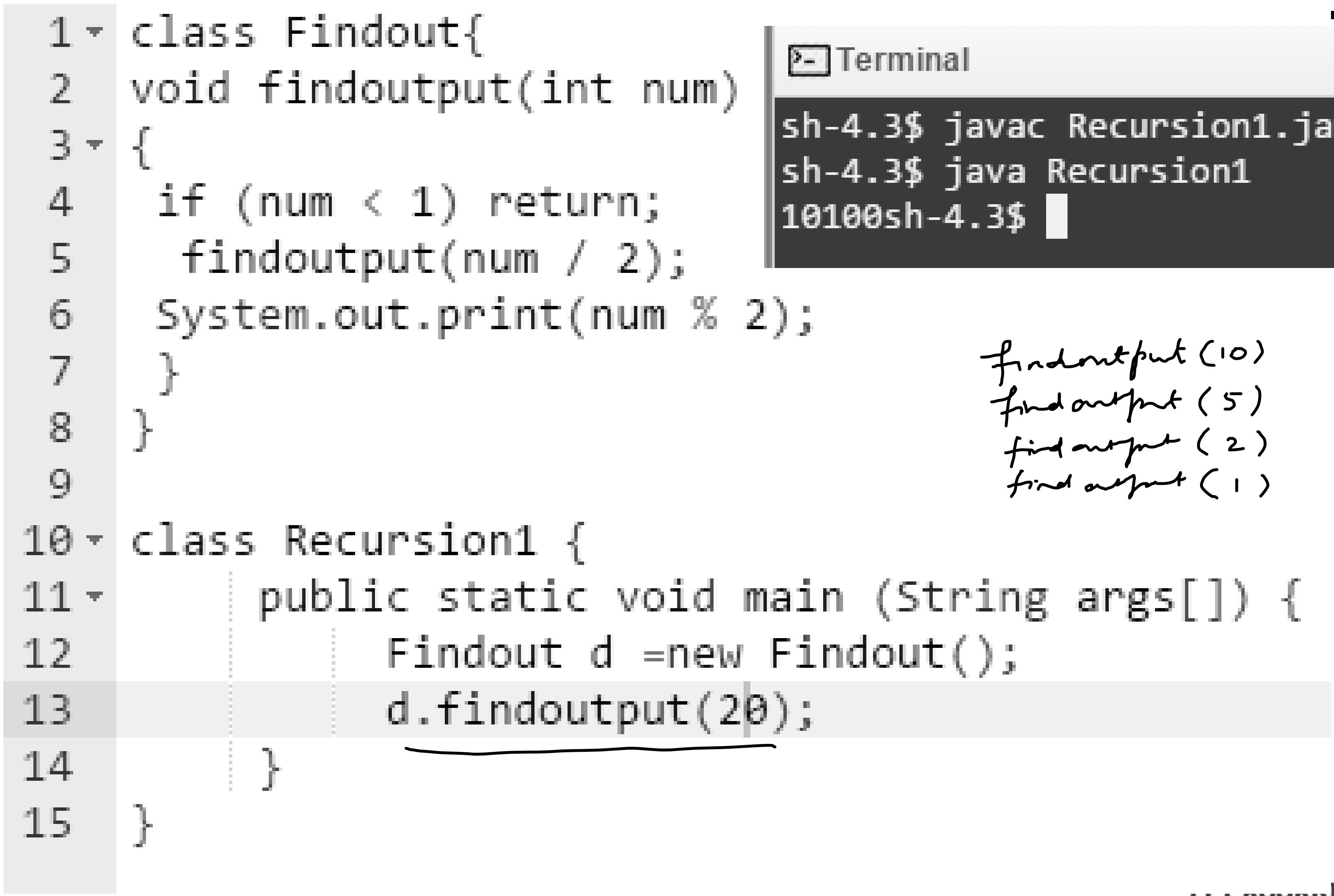


```
class Easyfun {
        void easy2(int n) {
           if ( n < 1) return;
        easy2(n-1);
        easy2(n-2);
        System.out.print(n);
        }
}

class TestEasy2{
        public static void main (String args[]) {
            Easyfun e =new Easyfun();
             e.easy2(5);
             }
}
```

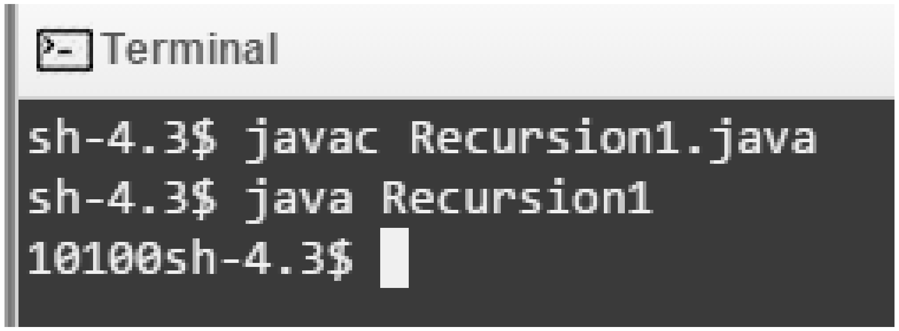
Terminal

```
sh-4.3$ javac TestEasy2.java
sh-4.3$ java TestEasy2
121312412135sh-4.3$
```

Recursion in JAVA



```
class Findout{
void findoutput(int num)
{
 if (num < 1) return;
  findoutput(num / 2);
 System.out.print(num % 2);
 }
}

class Recursion1 {
     public static void main (String args[]) {
          Findout d =new Findout();
          d.findoutput(20);
     }
}
```



Terminal

```
sh-4.3$ javac Recursion1.java
sh-4.3$ java Recursion1
10100sh-4.3$
```

findoutput (10)
findoutput (5)
findoutput (2)
findoutput (1)

Creating Objects

```
Student1 s1;
s1=new Student1();
```

✓

```
int [] a;
a = new int [100];
int [] a = new int [100];
```

- Creating an object is also referred to as instantiating an object.
- When we declare **s1**, it points to null.
- When we instantiate as given in second line **s1** is a reference to **Student1**.
- The method **Studnet1()** is the default constructor of the class.
- Same as

```
Student1 s1=new Student1();
```

✓

Creating Objects

- We can create any number of objects of Student1. For example
  - **`Student1 s1=new Student1();`**
  - // **`Student1 s2=new Student1();`**

Student1 s2 = s1;

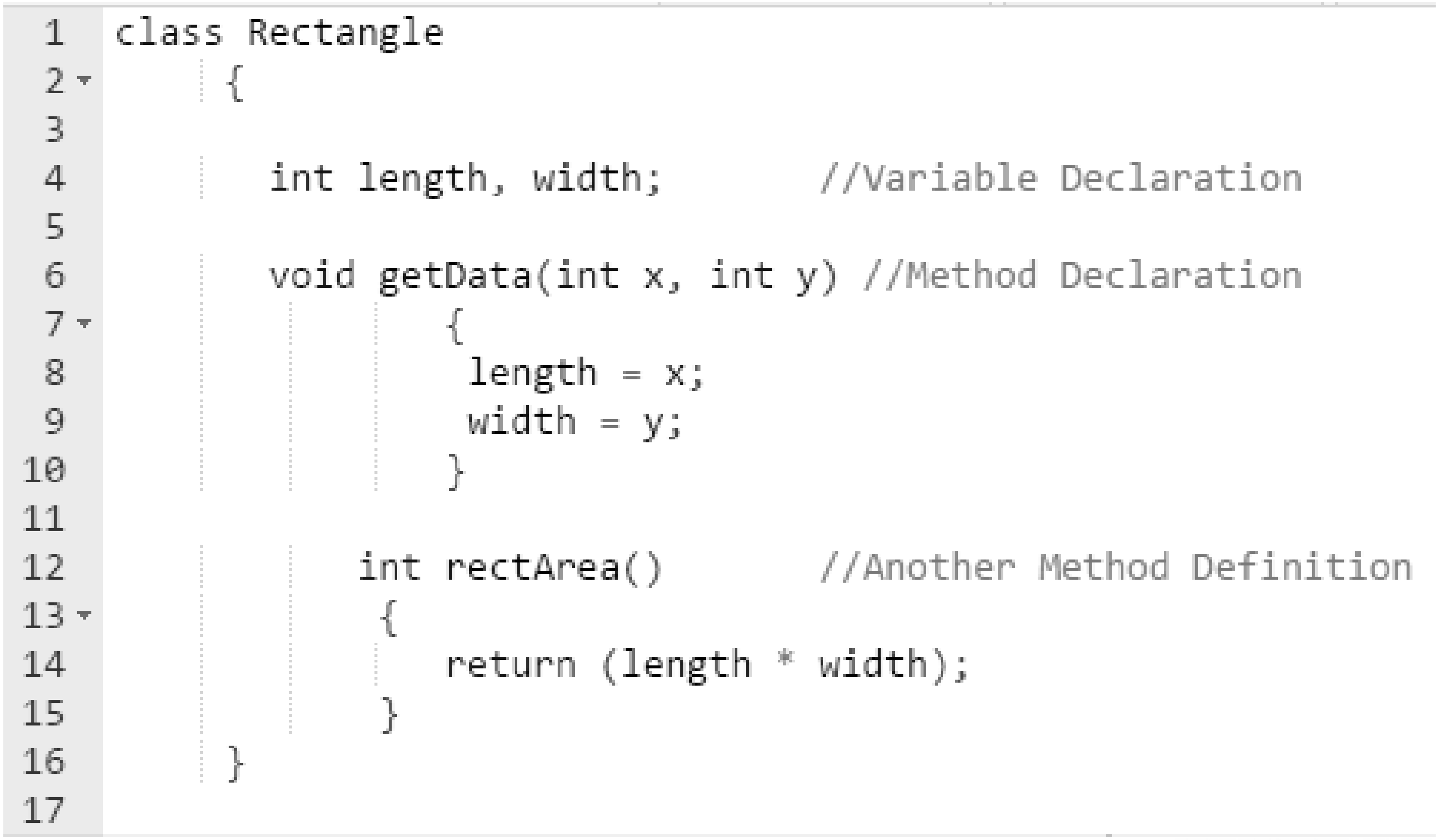
- Object References

Accessing Class members

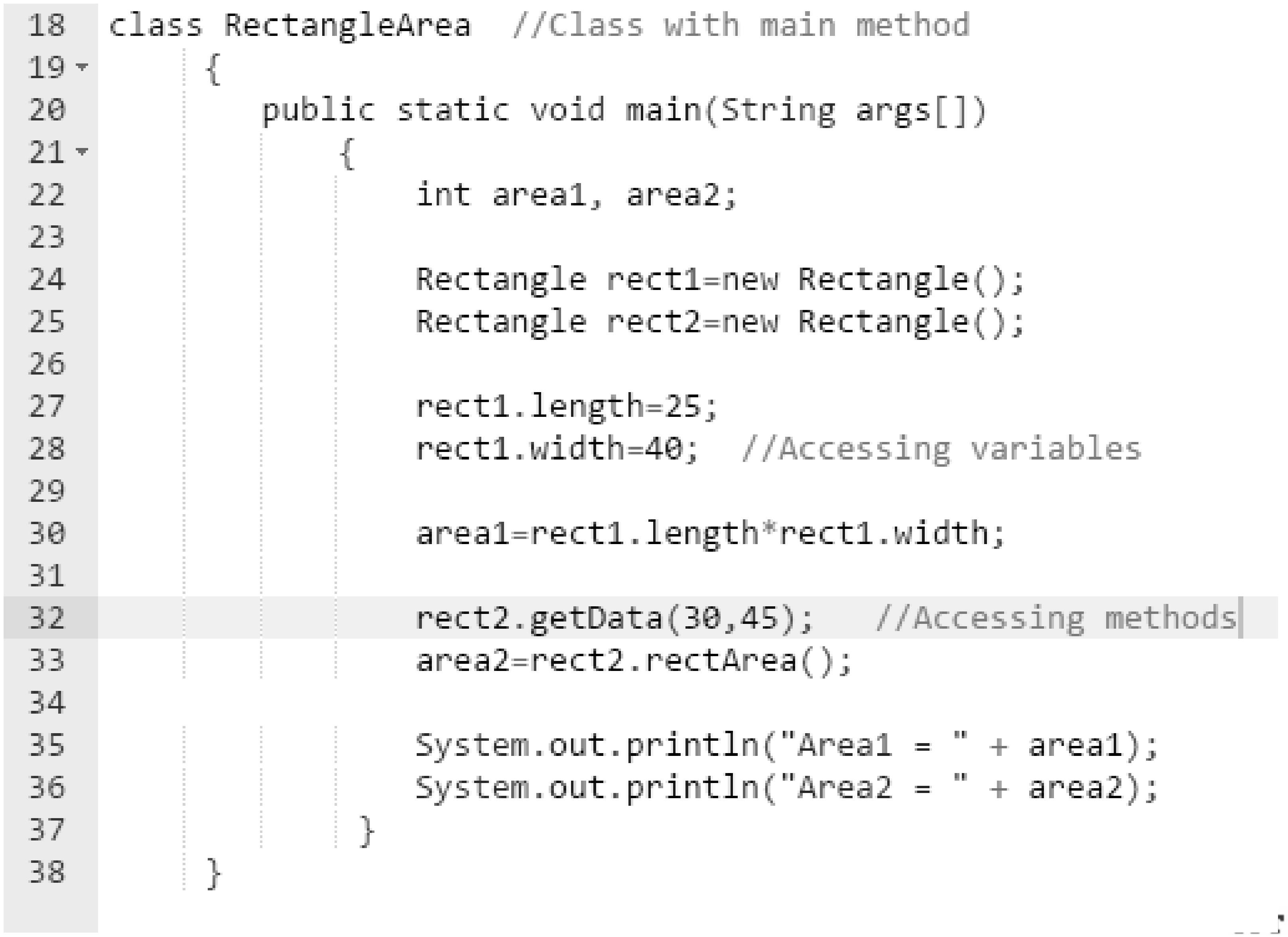
- Object and dot operator

```
Objectname.variablename=value;
Objectname.methodname(parameterList);
```

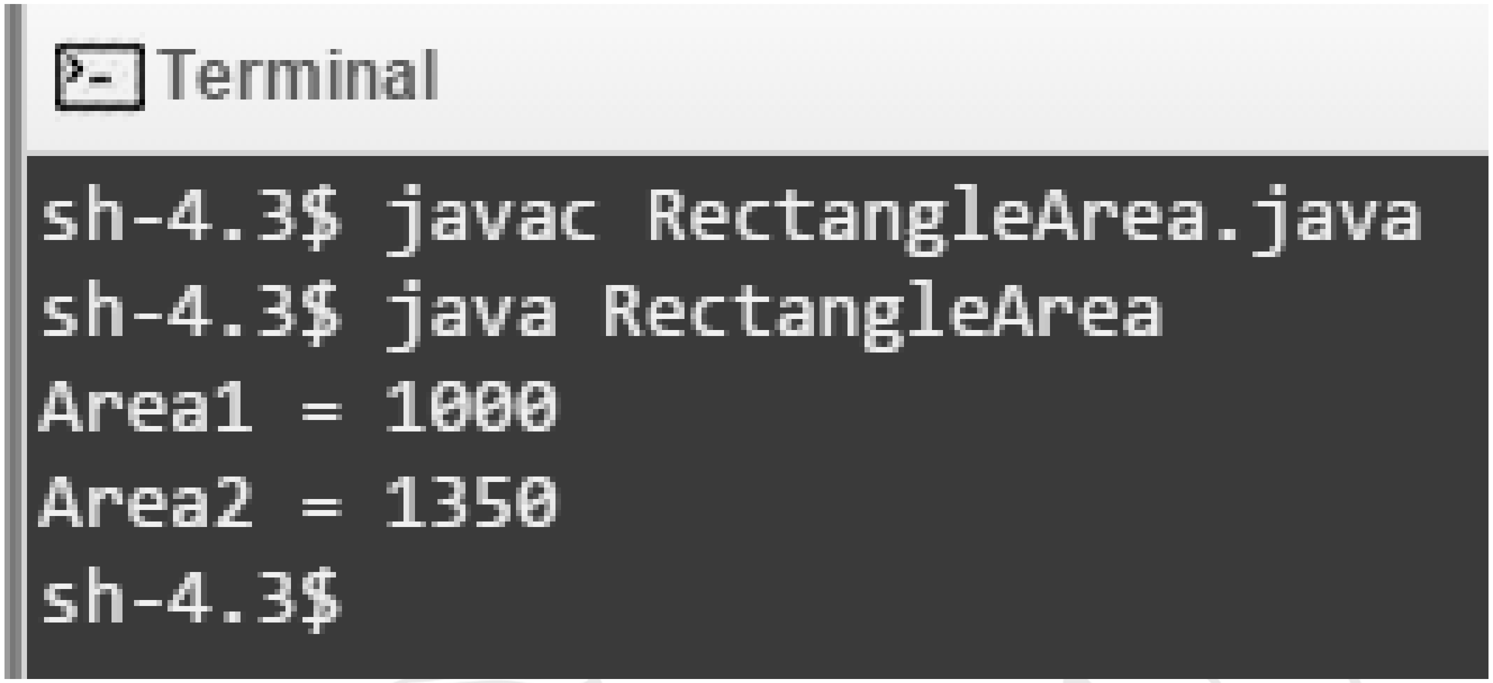
Example



```
class Rectangle
    {

      int length, width;      //Variable Declaration

      void getData(int x, int y) //Method Declaration
              {
               length = x;
               width = y;
              }

          int rectArea()      //Another Method Definition
           {
              return (length * width);
           }
    }

```



```java
class RectangleArea  //Class with main method
    {
        public static void main(String args[])
            {
                int area1, area2;

                Rectangle rect1=new Rectangle();
                Rectangle rect2=new Rectangle();

                rect1.length=25;
                rect1.width=40;  //Accessing variables

                area1=rect1.length*rect1.width;

                rect2.getData(30,45);   //Accessing methods
                area2=rect2.rectArea();

                System.out.println("Area1 = " + area1);
                System.out.println("Area2 = " + area2);
             }
    }
```



```
sh-4.3$ javac RectangleArea.java
sh-4.3$ java RectangleArea
Area1 = 1000
Area2 = 1350
sh-4.3$
```