JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

1. Classes and Instances
2. The Methods
3. The Constructors
4. Static elements
5. Initialization sections
6. Package
7. Inheritance and Polymorphism
8. Abstract classes and interfaces
9. String processing
10. Exceptions and Assertions
11. Nested classes
12. Enums
13. Wrapper classes for primitive types
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java Classes
19. Object Oriented Design

Module contents

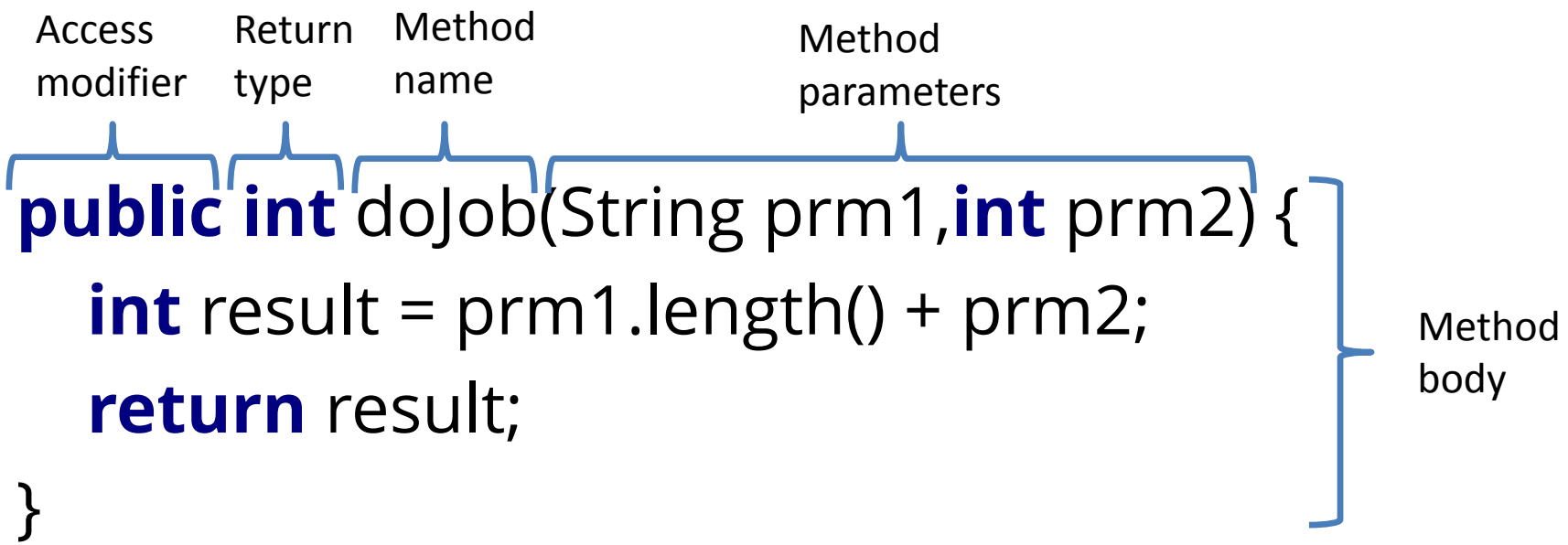
- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

Module contents

- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

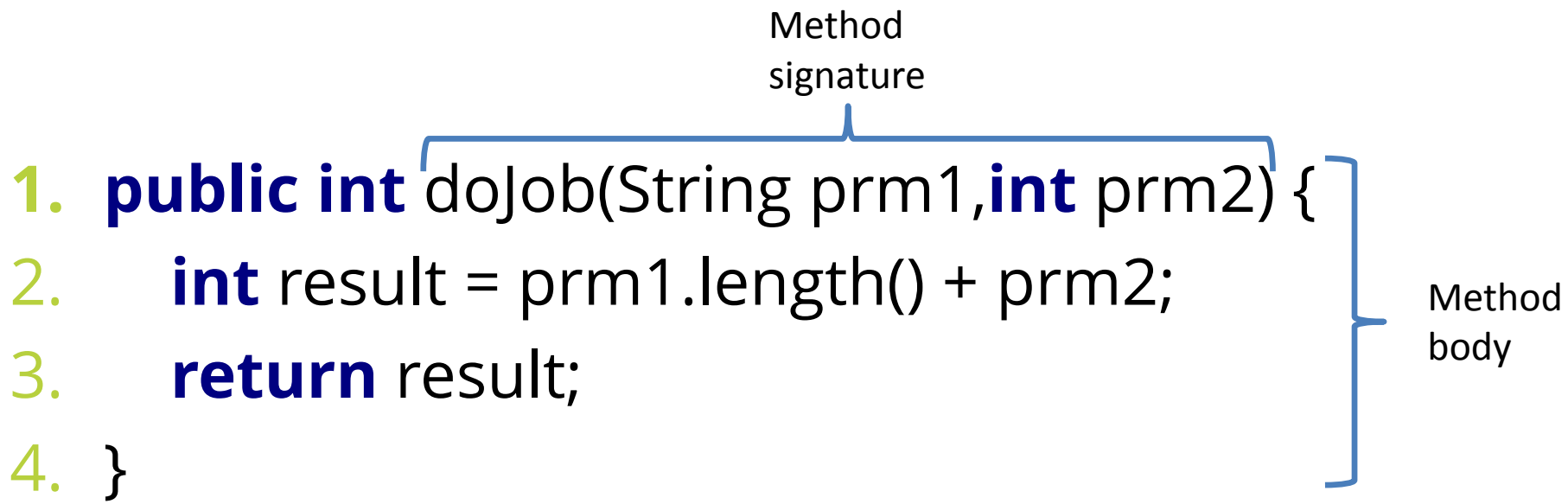
Methods declaration 1/3

- A *method* in Java is a block of statements that has a name and can be executed by calling (*invoking)* it from some other place in your program



Methods declaration 2/3

- Two of the components of a method declaration comprise the *method signature*—the method's name and the parameter types



Methods declaration 3/3

```
1.  public class Car {
2.      //...
3.      public int speed;
4.      //...
5.      public int getSpeed() {
6.          return speed;
7.      }
8.      public void setSpeed(int speed) {
9.          this.speed = speed;
10.     }
11. }
```

Module contents

- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

Passing arguments to a method 1/4

- *Parameters* refers to the list of variables in a method declaration. *Arguments* are the actual values that are passed in when the method is invoked. When you invoke a method, the arguments used must match the declaration's parameters in type and order.

```
1.  public static void main(String[] arg) {
2.          Car car1 = new Car();
3.          // call method "getSpeed" of car1 (no parameters)
4.          int speed1 = car1.getSpeed();
5.          // call method "setSpeed" of car1 (int parameter)
6.          car1.setSpeed(120);
7.  }
```

Passing arguments to a method 2/4

```
1.  public class Car {
2.      //...
3.      public int speed;
4.      //...
5.      public int getSpeed() {
6.          return speed;
7.      }
8.      public void setSpeed(int speed) {
9.          this.speed = speed;
10.     }
11. }
```

Passing arguments to a method 3/4

- Primitive arguments, are passed into methods *by value*

```
1.  public class Main {
2.      public static void testPrm(int x) {
3.          x = 123;
4.      }
5.      public static void main(String[] arg) {
6.          int x = 1;
7.          testPrm(x);
8.          System.out.print(x);
9.      }
10. }
```

Passing arguments to a method 4/4

- Reference data type parameters, such as objects, are also passed into methods *by reference value*.

```
1.  public class Main {
2.      public static void testPrm(int[] arr) {
3.          arr[1] = 123;
4.      }
5.      public static void main(String[] arg) {
6.          int[] arr = {1,2,3,4,5};
7.          testPrm(arr);
8.          System.out.print(Arrays.toString(arr));
9.      }
10. }
```

Module contents

- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

Methods with variable arguments 1/2

```
1. public static void main(String[] arg) {
2.     TestVarArg tstvarg = new TestVarArg();
3.     int sum0 = tstvarg.calcSum();
4.     int sum1 = tstvarg.calcSum(3);
5.     int sum2 = tstvarg.calcSum(55, 66);
6.     int sum5 = tstvarg.calcSum(77, 55, 33, 11, 99);
7. }
```

Methods with variable arguments 2/2

```
1. public class TestVarArg {
2.     public int calcSum(int... values) {
3.         int res = 0;
4.         for(int x : values){
5.             res+=x;
6.         }
7.         return res;
8.     }
9. }
```

Module contents

- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

Overloading methods 1/3

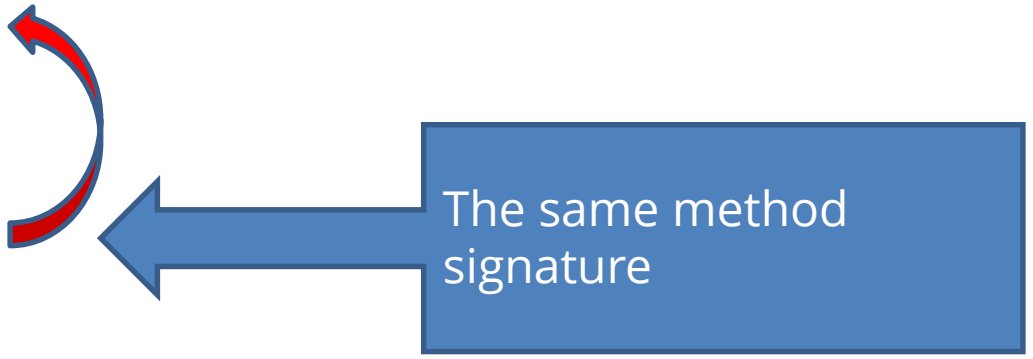
```
1.  public void test(String s) {
2.     s = "abcd";
3.     System.out.println("test(String s)");
4.  }
5.  public void test(int i) {
6.     System.out.println("test(int i)");
7.  }
8.  public void test(int[] arr) {
9.     System.out.println("test(int[] arr)");
10. }
11. public int test(int i, double f) {
12.    System.out.println("test(int i, double f)");
13.    return i+10;
14. }
```

Overloading methods 2/3

```
1. public static void main(String[] arg) {
2.     Car myCar = new Car();
3.     myCar.test(100);
4.     myCar.test("AAA");
5.     myCar.test(new int[]{1,2,3});
6.     myCar.test(10,3.5);
7. }
```

Overloading methods 3/3

```
1.  public void test(String s) {
2.      // ...
3.  }
4.  public void test(int i) {
5.      // ...
6.  }
7.  public int test(int i) {
8.      // ...
9.  }
10. public int test(int i, double f) {
11.     // ...
12.     return i;
13. }
```



Module contents

- The Methods
  - Methods declaration
  - Passing arguments to a method
  - Methods with variable arguments
  - Overloading methods
  - Final modifier for method arguments

Final modifier for method arguments

```
1.  public void test(final String s) {
2.      s = "abcd";
3.      // ...
4.  }
5.  public void test(final int i) {
6.      i++;
7.      // ...
8.  }
9.  public void test(final int[] arr) {
10.     arr[0] = 100;
11.     arr = new int[10];
12.     // ...
13. }
```