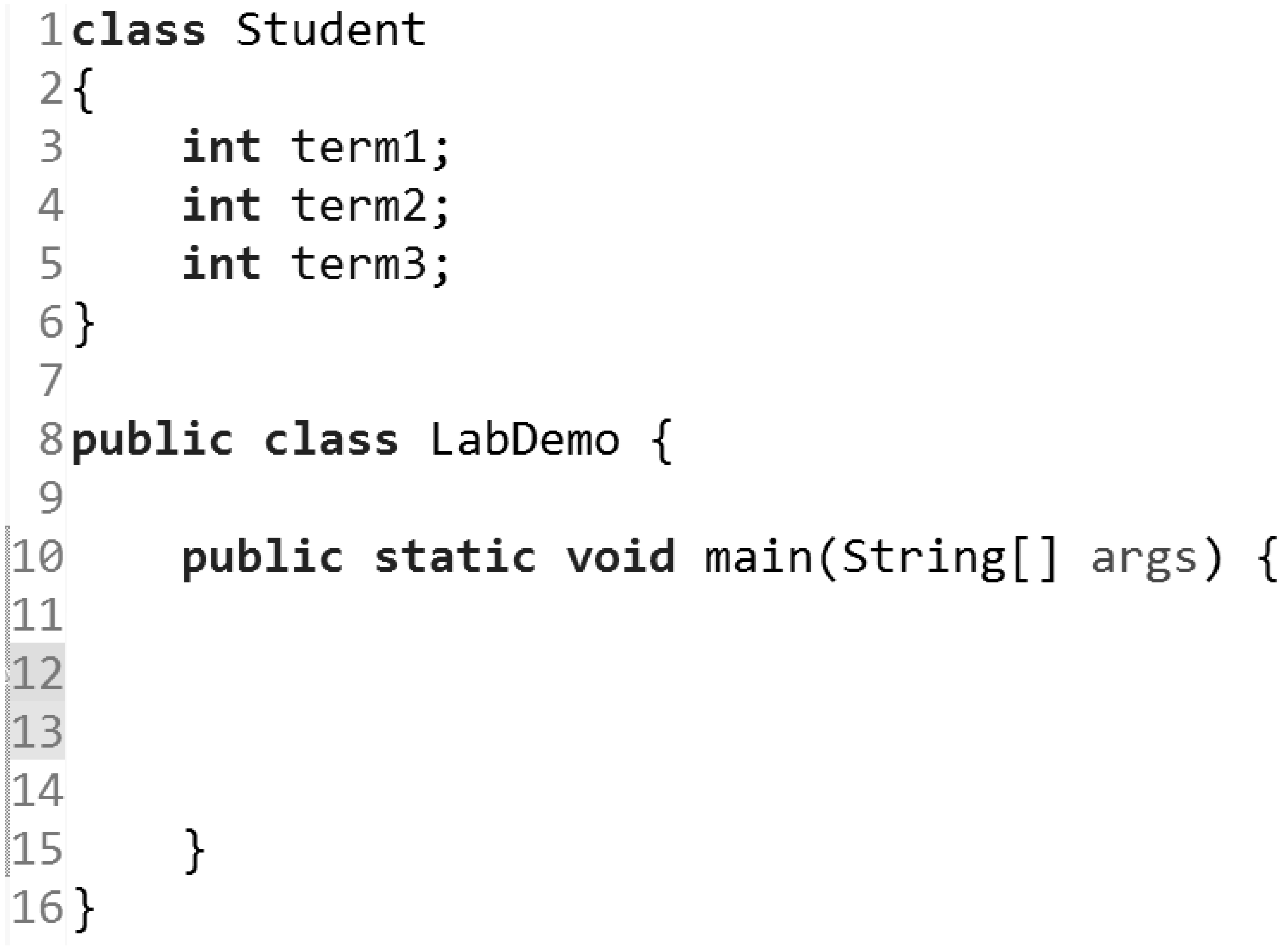
Java Classes and References

CSE-220

Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

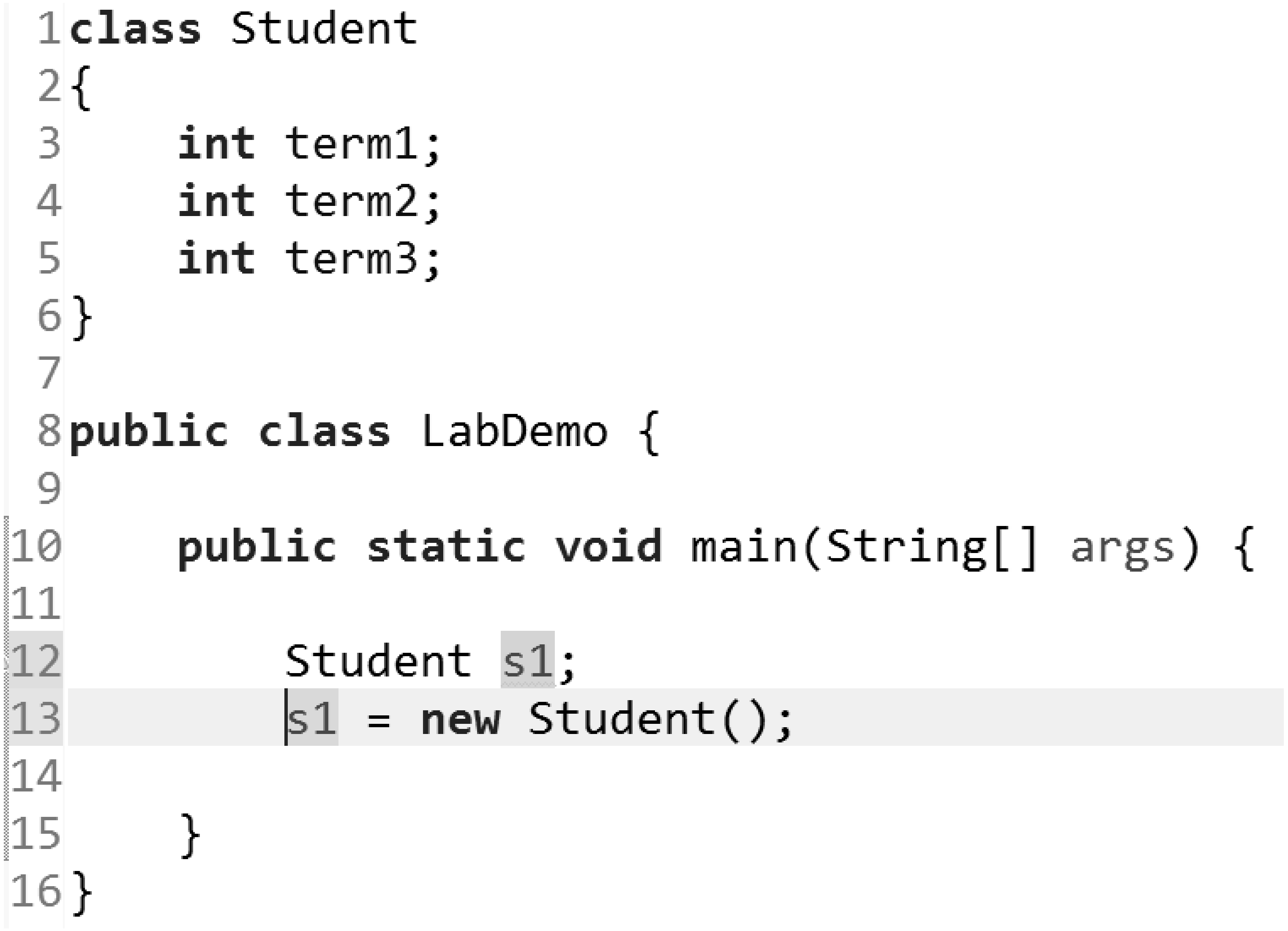
Reference: Java, The complete Reference
8th ed (Herb Schildt) – Chapter 6

Creating a class



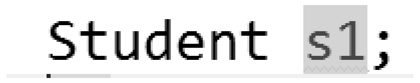
```java
class Student
{
    int term1;
    int term2;
    int term3;
}

public class LabDemo {

    public static void main(String[] args) {




    }
}
```

Declaring an object

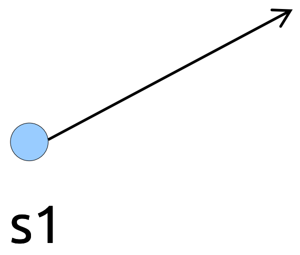


```
class Student
{
    int term1;
    int term2;
    int term3;
}

public class LabDemo {

    public static void main(String[] args) {

        Student s1;
        s1 = new Student();

    }
}
```

Taking a look at the reference

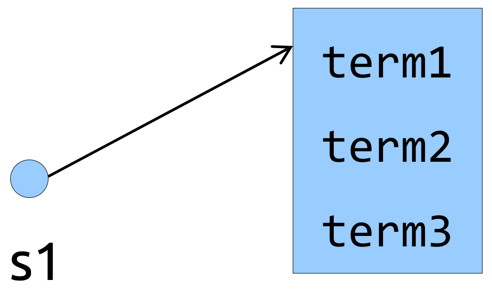


```
Student s1;
```



Taking a look at the reference

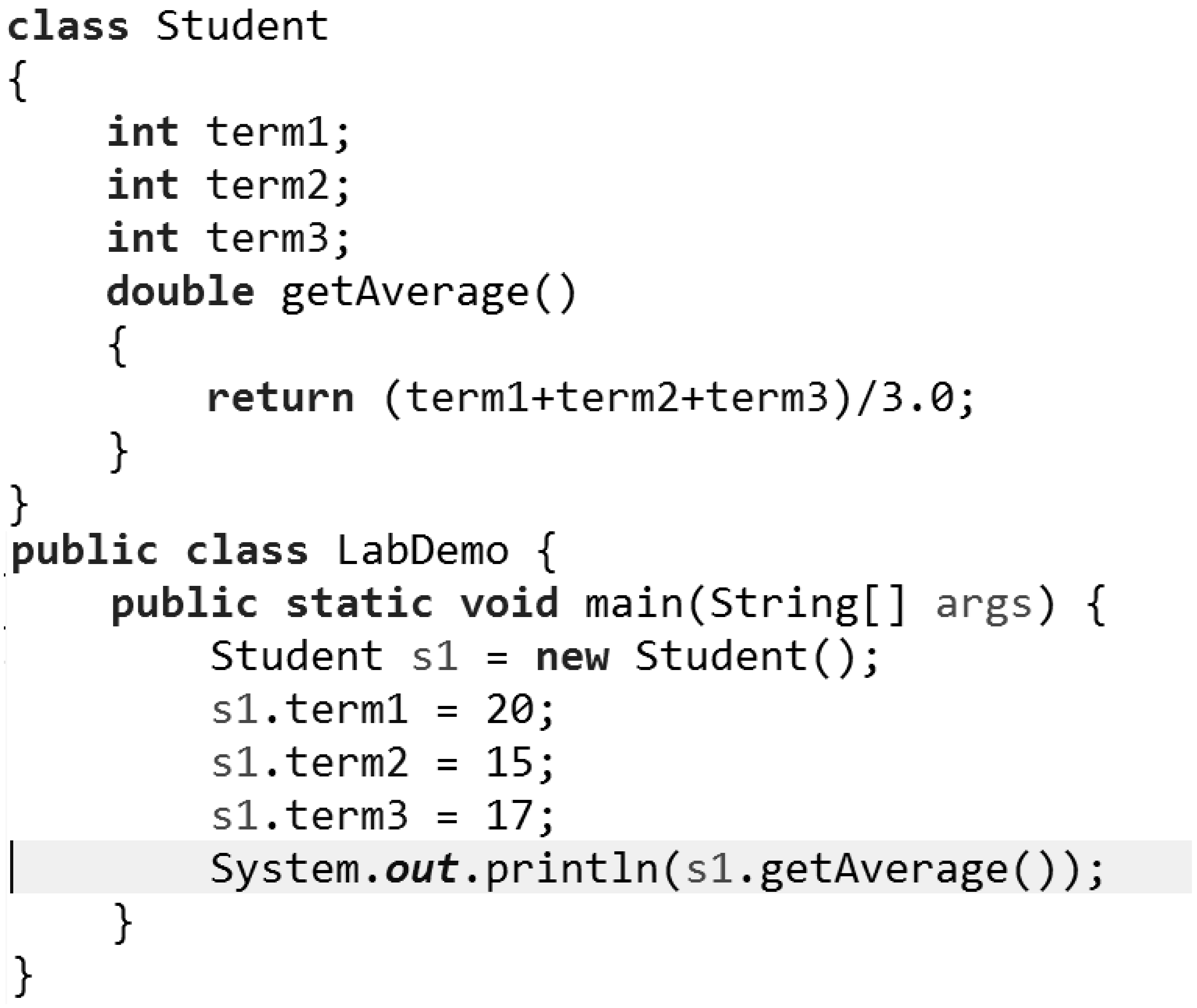
```
Student s1;

s1 = new Student();
```



Assigning values

```
public class LabDemo {

    public static void main(String[] args) {

        Student s1 = new Student();
        s1.term1 = 20;
        s1.term2 = 15;
        s1.term3 = 17;

    }
}
```

Adding a method



```
class Student
{
    int term1;
    int term2;
    int term3;
    double getAverage()
    {
        return (term1+term2+term3)/3.0;
    }
}
public class LabDemo {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.term1 = 20;
        s1.term2 = 15;
        s1.term3 = 17;
        System.out.println(s1.getAverage());
    }
}
```

Parameterized methods

```
class Student
{
    int roll;
    int term1;
    int term2;
    int term3;
    void setRoll(int r)
    {
        roll = r;
    }
    double getAverage()
    {
        return (term1+term2+term3)/3.0;
    }
}
```

Constructor

```
public Student(int r, int t1, int t2, int t3)
{
    roll = r;
    term1 = t1;
    term2 = t2;
    term3 = t3;
}
```

Constructor Calling

```
public Student(int r, int t1, int t2, int t3)
{
    roll = r;
    term1 = t1;
    term2 = t2;
    term3 = t3;
}
```

```
Student s1 = new Student(1, 20, 15, 17);
```

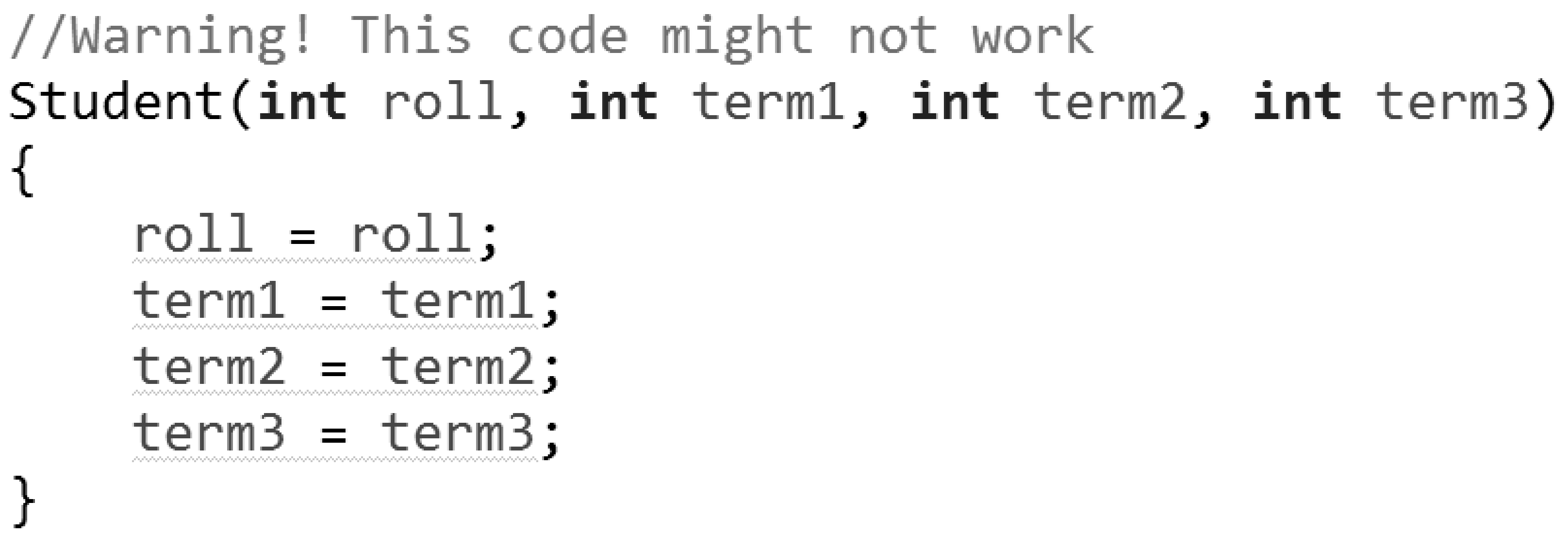
Overloading Constructor

```
Student()
{
    roll = term1 = term2 = term3 = 0;
}
```

```
public Student(int r, int t1, int t2, int t3)
{
    roll = r;
    term1 = t1;
    term2 = t2;
    term3 = t3;
}
```

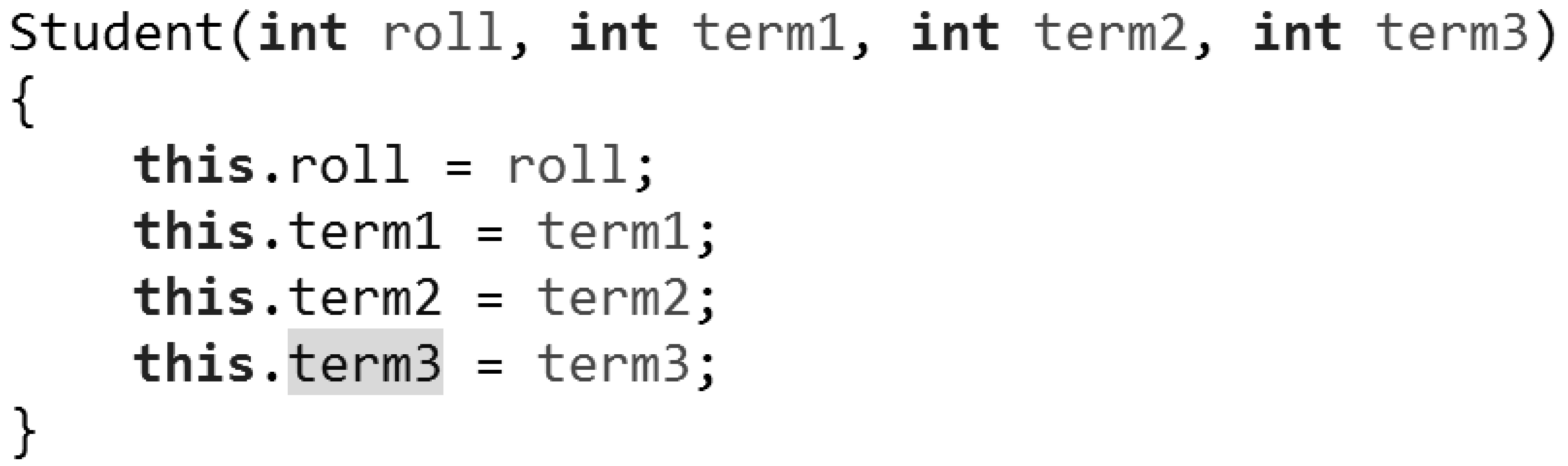
```
Student s1 = new Student(1, 20, 15, 17);
```

Using this keyword



```
//Warning! This code might not work
Student(int roll, int term1, int term2, int term3)
{
    roll = roll;
    term1 = term1;
    term2 = term2;
    term3 = term3;
}
```

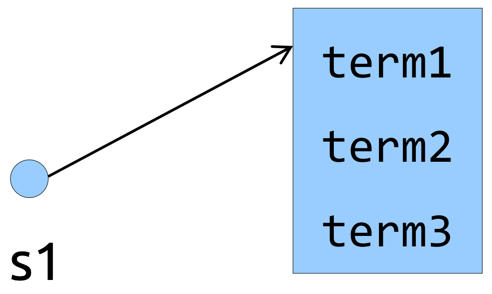
Using this keyword



```
Student(int roll, int term1, int term2, int term3)
{
    this.roll = roll;
    this.term1 = term1;
    this.term2 = term2;
    this.term3 = term3;
}
```

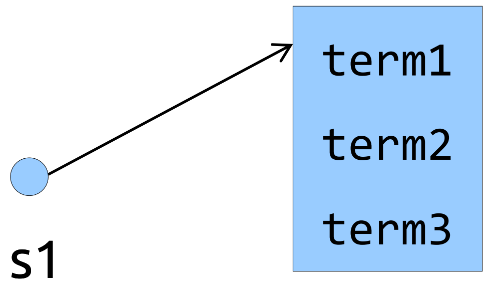
Destructor?

```
Student s1;

s1 = new Student();
```

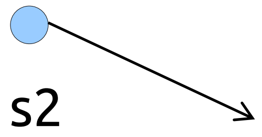


Destructor?

```
Student s1;
s1 = new Student();
```

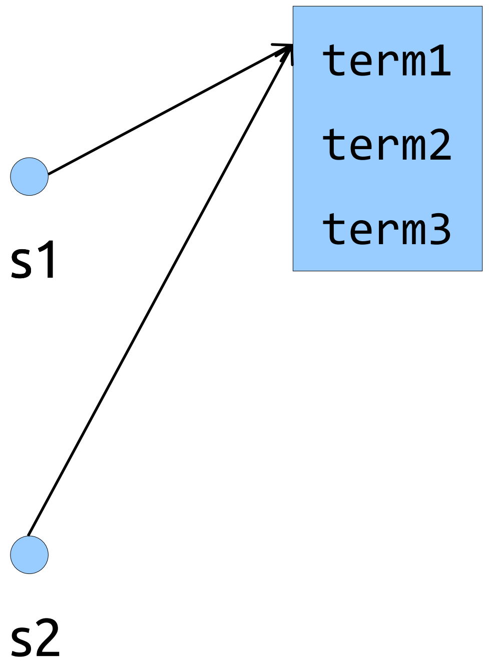


```
Student s2;
```



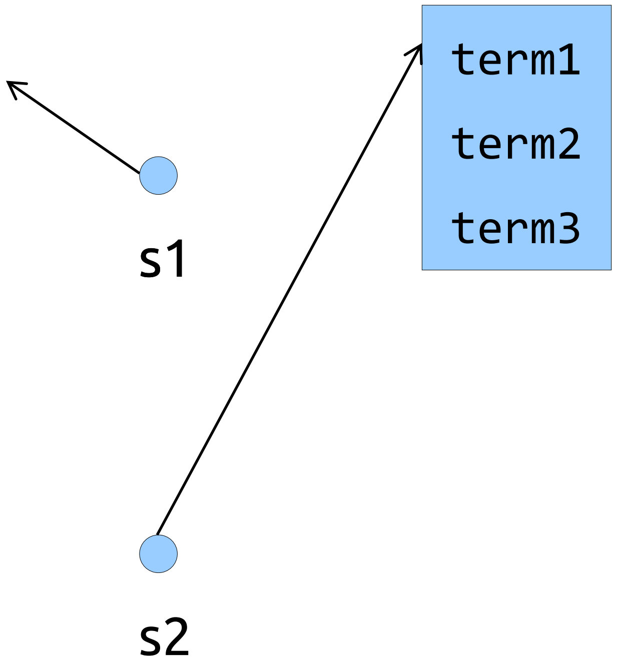
Destructor?

```
Student s1;
s1 = new Student();


Student s2;
s2 = s1;
```



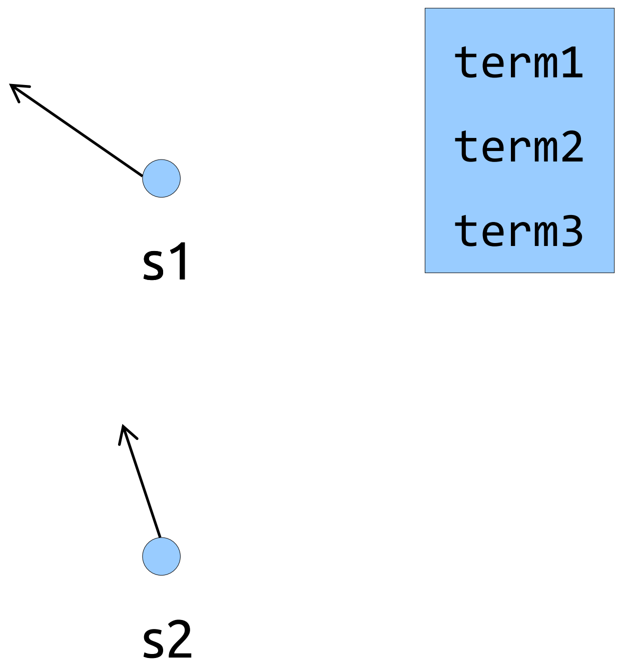
Destructor?

```
Student s1;
s1 = new Student();


Student s2;
s2 = s1;

s1 = null;
```



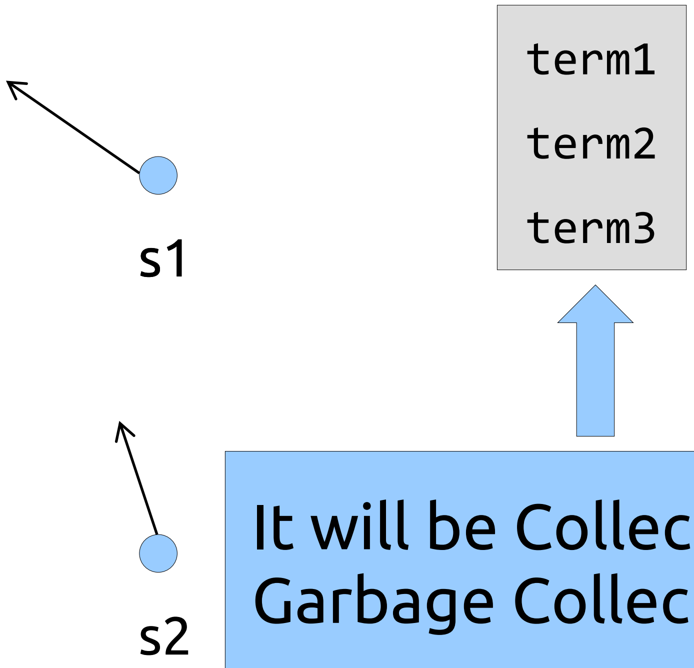
Destructor?

```
Student s1;
s1 = new Student();


Student s2;
s2 = s1;

s1 = null;

s2 = null;
```



Destructor?

```
Student s1;
s1 = new Student();


Student s2;
s2 = s1;

s1 = null;
s2 = null;
```

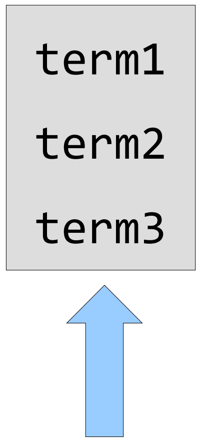


It will be Collected by a Garbage Collector.

It will not necessarily be immediate.

Garbage Collector

When GC collects an object, `finalize()` method is called.

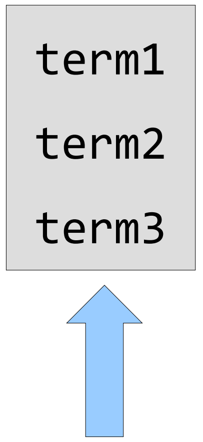


It will be Collected by a Garbage Collector.

It will <u>not</u> necessarily be immediate.

Garbage Collector

When GC collects an object,
`finalize()` method is called.



It will be Collected by a Garbage Collector.

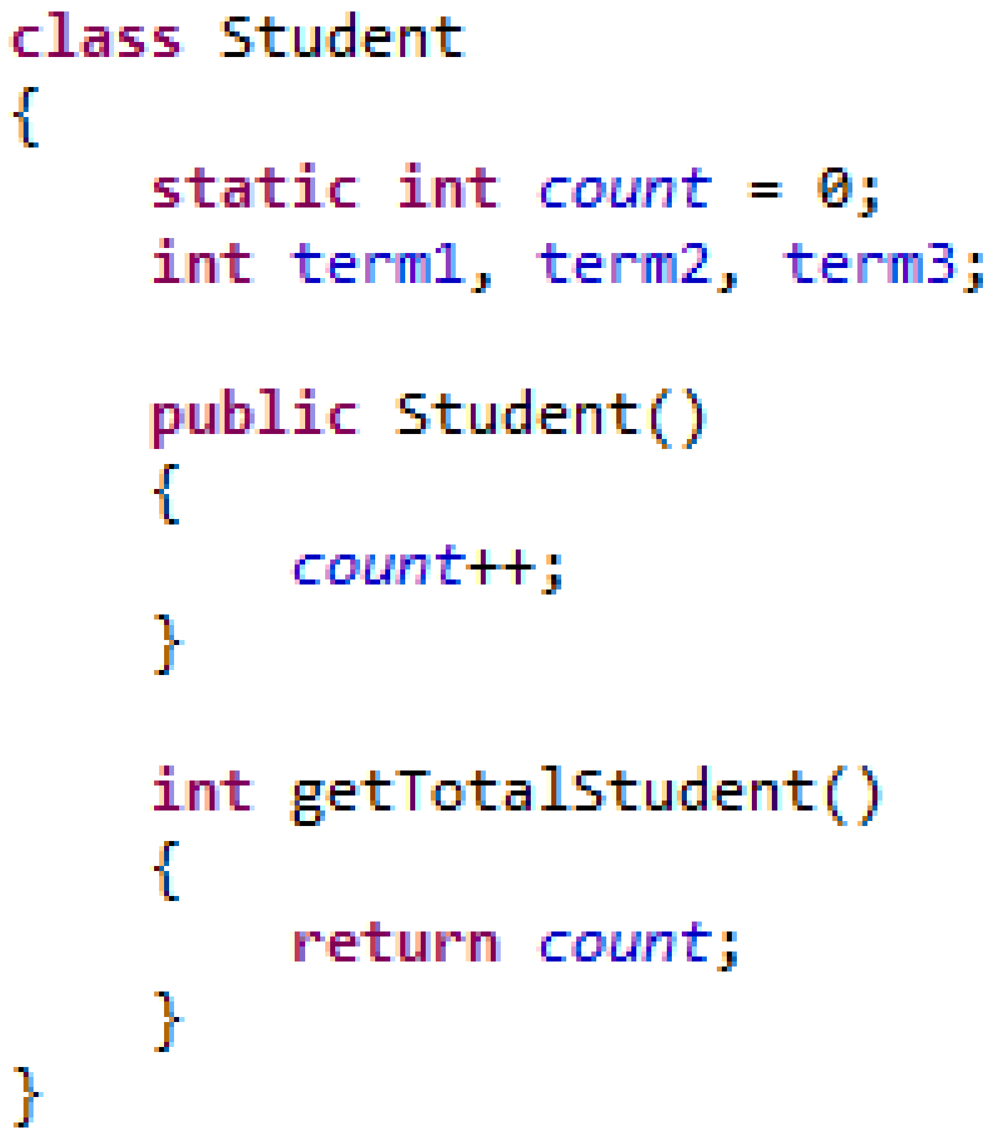
It will <u>not</u> necessarily be immediate.

The `static` keyword in class member

Only one instance is created

The `static` keyword in class member

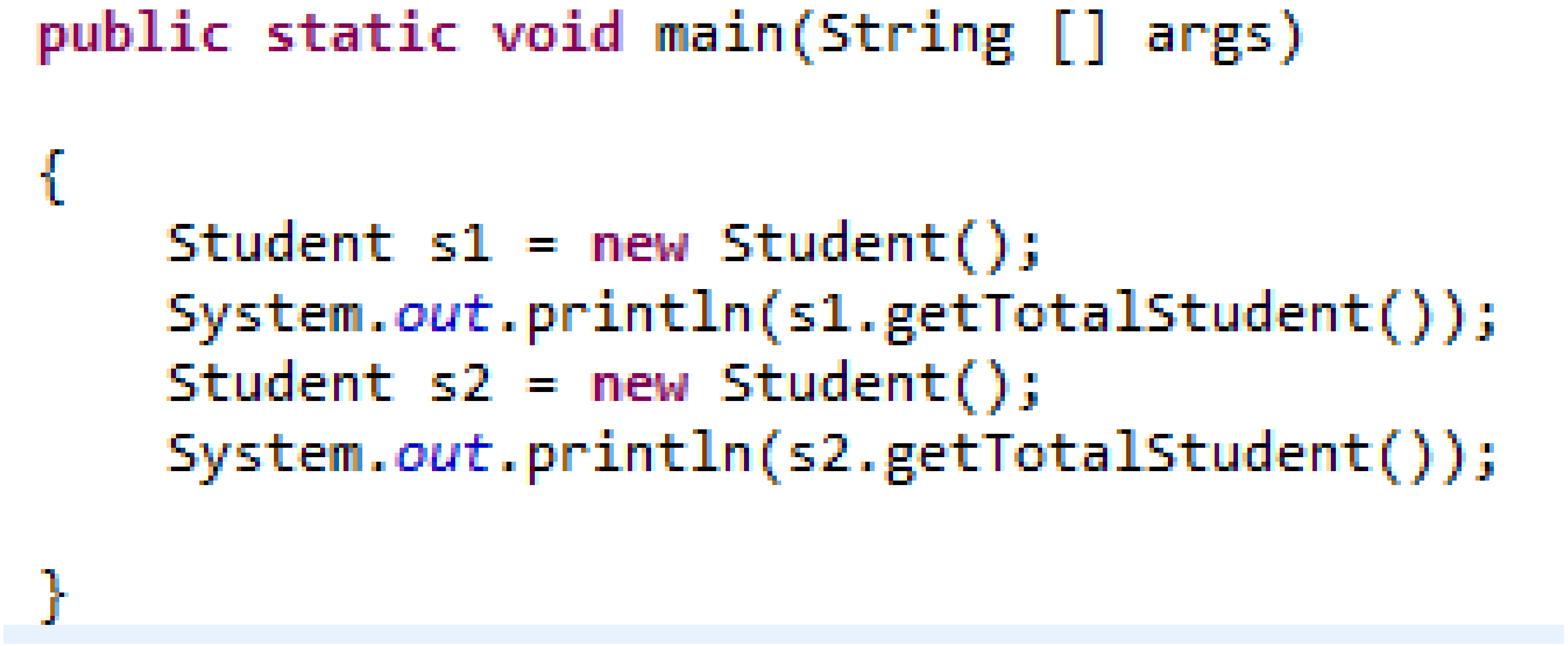
Only one instance is created



```
class Student
{
    static int count = 0;
    int term1, term2, term3;

    public Student()
    {
        count++;
    }

    int getTotalStudent()
    {
        return count;
    }
}
```

The static keyword in class member

Only one instance is created



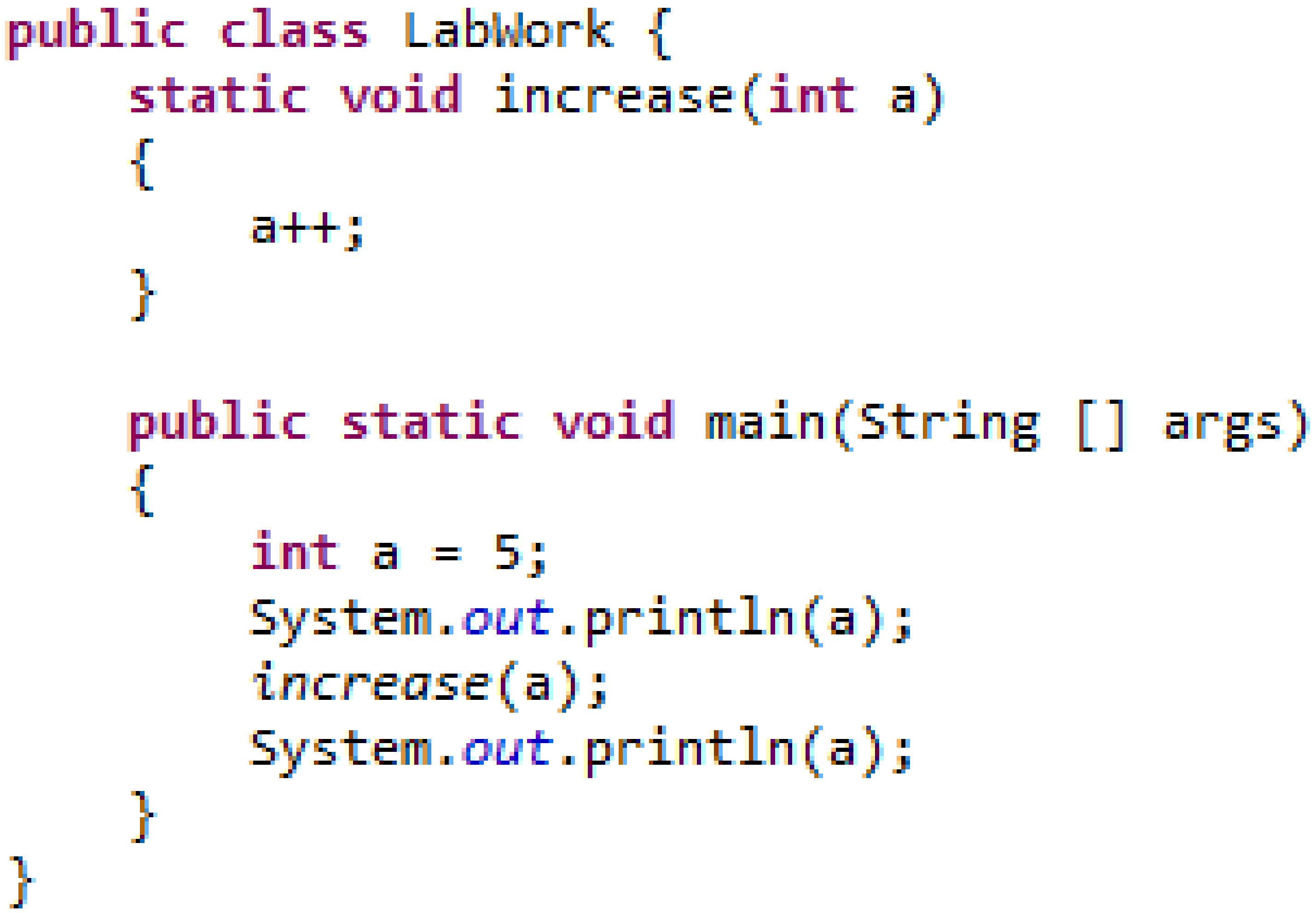
```
public static void main(String [] args)

{
        Student s1 = new Student();
        System.out.println(s1.getTotalStudent());
        Student s2 = new Student();
        System.out.println(s2.getTotalStudent());

}
```

Methods

Pass by value vs. Pass by reference

Methods

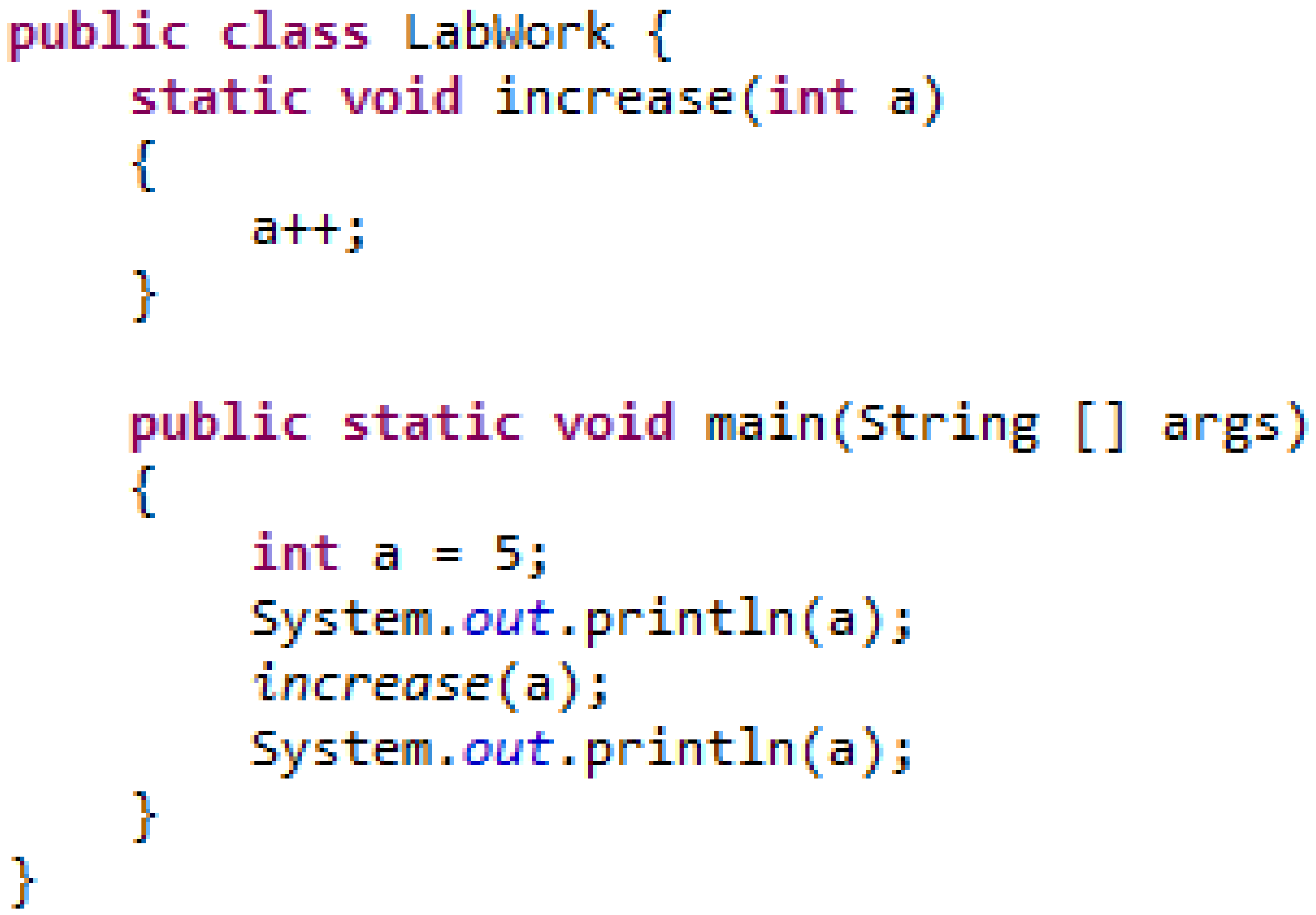
Pass by value vs. Pass by reference



```
public class LabWork {
    static void increase(int a)
    {
        a++;
    }

    public static void main(String [] args)
    {
        int a = 5;
        System.out.println(a);
        increase(a);
        System.out.println(a);
    }
}
```

Methods

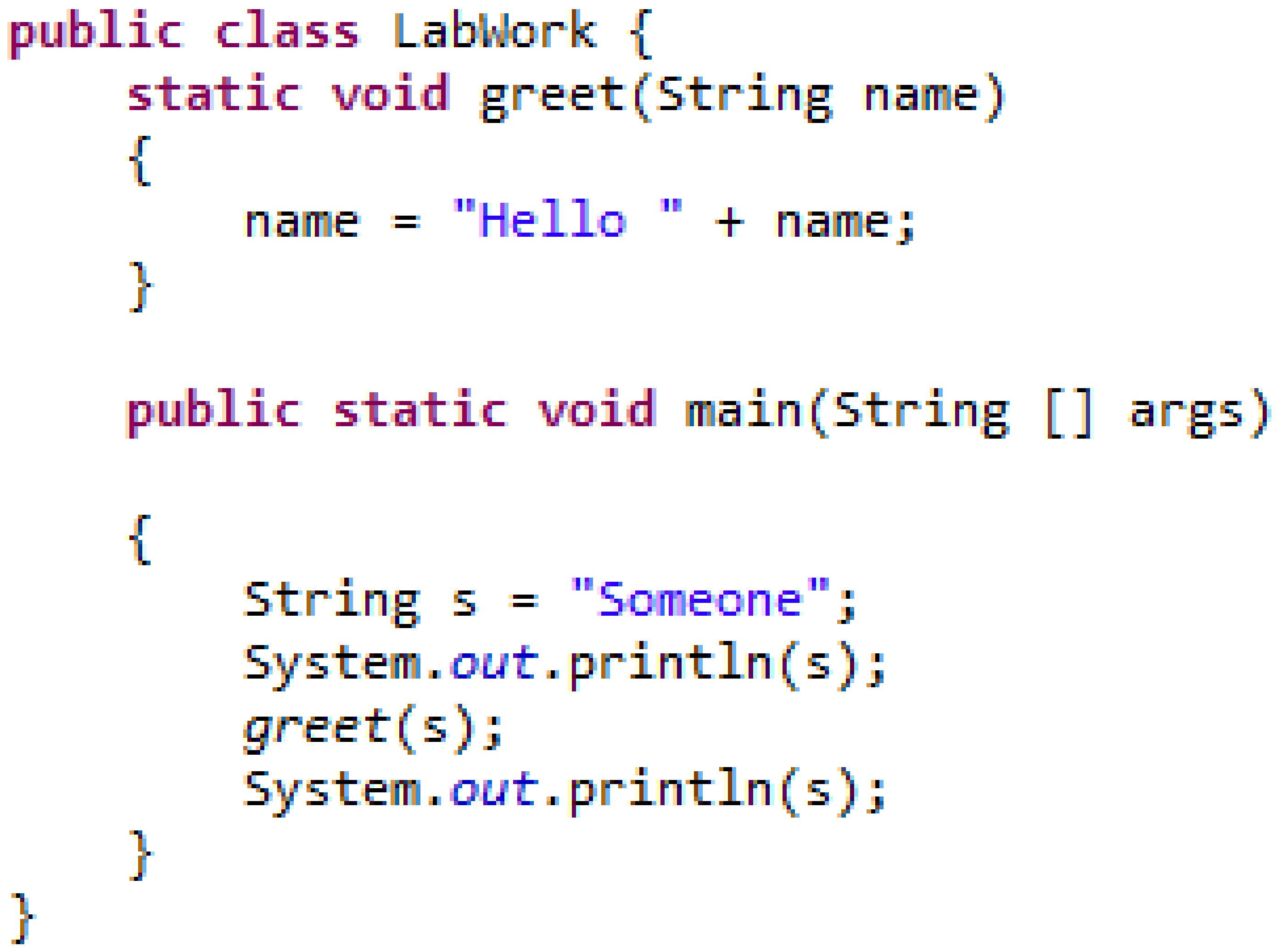
Pass by value ~~vs. Pass by reference~~



```
public class LabWork {
    static void increase(int a)
    {
        a++;
    }

    public static void main(String [] args)
    {
        int a = 5;
        System.out.println(a);
        increase(a);
        System.out.println(a);
    }
}
```

Methods

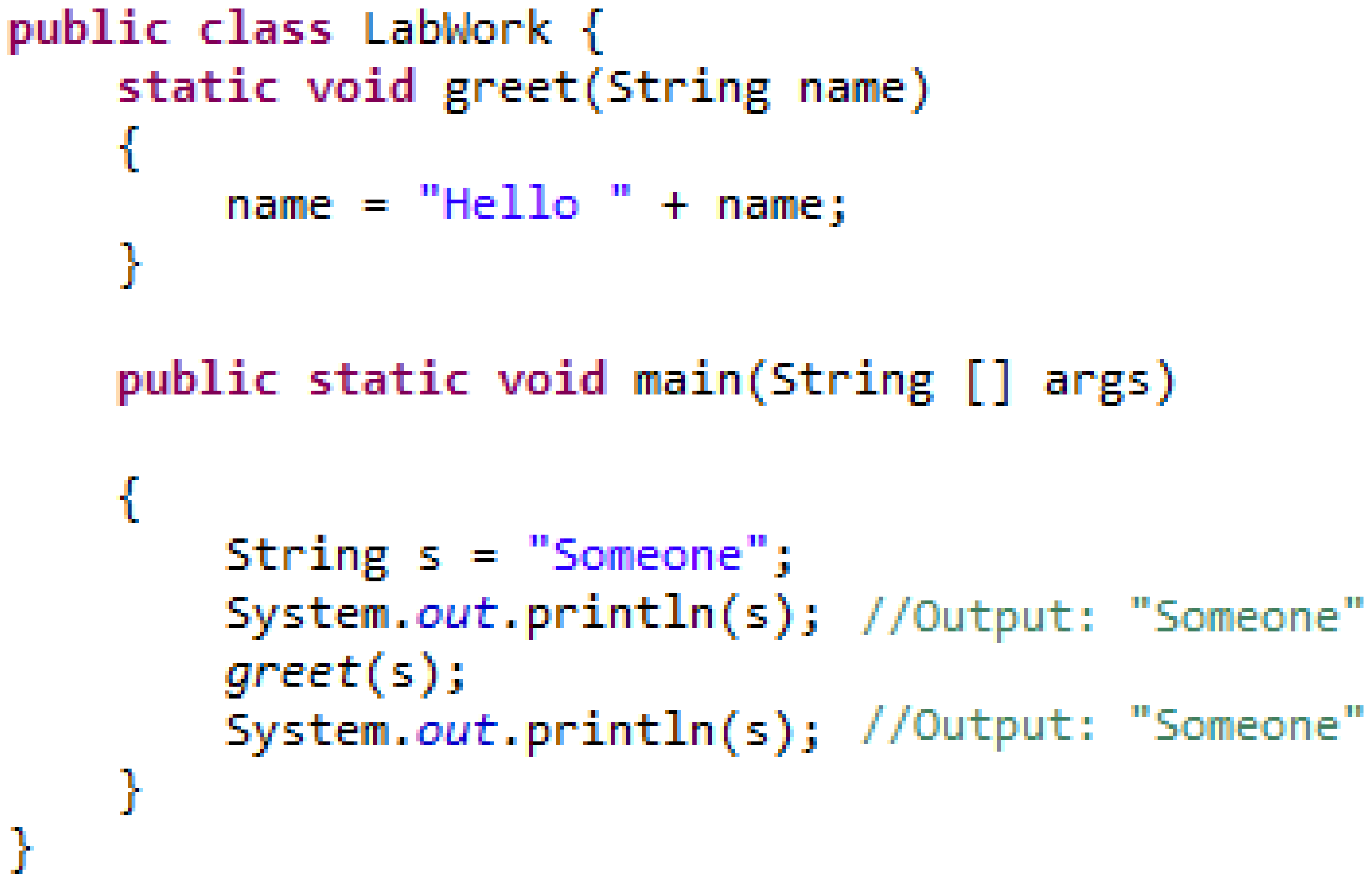
Passing object. Is it the reference that is passed?



```
public class LabWork {
    static void greet(String name)
    {
        name = "Hello " + name;
    }

    public static void main(String [] args)

    {
        String s = "Someone";
        System.out.println(s);
        greet(s);
        System.out.println(s);
    }
}
```

Methods

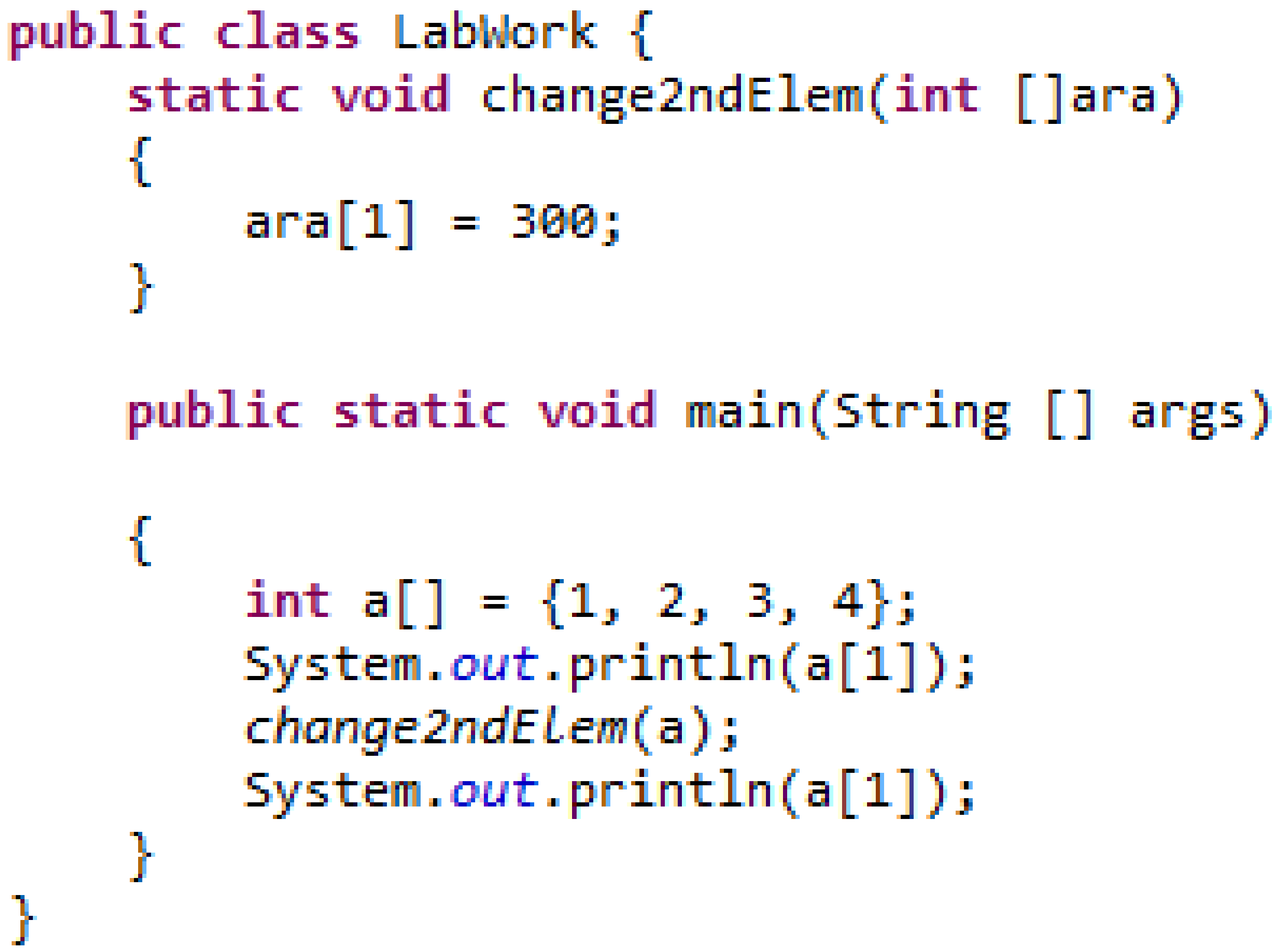
Only the ‘value’ of reference is passed



```
public class LabWork {
    static void greet(String name)
    {
        name = "Hello " + name;
    }

    public static void main(String [] args)

    {
        String s = "Someone";
        System.out.println(s); //Output: "Someone"
        greet(s);
        System.out.println(s); //Output: "Someone"
    }
}
```

Methods

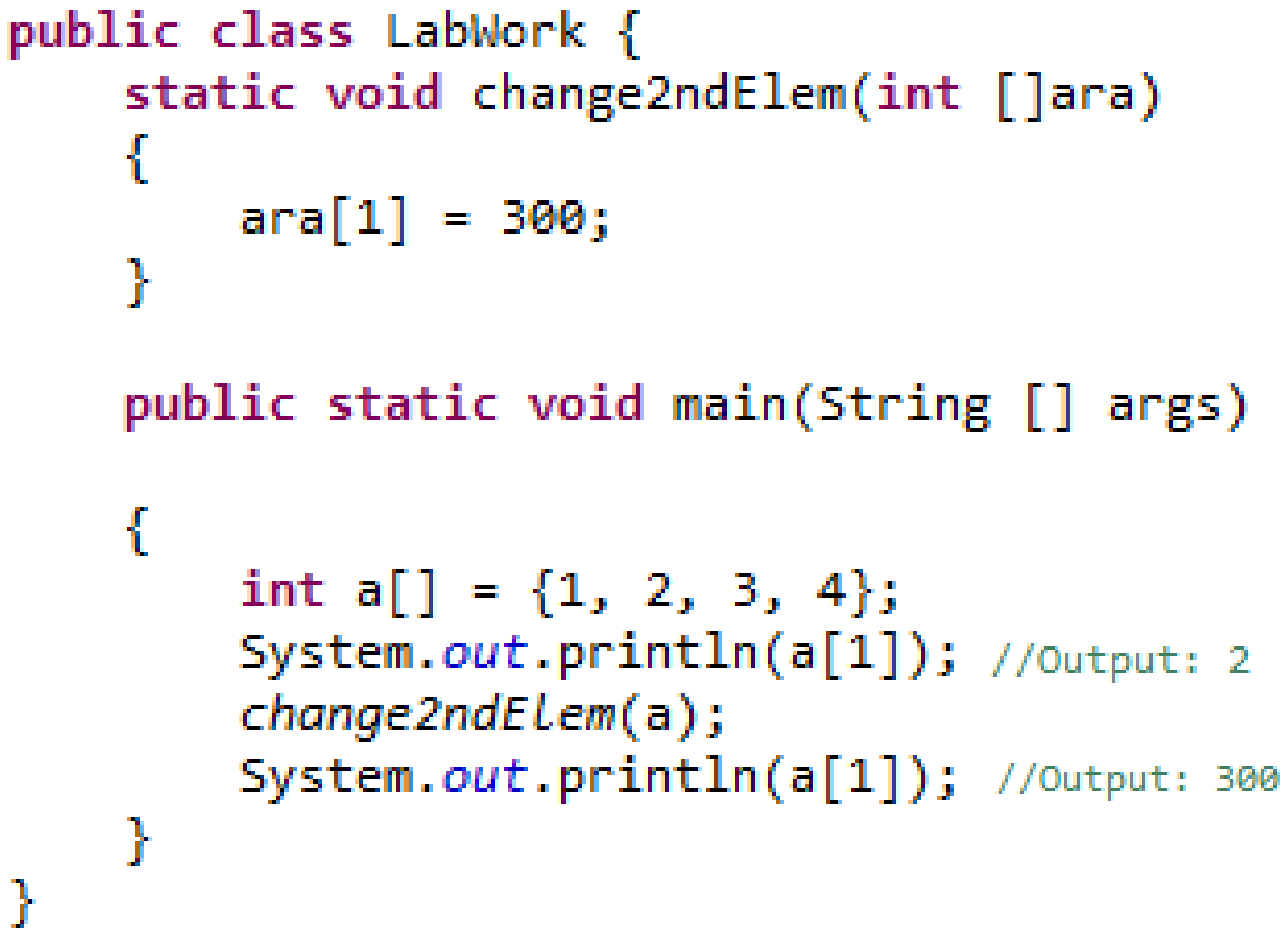
Passing an array. What will be the outputs?



```
public class LabWork {
    static void change2ndElem(int []ara)
    {
        ara[1] = 300;
    }

    public static void main(String [] args)

    {
        int a[] = {1, 2, 3, 4};
        System.out.println(a[1]);
        change2ndElem(a);
        System.out.println(a[1]);
    }
}
```

Methods

Passing an array.



```
public class LabWork {
    static void change2ndElem(int []ara)
    {
        ara[1] = 300;
    }

    public static void main(String [] args)

    {
        int a[] = {1, 2, 3, 4};
        System.out.println(a[1]); //Output: 2
        change2ndElem(a);
        System.out.println(a[1]); //Output: 300
    }
}
```

Two types of Object

Mutable vs. Immutable

Array Object →

String Object →

Two types of Object

Mutable vs. Immutable

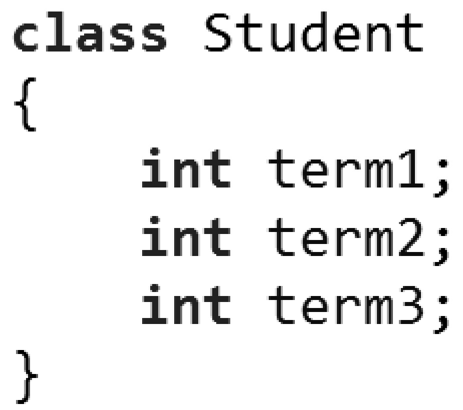
Array Object → Mutable

String Object → Immutable

Mutable or Immutable?

```
class Student
{
    int term1;
    int term2;
    int term3;
}
```

Mutable or Immutable?



```
class Student
{
    int term1;
    int term2;
    int term3;
}
```

Mutable

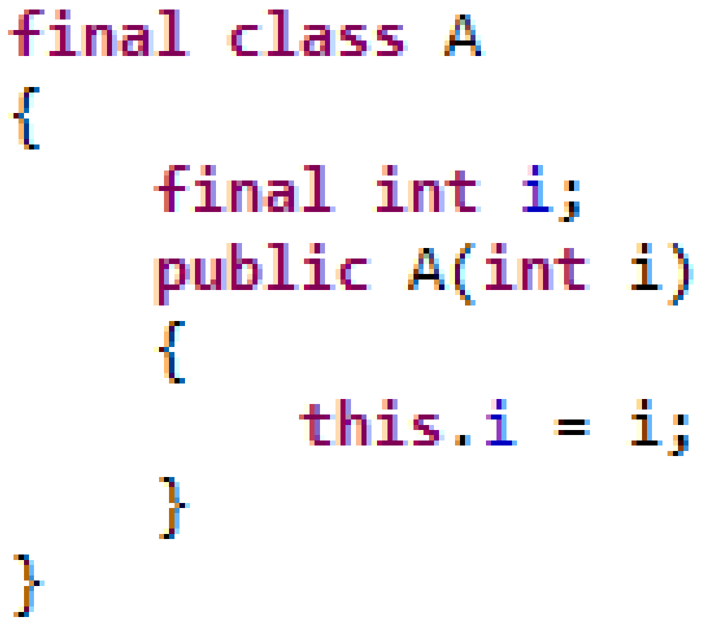
How to create Immutable Class?

How to create Immutable Class?

Use the `final` keyword

How to create Immutable Class?

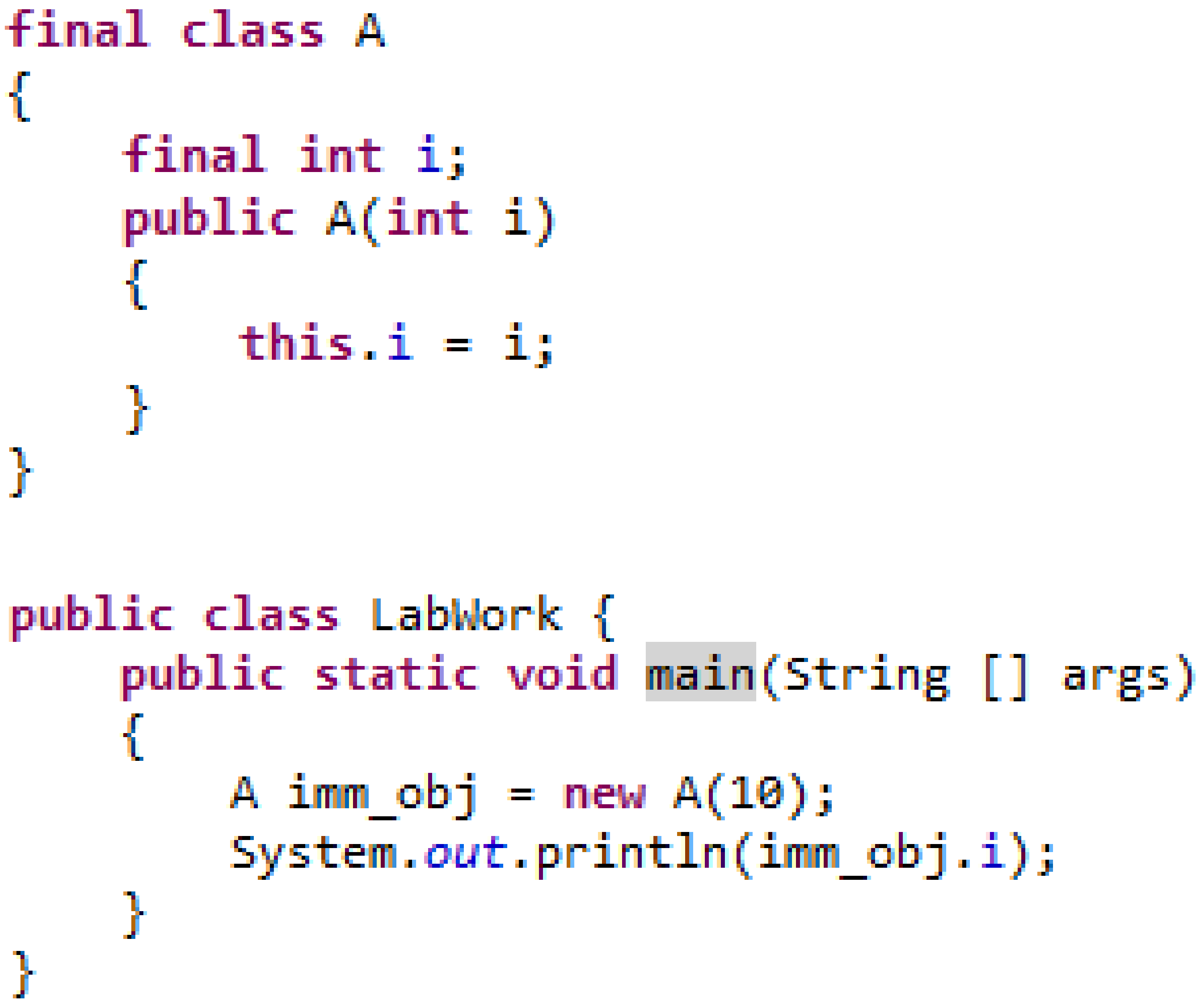
Use the `final` keyword



```
final class A
{
    final int i;
    public A(int i)
    {
        this.i = i;
    }
}
```

How to create Immutable Class?

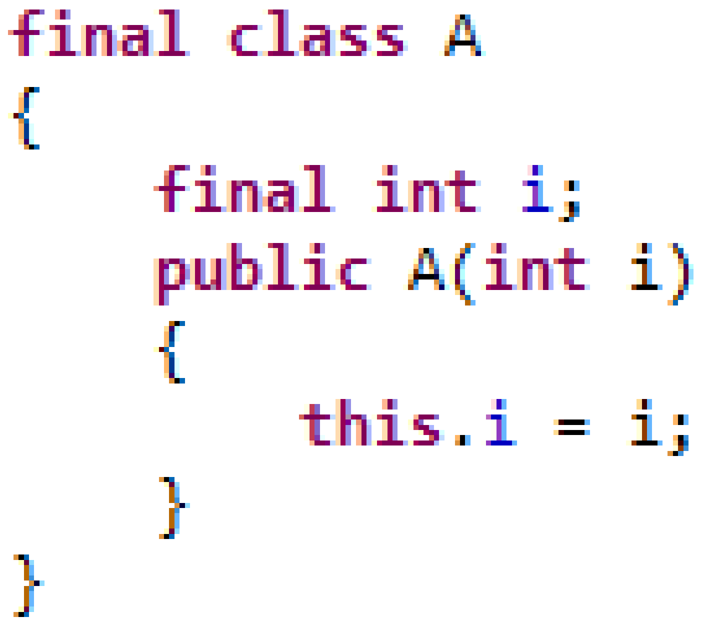
Use the `final` keyword



```
final class A
{
    final int i;
    public A(int i)
    {
        this.i = i;
    }
}

public class LabWork {
    public static void main(String [] args)
    {
        A imm_obj = new A(10);
        System.out.println(imm_obj.i);
    }
}
```

How to create Immutable Class?

Use the `final` keyword



```java
final class A
{
    final int i;
    public A(int i)
    {
        this.i = i;
    }
}
```

The above class is immutable because:

- The instance variable of the class is final i.e. we cannot change the value of it after creating an object.
- The class is final so we cannot create the subclass.
- There is no setter methods i.e. we have no option to change the value of the instance variable.

Reference: https://www.javatpoint.com/how-to-create-immutable-class

How to create Immutable Class?

Use the `final` keyword

Further study:

https://www.javatpoint.com/final-keyword