

Constructor

PREVIEW / RECAP:

We have seen how to create or call a method, static/non-static methods. Now let's understand the concept of Constructor.

What is a Constructor?

A constructor is a block of codes similar to the method. It is called when an instance of the class is created. It is a special type of method which is used to initialize the object.

Now you might have a question in your mind: if a constructor is similar to method why is it called a constructor? Because it constructs the values at the time of object creation.

Is it compulsory to create a Constructor in java program? Like the Main Method?

NO. It's ok if we don't create constructor in java programs. As java creates a **Default Constructor** for each class. So every class has at least 1 constructor which is **Default Constructor**.

Rules of Creating Constructor

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Types of Constructor

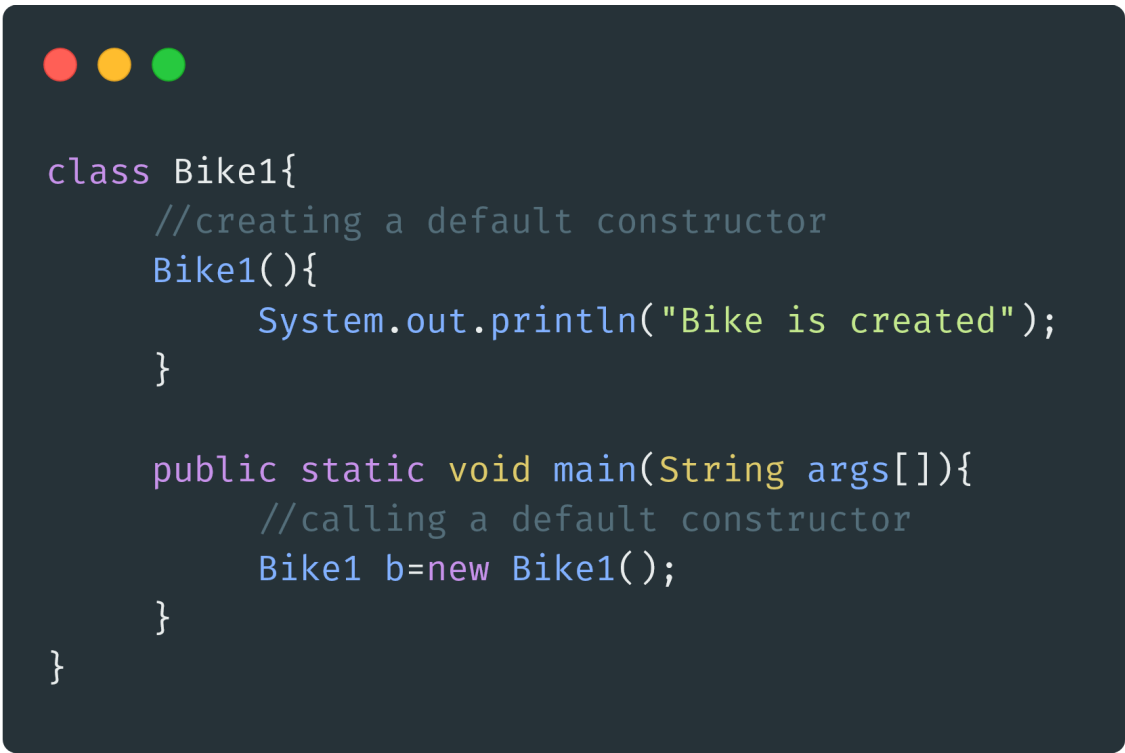
There are 2 types of constructors:

1. No Argument Constructor (Default constructor)
2. Parameterized constructor

No Argument Constructor

In this type of constructor, we create a constructor but we don't pass any arguments in it.

E.g.



```
class Bike1{
    //creating a default constructor
    Bike1(){
        System.out.println("Bike is created");
    }

    public static void main(String args[]){
        //calling a default constructor
        Bike1 b=new Bike1();
    }
}
```

Output

Bike is created

As, we can see we have created a Constructor for Bike1 class. Note that our Constructor has the same name as that of the class. Also we have not passed any arguments to the constructor.

Also, we just created an instance of class and the constructor has been called. We didn't called the constructor explicitly.

Parameterized Constructor

In this type of constructor, we create a constructor but we don't pass any arguments in it.

E.g.

```
//Java Program to demonstrate the use of the parameterized constructor.
class Student4{

    int id;
    String name;

    //creating a parameterized constructor
    Student4(int i,String n){
        id = i;
        name = n;
    }

    //method to display the values
    void display(){
        System.out.println(id+" "+name);
    }

    public static void main(String args[]){
        //creating objects and passing values
        Student4 s1 = new Student4(111,"Karan");
        Student4 s2 = new Student4(222,"Aryan");
        //calling method to display the values of object
        s1.display();
        s2.display();
    }
}
```

Output

111 Karan

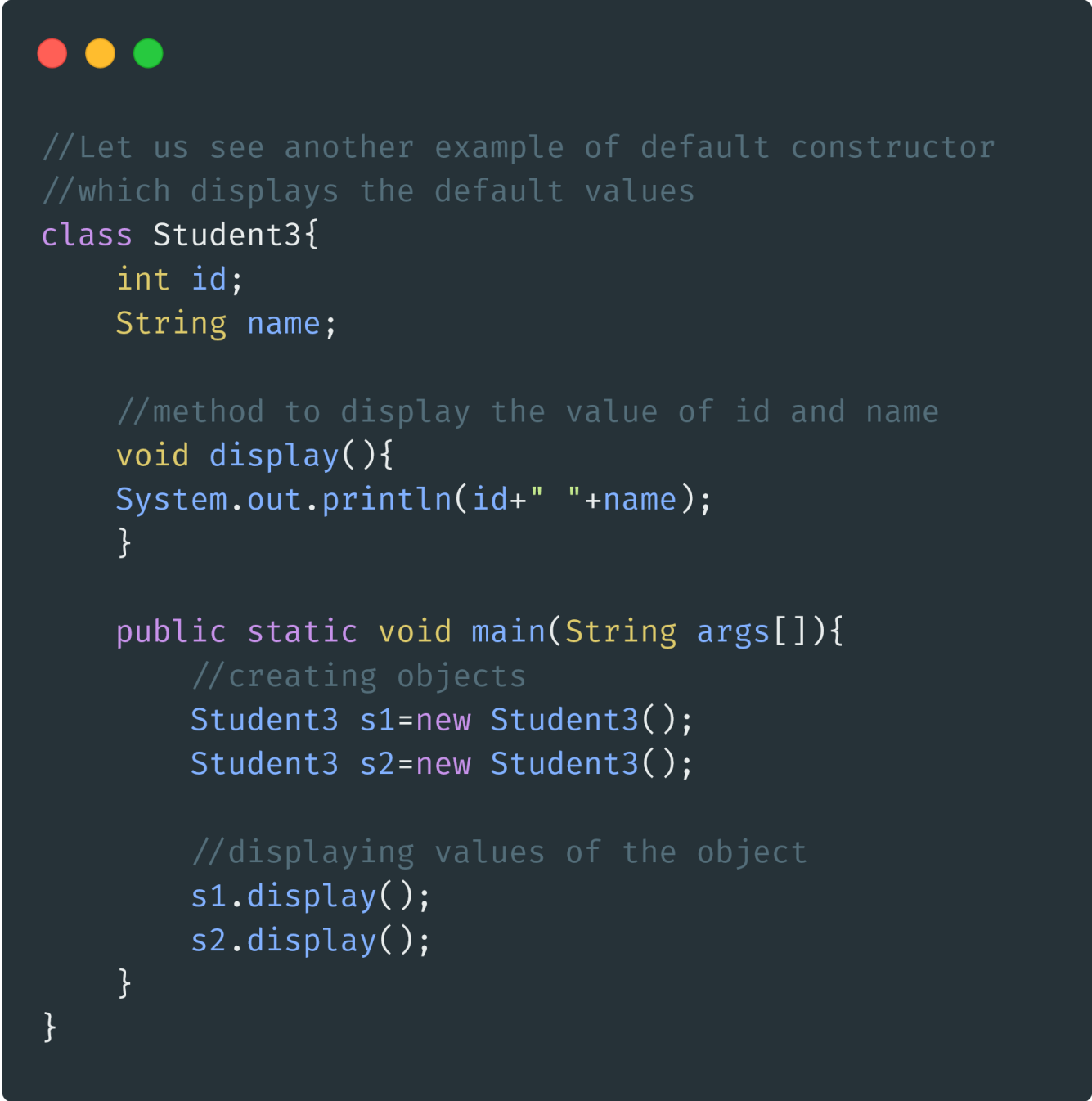
222 Aryan

We have created 2 instances of class student4 as s1 and s2 respectively. We passed different arguments to different objects and as we can see both objects (S1 and S2) are printing the values passed to them as parameters.

Default Constructor

In this example, we will show you that even if we don't create a constructor. Some value gets assigned to instance variables. This task of assigning default values to instance variables is done by Default Constructor.

E.g.



```
//Let us see another example of default constructor
//which displays the default values
class Student3{
    int id;
    String name;

    //method to display the value of id and name
    void display(){
        System.out.println(id+" "+name);
    }

    public static void main(String args[]){
        //creating objects
        Student3 s1=new Student3();
        Student3 s2=new Student3();

        //displaying values of the object
        s1.display();
        s2.display();
    }
}
```

Output

0 null

0 null

As we can see, we have created 2 objects of class Student3. But as we have not created any constructor, the default constructor gets called and it assigned default values to each instance variable in both the objects.

Great!! We are done with the basics of constructor. But wait... As we know we can overload a method. So can we also override a constructor?? Yes. We can do constructor overriding. Let's have a look at it...

Constructor Overloading

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

E.g.

```
//Java program to overload constructors
class Student5{
    int id;
    String name;
    int age;

    //creating two arg constructor
    Student5(int i,String n){
        id = i;
        name = n;
    }

    //creating three arg constructor
    Student5(int i,String n,int a){
        id = i;
        name = n;
        age=a;
    }

    void display(){
        System.out.println(id+" "+name+" "+age);
    }

    public static void main(String args[]){
        Student5 s1 = new Student5(111,"Karan");
        Student5 s2 = new Student5(222,"Aryan",25);
        s1.display();
        s2.display();
    }
}
```

Output

111 Karan 0

222 Aryan 25

Great!!! So now you know the constructor.