OBJECT ORIENTED PROGRAMMING USING JAVA

OUTLINE

- Constructors in java

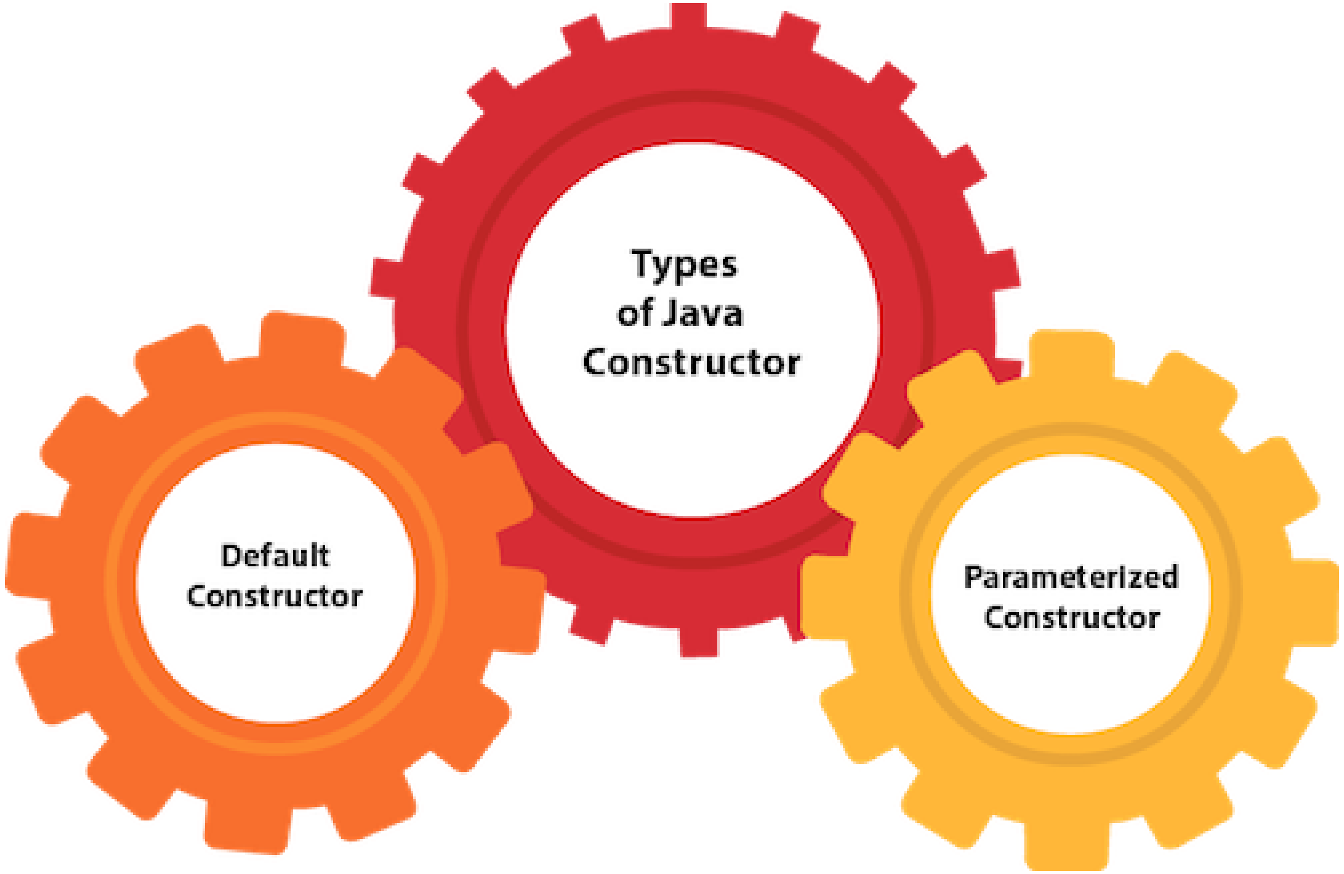
CONSTRUCTOR

- In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.
- It is a special type of method which is used to initialize the object.
- Every time an object is created using the new() keyword, at least one constructor is called.
- It calls a default constructor if there is no constructor available in the class.
- There are two types of constructors in Java:
    - Default (no-arg constructor)
    - parameterized constructor

RULES FOR CREATING JAVA CONSTRUCTOR

- There are two rules defined for the 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.

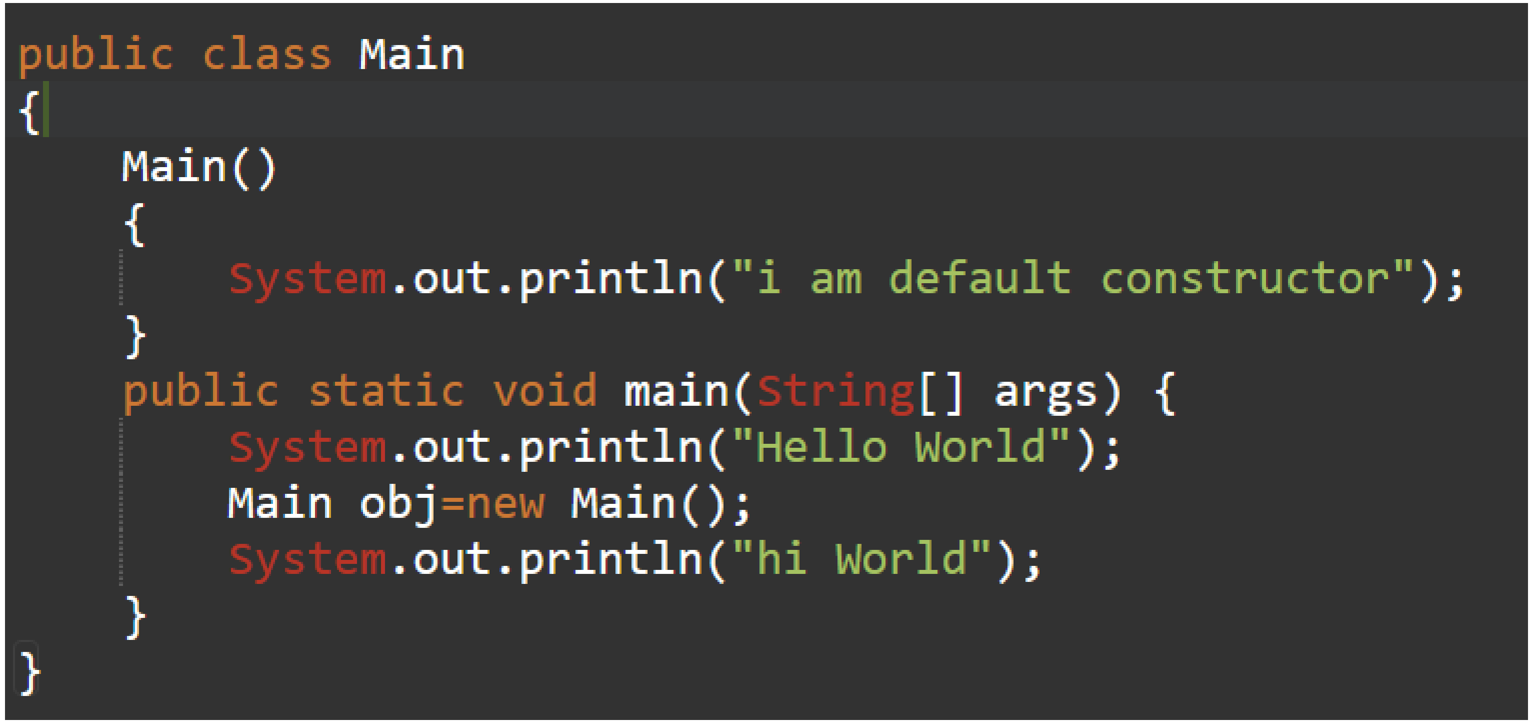
TYPES OF JAVA CONSTRUCTORS



JAVA DEFAULT CONSTRUCTOR

- A constructor is called "Default Constructor" when it doesn't have any parameter.

Example of default constructor

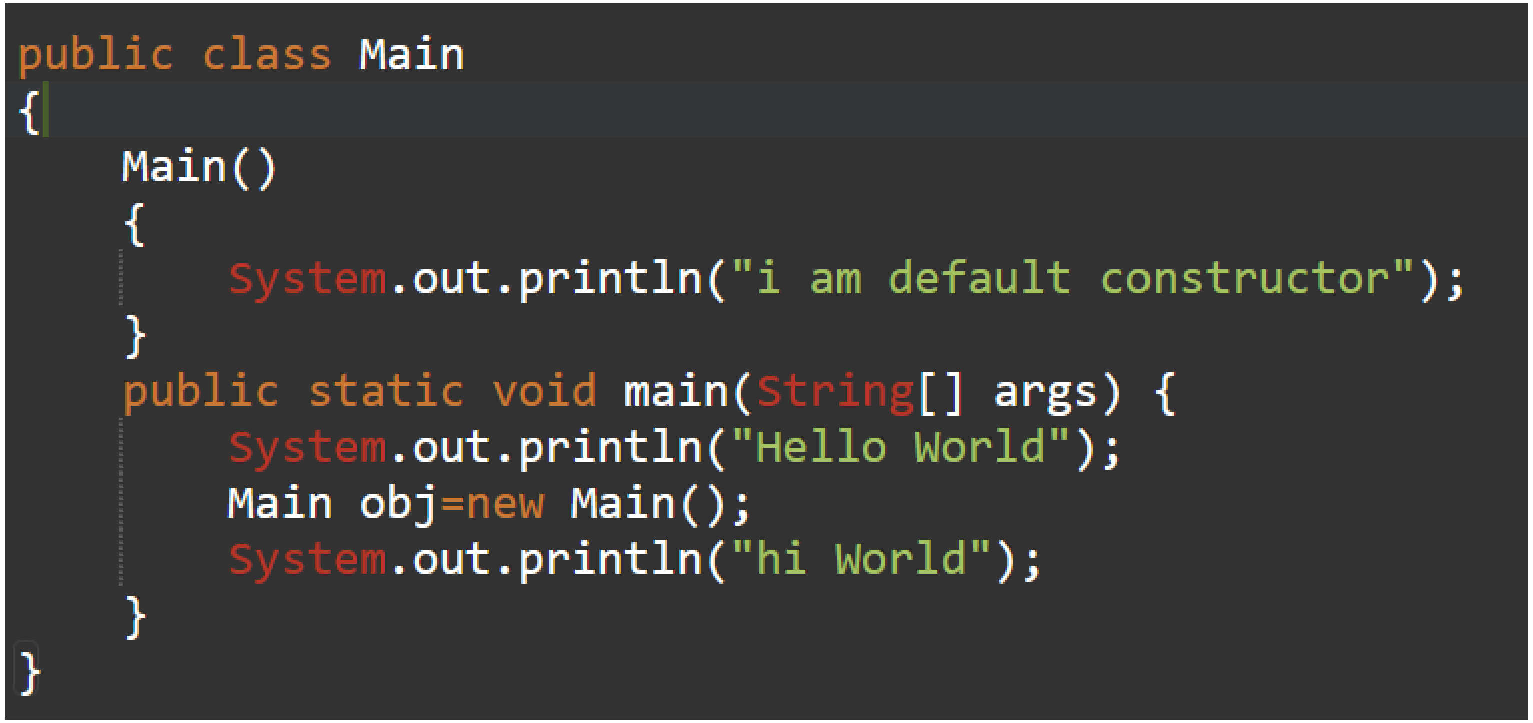


```
public class Main
{
    Main()
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main();
        System.out.println("hi World");
    }
}
```

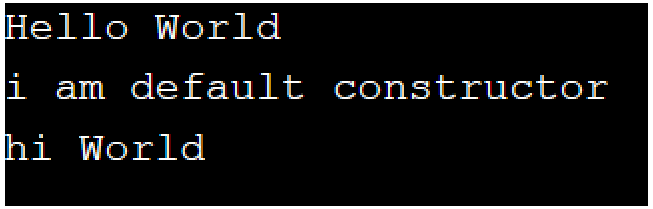
JAVA DEFAULT CONSTRUCTOR

- A constructor is called "Default Constructor" when it doesn't have any parameter.

Example of default constructor



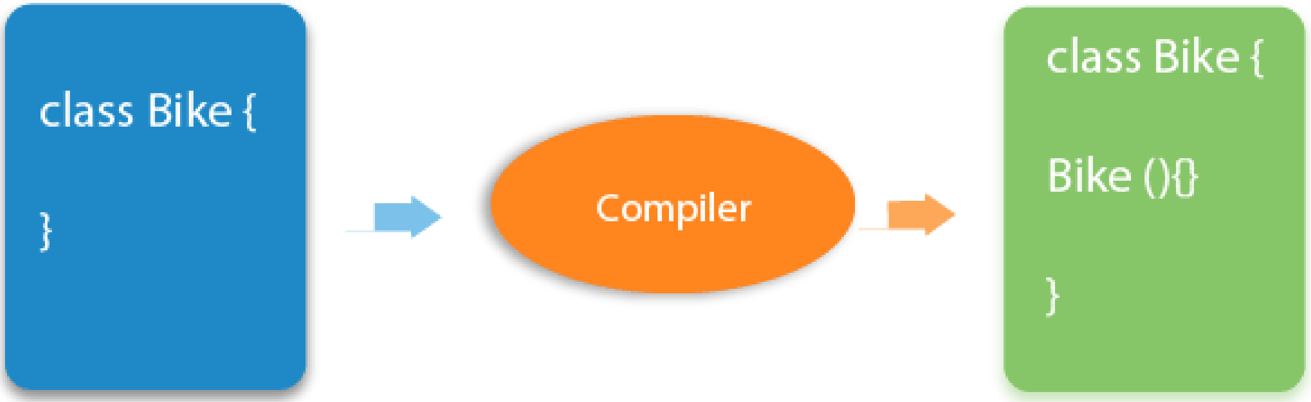
```
public class Main
{
    Main()
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main();
        System.out.println("hi World");
    }
}
```



```
Hello World
i am default constructor
hi World
```

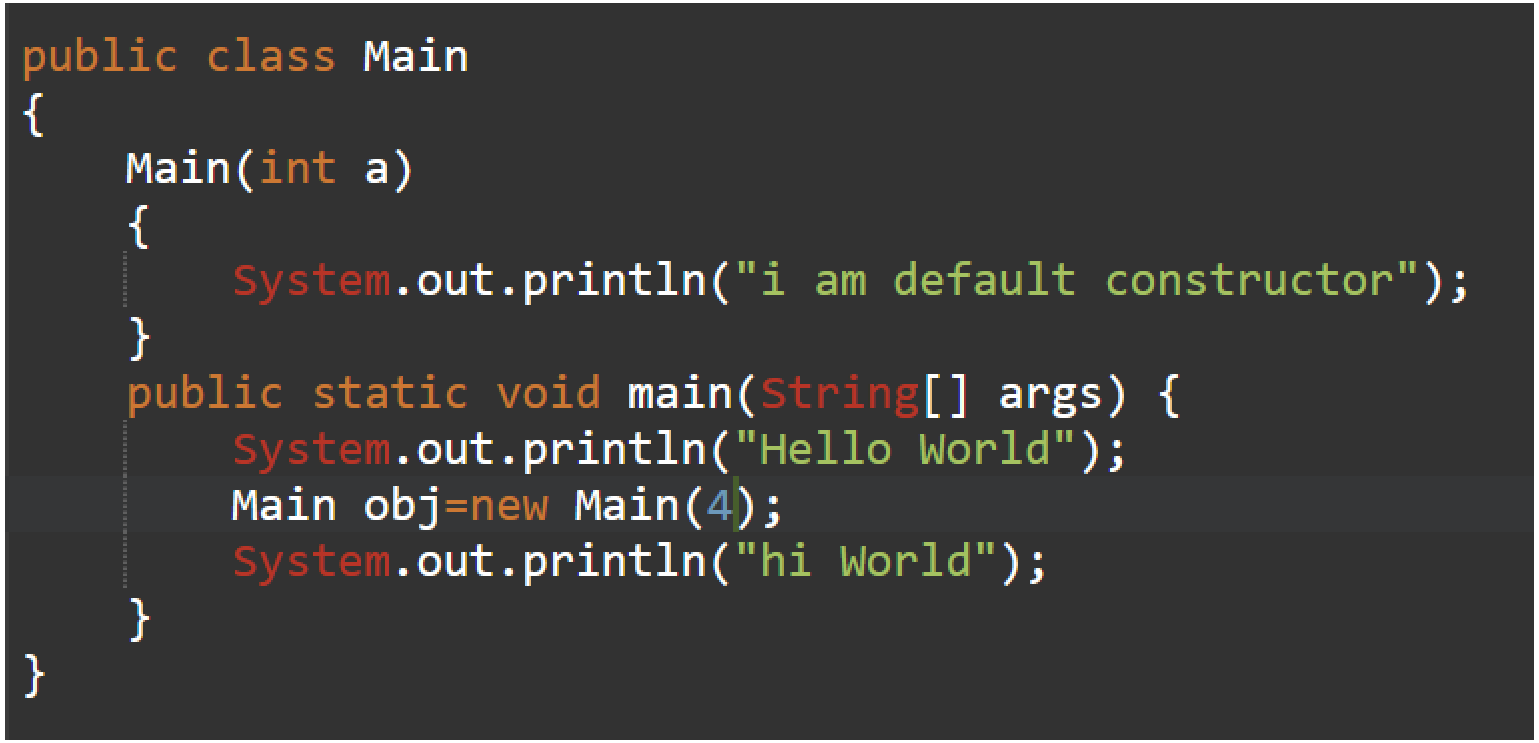
JAVA DEFAULT CONSTRUCTOR

- If there is no constructor in a class, compiler automatically creates a default constructor..



JAVA PARAMETERIZED CONSTRUCTOR

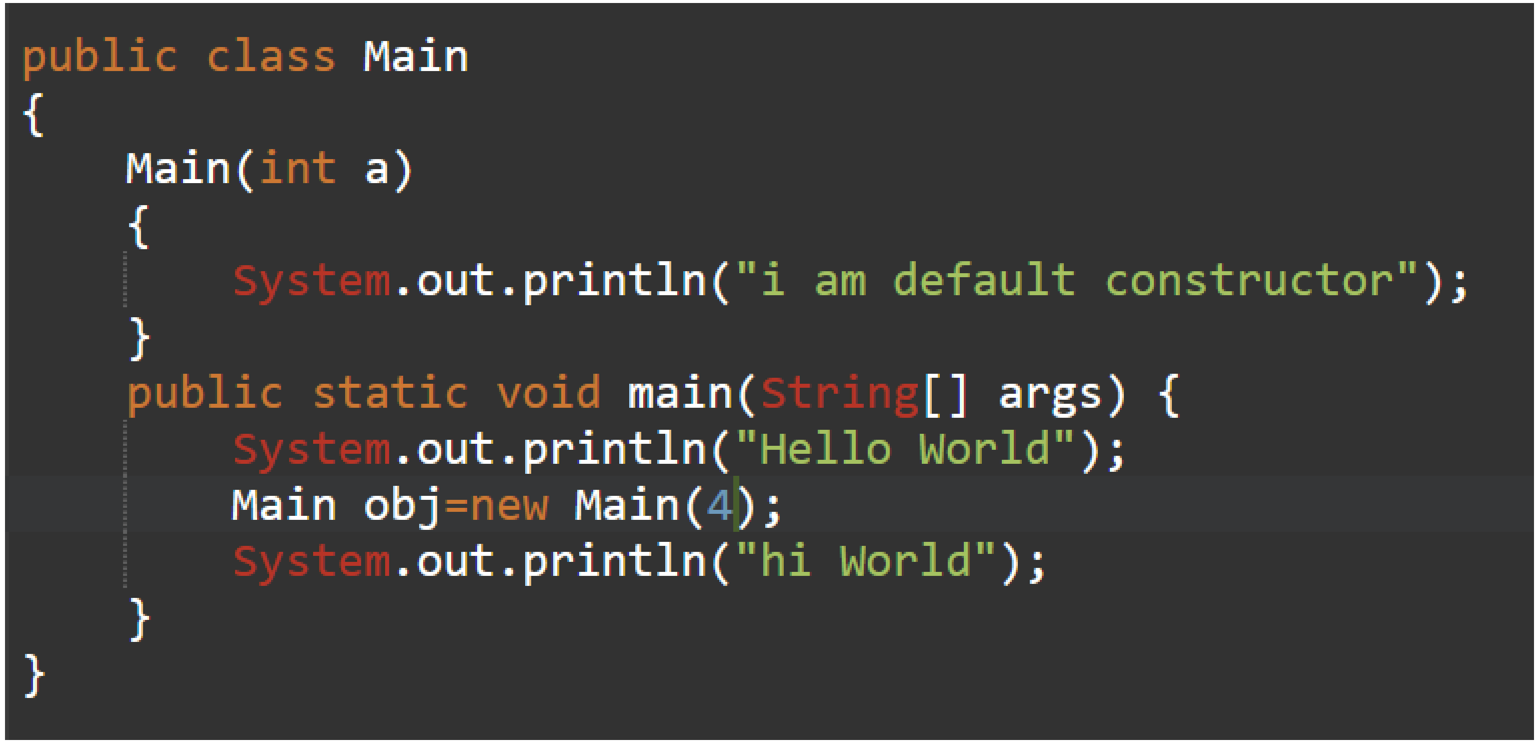
- A constructor which has a specific number of parameters is called a parameterized constructor.
- The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.



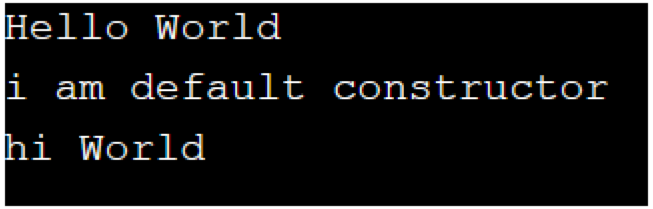
```
public class Main
{
    Main(int a)
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main(4);
        System.out.println("hi World");
    }
}
```

JAVA PARAMETERIZED CONSTRUCTOR

- A constructor which has a specific number of parameters is called a parameterized constructor.
- The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.



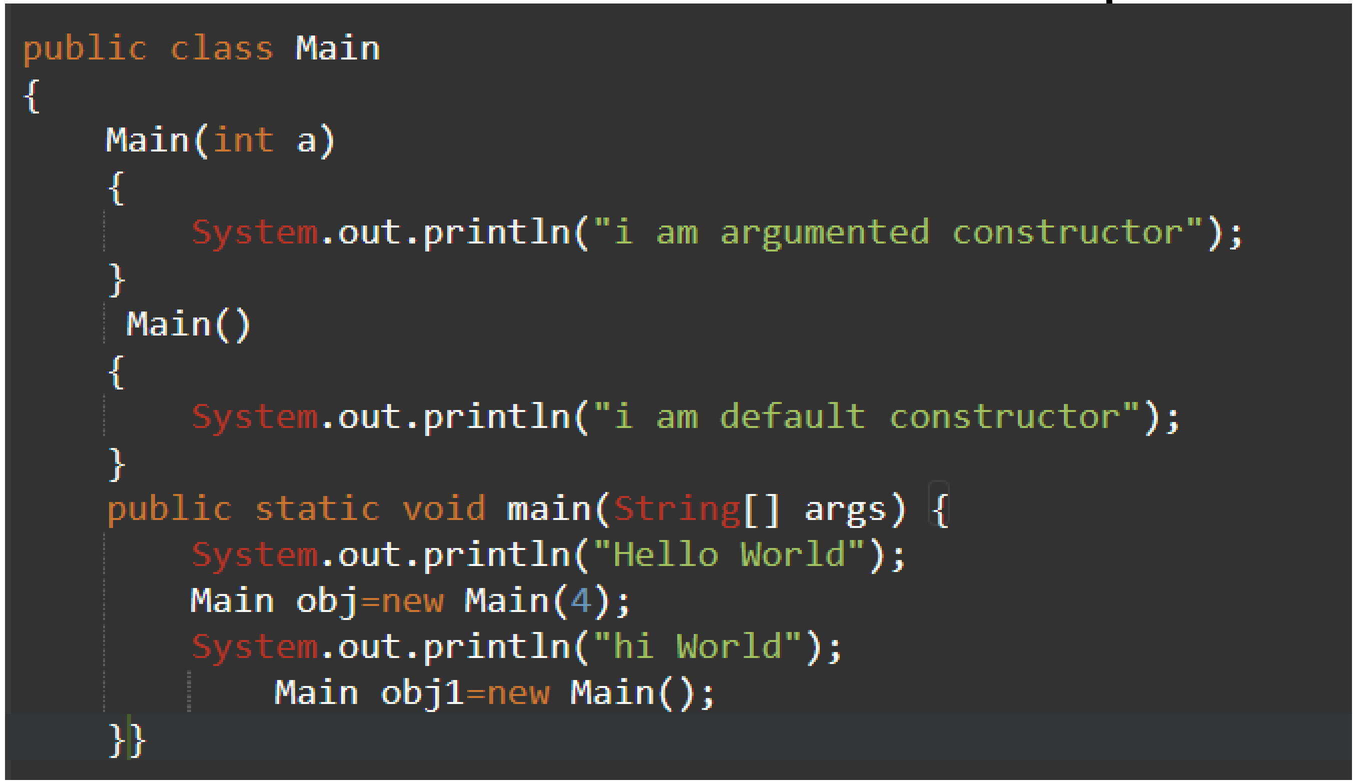
```java
public class Main
{
    Main(int a)
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main(4);
        System.out.println("hi World");
    }
}
```



```
Hello World
i am default constructor
hi World
```

CONSTRUCTOR OVERLOADING IN JAVA

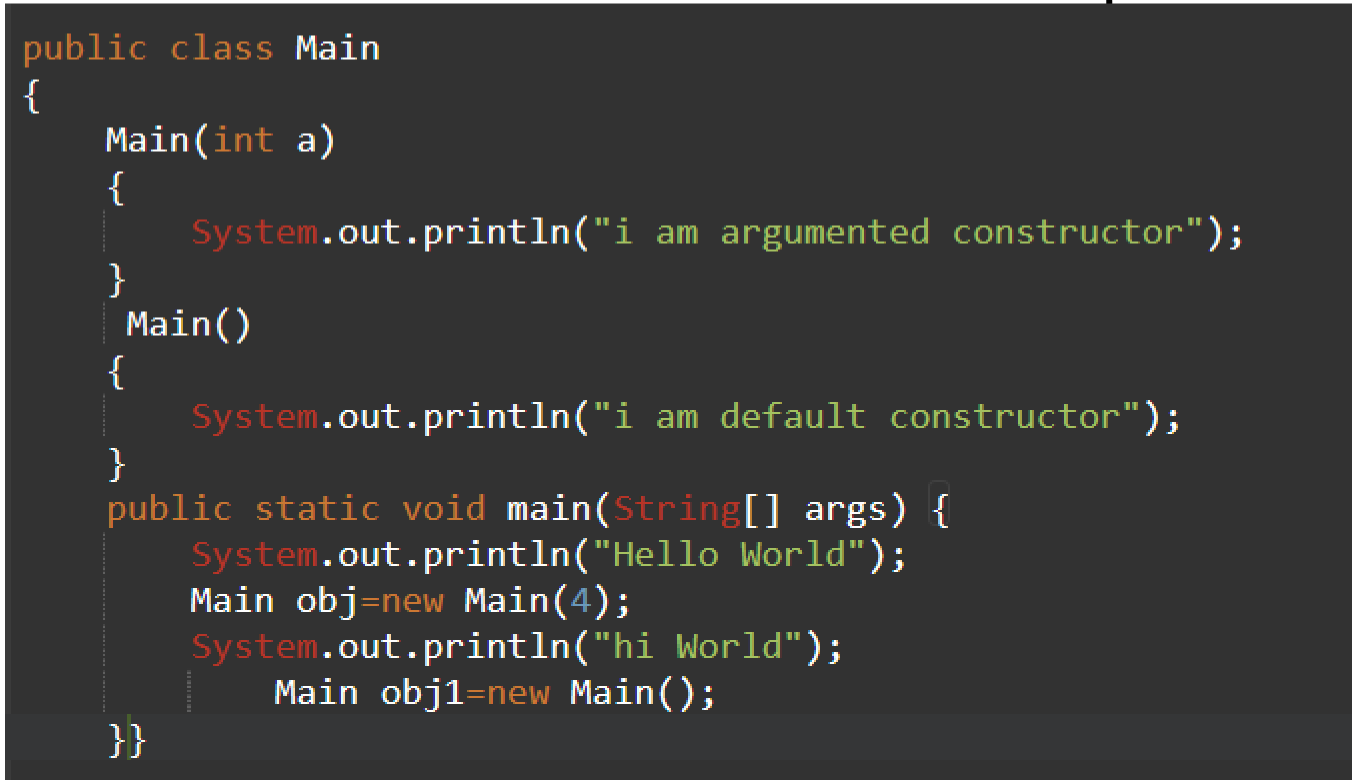
- In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.
- Constructor overloading in Java is a technique of having more than one constructor with different parameter lists.



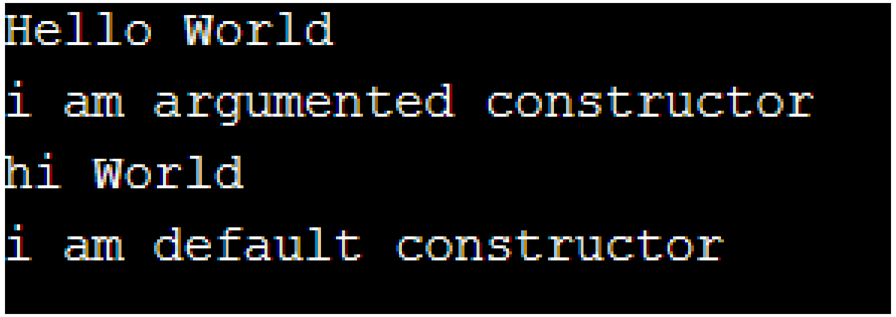
```
public class Main
{
    Main(int a)
    {
        System.out.println("i am argumented constructor");
    }
     Main()
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main(4);
        System.out.println("hi World");
            Main obj1=new Main();
    }}
```

CONSTRUCTOR OVERLOADING IN JAVA

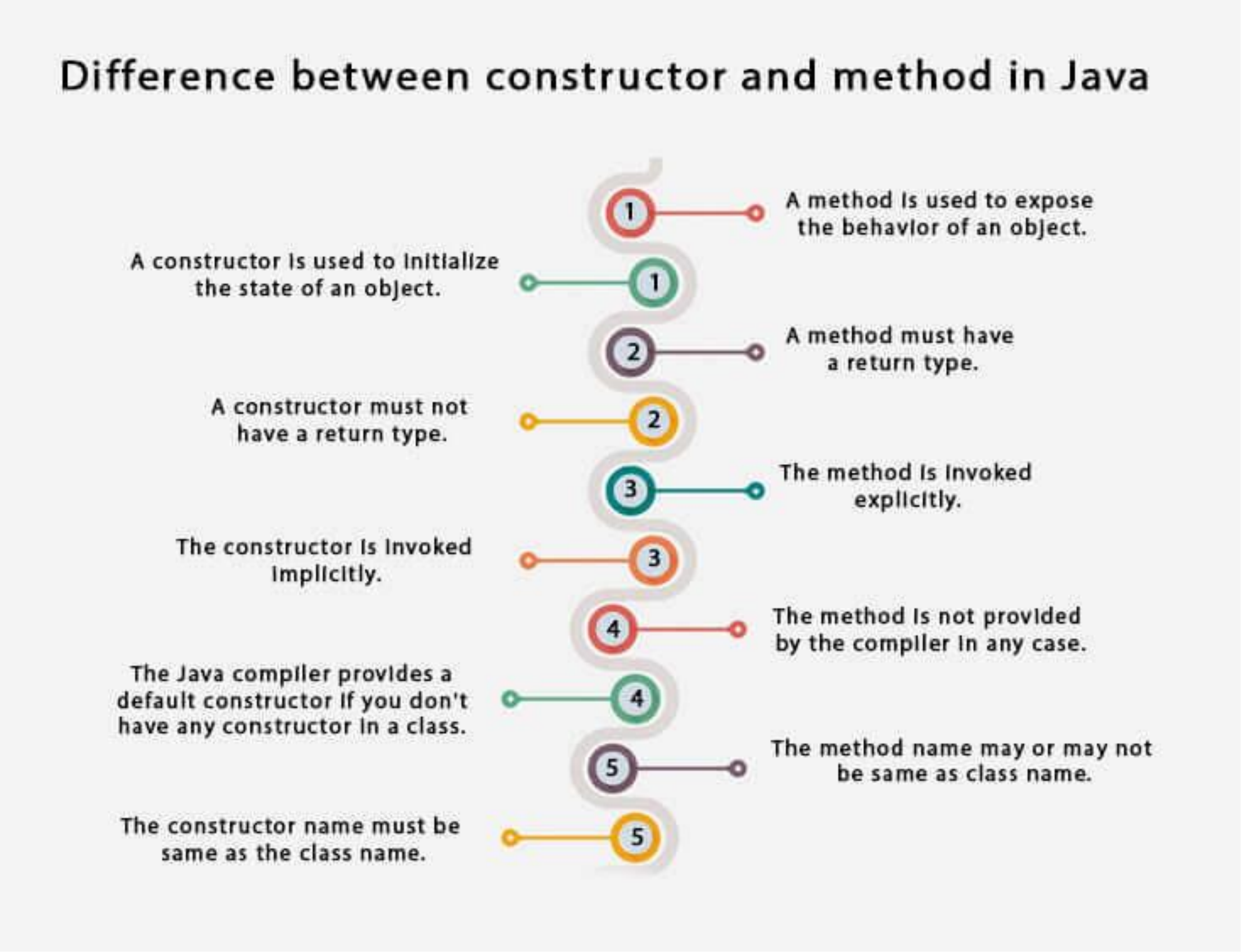
- In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.
- Constructor overloading in Java is a technique of having more than one constructor with different parameter lists.



```
public class Main
{
    Main(int a)
    {
        System.out.println("i am argumented constructor");
    }
     Main()
    {
        System.out.println("i am default constructor");
    }
    public static void main(String[] args) {
        System.out.println("Hello World");
        Main obj=new Main(4);
        System.out.println("hi World");
            Main obj1=new Main();
    }}
```



```
Hello World
i am argumented constructor
hi World
i am default constructor
```



Difference between constructor and method in Java

1 A method is used to expose the behavior of an object.

1 A constructor is used to initialize the state of an object.

2 A method must have a return type.

2 A constructor must not have a return type.

3 The method is invoked explicitly.

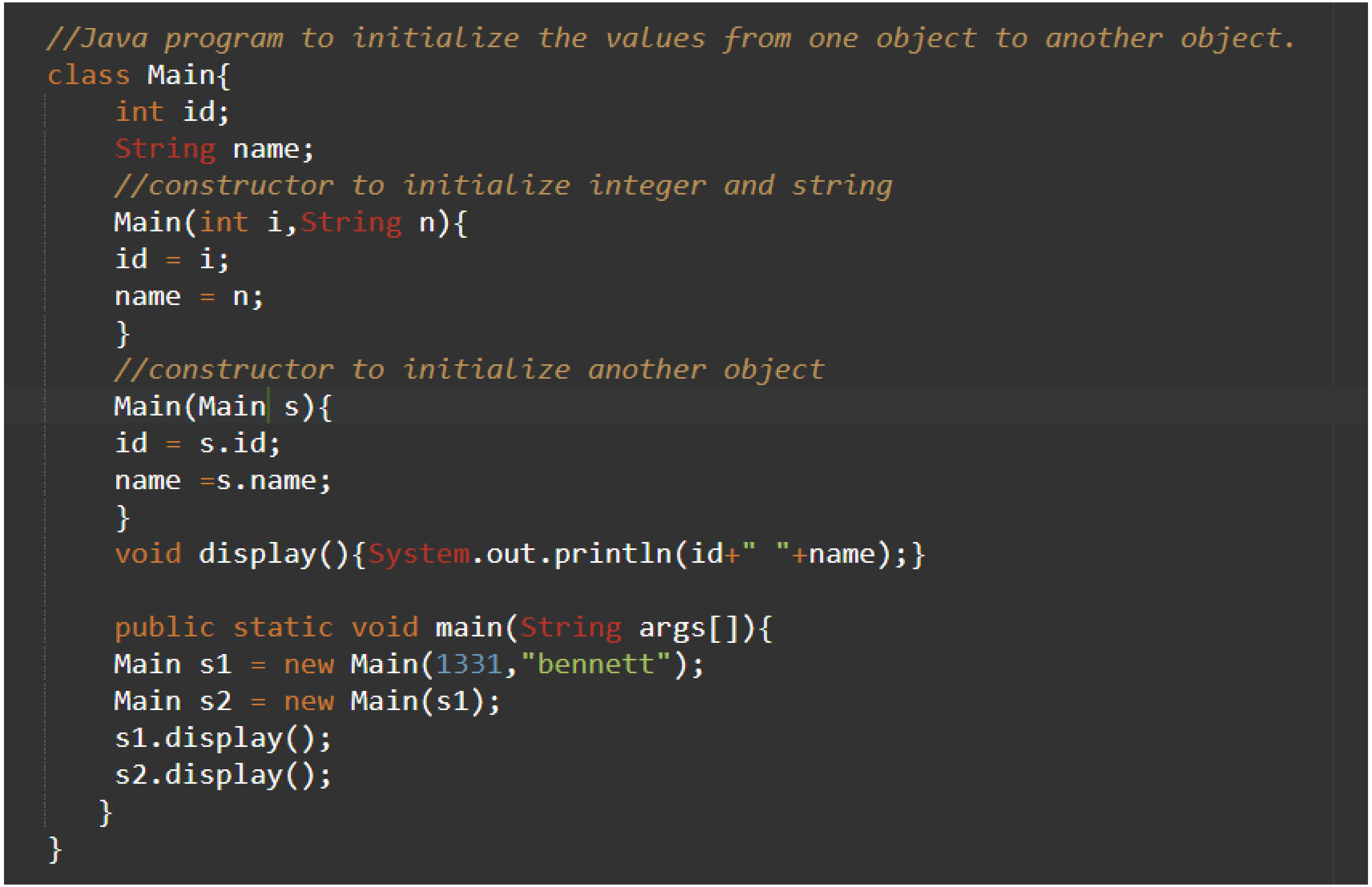
3 The constructor is invoked implicitly.

4 The method is not provided by the compiler in any case.

4 The Java compiler provides a default constructor if you don't have any constructor in a class.

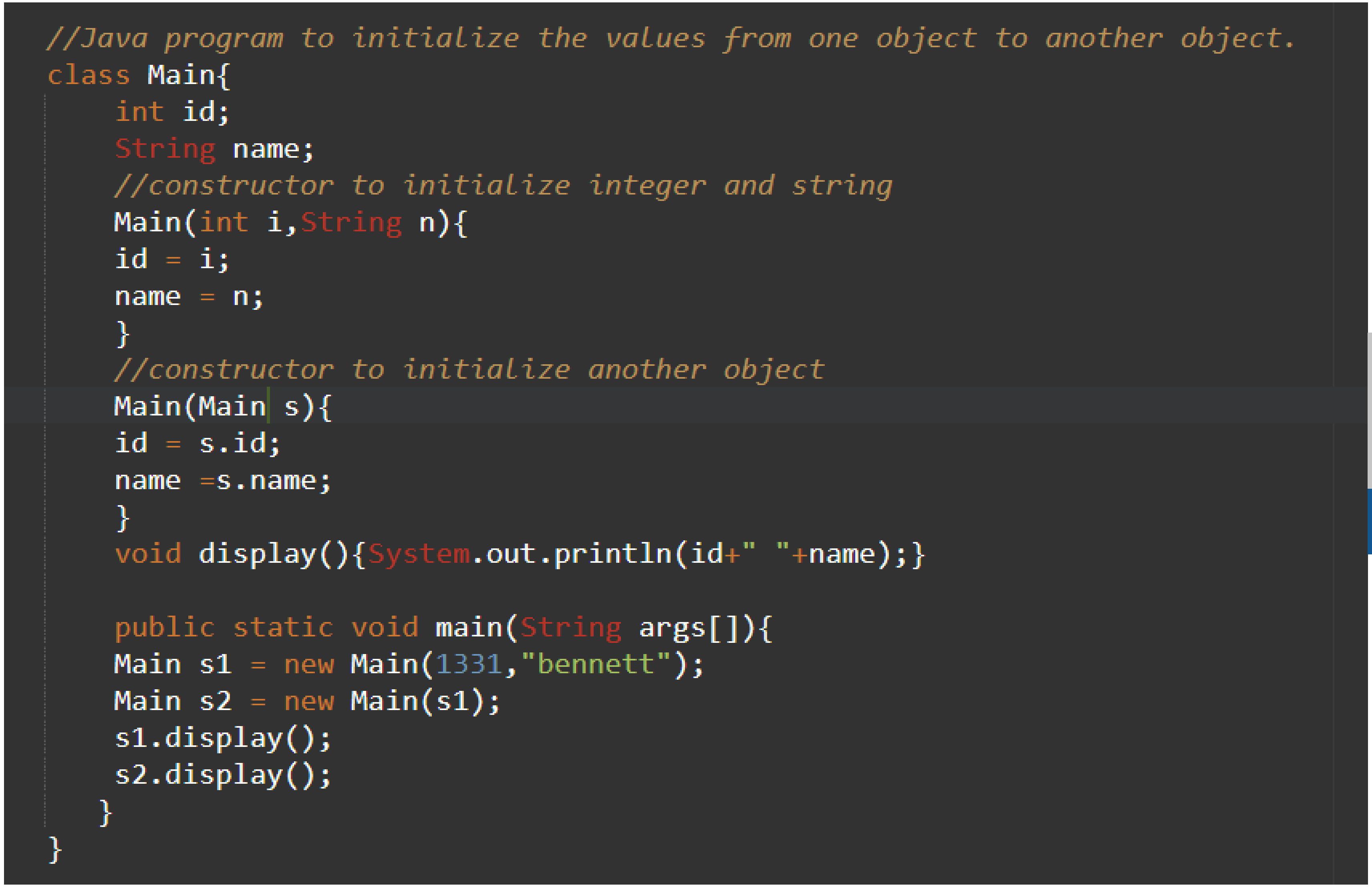
5 The method name may or may not be same as class name.

5 The constructor name must be same as the class name.



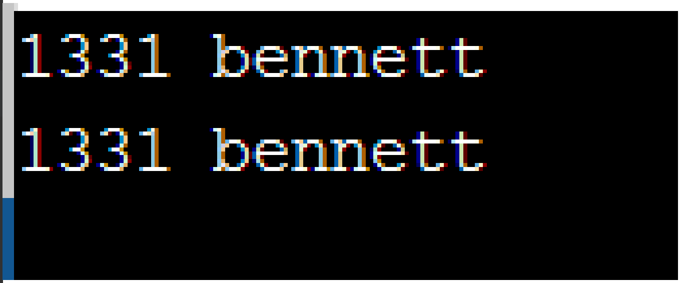
```
//Java program to initialize the values from one object to another object.
class Main{
    int id;
    String name;
    //constructor to initialize integer and string
    Main(int i,String n){
    id = i;
    name = n;
    }
    //constructor to initialize another object
    Main(Main s){
    id = s.id;
    name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Main s1 = new Main(1331,"bennett");
    Main s2 = new Main(s1);
    s1.display();
    s2.display();
   }
}
```



```java
//Java program to initialize the values from one object to another object.
class Main{
    int id;
    String name;
    //constructor to initialize integer and string
    Main(int i,String n){
    id = i;
    name = n;
    }
    //constructor to initialize another object
    Main(Main s){
    id = s.id;
    name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Main s1 = new Main(1331,"bennett");
    Main s2 = new Main(s1);
    s1.display();
    s2.display();
   }
}
```



```
1331 bennett
1331 bennett
```

THANK YOU

?