Object Oriented Programming

Java: Constructors

Bikash Karmokar
Lecturer,
Department of CSE,
Bangladesh University

The Constructor

When you create an object of a class using the new operator, you invoke a special kind of method called constructor.

A constructor is used to perform initializations of instance variables of a class

Constructor is a special method that is called when you use the new operator

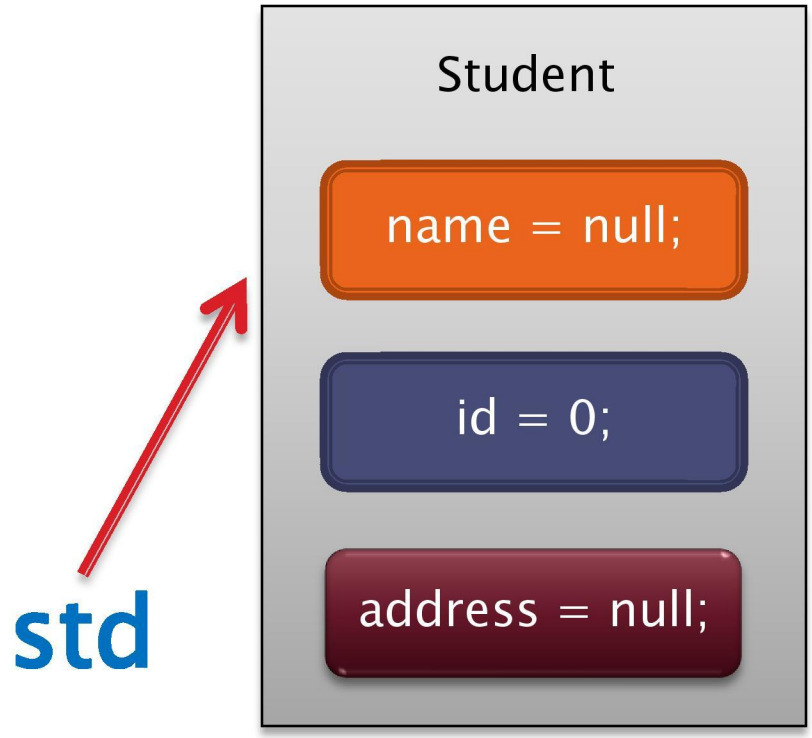
Circle c = new Circle();

If you don't define a constructor, java creates a default constructor that can assign default values for all your variables in the class

What does default constructor do?

```
class Student
{
        String name;
        int id;
        String address;
 }

Student std = new Student();
```



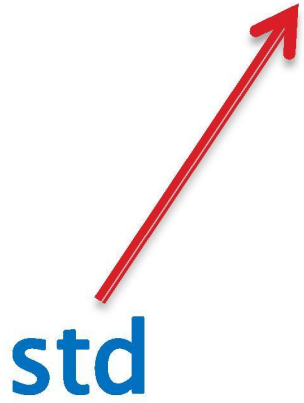
Constructor create objects and give their instance variables default values.

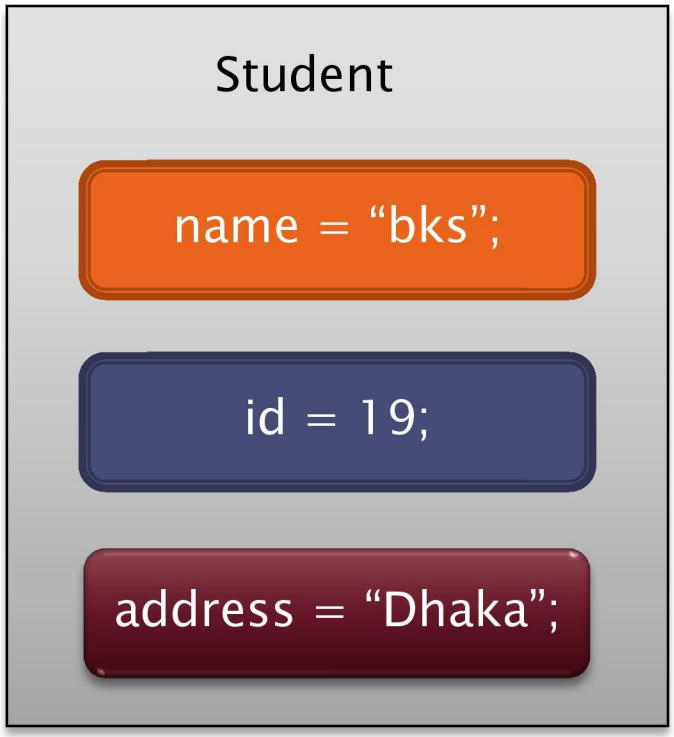
The Constructor

```
class Student
{
        String name;
        int id;
        String address;

        Student()
        {
          name = "bks"; id =19;
          address = "Dhaka";
        }

 }
```



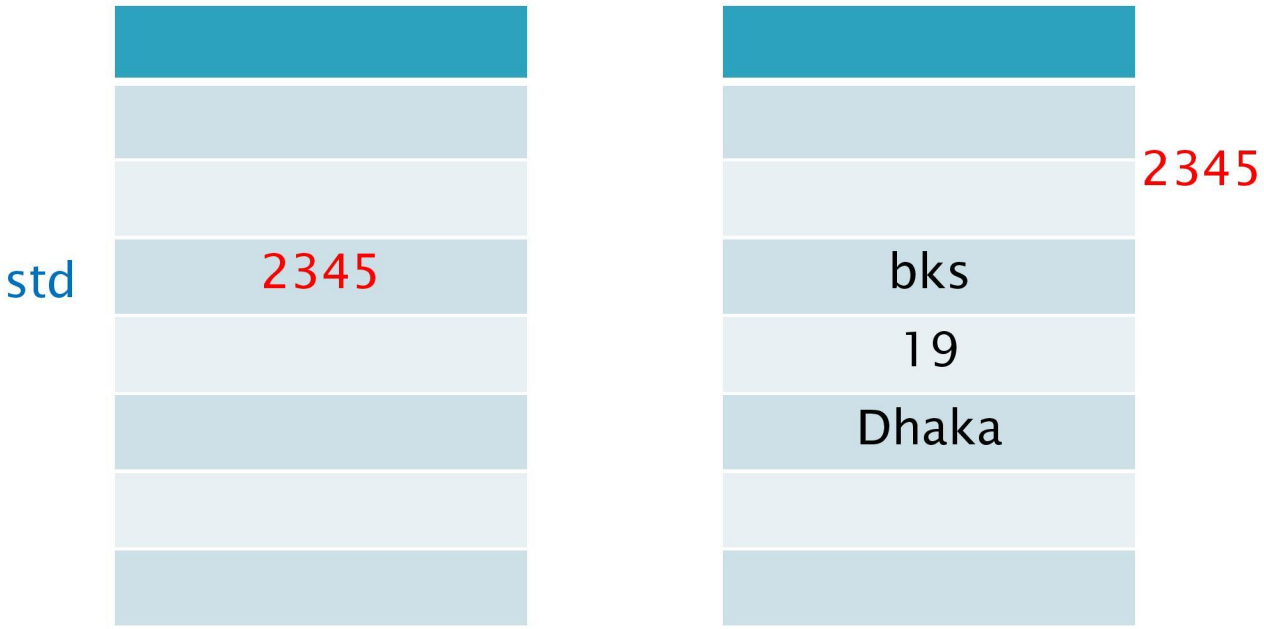



```
Student std = new Student();
```

A constructor is a special method with the same name as the class.

Memory details behind a constructor

Student std = new Student();



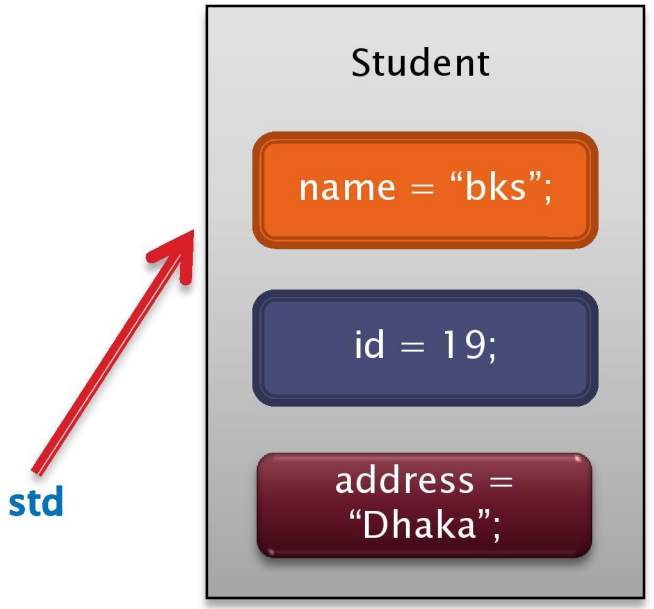
A parameterized Constructor

```
class Student
{
        String name;
        int id;
        String address;

        Student(String s, int a, String add)
        {
          name = s; id =a;
          address = add;
        }

 }
                    public static void main(String[ ] args)
                    {

                        Student std = new Student("bks",19,"Dhaka");
                    }
```



Constructor

When you create an object, its constructor is called before any methods.

When you don't define a constructor, java provides its own default constructor.

You define your own constructor to decide what values should be given to the instance variables.

Constructor's don't return values, not even void.

If you define your constructor, java does not provide the default constructor.

The constructor name must be same as class name.

If you define a constructor which is parameterized constructor and if you do not define a default constructor, then java will not generate any default constructor for you.

Constructor Overloading

```
class Student
{
    String name;
    int id;
    String address;

    Student()
    {
      name="NA"; id=19; address="NA"
    }

    Student(String s)
    {
       name=s; id=0; address=null;
    }

    Student(String s, int a, String add)
    {
       name=s; id=a; address=add;
    }

}
```

```
public static void main(String[] args)
{
Student s1 = new Student();

Student s2 = new Student("std");

Student s3 = new Student("bks",19,"Dhaka");
}
```

All the constructors in the class have the same name and so are overloaded.

Each constructor has either a different number of parameters or a parameter whose type is different from those in the other constructors.

A small issue

```
class Student
{
    String firstName;

    String lastName;


    Student(String  f, String  l)
    {
     firstName =f; lastName =l;
    }


    Student(String  f)
    {
     firstName ="Mr. "+f; lastName ="NA";
    }

 }
```

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

  Student s2 = new Student("R", "P");

  Student s3 = new Student("R");

}
```

The firstName is not consistent in both the constructors.

A small issue

```
class Student
{
    String firstName;

    String lastName;


    Student(String  f, String  l)
    {
     firstName =f; lastName =l;
    }


    Student(String  f)
    {
     this( f, "NA" );
    }

}
```

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

  Student s2 = new Student("R", "P");

  Student s3 = new Student("R");

}
```

No matter how many constructor you define, ensure they are consistent.

Reference:

Java : The Complete Reference
Seventh Edition
Herbert Schildt

Thank You