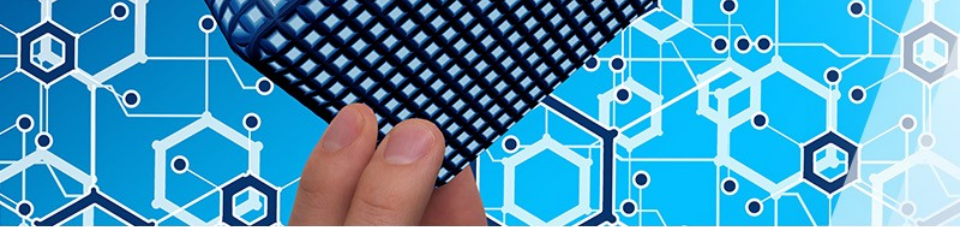




Chapter 5: Constructors

- Definitions of Constructors
- Rules of using Constructors
- Types of constructors
 - Default Constructors
 - Parametrized Constructors
 - Overloaded Constructors



Definitions of Constructors

- A constructor that can be **used to initialize an object** of a class when the object is created.
- In fact, Java *requires* a constructor call for *every* object that's created.
- **Keyword new requests memory from the system to store an object**, then calls the corresponding class's constructor to initialize the object.
- The call is indicated by the parentheses after the class name. A constructor *must* have the *same name* as the class.



- By default, the compiler provides a **default constructor** with *no parameters* in any class that does *not* explicitly include a constructor.
- When a class has only the default constructor, its instance variables are initialized to their *default values*.



Uses of Constructors

- For initializing instance variables.

```
public class Hi{
    String name;
    public Hi () {
        this.name="CoSc2044 class\n";
    }
    public static void main(String[] args)
    {

        Hi hiobj=new Hi ();
        System.out.println(hiobj);
    }
}
```



Rules of using Constructors

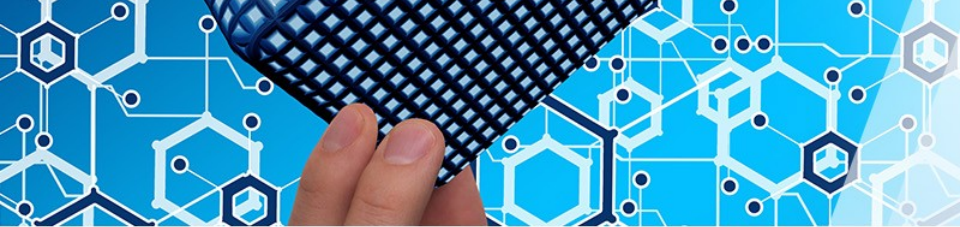
- Constructors have the **same name as the class name**.
- Constructors have access modifiers(package, private, protected, public)
- Constructors **do not have return types**.
- Constructors can be invoked only during object creation or from other constructors using this keyword.



Types of Constructors

There are three types of constructors:

1. Default Constructors
2. No-argument Constructors
3. Parameterized Constructors



Default Constructors

- If you do not implement any constructor in your class , Java compiler inserts a default constructor into your code on your behalf.
- By default, the compiler provides a **default constructor** with *no parameters* in any class that does *not* explicitly include a constructor.
- When a class has only the default constructor, its **instance variables are initialized to their *default values*.**

```
public class DefaultConst
{
    public static void main(String[]arg)
    {
        DefaultConst dc = new DefaultConst();
        System.out.println(Default Constructor");
    }
}
```

DefaultConst.java

Compiler creates
default constructor

```
public class DefaultConst
{
    public DefaultConst() {}

    public static void main(String[]arg)
    {
        DefaultConst dc = new DefaultConst();
        System.out.println(Default Constructor");
    }
}
```

DefaultConst.class



- *If you declare any constructor for a class, the Java compiler will not create a default constructor for that class.*
- The body of default constructor is empty.



No-Arg Constructor

- Constructor with no arguments is known as no-arg constructor.
- The signature is same as default constructor, however the body can have any code



```
public class NoArg {  
    public NoArg(){  
        //it can be empty or  
        //you can write any code  
        System.out.println("This is a no argument constructor");  
    }  
    public static void main(String [] args) {  
        new NoArg();  
    }  
}
```

```
C:\MyJavaApplications\Chapter 5>javac NoArg.java
```

```
C:\MyJavaApplications\Chapter 5>java NoArg
```

```
This is a no argument constructor
```

- Constructor with arguments (or Parameters) is known as **Parameterized constructor**.
- Constructors gets invoked after creating objects.

Parametrized Constructors example

```
public class Student{
    int age;
    String name;
    double cgpa;
    //parametrized constructor
    public Student(int sage,String sname,double scgpa){
        age=sage;
        name=sname;
        cgpa=scgpa;
    }
    // method to display information
    public void info(){
        System.out.println("Stud age:"+age+" Stud name:"+name+" Stud age:"+cgpa);
    }

    public static void main(String [] args){
        Student s1=new Student(19,"Yared", 3.56);
        s1.info();
    }
}
```



- It is possible to have more than one constructors in one program.

Constructor Overloading

- Constructor overloading is a concept of having more than one constructor with different parameters list.

```
public class Demo{  
    Demo(){  
        .....  
    }  
    Demo(String s){  
        .....  
    }  
    Demo(int i){  
        .....  
    }  
}
```

Constructor
without
Argument

Constructor
with String
Argument

Constructor
with int
Argument



End of Chapter V



CHAPTER SIX