Constructors

Constructors are class members used to initialize non-static member variable during object creation.

There are of two types:
1. Compiler defined.
2. User defined.

Complier defined constructor:

Constructor defined by the compiler is known as compiler defined constructor. Compiler will define the constructor only when if user has not defined the constructor. Compiler defined constructor is also called as default constructor.

User-Defined constructor:

Constructor defined by the user is known as user-defined constructor.
There are two types of user-defined constructor:
1. No argument constructor.
2. Parameterised constructor.

No argument constructor

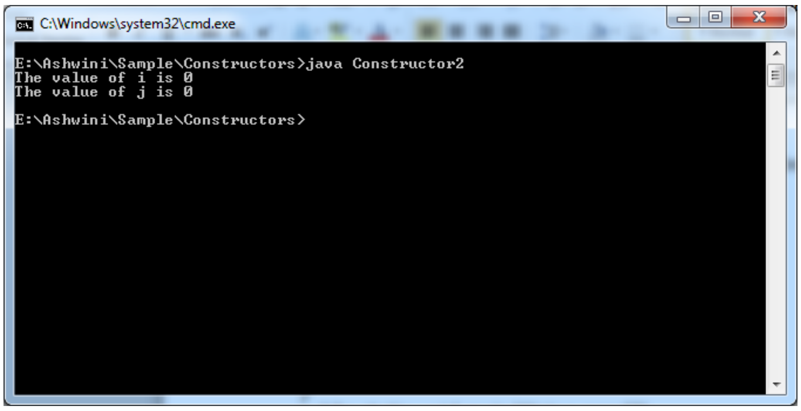
Constructor defined by the used without parameter is known as no argument constructor.

Parameterised constructor

Constructor defined with parameter are known as parameterised constructor. Constructor name and class name must be same. Constructor doesn't return a value. If we declare constructor with return type, then it is considered as a method.

Syntax:
```
Constructor_name(Parameters)
{
--
---
}
```

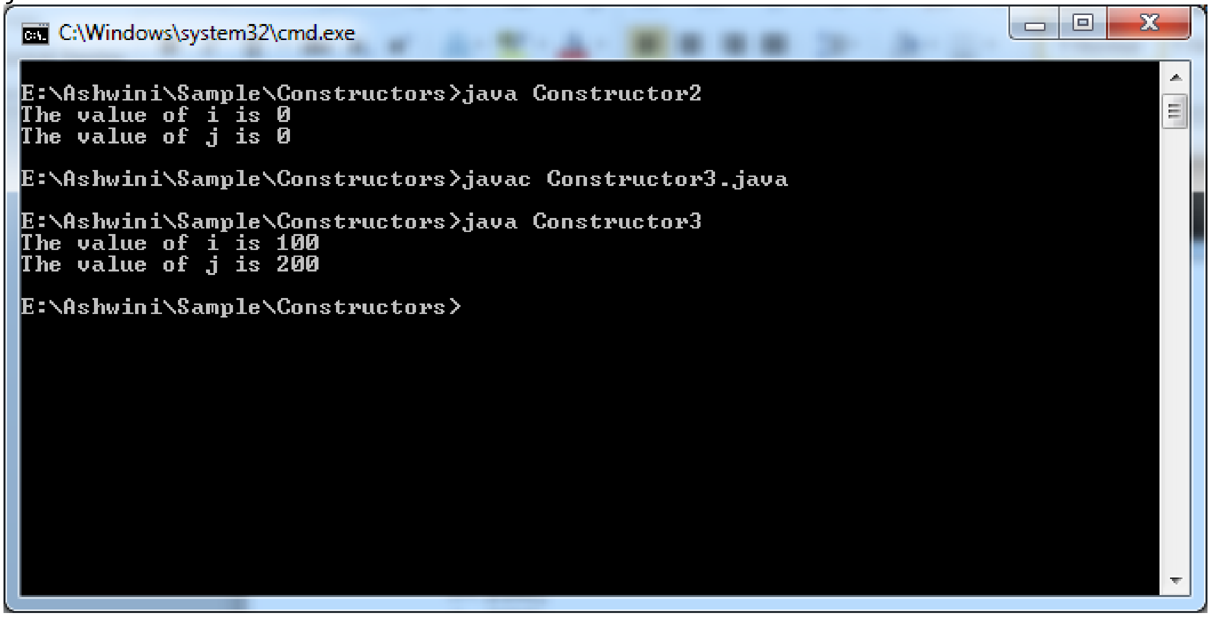
New will allocate memory, load all non-static members and constructor is called.
Example for compiler defined constructor:
```
class Constructor2
{
int i;
int j;
void display()
{
System.out.println("The value of i is "+i);
System.out.println("The value of j is "+j);
}
public static void main(String args[])
{
Constructor2 c=new Constructor2();
c.display();
}
}
```



```
E:\Ashwini\Sample\Constructors>java Constructor2
The value of i is 0
The value of j is 0

E:\Ashwini\Sample\Constructors>
```

Example for no argument constructor

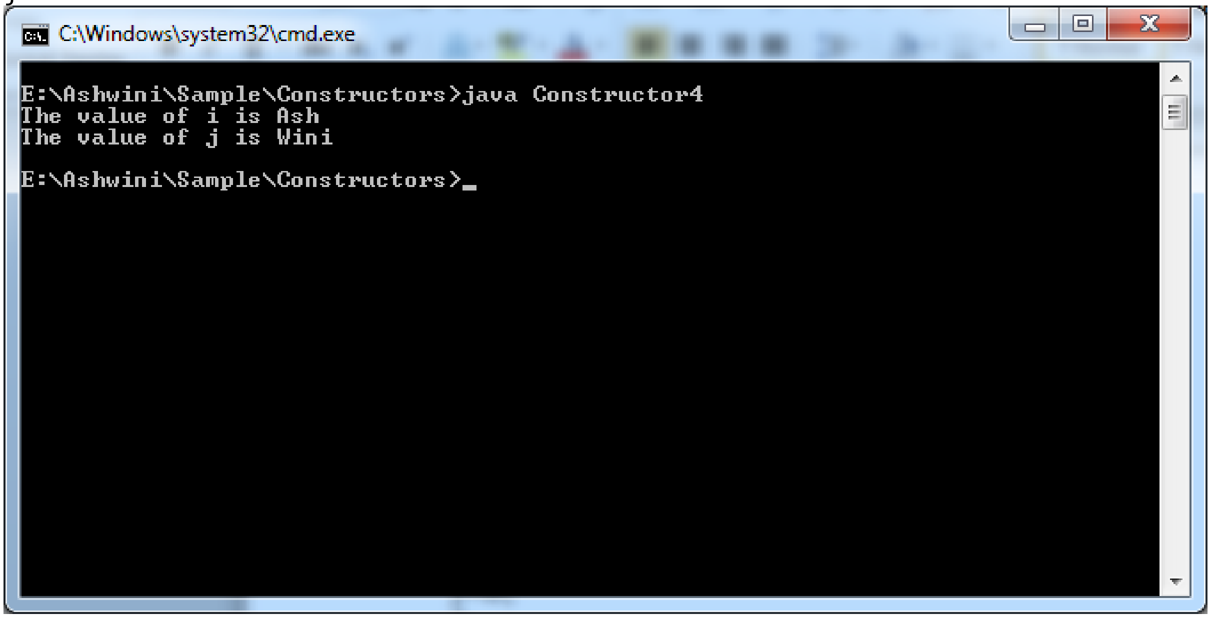
```
class Constructor3
{
int i;
int j;

Constructor3()
{
i=100;
j=200;
}

public static void main(String args[])
{
Constructor3 c=new Constructor3();
System.out.println("The value of i is "+c.i);
System.out.println("The value of j is "+c.j);
}
}
```



```
E:\Ashwini\Sample\Constructors>java Constructor2
The value of i is 0
The value of j is 0

E:\Ashwini\Sample\Constructors>javac Constructor3.java

E:\Ashwini\Sample\Constructors>java Constructor3
The value of i is 100
The value of j is 200

E:\Ashwini\Sample\Constructors>
```

Parameterised constructor

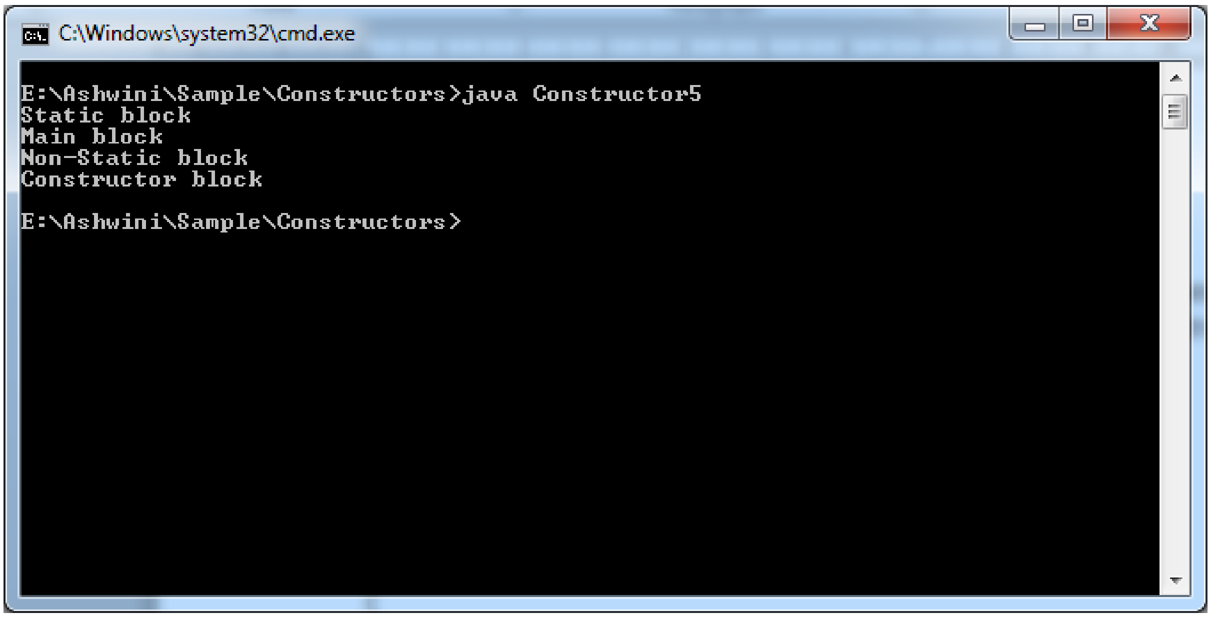
```
class Constructor4
{
String i;
String j;

Constructor4(String a, String b)
{
i=a;
j=b;
}
public static void main(String args[])
{
Constructor4 c=new Constructor4("Ash","Wini");
System.out.println("The value of i is "+c.i);
System.out.println("The value of j is "+c.j);
}
}
```



```
E:\Ashwini\Sample\Constructors>java Constructor4
The value of i is Ash
The value of j is Wini

E:\Ashwini\Sample\Constructors>_
```

The order of execution if a program contains static, non-static, main and constructor.
1. static
2. main
3. non-static
4. Constructor

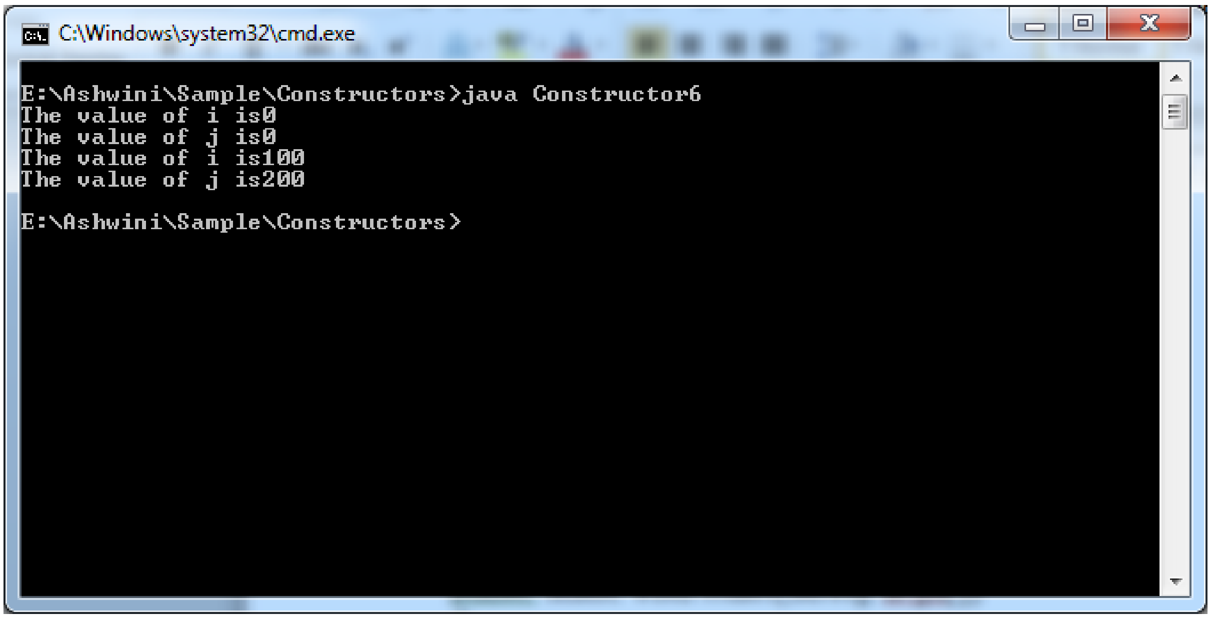
```
class Constructor5
{
Constructor5()
{
System.out.println("Constructor block");
}

static
{
System.out.println("Static block");
}
{
System.out.println("Non-Static block");
}

public static void main(String args[])
{
System.out.println("Main block");
Constructor5 c=new Constructor5();
}
}
```



```
C:\Windows\system32\cmd.exe

E:\Ashwini\Sample\Constructors>java Constructor5
Static block
Main block
Non-Static block
Constructor block

E:\Ashwini\Sample\Constructors>
```

If we declare constructor with reference type, it will be considered as a method. Method name and class name can be same or different. But constructor and class should be same.

```
class Constructor6
{
int i;
int j;
void Constructor6()
{
i=100;
j=200;
}
public static void main(String args[])
{
Constructor6 c=new Constructor6();
System.out.println("The value of i is"+c.i);
System.out.println("The value of j is"+c.j);
c.Constructor6();
System.out.println("The value of i is"+c.i);
System.out.println("The value of j is"+c.j);
}
}
```



```
E:\Ashwini\Sample\Constructors>java Constructor6
The value of i is0
The value of j is0
The value of i is100
The value of j is200

E:\Ashwini\Sample\Constructors>
```

Constructor Overloading

The process of declaring multiple constructor with same name and different parameter is known as constructor overloading. Parameter should differ either in number or type of parameters.

Eg:

```
Class Demo
{
Demo(int x)
{
}
Demo(Char a)
{
}
}
class Constructor7
{
int i;
char ch;
Constructor7( int a)
{
 i=a;
}
Constructor7( char b)
{
 ch=b;
}
public static void main(String args[])
{
Constructor7 c=new Constructor7(1);
Constructor7 c1=new Constructor7('z');
System.out.println(c.i);
System.out.println(c1.ch);
}
}
```

Write a prog to read constructor overloading:

```
class Constructor1
{
String Pen_Name;
String Pen_type;
double Pen_Price;
Constructor1( String a, String b)
{
Pen_Name=a;
Pen_type=b;
}
Constructor1( String a, String b, double c)
{
Pen_Name=a;
Pen_type=b;
Pen_Price=c;
}
public static void main(String args[])
{
Constructor1 c=new Constructor1("Cello","ballpark");
System.out.println("Name of the pen is "+c.Pen_Name);
System.out.println("Type of the pen is "+c.Pen_type);
Constructor1 c1=new Constructor1("Parker","ink",8.6);
System.out.println("Name of the pen is "+c1.Pen_Name);
System.out.println("Type of the pen is "+c1.Pen_type);
System.out.println("Price of the pen is "+c1.Pen_Price);
}
}
```

Constructor cannot be declared as a static it will throw compiletime error.