# Ch-2 - Basics of Objects and class in java.

* Explain Constructor with Overloading.

=> Constructor is a special method that is automatically called when object is created.

Constructor has same name of class.

Constructor is called automatically by a complier.

Constructor does not return any type of value.

There are three types of Constructor.

1) Default Constructor
2) Parametrized Constructor
3) Copy Constructor.

1 Default Constructor:

Default Constructor is also know as a no argument Constructor.

Default constructor is used to provides ~~vat~~ default values to the object.

Ex
```
class java
{
   int a, b;

   java()
  {
    int c = a+b;
      a = 5;
      b = 6;
  }

  public static void main
          (String args[])
  {
     java obj1 = new java()

          obj1.c;
   }
}
```

SMVS
Page No.
Date : / /

2 Parametrized Constructor :

The constructor which has parameter or argument it is called Parametrized Constructor.

```
Ex. class java
{
    int x, y;
    java(int a, int b)
{
    x = a;
    y = b;
    System.out.println(x+y);
}
public static void main
            (String args[])
{
    java obj = new java(5, 6);
}

}
```

3 Copy Constructor:

Copy Constructor is use to declare one object using other object.

Ex. class java

```
{
  int x, y;
  java (int a, int b)
  {
     x = a;
     y = b;
   System.out.println(x+y);
  }
  java (java &z)
  {
     x = z.x;
     y = z.y;
  }
  public static void main (
        String args[])
  {
    java obj1 = new java(5, 6);

    java obj2 = new java (obj1);
  }

}
```

=> Constructor Overloading:

If Constructor has same name and different parameter it is called Constructor Overloading.

In Constructor Overloading, always constructor has same name

Constructor Overloading can be done using two method:

1) Changing the number of argument

2) Changing the Data type.

Using Constructor Overloading we can use same name constructor with different argument.

Ex.

```
Class java
{
   int X, y ;
   int A, B, C ;
```

```
java (int P, int Q)
{
    X = P;
    Y = Q;
   int E = X + Y;

   System.out.println(E);
}

java (int F, int G, int H)
{
    A = F;
    B = G;
    C = H;
   int I = F + G + H;

   System.out.println(I);
}

public static void main
          (String args[])
{
   java obj1 = ~~java~~ new
                    java(5, 6);

   java obj2 = new java (7, 8, 9);

}

}
```

* Explain Visibility Modifiers.

=> Visibility Modifiers is called Access Modifiers.

There are four type of Visibility Modifiers.

1) Default
2) Private
3) Public
4) Protected.

1 Default Access Modifiers:

If we don't declare any access modifier for classes, method and variable this is called Default Access Modifiers.

Ex
```
class java
{
  void message()
  {
    System.out.println("Hello");
  }
public static void main(String
                    args[])
  {
    message();
  }
}
```

Page No.
Date : / /

2 Private access modifier :

If we declare any variable, method as a private then this method and variable can access only in class.

Ex.
```
class java
{
  Void message()
  {
    private string name;
  }
  public
}
class Main
{
  public static void main
      (String args[])
  {
    java obj = new java();
    obj.name = "Hello";
  }
}
```

3 Protected Access Modifier :

If we declare protected variable or method then this variable or method can access same class or subclass.

Ex. class java()
{protected
 void message()
 {
  System.out.println("Hello");
 }
}

class java1 extends java
{
 public static void main
        (String args[])
 {
  java1 obj = new java1();

  obj.message();

 }
}

4 Public Access Modifier:

If we declare any variable, or method public then we can access variable or method inside class and outside class.

Ex.

```
class java0()
{
  public void message()
  {
    System.out.println("Hello");
  }
}

class java1()
{
  public static void main
      (String args[])
  {
    java1 obj = new java1();

      obj.message();
  }

}
```

* Explain java Class and Object.

=> Java Class:

Class is a user-defined data types.

Class holds its own data member and member function.

When we create a class, there is no memory allocated to the class

Class is declare using class keyword.

Syntax:

```
class classname
{
    // class body
}
```

=> java Object:

Object is an instance of a class.

When we create object, then memory is allocated for class.

Object is created many time as requirement of user.

Using Object we can call class method or variable.

Object is a runtime and read-world entity.

Syntax:

class Object = new class ();
name name name

Using new keyword we can declare the object.

Ex class java
{
void AC)
{
System.out.println("Hello");
}
public static void main (String
args[])
{
java obj = new javac)
obj.AC);
}
}

~~* Explain~~

* Explain Java String.

=> String is called sequence of char values.

- There two types to Declare String in java.

1 Using String datatype

2 Using New Keyword

Page No.
Date : / /

1 Using String Datatype:

In this method, we can declare string using variable.

Syntax:

```
String Variable = "String";
       Name       Name
```

Ex.

```
class java
{
  public static void main
        (String args[])
  {
    String a = "Hello";

    System.out.println(a);

  }

}
```

2 Using New Keyword:

In this method, we can declare String using string object.

Syntax :

String Variable/Object Name = new String("string name");

Ex.

```
class java
{
  public static void main
                    (String args[])
  {
    String a = new String("Hello");
    System.Out.println(a);
  }
}
```

- Java String Can provides different String ~~buffer~~ method.

1 charAt(index) - Return String Particular index.

2 length() - Return String length

3 toLowercase() - Return String in lowercase

4 toUpperCase() - Return String in Uppercase.

5 indexof (ch, index) - Return Specify character of index String.

6 concat() - Connect String

7 isEmpty() - String is empty or not

8 replace(char Char old, new) = Replace character in string.

Ex, class java
{
public static void main (String args[])
{
String a = "Hello";
String b = "Khushi";

System.out.println(a.charAt(1));
System.out.println(a.length());

System.out.println(a.lowerCase());

```
System.out.println(b.UpperCase
                                ());
  c = a.Concet(b);
System.out.println(c);
 }

}
```

* Explain Java Stringbuffer.

=> Java StringBuffer class is used to create modifiable String Object.

The StringBuffer is almost similar to the java string.

A String that can be modified is know as mutable String.

Syntax:

```
String Object = new String("String");
Buffer Name           Buffer( name )
```

EX.

```
Class java
{
```

```
public static void main
        (String args[])
{
  StringBuffer a = new
                StringBuffer("Hello");

  System.out.println(a);
 }
}.
```

Java StringBuffer can provide different method.

- append(String s) - Add the string at the end of String.
- insert(index, String s) - Add the String at specify index.
- replace(Start index, End index, String s) - replace strind at between String.
- reverse() - Return Reverse string.

- lengh() - Return String length.

Ex, class java
{
public static void main(String
args[])
{
StringBuffer a = new
StringBuffer("Hello");

System.out.println(a.append
("khushi"));

System.out.println(a.insert
(5, "Khushi"));

System.out.println(a.replace
(1, 3, "ell"));

System.out.println(a.length());

System.out.println(a.reverse());
}

}

* Explain Java this keyword.

=> this keyword is use as a reference variable.

Using this keyword, we can call variable, method.

We can also use this keyword in constructor.

There are three way to use this keyword.

1) Using Variable

2) Using Method

3) Using Constructor.

1 Using Variable.

Ex
```
class java
{
  int name;

  display(int name)
  {
    this.name = name;
  }
```

```
    public static void main (String
                              args[])
    {
        display("Khushi");
    }

}
```

2 Using Method:

Ex.
```
class java
{
    int a, b;

    sum(int a, int b)
    {
        this.a = a;
        this.b = b;
        int c = a+b;
        System.out.println(c);
    }
    public static void main (String
                              args[])
    {
        sum(5, 6);
    }

}
```

3 Using Constructor

```
x. class java
{
    int a, b;
    int A, B, C;

    java(int x, int y)
    {
        this.a = x;
        this.b = y;
        int z = a + b;
        System.out.println(z);
    }

    java(int P, int Q, int R)
    {
        this.A = P;
        this.B = Q;
        this.C = R;
        int S = B + A + C;

        System.out.println(S);
    }

    public static void main
                (String args[])
    {
        java obj = new java();
```

```
obj.java(5,6);
obj.java(1,2,3);
}
}
```