**Class:**

* Class is a Blueprint/Architecture/template used to develop an application.
* it's a Logical Entity (i.e., it represents logics of an application)
* it's represented with "Class" Keyword.
* When Class is Created Memory will not be Allocated in "Heap Area".

In Every Program we can have 2 Types of Classes i.e.,

1.Business Logic Class (BL Class)

2.User Logic Class (UL Class)

**1.Business Logic Class (BL Class):** The Class which does not contains main () method it's called as "Business Logic Class (BL Class)"

* We can have Multiple Business Logic Classes in a Program.
* We usually didn't keep Business Logic Classes as "Public"

**2.User Logic Class (UL Class):** The Class which can contains main () method it's called as "User Logic Class (UL Class)"

* We can have only one User Logic Class in a Program.
* We Should keep only User Logic Class as "Public" and Save that File as "User Logic Class Name"

------------------------------------- \*\*\*\*\* -------------------------------------------------------------

**Object:**

* Object is a Reflection/Mirror image of a Class
* It's a Physical Entity (Because it contains Memory)
* Object can be Created by using "new" keyword
* For one class we can Create Multiple Objects & When Object gets Created memory will be allocated in "Heap Area"
* An Object is a real-world Entity which gets Created by using Class and it represents States & Behavior (i.e., States means "Variables (data members)" & Behavior means "Methods (member function)")

------------------------------------- \*\*\*\*\* -------------------------------------------------------------

**Constructor:**

Constructor is a special type of method which gets executed whenever we Create an Object.

The main Purpose of Constructor is to initialize the Non-Static Variable

1. Constructor name must be same as Class Name

2. Constructor will get called automatically, when we Create an Object

3. We can call One Constructor from another Constructor through "Constructor Chaining"

4. Constructor cannot inherited & Overridden

**Syntax:**

AccessModifier ConstructorName (args / No args)

{

//Body of the Constructor

}

**Rules for defining constructor:**

1. Constructor can be public, private, protected or default.

2. Constructor cannot be static, non-static, final or abstract.

3. Constructor name must be same as that of class name.

4. Constructor does not have any return type not even void.

5. Constructor can be with arguments or without arguments.

**There are 2 Types of Constructors:**

1. Non-Parameterized Constructor (or) No argument Constructor

2. Parameterized Constructor (or) Constructor with arguments

**1. Non-Parameterized Constructor (or) No argument Constructor**

It is the Constructor which does not have any arguments.

**Syntax:**

AccessModifier ConstructorName( ) //(i.e, No arguments)

{

//Body of the Constructor

}

**Drawbacks:**

1.One of the drawbacks of no argument constructor is, it provides same value for every object.

2.Therefore to overcome this drawback, we go for parameterized constructor.

**2. Parameterized Constructor (or) Constructor with arguments:**

It is the Constructor which will have the arguments. While Creating an Object we have to pass particular arguments to the Constructor.

**Syntax:**

AccessModifier ConstructorName( with args)

{

//Body of the Constructor

}

------------------------------------- \*\*\*\*\* -------------------------------------------------------------

**Constructor Chaining:**

The Process of Calling One Constructor from another Constructor is called as "Constructor Chaining"

Constructor chaining can be achieved in 2 ways

1. Call to this‐‐‐‐‐‐‐‐‐‐>represented as this()

2. Call to Super‐‐‐‐‐‐‐‐‐>represented as super()

**1.Call to this- this ():**

The process of calling one constructor from another constructor of same class is called as "call to this". it is represented with this()

The **main Rule** for this() is, Call to this must be the First statement of a constructor.

**2.Call to super-super():**

The process of calling one contructor from another constructor of different class is called as "call to super". it is represented with super()

Call to super must be the first statement in constructor.

Call to super is Mandatory, if we have any arguments in super class constructor.

Call to super is Optional, if we don't have any arguments in super class constructor.

------------------------------------- \*\*\*\*\* -------------------------------------------------------------

**Program for Reverse of a String:**

**public** **class** StringReverse

{

**public** **static** **void** main(String[] args)

{

String S1="gnireenigne tpecxe gnihtyna od nac sreenigne";

String S="";

**for**(**int** i=S1.length()-1; i>=0; i--)

{

S=S+S1 .charAt(i);

}

System.***out***.println("Reverse Of A String is: "+S);

}

}

**Output:** Reverse Of A String is: **engineers can do anything except engineering**

------------------------------------- \*\*\*\*\* -------------------------------------------------------------

**Program for Palindrome of a String:**

**public** **class** StringPalindrome

{

**public** **static** **void** main(String[] args)

{

String S1="Level";

String S="";

**for**(**int** i=S1.length()-1; i>=0; i--)

{

S=S+S1.charAt(i);

}

System.***out***.println("Reverse Of a String is : "+S);

**if**(S1.equalsIgnoreCase(S))

{

System.***out***.println("It's a palindrome string");

}

**else**

{

System.***out***.println("It's Non-palindrome string");

}

}

}

**Output:** Reverse Of a String is : **leveL**

**It's a palindrome string**

------------------------------------- \*\*\*\*\* -------------------------------------------------------------