Constructor :
****
- It is a specially design to perform initialization of an object.
- new keyword is responsible to create object.
- when object will create that time constructor will initialize
- how many time you will create object that much time constructor will execute.

Rules :
****
1) Name of the constructor and name of the class must be same
2) Return type concept us not applicable even void also. If you take return type void then it is treated
as a method.
3) The only modifier is allowed for constructor is : public, private, protected, default
private - anywhere
private - only within the class
protected - only within the package & outside where child class is present
default - only within the package
4) we can't declare static and other access modifier for constructor.
5) If we didn't declared constructor in our class there is default constructor is present
but if we declared customized constructor then default constructor will not be there.
6) Compiler is responsible for to declare default constructor.

Singleton class :
****
- To improve the class performance for any java class if we are allowed to create only one object
such type of classes are called as singleton classes.
- to implement singleton class we have to use private modifier for constructor.

Prototype of Default constructor :
****
- It's no arg constructor
- but every no arg constructor is not default constructor. ex., created by you.
- The access modifier of default constructor is same as class access modifier but this is applicable
for only public and default.
ex., if class is public constructor is also public
- It contains only one line i.e. super(). It is no-arg call to parent class constructor.

Case study of default constructor :
****
p's code (Programmers code) :

=============================
1) class Test {....}
2) public class Test {....}

```
3) class Test {
        void Test() {//it will treated as a method because we write down
void return type}
    }
4) class Test {
        Test() {// Compiler won't generate constructor (if you are
declaring constructor then
        there must be user super() or this(). If you didn't used in no-
args constructor the compiler always generate
        super() in that constructor.
        ex., Test() {super();
        }
    }
5) class Test() {
        Test(int i) { this(); }
        Test() {....}
    }
6) class Test {
        Test(int i) { super(); }
    }

c's code (Compiler code) :
===========================
1) Test() { super(); }
2) public Test() { super(); }
3) Test() { super(); }
4) Test() { super(); }
5) Test() { super(); }
6) won't generate default constructor. code will be same.

Various cases of this() and super() :
*************************************
Case-1 :
    public class Test {
        Test() {
            System.out.println("constructor");
            super();
        }
    }
    Here you will get compile time error because you wrote the super() in
2nd line.
    you can one use this() or super() in 1st line.
    you will get error : call to super() must be first statement in
constructor

Case-2 :
    public class Test {
        Test() {
            super();
            this();
            System.out.println("constructor");
        }
    }
    you can use super() or this() if you used both then you will get
error i.e., call to this must
    be 1st kube if constructor.

Case-3 :
    public class Test {
        Test() {
            super();
            System.out.println("constructor");
        }
    }
    you can use super() and this() only in constructor but not inside
method i.e. call to super()
    must be 1st statement in constructor.

super() => means we are calling parent class constructor
this() => calling same class or current class constructor

super() or this() :
*******************
- we can use only inside constructor
- we should use only in 1st line
- only one but not simultaneously

Note : this() and super() is a call to constructor & this and super is a
keyword

class Parent {
    String s = "parent";
}

class Child extends Parent {
    String s = "child";
    Child() {
        super(s);       //child
        this.s;     //child
        super(super.s);     //parent
    }
}
   - If you have same variable name in parent and child class and you
wan't to call parent class
    instance variable in child class then go for super(super.s)
   - If you want to refer super class instance variable then use "super"
keyword and if you want to use
   current class or child class variable contain same name
   - Inside static method or static area you can't use this or super
keyword except static we can
   use anywhere this and super keyword. If you tried to use you will get
compilation error


super(), this() VS super this :
*******************************
```

super(), this() :
1) these are constructor calls, to call super class and current class constructor
2) we can use only in constructor as 1st statement only.
3) we can use only one but not both simultaneously.

super, this :
1) these are keywords, to refer super and current class instance variable or member
2) we can use anywhere except static area.
3) we can use any number of time.

Constructor overloading :
****
within a class you can write any no. of constructor with different arguments that is called constructor
overloading.

Note : overriding and inheritance concept is not applicable for constructor.
- u can't use constructor in interface.

```
public class Test {
    public static void m1() {
        m2();
    }
    public static void m2() {
        m1();
    }
    public static void main(String[] args) {
        System.out.println("Hello");
    }
    //op : Hello
}
```
In the above program we are not calling m1 or m2 method in main method that's why we will not
get static overflow exception. It will simply print "Hello".

- recursive method call is always runtime exception saying stack overflow error if you called in main class.

- recursive constructor call is get error i.e. recursive constructor invocation