SINGLETON

SINGLETON CLASS:-

The java class that allows us to create only one object any situation is called singleton class.

- ⇨ Singleton java class in one object in all situation
- ⇨ When class is having read only state, sharable state and no state in these 3 situation creating multiple object having above 3 state is waste of memory waste of cpu time.
- ⇨ Singleton class comes under gangs of 4 pattern and it also comes under creational pattern because it is all about object creation
- ⇨ In gangs of 4 pattern there are 3 categories are there for design singleton class
    - Creational → (object creation process)
    - Structural
    - Behaviors

Problem :-

Creating multiple object for a java class in these situations gives memory issues, cpu utilization issues and performance issues:-

a. When class is not having any state
b. When class is having only read-only state
c. When class is having sharable state across the multiple other classes

Solution :-

Does not allow creating more than 1 object in any situation.

If we create only object for normal java class thought it allows creating more objects then that class not called as singleton class. Singleton class should not allow creating more than 1 object though we are trying to create more objects

Servlet container create only object for our servlet class it does not bean our servlet class is singleton class servlet container just not interested to create more then one object though our servlet class allow to create.

DO NOT COPY

SINGLETON

Predefined singleton class:-

Java.lang.Runtime, java.awt.Desktop,org,log4j.Logger

Ex:-

x--> user defined class (I want to make it as singleton class)

1. Class should be taken as public then only people can start using class outside of package.
2. Maximum object creation for that class will happen through **new operator** while using new operator internally constructor execution happen generally use take constructor as public so that **we can create object outside of the package** how many times we want
3. If we don't allow to create object outside the package then define constructor as **private constructor**

   If outsiders want to create object only one time then create as **static factory method()** then it will check if object are created or not if not created then it this factory method create new object ..if already created object then it will return reference of the existing object to static factory method. Then verification happen only in static factory method

Q:- why this method taken as static factory method?

Ans:- static factory method only one way to create the object to call that method people cannot provide object …so static method can call outside of the class because this method declared with static so that static method can called on outside the class without object

Q:- why are we calling as a factory ?

Ans:- its return same class object or own class object so that it is called factory method

Q:- inside factory method what logic will be use?

Ans:- it checked object is created or not if not created then its create object and return the object and if object is already created so that it not create object it is directly return reference of the object…….for this we should take

DO NOT COPY

SINGLETON

one refrence variable at the top of the class which is hold the current class object

Ex:- **private static X Instance=null;**

Q:- why this instance variable is private

Ans:- because it will access only in the class only .

Q:- why instance variable is declared with static keyword?

Ans:- its taking static so that we can use this variable inside the static method.

Q:- what the problem with if this instance variable taken as public?

Ans:- people will access this one directly wia class name they may change null after creating first object again if it will check this variable is null then factory method will check it is null then create new object more than one time so that it will recommended to restricted as private modifier then outsiders can access this variable.

Use case:- to make all the employee company using same printer, we must printer class as singleton java class

"developing printer class as singleton minimum standard"

Ex:-

```
package com.nit.dp;
public class printer {
	private static printer instence=null;
	private printer() {
		System.out.println("0 param constructor");
	}
	public static printer getInstance() {
		//static factory method
		if(instence==null) {
			instence =new printer();
		}
		return instence;
	}
	public void printDate(String data) {
		System.out.println(data);
	}
}
```

DO NOT COPY

SINGLETON

```
package com.nit.test;

import com.nit.dp.printer;

public class test1 {
     public static void main(String[] args) {
          printer print=null,print1=null;
          //not possoible to create object outside the class
          print=printer.getInstance();
          print1=printer.getInstance();
          System.out.println(print.hashCode());
          System.out.println(print1.hashCode());
          System.out.println(print==print1);
          System.out.println("============");
          print.printDate("vikas");
          print1.printDate("yadav");
     }
}
```

Output:-

```
0 param constructor
366712642
366712642
true
============
vikas
yadav
```

- ➔ Don't decide whether object is created or not based on constructor execution because when object are created through **deserialization and cloning** no constructor execute.
- ➔ When we create sub class object of abstract class along with sub class constructor the abstract class constructor executes but it does not mean object is created for abstract class.

DO NOT COPY

SINGLETON

2. where the declaration there only creating object singleton

Ex:-

```
package com.nit.dp;
public class printer {
private static printer instence=new printer();
private printer() {
     System.out.println("0 param constructor");
}
public static printer getinstence() {
     return instence;
}
/*public static printer getInstance() {
     //static factory method
     if(instence==null) {
          instence =new printer();
     }
     return instence;
}*/
public void printDate(String data) {
     System.out.println(data);
}
}
```

Test:-

```
package com.nit.test;

import com.nit.dp.printer;

public class test1 {
public static void main(String[] args) {
     printer print=null,print1=null;
     //not possoible to create object outside the class
     print=printer.getinstence();
     print1=printer.getinstence();
     System.out.println(print.hashCode());
     System.out.println(print1.hashCode());
     System.out.println(print==print1);
     System.out.println("============");
     print.printDate("vikas");
     print1.printDate("yadav");
}
}
```

DO NOT COPY

SINGLETON

EGAR INSTANTIATION:-

"it's also not best its worst …….. Because we are not asking for object but still object will be ready becouse the moment class is loaded though I don't ask for object …..object will be created and that object reference again will be return …..single object will be come but single object will be created though **egar instantiation** process "

Ex:-

```
package com.nit.dp;

public class printer {

//egar instantiation only if object creation at the time of declaration
	private static printer instence=new printer();

	private printer() {
		System.out.println("0 param constructor");
	}

	public static printer getinstence() {
		return instence;
	}

	//statif factory method
	/*public static printer getInstance() {
		//static factory method
		if(instence==null) {
			instence =new printer();
		}
		return instence;
	}*/

	public void printDate(String data) {
		System.out.println(data);
	}

}
```

DO NOT COPY

SINGLETON

Test:-

```
package com.nit.test;

import com.nit.dp.printer;

public class test1 {
public   static   void   main(String[]   args)   throws
ClassNotFoundException {
     printer print=null,print1=null;
     //not possoible to create object outside the class
     /*print=printer.getinstence();
     print1=printer.getinstence();
     System.out.println(print.hashCode());
     System.out.println(print1.hashCode());
     System.out.println(print==print1);
     System.out.println("============");
     print.printDate("vikas");
     print1.printDate("yadav");*/

     Class.forName("com.nit.dp.printer");
}
}
```

“here class.Forname is only meant for loading the class but here we are not asking for object but still object is created …… this object is creation is called egar instantiation and also we are not interested to use that object so its waste ”

DO NOT COPY

SINGLETON

LAZY INSTANTIATION:-

Until we not ask for object it not creating object its called lazy instantiation

Ex:-

```
package com.nit.dp;
public class printer {

    //private static printer instence=new printer();
//egar instantiation only if object creation at the time of declaration
    private static printer instence;
    private printer() {
        System.out.println("0 param constructor");
    }

    /*public static printer getinstence() {
        return instence;
    }*/

    //statif factory method
    public static printer getInstance() {
        //static factory method
        if(instence==null) {
            instence =new printer();  //lazy instantiation good practice
        }
        return instence;
    }
    public void printDate(String data) {
        System.out.println(data);
    }
}
```

DO NOT COPY

SINGLETON

Test:-
=====

```
public class test1 {
     public   static   void   main(String[]   args)   throws
ClassNotFoundException {
          printer print=null,print1=null;
          //not possoible to create object outside the class
          /*print=printer.getinstence();
          print1=printer.getinstence();
          System.out.println(print.hashCode());
          System.out.println(print1.hashCode());
          System.out.println(print==print1);
          System.out.println("============");
          print.printDate("vikas");
          print1.printDate("yadav");*/
          Class.forName("com.nit.dp.printer");
     }
}
//no object created
```

2:- test:-//recomended

Ex:-

```
package com.nit.test;

     import com.nit.dp.printer;

     public class test1 {
     public     static     void     main(String[]     args)     throws
     ClassNotFoundException {
          printer print=null,print1=null;
          //not possoible to create object outside the class
          print=printer.getInstance();
          print1=printer.getInstance();
          System.out.println(print.hashCode());
          System.out.println(print1.hashCode());
          System.out.println(print==print1);
          System.out.println("============");
          print.printDate("vikas");
          print1.printDate("yadav");

          //Class.forName("com.nit.dp.printer");
     }
}
```

"here we are asking for object then only object will be created"

DO NOT COPY

SINGLETON

***lazy instantiation is good practice for creating singleton object ***

==========minimum standard completed==========

New approach of creating second object for singleton by breaking this pattern:-

By using new operator 95% is closing object outside of class by minimum standard still 5 % is remaining .

==========

DO NOT COPY