Intern Report - Feb 24

Singleton class:

Only one instance can be created using the Singleton class.

Steps:

- ➔ Create a static instance.
- ➔ Create a private constructor.
- ➔ Create a static method to get the instance.

Normal method: (eager instantiation)

```
class alone
{
        static alone obj = new alone();                //eager instantiation
        private alone()
        {

        }
        public static alone getIns()
        {
                return obj;
        }
}

public class Sing
{
        public static void main(String args[])
        {
                alone temp = alone.getIns();
        }
}
```

Drawback:

- ➔ The instance is created as soon as the class is executed so that even if the object is not in need, it is instantiated automatically and memory is wasted.
- ➔ So we move to the lazy instantiation method in which the instance is created only at the time of getting the instance.

Method 2 : (lazy instantiation)

```
class alone
{
        static alone obj;                //lazy instantiation

        private alone()
        {
                System.out.println("Instance created");
        }
        public static alone getIns()
        {
                if(obj == null)
                {
                        obj = new alone();
                }
                return obj;
        }

}

public class Sing2
{
        public static void main(String args[])
        {
                alone temp = alone.getIns();
                alone temp2 = alone.getIns(); // this will not be created
        }
}
```

Drawback:

- ➔ If two threads call the instance at the same time, both the threads will execute the constructor.
- ➔ So we make the getIns() method synchronized to execute the threads one by one.

Synchronized method:

```
class alone
{
        static alone obj;
        private alone()
        {
                System.out.println("Instance created");
        }
        public static synchronized alone getIns()
        {
                if(obj == null)
                {
                        obj = new alone();
                }
                return obj;
        }
}

public class Sing3
{
        public static void main(String args[])
        {
                Thread t1 = new Thread( new Runnable()
                {
                        public void run()
                        {
                                alone temp = alone.getIns();
                        }
                });

                Thread t2 = new Thread ( new Runnable()
                {
                        public void run()
                        {
                                alone temp = alone.getIns();
                        }
                });
                t1.start();
                t2.start();
        }
}
```

Drawback:

- ➔ Using a Synchronized method would affect the performance time of the program.
- ➔ So we moved to a ‘double-checked locking’ mechanism.

Double checked locking method:

```
class alone
{
        static alone obj;

        private alone()
        {
                System.out.println("Instance created");
        }

        public static alone getIns()
        {
                if(obj == null)                                 // check 1
                {
                        synchronized(alone.class)
                        {
                                if(obj == null)                 // check 2
                                {
                                        obj = new alone();
                                }
                        }

                }
                return obj;
        }
}

public class Sing4
{
	public static void main(String args[])
	{
		Thread t1 = new Thread( new Runnable()
		{
			public void run()
			{
				alone temp = alone.getIns();
			}
		});

		Thread t2 = new Thread ( new Runnable()
		{
			public void run()
			{
				alone temp2 = alone.getIns();
			}
		});
		t1.start();
		t2.start();
	}
}
```

ENUM Instance method: (best and simple method)

- ➔ Enum is a special type of class.
- ➔ In enum, when we say INSTANCE, it implicitly creates a private constructor for the enum.

```
enum alone
{
	INSTANCE;
	int i;
	public void disp()
	{
		System.out.println(i);
	}
}
public class Sing5
{
	public static void main(String args[])
	{
		alone t1 = alone.INSTANCE;
		t1.i=1;
		t1.disp();					// i will be 1
```

```
                alone t2 = alone.INSTANCE;
                t2.i=2;
                t2.disp();                              // i will be 2

                t1.disp();                              // i will be 2
        }
}
```