Lab 5 : Singleton Design Pattern

1. **Eger – Lazy Singleton**

```
package singleton;
// Eager -> Lazy Singleton//
class student {
   public static student s1;
   private student()
   {
      System.out.println("Student created....");
   }

   public static student getInstance()
   {
      s1 = new student();
      return s1;
   }

   public static void main(String[] args) {
      student s2 = student.getInstance();
   }
}
```

Output:

```
C:\Users\harsh\.jdks\openjdk-19.0.1\bin\java.exe "-j
Student created....

Process finished with exit code 0
```

2. **Synchronized methods…**

```
package singleton;
//synchronizing methos//
class studentsync {
   public static studentsync s1;
   private studentsync()
   {
      System.out.println("Student created....");
   }

   public static synchronized studentsync getInstance()
   {
     if(s1==null)
     {
        s1 = new studentsync();
     }
      return s1;
   }

   public static void main(String[] args)
   {
      Thread t1 = new Thread(new Runnable() {
         @Override
         public void run() {
            studentsync s = studentsync.getInstance();
         }
      });
      Thread t2 = new Thread(new Runnable() {
         @Override
         public void run() {
            studentsync s = studentsync.getInstance();
         }
      });

      t1.start();
      t2.start();

   }
}
```

Output:

```
C:\Users\harsh\.jdks\openjdk-19.0.1\bin\java.exe "-
Student created....

Process finished with exit code 0
```

3. Double check locked

```
package singleton;
//double checked locking//
public class studentdclock {
   public static studentdclock obj1;
   private studentdclock()
   {
      System.out.println("Student created...");
   }
   public static studentdclock getInstance()
   {
      if(obj1==null)
      {
         synchronized(studentdclock.class)
         {
            if(obj1==null){
               obj1 = new studentdclock();
            }
         }
      }
      return obj1;
   }

   public static void main(String[] args) {
      Thread t1 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            studentdclock s = studentdclock.getInstance();
         }
      });
      Thread t2 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            studentdclock s = studentdclock.getInstance();
         }
      });
     t1.start();
      t2.start();
   }
}
```

Output: 

4. **Using enum**

```
package singleton;

//enum method//
public class studentenum {
   public static void main(String[] args) {
      students obj1 = students.INSTANCE;
      obj1.seti(2);
      System.out.println(obj1.geti());
      students obj2 = students.INSTANCE;
      obj2.seti(5);
      System.out.println(obj1.geti());

   }

}
enum students
{
   INSTANCE;
   int i;
   public int geti()
   {
      return i;
   }
   public void seti(int i)
   {
      this.i = i;
   }
}
```

Output:

```
C:\Users\harsh\.jdks\openjdk-19.0.1\bin\java.exe "-javaa
2
5

Process finished with exit code 0
```