*public class* Singleton {

 *// Singleton is a creational design pattern, which ensures that only one object of its kind exists
 // and provides a single point of access to it for any other code.

 /\*

 private static final Singleton instance = new Singleton(); // Eager

 public static Singleton getInstance() { //Eager
 return instance;
 }
 \*/
 private static volatile* Singleton *instance* = *null*;

 *private* Singleton() {}

 *// Lazy initialization

 public static* Singleton getInstance() {
 *if* (*instance* == *null*) {
 *instance* = *new* Singleton();
 }
 *return instance*;
 }

 *// static block initialization
 static* {
 *try* {
 *instance* = *new* Singleton();
 } *catch* (Exception e) {
 *throw new* RuntimeException("Exception occurred in creating singleton instance");
 }
 }

 *// Thread Safe

 public static synchronized* Singleton getInstanceWithThreadSafe() {
 *if* (*instance* == *null*) {
 *instance* = *new* Singleton();
 }
 *return instance*;
 }

 *// Thread Safe with double lock
 public static* Singleton getInstanceUsingDoubleLocking() {
 *if* (*instance* == *null*) {
 *synchronized* (Singleton.*class*) {
 *if* (*instance* == *null*) {
 *instance* = *new* Singleton();
 }
 }
 }
 *return instance*;
 }

 *// Using Bill Pugh inner static helper class.

 private static class* SingletonHelper {
 *private static final* Singleton *INSTANCE* = *new* Singleton();
 }

 *public static* Singleton getInstanceWithInnerStaticHelper() {
 *return* SingletonHelper.*INSTANCE*;
 }

}

*public class* SerializedSingleton *implements Serializable* {

 *private static final long serialVersionUID* = -7604766932017737115L;

 *private* SerializedSingleton(){}

 *private static class* SingletonHelper {
 *private static final* SerializedSingleton *instance* = *new* SerializedSingleton();
 }

 *public static* SerializedSingleton getInstance() {
 *return* SingletonHelper.*instance*;
 }
 *protected* Object readResolve() {
 *return getInstance*();
 }
}

*public class* SingletonSerializedTest {

 *public static void* main(String[] args) *throws* FileNotFoundException, IOException, ClassNotFoundException {
 SerializedSingleton instanceOne = SerializedSingleton.*getInstance*();
 *ObjectOutput* out = *new* ObjectOutputStream(*new* FileOutputStream(
 "filename.ser"));
 out.writeObject(instanceOne);
 out.close();

 *// deserialize from file to object
 ObjectInput* in = *new* ObjectInputStream(*new* FileInputStream(
 "filename.ser"));
 SerializedSingleton instanceTwo = (SerializedSingleton) in.readObject();
 in.close();

 System.*out*.println("instanceOne hashCode="+instanceOne.hashCode());
 System.*out*.println("instanceTwo hashCode="+instanceTwo.hashCode());

 }
}

