```
import java.io.Serializable;

/**
 * The perfect implementation of Singleton design pattern
 * Properly solves all the below mentioned problems in Singleton pattern
 * <p>
 * 1) Attack using Reflection API
 * 2) Problems from serialization/deserialization of your object
 * 3) Problems from cloning your object
 * 4) Uncertainty in a multi-threaded environment
 *
 * Problems with Garbage Collection have already been fixed in prior versions of Java
 * <p>
 * Created by aritraroy on 26/05/16.
 */
public class Singleton implements Serializable, Cloneable {

    // We would not eagerly initialize the singleton
    // The volatile keyword ensures that half-initialized objects are not published to other threads
    public static volatile Singleton INSTANCE = null;

    private Singleton() {
        // Preventing attack by Reflection APIs
        if (INSTANCE != null) {
            throw new RuntimeException("Cannot instantiate single object using constructor.
Use its #getInstance() method");
        }
        // Create your object here
    }

    /**
     * A global point of access for the singleton
     *
     * @return
     */
    public static Singleton getInstance() {

        // Implementing double-locking to prevent ambiguity in multi-threaded environment
        if (INSTANCE == null) {
            synchronized (Singleton.class) {
                if (INSTANCE == null) {
                    INSTANCE = new Singleton();
                }
            }
        }
        return INSTANCE;
    }

    /**
     * Ensuring that singleton contract is not violated by serialization/deserialization
     *
     * @return
     */
    public Object readResolve() {
        return Singleton.getInstance();
    }
    /**
     * Ideally we should not support cloning functionality as by definition
     * a singleton provides only a single instance
     *
     * @return
     * @throws CloneNotSupportedException
     */
    @Override
    protected Object clone() throws CloneNotSupportedException {
        throw new CloneNotSupportedException();
    }
}
```