Singleton class and enum in Java

Jay Chow

Software Engineer, Cisco eCommerce, IT

Cisco, San Jose

29th April 2020

Agenda

1. What is a singleton class?
2. Methods of making singletons
3. Singleton with public static final field
4. Singleton with public static factory method
5. Singleton with lazy initialization
6. Singleton with enum
7. Reference

What is a singleton class?

- In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time
- If we try to instantiate the Singleton class after the first instance is created, the new variable also points to the same first instance; the modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created
- Same instance of singleton can be reused by multiple threads
- Singletons represent system configurations and window managers, since those singletons should be common to all threads and objects within a JVM

Methods of Making Singletons

Method 1: Singleton With Public Static Final Field

```
public class Singleton {
    public static final Singleton INSTANCE = new Singleton();
    // private constructor
    private Singleton() {}
}
```

Method 2: Singleton With Public Static Factory Method

```
public class Singleton {
    private static final Singleton INSTANCE = new Singleton();
    // private constructor
    private Singleton() {}
    public static Singleton getInstance(){
        return INSTANCE;
    }
}
```

Method 3: Singleton With Lazy Initialization

```
public class Singleton {
    private static Singleton INSTANCE = null;
    // private constructor
    private Singleton() {}
    public static Singleton getInstance() {
        if (INSTANCE == null) {
            synchronized (Singleton.class) {
                if (INSTANCE == null) {
                    INSTANCE = new Singleton();
                }
            }
        }
        return INSTANCE;
    }
}
```

Method 4:Making Singletons with Enum in Java (Recommended)

Singleton with Enum:

```
public enum Singleton {
    INSTANCE;
}
```

How to use:

```
public enum SingletonEnum {
    INSTANCE;
    int value;
    public int getValue() {
        return value;
    }
    public void setValue(int value) {
        this.value = value;
    }
}
```

Main class implementation:

```
public class EnumDemo {
    public static void main(String[] args) {

        SingletonEnum singleton = SingletonEnum.INSTANCE;

        System.out.println(singleton.getValue());

        singleton.setValue(2);

        System.out.println(singleton.getValue());
    }
}
```

Note: when serializing an enum, field variables are not getting serialized. For example, if we serialize and deserialize the SingletonEnum class, we will lose the value of the int value field

References

https://dzone.com/articles/java-singletons-using-enum