Java
Immutable Class

By Dipak K.

Immutable Class
Learn How to create Immutable class in java?
What are the usages and benefits by using Immutable class?

INTRODUCTION

"Immutability means that something cannot be changed"

There are many immutable classes like String, Boolean, Byte, Short, Integer, Long, Float, Double etc. In short, all the wrapper classes. String class is immutable. We can also create an immutable class by creating a final class that has final data members.

Immutable class

Immutable class is very simple to understand, it has only one state. Immutable class is carefully instantiated by the constructor.

Immutable classes are thread safe. This is the biggest advantage of immutable classes, you don’t need to apply synchronization for immutable objects.

Immutable class can be useful while putting an object of immutable class in HashMap or it can be used for caching purposes because its value won’t change.

Immutable objects are by default thread safe.

Advantages

1. **Immutable Objects are thread safe, because the. values will never change.**

2. **Require no synchronization**

Create Immutable class

Make your class final

If you make your class final, no class will be able to extend it.hence will not be able to override methods of this class.

Declare all instance variable with private and final

If you make instance variables private, no outside class will be able to access instance variables. if you make them final, you can not change it.

Say no to setter methods

Don’t create a setter method for any instance variables, hence there will be no explicit way to change the state of instance variables.

Initialize all variables in constructor

You can initialize variables in the constructor. You need to take special care while working with mutable objects. You need to do deep copy in case of immutable objects.

Perform cloning of mutable objects while returning from getter method

If you return a clone of an object from the getter method, it won't return the original object, so your original object will remain intact.

Code Example

Demo.java

```
public class Demo {
    private final String name;
    private final int age;

 public Demo(String name , int age) {
        this.name = name;
        this.age = age;
    }

   public String getName() {
      return name;
    }
public int getAge() {
        return age;
    }

    @Override
   public String toString() {
       return "Demo [name=" + name + ",age=" + age + "]";
    }
}
```

Thank you...