

# Java: Singleton and Immutable Class

## "Concept && Coding" YT Video Notes

### Today's Topics:

---

- Singleton Class
- Immutable Class
- Wrapper Class

### Singleton Class:

---

This class objective is to create only 1 and 1 Object.

### Different Ways of creating Singleton Class:

- Eager Initialization
  - Lazy Initialization
  - Synchronization Block
  - Double Check Lock (there is a memory issue, resolved through Volatile instance variable)
  - Bill Pugh Solution
  - Enum Singleton
- 3



## Eager Initialization:

---

```
public class DBConnection {  
  
    private static DBConnection conObject = new DBConnection();  
  
    private DBConnection(){  
    }  
  
    public static DBConnection getInstance(){  
        return conObject;  
    }  
}
```

```
public class Main {  
  
    public static void main(String args[]) {  
  
        DBConnection connObject = DBConnection.getInstance();  
    }  
}
```

Lazy Initialization:

```
public class DBConnection {  
  
    private static DBConnection conObject;  
  
    private DBConnection(){  
    }  
  
    public static DBConnection getInstance(){  
  
        if(conObject == null){  
            conObject = new DBConnection();  
        }  
        return conObject;  
    }  
}
```

## Synchronized Method:

```
public class DBConnection {  
  
    private static DBConnection conObject;  
  
    private DBConnection(){  
    }  
  
    @Synchronized public static DBConnection getInstance(){  
  
        if(conObject == null){  
            conObject = new DBConnection();  
        }  
        return conObject;  
    }  
}
```

Synchronized  
↳ Lock  
↳ unlock

## Double Locking:

---

```
public class DatabaseConnection {  
    private static volatile DatabaseConnection conObject;  
  
    private DatabaseConnection() {}  
  
    public static DatabaseConnection getInstance() {  
        if(conObject == null){  
            synchronized (DatabaseConnection.class){  
                if(conObject == null) {  
                    conObject = new DatabaseConnection();  
                }  
            }  
        }  
        return conObject;  
    }  
}
```

## Bill Pugh Solution:

---

```
public class DatabaseConnection {  
    private DatabaseConnection() {}  
  
    private static class DBConnectionHelper {  
        private static final DatabaseConnection INSTANCE_OBJECT = new DatabaseConnection();  
    }  
  
    public static DatabaseConnection getInstance() {  
        return DBConnectionHelper.INSTANCE_OBJECT;  
    }  
}
```

ENUM:

```
enum DBConnection {  
    INSTANCE;  
}
```

~~private~~

### IMMUTABLE CLASS:

- We can not change the value of an object once it is created.
- Declare class as 'final' so that it can not be extended.
- All class members should be private. So that direct access can be avoided.
- And class members are initialized only once using constructor.
- There should not be any setter methods, which is generally used to change the value.
- Just getter methods. And returns Copy of the member variable.
- Example: String, Wrapper Classes etc.

```
final class MyImmutableClass {  
    private final String name;  
    private final List<Object> petNameList;  
  
    MyImmutableClass(String name, List<Object> petNameList){  
        this.name = name;  
        this.petNameList = petNameList;  
    }  
  
    public String getName(){  
        return name;  
    }  
  
    public List<Object> getPetNameList(){  
        //this is required, because making list final,  
        // means you can not now point it to new list, but still can add, delete values in it  
        //so thats why we send the copy of it.  
  
        return new ArrayList<>(petNameList);  
    }  
}
```

```
public class Main {  
  
    public static void main(String args[]){  
        List<Object> petNames = new ArrayList<>();  
        petNames.add("sj");  
        petNames.add("pj");  
  
        MyImmutableClass obj = new MyImmutableClass( name: "myName", petNames);  
        obj.getPetNameList().add("hello");  
        System.out.println(obj.getPetNameList());  
    }  
}
```

Output:

[sj, pj]

## WRAPPER CLASS:

---

Checkout the Video no:  
[6. Java Variables - Part2 | Reference/Non-Primitive Data Types in Depth]