TL;DR

> Only one instance of a singleton class exists. There are two ways to achieve singularity.
>
> 1. Single element enum.
> 2. Private constructor and static factory method.

1. Problem: Design a class so that only one instance of it exists.

The class whose only one instance is available all the time is called a singleton class.

Example: Earth, Mars etc.

2. Design

There are two designs to achieve singularity.

Design 1 - Have a single element enum

This method is reflection safe.

Though less used, this method is the best.

```
public enum Earth {

    INSTANCE;
```

```
    public void goRoundTheSun() {

    }
}
```

Design 2

This method is susceptible to reflection attacks.

Step 1 : Mark constructor private.

Step 2 : Declare Private static final reference var Earth

Step 3 : Declare static factory method which returns singleton

```
public class Earth{

    // Step 2. static final private ref var.

    private static final Earth instance = new Earth();

    //  Step1. private constructor

    private Earth(){};

    // Step 3 : static factory method

    public static Earth getInstance(){
        return instance;
    }
}
```

3. Real Time Use

Example : Logger class to log errors and events. There should only be one per system. Hence singleton.

Another Example : Manager type classes like WiFiManager. Or Controller.

4. Tester Class Code

```
public class SingletonTest{

        public static void main(String[] args){

                Earth e1 = Earth.getInstance();

        Earth e2 = Earth.getInstance();

        System.out.println(e1==e2);
        //Above statement should always return true for singleton.
        }
}
```