Creational Design Pattern: Singleton

When should this be Used:

When there is a class or a task common for entire project like EmailSending, SMSSend, Login etc we should use this pattern

What is this Pattern:

create a separate Utility class for common task and let all Service classes access them.

How to Implement:

- Make constructor private so that no one can create an object.
- Make instance private and instantiate (create object) in static block
- Give static getInstance() method to service classes

Code Snippet:

```
//Singleton Design Pattern
public class EmailUtility { //send an Email

    private static EmailUtility emailUtility;

    private EmailUtility(){ //step 1: make constructor private

    }

    static {
        emailUtility = new EmailUtility(); //in static mem : 10X
    }

    public void sendEmail(String message ) {
        //logic code of email send
        System.out.println("Email sent successfully... " + message);
    }

    public static EmailUtility getInstance() {
        return emailUtility;
    }
```