**Exercise 1: Implementing the Singleton Pattern**

**Design Patterns:** A reusable solution to common problems in software design

It's based on two principles:

Composition over interface

Code for interface, not for implementation

They are classified into three types:- Creational, Structural, Behavioural

**1.Singleton Pattern (Creational Pattern)**

**Purpose:** To instantiate the class only once.

**Program:**

package SingletonPattern;

class Browser{

private static Browser browse;

private Browser() { }

public synchronized static Browser getInstance(){

if(browse==null)

return new Browser();

return browse;

}

public void displayInfo(){

System.out.println(“Hello Everyone”);

}

}

package SingletonPattern;

public class Test{

public static void main(String args[]){

Browser b=Browser.getInstance();

b.displayInfo();

}}

**Output:** Hello Everyone