|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment Number: 11** | | | | | |
| **Date of Performance:** | | **23-09-2022** | | | |
| **Date of Submission:** | | **30-09-2022** | | | |
| **Program Execution/ formation/ correction/ ethical practices (07)** | **Documentation (02)** | **Timely Submission (03)** | **Viva Answer to sample questions (03)** | **Experiment Total (15)** | **Sign** |
| **7** | **2** | **2** | **3** | **14** |  |

Experiment No: 11

Aim: *Singleton Design Pattern*

**Lab Outcome:** Use computer-aided software engineering (CASE) tools.

# Design pattern:

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. Reusing design patterns helps to prevent subtle issues that can cause major problems and improves code readability for coders and architects familiar with the patterns.

Often, people only understand how to apply certain software design techniques to certain problems. These techniques are difficult to apply to a broader range of problems. Design patterns provide general solutions, documented in a format that doesn't require specifics tied to a particular problem.

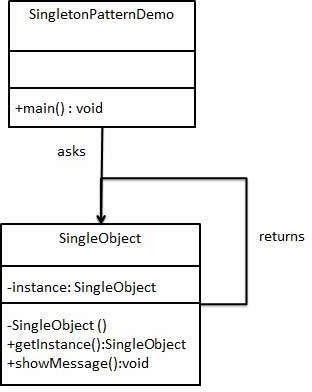
In addition, patterns allow developers to communicate using well-known, well understood names for software interactions. Common design patterns can be improved over time, making them more robust than ad-hoc designs.

# Singleton Pattern:

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

# Diagram of Singleton Pattern:



**Code of Singleton pattern:**

## Step 1:-

**Create a Singleton Class.**

**SingleObject.java**

**package** com.kratos;

**public** **class** SingleObject {

**private** **static** SingleObject *instance* = **new** SingleObject();

**private** SingleObject(){}

**public** **static** SingleObject getInstance(){

**return** *instance*;

}

**public** **void** showMessage(){

System.***out***.println("Welcome to the Gates of Hades..!");

}

**Step 2:-**

**Get the only object from the singleton class.**

**SingletonPatternDemo.java**

**package** com.kratos;

**public** **class** SinglePatternDemo {

**public** **static** **void** main(String[] args) {

SingleObject object = SingleObject.*getInstance*();

object.showMessage();

}

}

## Step 3:-

**Verify the output.**

