Report Group 5 Task01

Student Name: KhoaNCHE163718

Subject: PRN211

Topic: Singleton Pattern

Singleton Pattern, one of the many pattern belonging to the Creational Patterns, which ensure that a class has only one instance and provide a global point of access to it.

* UML Diagram:

Table

Description automatically generated

* **Singleton Pattern structural code in C#:**

This is a prime example of Singleton Pattern in its simplest form, which demonstrate its property of ensuring only a single instance (in this case Singleton) of the class can be created

Code:

Text

Description automatically generated

Output:



* Output Explanation: The line “Objects are the same instance” are printed because both S1 and S2 are of the same instance, this prove the property of Singleton Pattern which only one instance can be created.
* **Real World code in C#:**

This is one of many Real-time scenario where we can use Singleton Pattern, here we represent our Singleton as LoadBalancing object. Only a single class of this object can be created as the Server may periodically come on or off and every request must go through this one object that has knowledge about the state of the server.

Code:

Text

Description automatically generated

Text

Description automatically generated

Output:

Text

Description automatically generated

* Output Explanation: B1, B2, B3 and B4 are created which would result in the “Same Instance” line being printed, further proving the property of Singleton Pattern. The “\_server” List would have 5 different variable “ServerI”, “ServerII”, “ServerIII”, “ServerIV” and “ServerV” each represent a different server the Balancer would then Dispatch each request to each server randomly.
* **Other-real world appliance**
* **Service Proxies**: As we may have already know an welcome of the extensive operation in an application is invoking a service. This process usually take a very long time to create. If you create the Service proxy as Singleton then it will improve the performance of your application.
* **Data sharing**: If you have any constant values or configuration values then you can keep these values in Singleton So that these can be read by other components of the application.
* **Caching**: In such situations, the Singleton class can be used to handle the caching with thread synchronization in an efficient manner which drastically improves the performance of the application to eliminate the time-consuming process of fetching the data from a database.