Singleton Pattern

Overview

A singleton pattern ensures a class that has only one instance, and provides a global point of access to it.

Normal Class

- Multiple instances are created with new keyword for each object
- Constructor signature is public as default signature
- Each object creation instantiates new object class

Characters of Singleton class/pattern

- Only one instance is created of an object
- Constructor is private instead of public
- Object is created once, and it's instance is called from main method
- Default constructor is not called

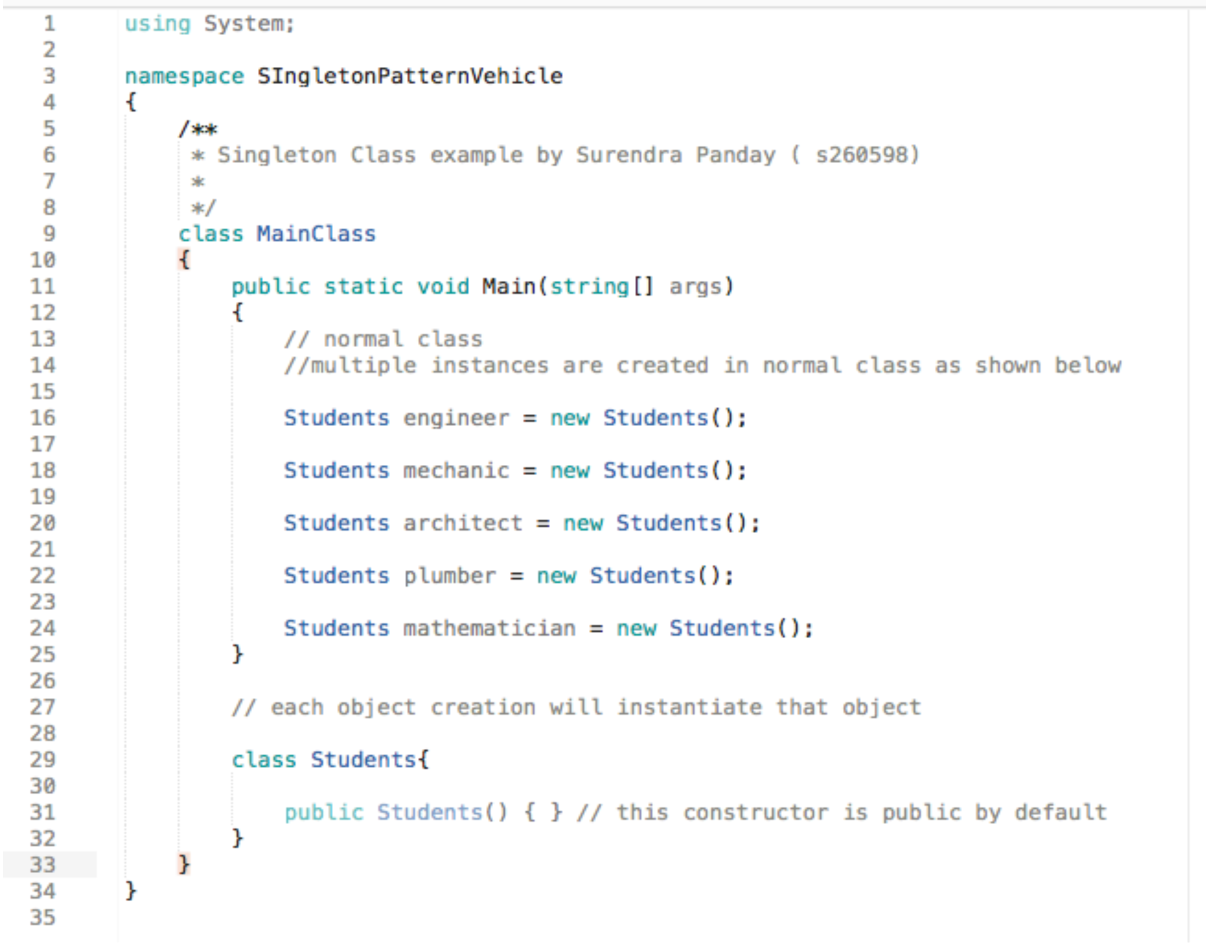
Scenario / Example

In a situation create a group of students where all students are identified by different professions such as Engineer, Doctor, Mechanic, Teacher. It can be done in two ways one is with normal way of instantiating each object with new keyword and continuing those classes for additional logic and other way with singleton design pattern transformation of those list of object with one instance as shown in before/after code snippet.

Benefits

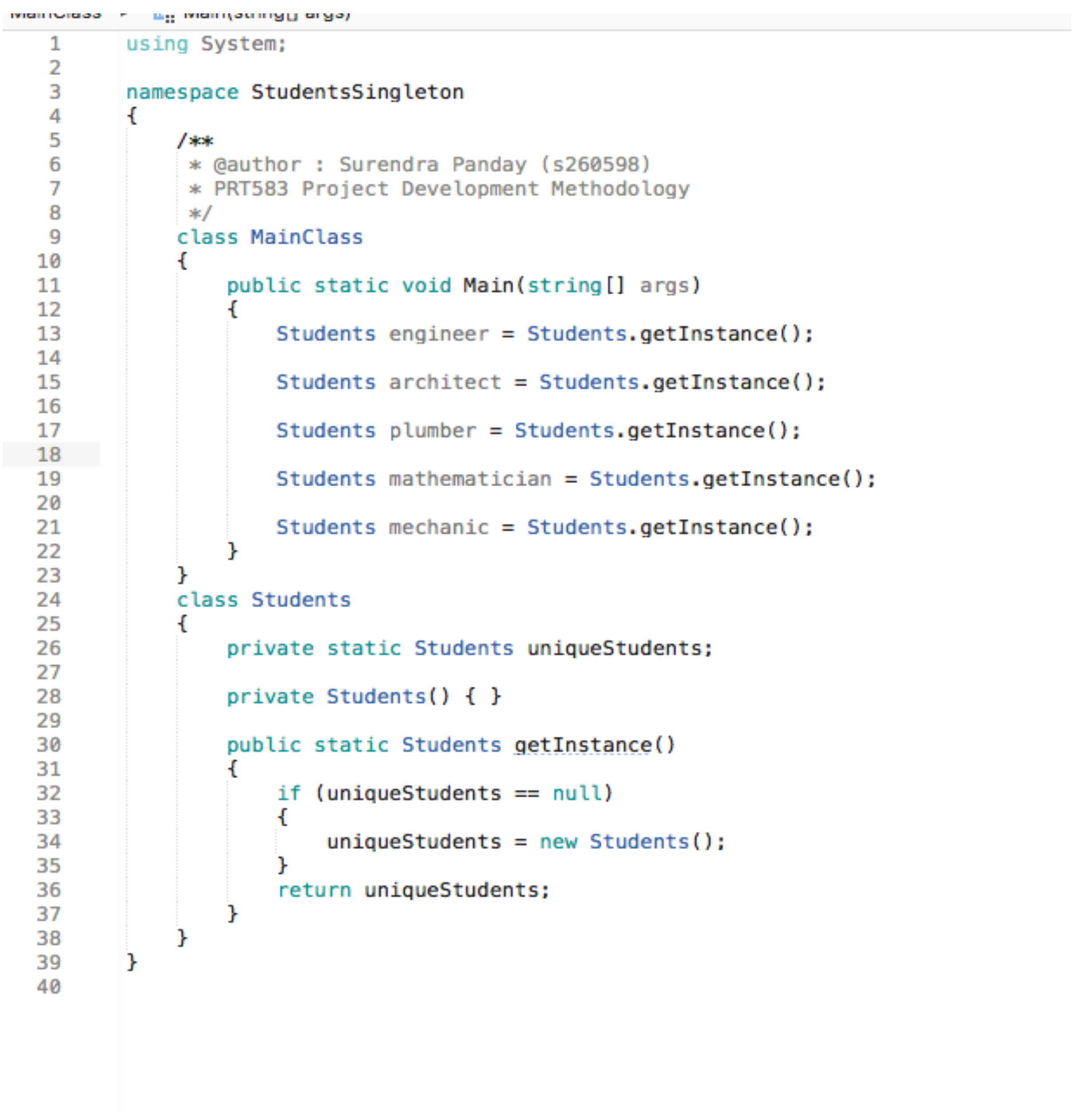
1) It prevents multiple classes create same instances and instead multiple classes would be forced to call same instance, so one object , one instance design is easy to read, eliminates duplicate. Freeman Eric., Freeman Elisa. (2004, p. 177)
2) It provides global access point to instance. When instance is required just query class and get that instance and create singleton class in lazy manner. It helps for resource intensive objects.
3) By using keyword 'synchronized' in singleton static method, it will support multithreading operation for resource intensive operations.

Before Refactor



```
using System;

namespace SIngletonPatternVehicle
{
    /**
     * Singleton Class example by Surendra Panday ( s260598)
     *
     */
    class MainClass
    {
        public static void Main(string[] args)
        {
            // normal class
            //multiple instances are created in normal class as shown below

            Students engineer = new Students();

            Students mechanic = new Students();

            Students architect = new Students();

            Students plumber = new Students();

            Students mathematician = new Students();
        }

        // each object creation will instantiate that object

        class Students{

            public Students() { } // this constructor is public by default
        }
    }
}

```

After Refactor



```
using System;

namespace StudentsSingleton
{
    /**
     * @author : Surendra Panday (s260598)
     * PRT583 Project Development Methodology
     */
    class MainClass
    {
        public static void Main(string[] args)
        {
            Students engineer = Students.getInstance();

            Students architect = Students.getInstance();

            Students plumber = Students.getInstance();

            Students mathematician = Students.getInstance();

            Students mechanic = Students.getInstance();
        }
    }
    class Students
    {
        private static Students uniqueStudents;

        private Students() { }

        public static Students getInstance()
        {
            if (uniqueStudents == null)
            {
                uniqueStudents = new Students();
            }
            return uniqueStudents;
        }
    }
}
```

UML Diagram

Please find it inside relevant project uml in gitlab

/StudentSingletonPatternUML.png

https://app.creately.com/diagram/tduuhLGm7gS/edit

Reference

Freeman Eric., Freeman Elisa. 2004. 1st ed. O'Reily Media Inc. 1995, Sebasttopo, CA