In this exercise you will be exploring more usage of the static keyword and the control of object creation.

# You must following the specifications exactly

# The Astronaut class

We are simulating a biosphere on the planet Mars. The biosphere is only able to sustain five persons. This application will try to mimic this condition.

|  |
| --- |
| **Astronaut**  Class |
| **Fields**  $- «property» ASTRONAUT\_COUNT : **int**  $- «property» THRESHOLD : **int**  **Properties**  + «property setter absent» Name : **string**  + «property setter absent» Nationality : **string** |
| **Methods**  - «constructor» Astronaut(name : **string**, nationality : **string**)  + ToString() : **string**  $+ CreateAstronaut(name : **string**, nationality : **string**) : Astronaut |

This class is composed of two fields, two properties, a constructor and two methods

#### Description of the members

##### Fields

**THRESHOLD** – this private field has class persistence. It represents the maximum number of astronauts that the station can safely accommodate. This is initialized to 5. This and **ASTRONAUT\_COUNT** is used in the **CreateAstronaut(string, string)** method to determine if another astronaut can be created.

**ASTRONAUT\_COUNT** – this private field has class persistence. It represents the number of astronauts that is currently living in the station. This is initialized to 0. This field and **THRESHOLD** fields are used in the **CreateAstronaut(string, string)** method to determine if another astronaut can be created.

##### Properties:

All the properties have public getter and private setters

**Name –** this string property represent the name of the astronaut. The getter is public and the setter is private.

**Nationality –** this string property represents the Country of origin of the astronaut. The getter is public and the setter is private.

##### Constructor:

**private Astronaut(string name, string nationality)** – This private constructor that takes two string arguments and assigns them to the respective properties. The class variable **ASTRONAUT\_COUNT** is incremented to reflect the correct count. This method is used by the CreateAstronaut() method.

This constructor is private; therefore the class may not be instantiated outside the class, hence the static method below is use to create an object.

A null is a stand in for ANY type.

##### Method:

**public static Astronaut CreateAstronaut(string name, string nationality)** – This public class method takes two string arguments. This method compares the **ASTRONAUT\_COUNT** and **THRESHOLD** to determine if there is room for another astronaut. If there is room then it instantiates a new astronaut and return it. Otherwise it returns **null**.

A null is a stand in for ANY type.

**public override string ToString()** – This public method overrides the ToString() method of the object class. It does not take any argument and returns a string representation of the object.

### Test Harness

Where would you put the code below?

You will need to write code to test all the features of your class:

You might create a collection and try to insert 6 objects into it. Then print all the objects. Because of the built-in constraints you will only be able to create 5 objects

You will need to code the method **SetThreshold**. From the usage, you should be able to deduce the method header.

List<Astronaut> astronauts = new List<Astronaut>();

Astronaut a = Astronaut.CreateAstronaut("Yuri Gagarin", "Russian"); // 1

if(a != null) astronauts.Add(a);

a = Astronaut.CreateAstronaut("Alan Shepard", "American"); // 2

if(a != null) astronauts.Add(a);

a = Astronaut.CreateAstronaut("Virgil Grissom", "American"); // 3

if(a != null) astronauts.Add(a);

a = Astronaut.CreateAstronaut("Gherman Titov", "Russian"); // 4

if(a != null) astronauts.Add(a);

a = Astronaut.CreateAstronaut("John Glenn", "American"); // 5

if(a != null) astronauts.Add(a);

a = Astronaut.CreateAstronaut("Scott Carpenter", "American"); // 6

if(a != null) astronauts.Add(a);

//only 5 astronaut created

Console.WriteLine("Only 5 astronauts created");

int count = 1;

foreach (var astronaut in astronauts)

{

Console.WriteLine($"{count++} {astronaut}");

}

/\*

Astronaut.SetThreshold(6);

astronauts.Add(Astronaut.CreateAstronaut("Scott Carpenter", "American")); // 6

//now 6 astronaut created

Console.WriteLine("Now 6 astronauts created");

count = 1;

foreach (var astronaut in astronauts)

{

Console.WriteLine(astronaut);

}

\*/

### Program output

Only 5 astronauts created

[Russian] Yuri Gagarin

[American] Alan Shepard

[American] Virgil Grissom

[Russian] Gherman Titov

[American] John Glenn

Now 6 astronauts created

[Russian] Yuri Gagarin

[American] Alan Shepard

[American] Virgil Grissom

[Russian] Gherman Titov

[American] John Glenn

[American] Scott Carpenter

Press any key to continue . . .

#### Additional functionality

Without modifying any of the data members, what would you add (method???) to be able to change the number of astronauts?

#### Further exploration

What happens if the properties Name and Nationality were decorated with the static keyword?