In this exercise you will be persisting objects by json serialization. We will start by serializing and deserializing a simple class and then we will try try the same with a list of objects

# You must following the specifications exactly

# The LizardColor enum

This type contains all the possible color that a lizard can have. This type consist of three members

|  |
| --- |
| **LizardColor**  Enum |
| **Members**  Black  Brown  Green |

# The Lizard class

This class is a simplistic representation of a lizard. There are three public properties and a single method.

|  |
| --- |
| **Astronaut**  Class |
| **Properties**  + «property» Name : **string**  + «property» Age : **int**  + «property» Color : LizardColor |
| **Methods**  + ToString() : **string** |

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

#### Description of the members

##### 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:

There is not constructor. Because all the properties are public, you will used object initialization.

##### Method:

**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 serialization and deserialization of a single lizard object and a list of liall 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?