* **C# OOP Retake Exam - 15 August 2019**
* **Overview**

Space Missions are very interesting and you have been sent to such one. Your mission is create a **Space Station** project, which navigates astronauts missions for collecting items from a foreign planet. The Space Station has **Astronauts** with different professional speacialties and their capability to survive in open space differs according to their essential needs, like the need for oxygen. Your task is to send them on missions and collect items from the different planets.

* **Setup**
* Upload **only the SpaceStation** project in every problem **except** **Unit Tests**
* **Do not modify the interfaces or their namespaces**
* Use **strong cohesion** and **loose coupling**
* **Use inheritance and the provided interfaces wherever possible**.
* This includes **constructors**, **method parameters** and **return types**
* **Do not** violate your **interface** **implementations** by adding **more public methods** or **properties** in the concrete class than the interface has defined
* Make sure you have **no public fields** anywhere
* **Task 1: Structure (50 points)**

You are given **5** interfaces, and you have to implement their functionality in the **correct classes**.

There are **5** types of entities in the application: **Astronaut, Backpack, Mission, Planet**. There should also have be a **AstronautRepository**, as well as **PlanetRepository**.

В приложението има 5 вида субекти: Астронавт, Раница, Мисия, Планета. Трябва да има и AstronautRepository, както и PlanetRepository.

**Astronaut**

**Astronaut** is a **base class** or any **type of astronaut** and it **should not be able to be instantiated**.

Астронавтът е базов клас или какъвто и да е вид астронавт и не трябва да може да бъде инстанциран.

**Data**

* **Name** – **string**
* If the name **is null or whitespace,** throw an **ArgumentNullException** with message: "Astronaut name cannot be null or empty."
* All names are unique
* **Oxygen** – **double**
* The oxygen of аn astronaut
* If the oxygen is below **0,** throw an **ArgumentException** with message:

"Cannot create Astronaut with negative oxygen!"

* **CanBreath** – calculated property, which returns **bool**
* **Bag – IBag**
* A property of type **Backpack**
* **Име - низ**
* **Ако името е нулево или бяло пространство, хвърлете ArgumentNullException със съобщение: "Името на космонавта не може да бъде нулево или празно."**
* **Всички имена са уникални**
* **Кислород - двоен**
* **Кислородът на астронавта**
* **Ако кислородът е под 0, хвърлете ArgumentException със съобщение:**
* **„Не мога да създам Астронавт с отрицателен кислород!“**
* **CanBreath - изчислено свойство, което връща bool**
* **Чанта - IBag**
* **Свойство от тип Backpack**

**Behavior**

**virtual void Breath()**

The **Breath()** method decreases astronauts' oxygen. Keep in mind that some types of Astronaut can implement the method in a different way.

* The method **decreases** the astronauts' oxygen by **10 units**.
* Astronaut's oxygen should not drop below zero
* Методът Breath () намалява кислорода на астронавтите. Имайте предвид, че някои видове Astronaut могат да прилагат метода по различен начин.
* Методът намалява кислорода на астронавтите с 10 единици.
* Кислородът на астронавта не трябва да пада под нулата

**Constructor**

An **Astronaut** should take the following values upon initialization:

string name, double oxygen

Астронавтът трябва да приеме следните стойности при инициализация:

име на низ, двоен кислород

**Child Classes**

There are several concrete types of **Astronaut**:

**Biologist**

Has **70 initial units of oxygen**.

Constructorshould take the following values upon initialization:

string name

Има 70 първоначални единици кислород.

Конструктор трябва да вземе следните стойности при инициализация:

име на низ

**Behavior**

**void Breath()**

The breath method **decreases the biologists' oxygen by 5** units.

Методът на дишане намалява кислорода на биолозите с 5 единици.

**Geodesist**

Has **50 initial units of ogyxen**.

Constructorshould take the following values upon initialization:

string name

Има 50 начални единици ogyxen.

Конструктор трябва да вземе следните стойности при инициализация:

име на низ

**Meteorologist**

Has initial **90** **units** **of oxygen**.

Constructorshould take the following values upon initialization:

string name

Има първоначални 90 единици кислород.

Конструктор трябва да вземе следните стойности при инициализация:

име на низ

**Backpack**

The **Backpack** is class that holds collection of items. **It should** be able to be **instantiated**.

Раницата е клас, който съдържа колекция от предмети. Тя трябва да може да бъде инстанцирана.

**Data**

* **Items –** a collection of **strings**
* **Предмети - колекция от струни**

**Constructor**

The constructor should not take any values upon initialization.

Конструкторът не трябва да приема стойности при инициализация.

**Planet**

The Planet is a class that holds information about the items that can be found on its surface. **It should** be able to be **instantiated**.

Планетата е клас, който съхранява информация за предметите, които могат да бъдат намерени на нейната повърхност. Тя трябва да може да бъде инстанцирана.

**Data**

* **Name – string**
* If the name **is null or whitespace,** throw an **ArgumentNullException** with message: "Invalid name!"
* **Items** – a collection of strings
* Име - низ
* Ако името е нулево или бяло пространство, хвърлете ArgumentNullException със съобщение: "Невалидно име!"
* Предмети - колекция от струни

**Constructor**

The constructor should take the following values upon initialization:

Конструкторът трябва да вземе следните стойности при инициализация:

string name

**Mission**

The **Mission** class holds the main action, which is the **Explore** method.

Класът Mission притежава основното действие, което е методът Explore.

**Behavior**

**void Explore(IPlanet planet, ICollection<IAstronaut> astronauts)**

Here is how the **Explore** method works:

* The astronauts start going out in open space one by one. They **can't go**, if they don't have **any** **oxygen** left.
* An astronaut lands on a planet and **starts collecting its items one by one**.
* He **finds an item** and he **takes a breath**.
* He **adds the item** to his **backpack** and respectively the item **must be removed** from the planet.
* **Astronauts** **can't keep collecting** **items** if their **oxygen becomes 0**.
* **If it becomes 0, the next astronaut starts exploring.**
* Ето как работи методът Explore:
* Астронавтите започват да излизат в открито пространство един по един. Те не могат да отидат, ако нямат кислород.
* Астронавт каца на планета и започва да събира предметите си един по един.
* Той намира предмет и си пое дъх.
* Той добавя предмета в раницата си и съответно предметът трябва да бъде изваден от планетата.
* Астронавтите не могат да продължат да събират предмети, ако кислородът им стане 0.
* Ако стане 0, следващият астронавт започва да изследва.

**AstronautRepository**

The astronaut repository is a repository for the astronauts that are on the Space Station.

Хранилището на астронавтите е хранилище за астронавтите, които са на Космическата станция.

**Data**

* **Models** – **a** **collection of astronauts (unmodifiable)**
* **Модели - колекция от астронавти (немодифицируема)**

**Behavior**

**void Add(IAstronaut astronaut)**

* Adds an astronaut in the Space Station.
* Every astronaut is unique and it is guaranteed that there will not be an astronaut with the same name.

Добавя астронавт в космическата станция.

Всеки астронавт е уникален и е гарантирано, че няма да има астронавт със същото име.

**bool Remove(IAstronaut astronaut)**

* Removes an astronaut from the collection. Returns true if the deletion was sucessful.

**IAstronaut FindByName(string name)**

* Returns an astronaut with that name, if he exists. If he doesn't, returns null.
* Връща астронавт с това име, ако съществува. Ако няма, връща нула.

**PlanetRepository**

The planet repository is a repository for planets that await to be explored.

Планетарното хранилище е хранилище за планети, които чакат да бъдат проучени.

**Data**

* **Models** – **a** **collection of planets (unmodifiable)**
* **Модели - колекция от планети (немодифицируема)**

**Behavior**

**void Add(IPlanet planet)**

* Adds a planet for exploration.
* Every planet is unique and it is guaranteed that there will not be a planet with the same name.
* Добавя планета за изследване.
* Всяка планета е уникална и е гарантирано, че няма да има планета със същото име.

**bool Remove(IPlanet planet)**

* Removes a planet from the collection. Returns true if the deletion was sucessful.
* Премахва планета от колекцията. Връща истина, ако изтриването беше успешно.

**IPlanet FindByName(string name)**

* Returns a planet with that name.
* It is guaranteed that the planet exists in the collection.
* Връща планета с това име.
* Гарантирано е, че планетата съществува в колекцията
* **Task 2: Business Logic (150 points)**

**The Controller Class**

The business logic of the program should be concentrated around several **commands**. You are given interfaces, which you have to implement in the correct classes.

**Note: The Controller class SHOULD NOT handle exceptions! The tests are designed to expect exceptions, not messages!**

The first interface is **IController**. You must create a **Controller** class, which implements the interface and implements all of its methods. The constructor of **Controller** does not take any arguments. The given methods should have the following logic:

Първият интерфейс е IController. Трябва да създадете клас Controller, който реализира интерфейса и реализира всичките му методи. Конструкторът на Controller не взема никакви аргументи. Дадените методи трябва да имат следната логика:

**Commands**

There are several commands, which control the business logic of the application. They are stated below.

Има няколко команди, които контролират бизнес логиката на приложението. Те са посочени по-долу.

**AddAstronaut Command**

**Parameters**

* **type** – **string**
* **astronautName - string**

**Functionality**

Creates an astronaut with the given name of the given type. If the astronaut is invalid, throw an **InvalidOperationException** with message:

"Astronaut type doesn't exists!"

The method should **return** the following message:

* "Successfully added {astronautType}: {astronautName}!"
* Създава астронавт с даденото име на дадения тип. Ако астронавтът е невалиден, хвърлете InvalidOperationException със съобщение:
* „Тип астронавт не съществува!“
* Методът трябва да върне следното съобщение:
* „Успешно добавено {astronautType}: {astronautName}!“

**AddPlanet Command**

**Parameters**

* **planetName - string**
* **items – params[] string**

**Functionality**

Creates a **planet** with the provided **items** and **name**.

When the planet is created, keep it and **return** the following message:

* "Successfully added Planet: {planetName}!".
* Създава планета с предоставените елементи и име.
* Когато планетата е създадена, запазете я и върнете следното съобщение:
* „Успешно добавен планета: {planetName}!“.

**RetireAstronaut Command**

**Parameters**

* **astronautName – string**

**Functionality**

**Retires the astronaut** from the space station by **removing** it from **its repository**. If an astronaut with that name doesn't exist, throw **InvalidOperationException** with return the following message:

* "Astronaut {astronautName} doesn't exists!"

If an astronaut is successfully retired, **remove it from the repository** and **return** the following **message**:

* "Astronaut {astronautName} was retired!"
* Пенсионира астронавта от космическата станция, като я изважда от хранилището си. Ако астронавт с това име не съществува, хвърлете InvalidOperationException, като върнете следното съобщение:
* „Астронавт {astronautName} не съществува!“
* Ако астронавтът се пенсионира успешно, го извадете от хранилището и върнете следното съобщение:
* „Астронавтът {astronautName} беше пенсиониран!“

**ExplorePlanet Command**

**Parameters**

* **planetName - string**

**Functionality**

When the explore command is called, the action happens. You should start exploring the given planet, by sending the astronauts that are most suitable for the mission:

* You call each of the astronauts and pick only the ones that have oxygen above 60 units.
* You send the suitable astronauts on a mission to explore the planet.
* If you **don't have any** **suitable** **astronauts**, throw an **InvalidOperationException** with the following message:

"You need at least one astronaut to explore the planet"

* After a mission, you must **return the following message**, with the **name of the explored planet** and the **count** of the **astronauts** that **had given their lives** for the mission:

"Planet: {planetName} was explored! Exploration finished with {deadAstronauts} dead astronauts!"

Когато се извика командата за изследване, действието се случва. Трябва да започнете да изследвате дадената планета, като изпратите астронавтите, които са най-подходящи за мисията:

Обаждате се на всеки от астронавтите и избирате само онези, които имат кислород над 60 единици.

Изпращате подходящите астронавти на мисия за изследване на планетата.

Ако нямате подходящи астронавти, хвърлете InvalidOperationException със следното съобщение:

„Имате нужда от поне един астронавт, за да изследвате планетата“

След мисия трябва да върнете следното съобщение с името на изследваната планета и броя на астронавтите, които са дали живота си за мисията:

"Планета: {planetName} беше проучена! Проучването приключи с {deadAstronauts} мъртви астронавти!"

**Report Command**

**Functionality**

Returns the information about the astronauts. If any of them **doesn't have bag items**, print **"none"** instead.

Връща информацията за астронавтите. Ако някой от тях няма елементи на чантата, отпечатайте вместо „няма“.

"{exploredPlanetsCount} planets were explored!  
Astronauts info:  
Name: {astronautName}  
Oxygen: {astronautOxygen}  
Bag items: {bagItem1, bagItem2, …, bagItemn} / none  
…  
Name: {astronautName}  
Oxygen: {astronautOxygen}  
Bag items: {bagItem1, bagItem2, …, bagItemn} / none"

**Note: Use \r\n or Environment.NewLine for a new line.**

**Exit Command**

**Functionality**

Ends the program.

**Input / Output**

You are provided with one interface, which will help you with the correct execution process of your program. The interface is **IEngine** and the class implementing this interface should read the input and when the program finishes, this class should print the output.

Осигурен ви е един интерфейс, който ще ви помогне с правилния процес на изпълнение на вашата програма. Интерфейсът е IEngine и класът, реализиращ този интерфейс, трябва да чете входа и когато програмата завърши, този клас трябва да отпечата изхода.

**Input**

Below, you can see the **format** in which **each command** will be given in the input:

* **AddAstronaut** **{astronautType} {astronautName}**
* **AddPlanet** **{planetName} {item1} {item2}… {itemN}**
* **RetireAstronaut** **{astronautName}**
* **ExplorePlanet {planetName}**
* **Report**
* **Exit**

**Output**

Print the output from each command when issued. If an exception is thrown during any of the commands' execution, print the exception message.

Отпечатайте изхода от всяка команда, когато бъде издадена. Ако по време на изпълнение на която и да е от командите е хвърлено изключение, отпечатайте съобщението за изключение.

**Examples**

|  |
| --- |
| **Input** |
| **AddAstronaut Biologist Oliver**  **AddAstronaut Geodesist Jake**  **AddAstronaut Meteorologist James**  **AddAstronaut Biologist Michael**  **AddAstronaut Meteorologist David**  **AddAstronaut Meteorologist Thomas**  **AddAstronaut Engineer Alexander**  **AddPlanet Mercury**  **AddPlanet Mars Carbon**  **RetireAstronaut David**  **ExplorePlanet Mars**  **Report**  **Exit** |
| **Output** |
| **Successfully added Biologist: Oliver!**  **Successfully added Geodesist: Jake!**  **Successfully added Meteorologist: James!**  **Successfully added Biologist: Michael!**  **Successfully added Meteorologist: David!**  **Successfully added Meteorologist: Thomas!**  **Astronaut type doesn't exists!**  **Successfully added Planet: Mercury!**  **Successfully added Planet: Mars!**  **Astronaut David was retired!**  **Planet: Mars was explored! Exploration finished with 0 dead astronauts!**  **1 planets were explored!**  **Astronauts info:**  **Name: Oliver**  **Oxygen: 65**  **Bag items: Carbon**  **Name: Jake**  **Oxygen: 50**  **Bag items: none**  **Name: James**  **Oxygen: 90**  **Bag items: none**  **Name: Michael**  **Oxygen: 70**  **Bag items: none**  **Name: Thomas**  **Oxygen: 90**  **Bag items: none** |

|  |
| --- |
| **Input** |
| **AddAstronaut Geodesist Jake**  **AddPlanet Mars Carbon**  **ExplorePlanet Mars**  **Report**  **AddAstronaut Biologist Jack**  **AddAstronaut Meteorologist Liam**  **AddAstronaut Biologist Michael**  **AddAstronaut Meteorologist David**  **AddAstronaut Meteorologist Thomas**  **AddPlanet Jupiter Titanium Quartz Aluminium Azurnium Cobalt Copper Iron Lead Lithium Plutonium Mercury Nickel Magnesium Diamond Gold Carbon**  **RetireAstronaut David**  **RetireAstronaut William**  **ExplorePlanet Jupiter**  **Report**  **Exit** |
| **Output** |
| **Successfully added Geodesist: Jake!**  **Successfully added Planet: Mars!**  **You need at least one astronaut to explore the planet!**  **0 planets were explored!**  **Astronauts info:**  **Name: Jake**  **Oxygen: 50**  **Bag items: none**  **Successfully added Biologist: Jack!**  **Successfully added Meteorologist: Liam!**  **Successfully added Biologist: Michael!**  **Successfully added Meteorologist: David!**  **Successfully added Meteorologist: Thomas!**  **Successfully added Planet: Jupiter!**  **Astronaut David was retired!**  **Astronaut William doesn't exists!**  **Planet: Jupiter was explored! Exploration finished with 1 dead astronauts!**  **1 planets were explored!**  **Astronauts info:**  **Name: Jake**  **Oxygen: 50**  **Bag items: none**  **Name: Jack**  **Oxygen: 0**  **Bag items: Titanium, Quartz, Aluminium, Azurnium, Cobalt, Copper, Iron, Lead, Lithium, Plutonium, Mercury, Nickel, Magnesium, Diamond**  **Name: Liam**  **Oxygen: 70**  **Bag items: Gold, Carbon**  **Name: Michael**  **Oxygen: 70**  **Bag items: none**  **Name: Thomas**  **Oxygen: 90**  **Bag items: none** |

* **Task 3: Unit Tests (100 points)**

You will receive a skeleton with **Hero** and **HeroRepository** classes inside. The class will have some methods, fields and one constructor, which are working properly. You are **NOT ALLOWED** to change any class. Cover the whole class with unit tests to make sure that the class is working as intended.

You are provided with a **unit test project** in the **project skeleton**.

Do **NOT** use **Mocking** in your unit tests!

Ще получите скелет с класове Hero и HeroRepository вътре. Класът ще има някои методи, полета и един конструктор, които работят правилно. НЕ СЕ ИЗПРАТЕН да променяте никой клас. Покрийте целия клас с единични тестове, за да сте сигурни, че класът работи по предназначение.

Осигурен ви е тестов проект в скелета на проекта.

НЕ използвайте подигравки в тестовете на вашето устройство!