**C# OOP Exam - 11 August 2019**

* **Overview**
* **Преглед**

GTA Vice city is one of the greatest games ever. It's a action-adventure game with main player Tommy Vercetti. You have been asked from "Rockstar Games" to create an extension. Your task is to create a entity which will be tha main player and one more which will be the same - civil player. You GTA witout guns is very boring, so that's why you need to crete enities which will represent guns. Finally the fight is happen in the gangsta neighbourhood.

GTA Vice city е една от най-великите игри някога. Това е екшън-приключенска игра с главния играч Томи Верчети. От вас поискахте от „Rockstar Games“ да създадете разширение. Вашата задача е да създадете образувание, което ще бъде главен играч и още едно, което ще бъде същото - граждански играч. Вие GTA без оръжия е много скучен, така че затова трябва да създадете хора, които ще представляват оръжия. Накрая двубоят се случва в махалата на гангстата.

* **Setup**
* **Настройвам**
* Upload **only the ViceCity** project in every problem **except** **Unit Tests**
* **Качете само проекта ViceCity във всеки проблем, с изключение на тестовете на единица**
* **Do not modify the interfaces or their namespaces**
* **Не променяйте интерфейсите или техните пространства от имена**
* Use **strong cohesion** and **loose coupling**
* **Използвайте силно сцепление и разхлабена връзка**
* **Use inheritance and the provided interfaces wherever possible**.
* Използвайте наследяване и предоставените интерфейси, когато е възможно.
* This includes **constructors**, **method parameters** and **return types**
* **Това включва конструктори, параметри на метода и видове връщане**
* **Do not** violate your **interface** **implementations** by adding **more public methods** or **properties** in the concrete class than the interface has defined
* Не нарушавайте реализациите на вашия интерфейс, като добавите повече публични методи или свойства в конкретния клас, отколкото интерфейсът е определил
* Make sure you have **no public fields** anywhere
* Уверете се, че нямате публични полета никъде
* **Task 1: Structure (50 points)**

You are given **4** interfaces, and you have to implement their functionality in the **correct classes**.

Дават ви 4 интерфейса и трябва да внедрите тяхната функционалност в правилните класове.

There are **4** types of entities in the application: **Player, Gun, Neighborhood** and **GunRepository**:

В приложението има 4 вида субекти: Player, Gun, Neighborhood и GunRepository:

**Player**

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

Играчът е базов клас или всякакъв тип играчи и не трябва да може да бъде инстанциран.

**Data**

* **Name** – **string**
* If the username **is null or whitespace,** throw an **ArgumentNullException** with message: "Player'sname cannot be null or a whitespace!"
* All names are unique
* Ако потребителското име е нулево или бяло пространство, хвърлете ArgumentNullException със съобщение: "Името на играча не може да бъде нулево или бяло пространство!"
* Всички имена са уникални
* **LifePoints** – **int**
* The health of а player
* If the health is below **0,** throw an **ArgumentException** with message:

"Player life points cannot be below zero!"

Здравето на играч

Ако здравето е под 0, хвърлете ArgumentException със съобщение:

„Жизнените точки на играча не могат да бъдат под нулата!“

* **GunRepository - IRepository<Gun>**
* Generic repository of all **player's** guns
* Общо хранилище на всички пушки на играча
* **IsAlive –** calculated property, which returns **bool**
* **IsAlive - изчислено свойство, което връща bool**

**Behavior**

**void TakeLifePoints(int points)**

The **TakeLifePoints** method decreases players' life points.

Методът TakeLifePoints намалява жизнените точки на играчите.

* Player's life points should not drop below zero
* Points Жизнените точки на играча не трябва да падат под нулата

**Constructor**

A **Player** should take the following values upon initialization:

Играчът трябва да приеме следните стойности при инициализация:

string name, int lifePoints

**Child Classes**

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

Има няколко конкретни типа играчи:

**MainPlayer**

Has **100 initial life points** and the main player has only one name **"Tommy Vercetti".** The constructorshould not take name and life points values upon initialization.

Има 100 начални житейски точки, а основният играч има само едно име "Томи Версети". Конструкторът не трябва да приема стойности за име и жизнени точки при инициализация.

**CivilPlayer**

Has **50 initial life points**.

Има 50 начални жизнени точки

Constructorshould take the following values upon initialization:

Конструктор трябва да вземе следните стойности при инициализация:

string name

**Gun**

The **Gun** is a base class for any type of gun and it should not be able to be instantiated.

Пистолетът е основен клас за всеки тип пистолет и не трябва да може да бъде инстанциран.

**Data**

* **Name – string**
* If the name of the gun **is null or empty,** throw an **ArgumentException** with message:

"Name cannot be null or a white space!"

* All names are unique
* Ако името на пистолета е нищожно или празно, хвърлете ArgumentException със съобщение:
* „Името не може да бъде нулево или бяло интервал!“
* ⦁ Всички имена са уникални
* **BulletsPerBarrel – int**
* If the bullets are **below zero,** throw an **ArgumentException** with message:

"Bullets cannot be below zero!"

* The **initial** **BulletsPerBarrel** **count** is the actual **capacity** of the **barrel**!
* Ако куршумите са под нулата, хвърлете ArgumentException със съобщение:
* „Куршумите не могат да бъдат под нулата!“
* ⦁ Първоначалният брой на BulletsPerBarrel е действителният капацитет на цевта!
* **TotalBullets - int**
* If the total bullets are **below zero,** throw an **ArgumentException** with message:

"Total bullets cannot be below zero!"

Ако общите куршуми са под нулата, хвърлете ArgumentException със съобщение:

„Общите куршуми не могат да бъдат под нулата!“

* **CanFire –** calculated property, which returns **bool**
* **CanFire - изчислено свойство, което връща bool**

**Behavior**

**int Fire()**

The Fire method acts **different** in all **child classes**. It shoots bullets and returns the number of bullets that were shot. Here is how it works:

Методът Огън действа различно във всички класове на деца. Стреля куршуми и връща броя на изстреляните куршуми. Ето как работи:

* Your guns have a **barrel**, which have a certain **capacity** **for bullets** and you shoot a different **count of bullets** when you **pull the trigger.**
* **Пушките ви имат цев, която има определен капацитет за куршуми и вие изстрелвате различен брой куршуми, когато дръпнете спусъка.**
* **If your barrel becomes empty**, you need to **reload** before you can shoot again.
* Ако цевта ви се изпразни, трябва да презаредите, преди да можете да стреляте отново.
* **Reloading** is done by refilling the **whole** **barrel of the gun (Keep in mind, that every barrel has capacity).**
* **Презареждането се извършва чрез презареждане на цялата цев на пистолета (имайте предвид, че всяка цев има капацитет)**
* The bullets you **take for reloading** are **taken** from the gun's **total bullets stock**.
* Куршумите, които вземате за презареждане, са взети от общия запас от куршуми.

Keep in mind, that every type of gun shoots **different count** of **bullets**, when you **pull the trigger**!

Имайте предвид, че всеки тип пистолет изстрелва различен брой куршуми, когато дръпнете спусъка!

**Constructor**

A **Gun** should take the following values upon initialization:

Пистолет трябва да вземе следните стойности при инициализация:

string name, int bulletsPerBarrel, int totalBullets

**Child Classes**

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

Има няколко конкретни вида пушки:

**Pistol**

Has **10 bullets per barrel** and **100 total bullets**.

Има 10 куршума на барел и 100 куршума.

**Behavior**

**int Fire()**

The pistol shoots only **one** **bullet**.

Пистолетът изстрелва само един куршум.

**Constructor**

Constructorshould take the following values upon initialization:

Конструктор трябва да вземе следните стойности при инициализация:

string name

**Rifle**

Has **50 bullets** per **barrel** and **500** **total bullets**.

Има 50 куршума на барел и 500 куршума.

**Behavior**

**int Fire()**

The rifle can shoot with **5 bullets**.

Пушката може да стреля с 5 куршума.

**Constructor**

Constructorshould take the following values upon initialization:

Конструктор трябва да вземе следните стойности при инициализация:

string name

**GangNeighbourhood**

The gang neighbourhood is the place where the shooting happens. It should inherit the **INeighbourhood** interface.

Кварталът на бандата е мястото, където се случва стрелбата. Той трябва да наследи интерфейса INeighbourhood.

**Behavior**

**void Action(IPlayer mainPlayer, ICollection<IPlayer> civilPlayers)**

That's the most interesting method.

Това е най-интересният метод.

**The main player** starts shooting at all the civil players. When he starts shooting at a civil player, the following **rules** apply:

Основният играч започва да стреля по всички граждански играчи. Когато започне да стреля по граждански играч, се прилагат следните правила:

* He takes a gun from his guns.
* Взема пистолет от пистолетите си.
* Every time he shoots, he **takes life points** from the civil player, which are **equal** to the **bullets that the current gun** shoots when the trigger is pulled.
* Всеки път, когато стреля, той взема жизнени точки от гражданския играч, които са равни на куршумите, които настоящият пистолет изстрелва при натискане на спусъка.
* If the **barrel of his gun becomes empty**, he **reloads** from his **bullets stock** and **continues** **shooting** with the **current gun**, **until he uses all of its bullets**.
* Ако цевта на пистолета му се изпразни, той презарежда от запасите си от куршуми и продължава да стреля с текущия пистолет, докато не използва всичките му куршуми.
* If his **gun runs out** of total **bullets**, he takes **the next gun** he has and **continues** shooting.
* Ако пистолетът му изчерпи общите куршуми, той взема следващия пистолет, който има и продължава да стреля
* He shoots at the **current civil** player **until he / she is alive**.
* Той стреля по сегашния граждански играч, докато не е жив.
* If the civil player **dies**, he starts **shooting at the next one**.
* Ако гражданският играч умре, той започва да стреля в следващия.
* The **main player stops shooting only if he runs out of guns** or **until all the civil players are dead**.
* Основният играч спира да стреля само ако му изтече пистолет или докато всички граждански играчи не са мъртви

**The civil players** (**the ones that have stayed alive after the main player's attack**) attack second. They start shooting at him **one after another** and the following **rules** apply:

Гражданските играчи (тези, които са останали живи след атаката на основния играч) атакуват втори. Те започват да стрелят по него един след друг и важат следните правила:

* A civil player takes one of his guns and starts shooting at the main player.
* Граждански играч взема едно от пушките си и започва да стреля по главния играч.
* Every time he shoots, he **takes life points** from the main player, which are **equal** to the **bullets that the current gun shoots** when the trigger is pulled.
* Всеки път, когато стреля, той взема жизнени точки от основния играч, които са равни на куршумите, които настоящият пистолет изстрелва при натискане на спусъка.
* If the **barrel of his gun becomes empty**, he **reloads** from his **bullets stock** and **continues** **shooting** with the **current gun**, **until he uses all of its bullets**.
* Ако цевта на пистолета му се изпразни, той презарежда от запасите си от куршуми и продължава да стреля с текущия пистолет, докато не използва всичките му куршуми.
* If his current **gun runs out of all its bullets**, he takes **the next gun** he has and **continues** shooting.
* Ако сегашният му пистолет изтече от всичките му куршуми, той взема следващия пистолет, който има, и продължава да стреля.
* If a **civil player** runs out of **guns**, the **next civil player begins shooting**.
* Ако гражданският играч свърши с пушки, следващият граждански играч започва да стреля.
* If the main player **dies**, the whole **action** ends.
* Ако основният играч умре, цялото действие приключва.

**GunRepository**

The gun repository holds information about a player's guns.

В хранилището на пистолета има информация за оръжията на играча.

**Data**

* **Models** – **collection of guns (unmodifiable)**

**Behavior**

**void Add(IGun model)**

* Adds a gun in the collection.
* Добавя пистолет в колекцията.
* If the gun already exists in the player's collection of guns, don't add it.
* Ако пистолетът вече съществува в колекцията от пистолети, не го добавяйте.
* Every gun is unique.
* Всеки пистолет е уникален.

**bool Remove(IGun model)**

* Removes a gun from the collection.
* Премахва пистолет от колекцията.

**IGun Find(string name)**

* Returns a gun with that name.
* Връща пистолет с това име.
* It is guaranteed that the guns exists in the collection.
* Гарантирано е наличието на пистолети 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.

Бизнес логиката на програмата трябва да се концентрира около няколко команди. Дават ви интерфейси, които трябва да внедрите в правилните класове.

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

**Забележка: Класът на контролера НЕ трябва да обработва изключения! Тестовете са създадени да очакват изключения, а не съобщения!**

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. When a controller is initialized, the main player is created. 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. Когато се инициализира контролер, се създава основният плейър. 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.

**AddPlayer Command**

**Parameters**

* **Name** – **string**

**Functionality**

Creates a civil player with the given name.

Създава граждански играч с даденото име.

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

Методът трябва да върне следното съобщение:

* "Successfully added civil player: {civilPlayerName}!"

**AddGun Command**

**Parameters**

* **Type - string**
* **Name - string**

**Functionality**

Creates a **gun** with the provided **type** and **name**.

Създава пистолет с предоставения тип и име.

If the gun type is invalid, the method should return the following message:

Ако типът пистолет е невалиден, методът трябва да върне следното съобщение:

* "Invalid gun type!"

If the gun type is correct, keep the gun and **return** the following message:

Ако типът на пистолета е правилен, запазете пистолета и върнете следното съобщение:

* "Successfully added {name} of type: {type}".

**AddGunToPlayer Command**

**Parameters**

* **Name – string (player's name)**

**Functionality**

Adds the first added gun to the **player's** gun **repository**.

Добавя първия добавен пистолет в хранилището на пистолета на играча.

* If there no guns in the queue, return the following message:
* Ако няма опашки в опашката, върнете следното съобщение:

"There are no guns in the queue!"

* If the name of the player is **"Vercetti"**, you need to add the gun to the main player's repository and return the following message:
* Ако името на играча е "Vercetti", трябва да добавите пистолета в основното хранилище на играча и да върнете следното съобщение:

"Successfully added {gunName} to the Main Player: Tommy Vercetti"

* If you receive a name which doesn't exist, you should return the following message:
* Ако получите име, което не съществува, трябва да върнете следното съобщение:

"Civil player with that name doesn't exists!"

* If everything is successful, you should add the gun to the player's repository and return the following message:
* Ако всичко е успешно, трябва да добавите пистолета в хранилището на играча и да върнете следното съобщение:

"Successfully added {gunName} to the Civil Player: {playerName}"

**Fight Command**

**Functionality**

When the fight command is called, the action happens. You should start the action between the main player and all the civil players. When a civil player is killed, it should be removed from the repository.

Когато се извика командата за борба, действието се случва. Трябва да започнете действието между основния играч и всички граждански играчи. Когато граждански играч е убит, той трябва да бъде изваден от хранилището.

* If the main player still has all of his points and no one is dead or harmed from the civil players, you should return the following messages:
* Ако основният играч все още има всичките си точки и никой не е мъртъв или наранен от гражданските играчи, трябва да върнете следните съобщения:

"Everything is okay!"

* If any of the players has different life points, you should return the following message:
* Ако някой от играчите има различни житейски точки, трябва да върнете следното съобщение:

"A fight happened:"

"Tommy live points: {mainPlayerLifePoints}!"

"Tommy has killed: {deadCivilPlayers} players!"

"Left Civil Players: {civilPlayersCount}!"

**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, осъществяващ 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:

По-долу можете да видите формата, в който всяка команда ще бъде дадена във входа:

* **AddPlayer** **{player username}**
* **AddGun** **{gun type} {gun name}**
* **AddGunToPlayer** **{player name}**
* **Fight**

**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 при издаването. Ако по време на изпълнение на която и да е от командите е хвърлено изключение, отпечатайте съобщението за изключение.

**Examples**

|  |
| --- |
| **Input** |
| **AddGun Pistol Colt**  **AddGun Rifle SniperRifle**  **AddPlayer Alfie**  **AddPlayer Alexis**  **AddPlayer Bean**  **AddPlayer Beck**  **AddPlayer Camber**  **AddPlayer Burney**  **Fight**  **AddGunToPlayer Vercetti**  **AddGunToPlayer Vercetti**  **AddGunToPlayer Vercetti**  **Fight**  **Exit** |
| **Output** |
| **Successfully added Colt of type: Pistol**  **Successfully added SniperRifle of type: Rifle**  **Successfully added civil player: Alfie!**  **Successfully added civil player: Alexis!**  **Successfully added civil player: Bean!**  **Successfully added civil player: Beck!**  **Successfully added civil player: Camber!**  **Successfully added civil player: Burney!**  **Everything is okay!**  **Successfully added Colt to the Main Player: Tommy Vercetti**  **Successfully added SniperRifle to the Main Player: Tommy Vercetti**  **There are no guns in the queue!**  **A fight happened:**  **Tommy live points: 100!**  **Tommy has killed: 6 players!**  **Left Civil Players: 0!** |

|  |
| --- |
| **Input** |
| **AddGun Pistol Colt**  **AddGun Pistol ColtPython**  **AddGun Rifle SniperRifle**  **AddGun Rifle PSGSniper**  **AddGun Shotgun Spaz**  **AddPlayer Alfie**  **AddPlayer Alexis**  **AddPlayer Bean**  **AddPlayer Beck**  **AddPlayer Camber**  **AddPlayer Burney**  **AddGunToPlayer Bean**  **AddGunToPlayer Vercetti**  **AddGunToPlayer Alfie**  **AddGunToPlayer Arthur**  **AddGunToPlayer Alfie**  **AddGunToPlayer Burney**  **Fight**  **Exit** |
| **Output** |
| **Successfully added Colt of type: Pistol**  **Successfully added ColtPython of type: Pistol**  **Successfully added SniperRifle of type: Rifle**  **Successfully added PSGSniper of type: Rifle**  **Invalid gun type!**  **Successfully added civil player: Alfie!**  **Successfully added civil player: Alexis!**  **Successfully added civil player: Bean!**  **Successfully added civil player: Beck!**  **Successfully added civil player: Camber!**  **Successfully added civil player: Burney!**  **Successfully added Colt to the Civil Player: Bean**  **Successfully added ColtPython to the Main Player: Tommy Vercetti**  **Successfully added SniperRifle to the Civil Player: Alfie**  **Civil player with that name doesn't exists!**  **Successfully added PSGSniper to the Civil Player: Alfie**  **There are no guns in the queue!**  **A fight happened:**  **Tommy live points: 0!**  **Tommy has killed: 2 players!**  **Left Civil Players: 4!** |

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

You will receive a skeleton with **Astronaut** and **Spaceship** 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.

Ще получите скелет с класове Астронавт и Космически кораб 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.

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

Осигурен ви е тестов проект в скелета на проекта.

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

НЕ използвайте подигравки в тестовете на вашето устройство!