# C# OOP Regular Exam – 10 April 2024



**Influencer Manager App**

1. **Overview**

*In the fast-paced digital world, our Influencer Manager App is made to help organize and manage influencer marketing campaigns. This app lets you register influencers, create different campaigns with varying budgets and engagement needs, and keep track of their involvement and payments. It allows influencers to join campaigns, calculates payments according to set rules, and gives tools to monitor campaign stats. The app acts as a central hub for coordinating influencer marketing, making sure brands and influencers work together seamlessly.*

## Setup

* Upload **only the InfluencerManagerApp** project in every task **except** **Unit Tests.**
* **Do not modify the interfaces or their packages.**
* 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** in the concrete class than the interface has defined.
* Make sure you have **no public fields** anywhere.
* **Exception messages** and **output messages** can be found in the **"Utilities"** folder.
* To solve this problem use **Visual Studio 2022** and **netcoreapp 6.0**
* **Do not use** "\r\n" **for a new line.**

## Task 1: Structure (50 points)

**For this task’s evaluation logic in the methods isn’t included.**

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

There are **2** types of entities: **Influencer** and **Campaign**. There should also be an **InfluencerRepository** and **CampaignRepository**.

### Influencer

The Influencer is a **base class** of any **type of influencer,** and it **should not be able to be instantiated**.

#### Data

* **Username** - **string**
  + If the name **is null or whitespace,** throw a new **ArgumentException** with the following message:

"Username is required."

* **Followers -** **int**
  + Represents the **number of followers** the influencer has.
  + The value must be a **non-negative number,** or elsethrow a new **ArgumentException** with the message: "Followers count cannot be a negative number."
* **EngagementRate –** **double**
  + Represents the percentage value of the **engagement rate of the influencer** as a floating-point number.
* **Income – double**
  + Represents the **total monetary earnings** of the influencer from their influencer activities.
  + The value should be **set to zero** upon initializing a new instance of the influencer, indicating that they have not yet earned any income from their activities as an influencer.
* **Participations – IReadOnlyCollection<string>**
  + A string collection of the **brands' campaigns in which an influencer has participated**.

#### Override ToString() method:

Overrides the existing method ToString()and modify it, so the returned **string must be on a single line**, in the following format:

**"{Username}** - Followers: **{Followers}**, Total Income: **{Income}"**

#### Constructor

An Influencer should take the following values upon initialization:

string username, int followers, double engagementRate

The constructor should initialize a new instance of the Participations collection.

#### Behavior

##### void EarnFee(double amount)

This method **increases the influencer's Income** by the specified **amount.**

##### void EnrollCampaign(string brand)

This method allows an influencer to enroll in a campaign. When called, the name of the **brand** associated with the campaign **is added** to the influencer's **Participations** collection.

##### void EndParticipation(string brand)

This method allows an **influencer to end their participation in a campaign**. When called, the **name of the brand associated with the campaign is removed from the influencer's Participations collection**.

##### int CalculateCampaignPrice()

It **multiplies the product** of theinfluencer's **follower count** and **engagement rate** by a **predefined constant** **factor**:

**Followers \* EngagementRate \* factor**

* **The result** represents the campaign fee andshould be **rounded down** **to the nearest integer value.**
* **Each child class** (BusinessInfluencer, FashionInfluencer, BloggerInfluencer) **implements the CalculateCampaignPrice() method differently** according to their custom logic.

#### Child Classes

There are three specific types of Influencer:

**BusinessInfluencer**

It has **an** **EngagementRate value of 3.0**

It can **contribute** to **all campaigns**. Keep that in mind for the business logic.

The Constructor of the **BusinessInfluencer** should take the following parameters upon initialization:

string username, int followers

The **CalculateCampaignPrice()** method is implemented with a **factor of 0.15**

**FashionInfluencer**

It has **an EngagementRate value of 4.0**

It can **contribute** to **product campaigns**. Keep that in mind for the business logic.

The Constructor of the **FashionInfluencer** should take the following parameters upon initialization:

string username, int followers

The **CalculateCampaignPrice()** method is implemented with a **factor of 0.1**

**BloggerInfluencer**

It has **an EngagementRate value of 2.0**

It can **contribute** to **service campaigns**. Keep that in mind for the business logic.

The Constructor of the **BloggerInfluencer** should take the following parameters upon initialization:

stringusername, intfollowers

The **CalculateCampaignPrice()** method is implemented with a **factor of 0.2**

### Campaign

The Campaign is a **base class** of any **type of campaign,** and it **should not be able to be instantiated**.

#### Data

* **Brand** - **string**
  + If the **Brand** is **null or whitespace,** throw a new **ArgumentException** with the message:

"Brand is required."

* **Budget – double**
  + Representing the **budget available** for the campaign.
* **Contributors – IReadOnlyCollection<string>**
  + A string collection of the **usernames of influencers who participated in the campaign**.

#### Behavior

##### void Gain(double amount)

This method **increases the campaign’s budget** by the specified amount.

##### void Engage(IInfluencer influencer)

This method enables the campaign to collaborate with influencers. When invoked, it **includes the username** of the influencer **in the collection of contributors. You have to take the campaign price calculated amount. Decreases the campaign’s budget** by the amount, representing expenditures such as influencer fees.

#### Override ToString() method:

Overrides the existing method ToString()and modifies it, so the returned string **must be on a single line**, in the following format:

**"{campaignTypeName}** - Brand: **{Brand}**, Budget: **{Budget}**, Contributors: **{count}"**

#### Constructor

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

string brand, double budget

The constructor should initialize a new instance of the Contributors collection.

#### Child Classes

There are two concrete types of **Campaign**:

##### ProductCampaign

It has **a** **Budget value of $60,000**

It can **contribute** to **business** and **fashion** **influencers**. Keep that in mind for the business logic.

The Constructor of the **ProductCampaign** should take the following parameters upon initialization:

stringbrand

##### ServiceCampaign

It has **a Budget value of $30,000**

It can **contribute** to **business** and **blogger** **influencers**. Keep that in mind for the business logic.

The Constructor of the **ServiceCampaign** should take the following parameters upon initialization:

stringbrand

## InfluencerRepository

The **InfluencerRepository** is an **IRepository<IInfluencer>. Collection** of all created **influencers**.

#### Data

* **Models – a collection of influencers (unmodifiable)**

#### Behavior

**void AddModel(IInfluencer influencer)**

* **Adds** an **influencer** to the repository.

**bool Remove(IInfluencer influencer)**

* **Removes an influencer** from the repository. **Returns true** **if the deletion was successful**,   
  **otherwise returns false**.

**IInfluencer FindByName(string username)**

* Returns an **influencer with a username** **matching the given parameter**, if such exists. Otherwise, it returns **null**.

## CampaignRepository

The **CampaignRepository** is an **IRepository<ICampaign>. Collection** of all created **campaigns**.

#### Data

* **Models – a collection of campaigns (unmodifiable)**

#### Behavior

**void AddModel(ICampaign campaign)**

* **Adds** a **campaign** to the repository.

**bool Remove(ICampaign campaign)**

* **Removes a campaign** from the repository. **Returns true if the deletion was successful**,   
  **otherwise returns false**.

**ICampaign FindByName(string brand)**

* Returns a **campaign from the brand** **matching the given parameter**, if such exists. Otherwise, it returns **null**.

## 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: Do not use** "\r\n" **for a new line.**

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 logic described for each in the Commands section. When you create the **Controller** class, go into the **Engine** class constructor and uncomment the "this.controller = new Controller();" line.

**Data**

You need to keep track of some sequences, this is why you need some private fields in your controller class:

**Example:**

* **influencers – IRepository<****IInfluencer>**
* **campaigns – IRepository<ICampaign>**

**Commands**

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

#### RegisterInfluencer Command

##### Parameters

* **typeName – string**
* **username – string**
* **followers - int**

##### Functionality

The method should **create and add** a new **IInfluencer** to the **influencers** collection.

* If the given typeName is NOT presented as a valid **Influencer's** child class (BusinessInfluencer, FashionInfluencer, BloggerInfluencer), return the following message:   
  "{typeName**}** has not passed validation."
* If an influencer with the same **username** is already added to the repository, do not duplicate records, return the following message:   
  "{**userName}** is already registered in {**correctRepositoryTypeName}**."
* If none of the above cases is reached, the IInfluencer is successfully created. Store the influencer to the appropriate collection and return: "{**username}** registered successfully to the application."

#### BeginCampaign Command

##### Parameters

* **typeName - string**
* **brand - string**

##### Functionality

The method should **create and add** a new **ICampaign** to the **campaigns** collection.

* If the given **typeName** is NOT presented as a valid **Campaign's** child class (**ProductCampaign**, **ServiceCampaign**), return the following message:   
  "{typeName**}** is not a valid campaign in the application."
* If a campaign with the same **Brand** is already added to the repository, do not duplicate records, return the following message: "{**brand}** campaign cannot be duplicated."
* If the above case is not reached, create the correct type of **ICampaign** and add it to the appropriate collection. Return the following message: "{brand**}** started a {typeName**}**."

#### AttractInfluencer Command

##### Parameters

* **brand - string**
* **username – string**

##### Functionality

The method is responsible for attracting a **specific influencer** to a campaign started by the given brand:

* Validates if an **influencer** **with the given username** **exists** in the influencers repository. If **no influencer with the provided name is found**, return the following message: **"**{correctRepositoryTypeName} has no **{username}** registered in the application.**"**
* Validates if a **campaign** **with the given brand exists** in the campaigns repository. If **no campaign with the provided brand is found**, return the following message: **"**There is no campaign from **{brand}** in the application.**"**
* **Check if the influencer is already engaged with the specified campaign**. If the influencer **is already part** of the campaign's **Contributors**, return the following message: **"**{username} is already engaged for the **{brand}** campaign.**"**
* **Checks the type of campaign and whether the influencer's type is eligible to participate** based on the campaign's rules:
  + For a **ProductCampaign**, **only** **business** and **fashion** **influencers** can contribute
  + For a **ServiceCampaign**, **only business** and **blogger influencers** can contribute
  + If the **influencer is not eligible for the campaign due to type restrictions**, return the following message: **"**{username} is not eligible for the **{brand}** campaign.**"**
* If the **campaign's budget does not allow for additional influencer engagement**, the following message should be returned: **"**The budget for **{brand}** is insufficient to engage **{username}**.**"**
* If the influencer successfully joins the campaign, **update the influencer's** **Income** and **Participations**, **update the campaign's** **Contributors** and **Budget** accordingly, and return the following message: **"{username}** has been successfully attracted to the **{brand}** campaign.**"**

#### FundCampaign Command

##### Parameters

* **brand - string**
* **amount – string**

##### Functionality

The method is responsible for **adding additional funds to an existing campaign’s budget**:

* Validates if a **campaign with the given brand exists** in the campaigns repository. **If** **no campaign with the provided brand is found**, return the following message: **"**Trying to fund an invalid campaign.**"**
* Validate the **amount to ensure it is a positive number**. If the **amount is** **not a positive number**, return the following message: **"**Funding amount must be greater than zero.**"**
* **If both validations pass, the campaign's budget is increased by the specified amount.**Returnsthe following message: **"{brand}** campaign has been successfully funded with **{amount}** $**"**

#### CloseCampaign Command

##### Parameters

* **brand – string**

##### Functionality

The method manages the **process** of concluding a campaign:

* **Validates if a** **campaign with the given brand exists** in the campaigns repository. If **no campaign with the provided brand is found**, return the following message: **"**Trying to close an invalid campaign.**"**
* **Checks if the campaign’s budget exceeds $10,000**. This **threshold determines whether a campaign has met its targets** and can be considered for closure.
  + **If the campaign’s budget is $10,000 or less**, it is **deemed not to have met its targets**, and the following message is returned: **"{brand}** campaign cannot be closed as it has not met its financial targets.**"**
* If the campaign meets the closure criteria (**budget over $10,000**), **each influencer** who participated in the campaign **receives a bonus of $2,000, no matter the budget amount**. This bonus is added to each influencer's **Income** property.
* Additionally, for **each influencer** who participated in the campaign, **the campaign’s brand is removed from their Participations collection**.
* After **distributing the bonuses** and **updating the participations collections**, the **campaign is removed** from the campaigns repository, and the following message is returned: **"{brand}** campaign has reached its target.**"**

#### ConcludeAppContract Command

##### Parameters

* **username – string**

##### Functionality

This method **allows an influencer** **to** conclude their contract and be **removed from the application**, given certain conditions are met:

* **Validates if an influencer with the given username exists** in the influencers repository. **If no influencer with the provided username is found**, return the following message: **"{username}** has still not signed a contract.**"**
* **Checks if the influencer currently has any active campaigns in the participations collection**. If the influencer is part of any campaign's participations collection, they **cannot conclude their contract**. In this case, return the following message: **"{username}** cannot conclude the contract while enrolled in active campaigns.**"**
* **If the influencer has no active campaign participations**, they are **eligible to conclude their contract**. The **influencer is** then **removed from the influencers repository**. In this case, return the following message: **"{username}** concluded their contract.**"**

#### ApplicationReport Command

##### Functionality

This method generates a **detailed report for all influencers in the application**, providing insights into their **earnings**, **follower counts**, and **campaign participations**. To receive the correct result, **use the overridden ToString()** methods **for influencers and campaigns**. The report is structured as follows**:**

* **Influencers are ordered** **first by their total income in descending order**, and **then by their number of followers in descending order**.
* **For each influencer**, the **campaigns they have participated in are listed**. These campaign details **are sorted alphabetically** by the **campaign brand** name.
* **If the influencer has not participated in any campaigns, skip the active campaigns report.**

"**{influencer1}**

Active Campaigns:

--**{campaign1}**

--**{campaign2}**

**…**

--**{campaignn}**

**…**

**{influencern}**

Active Campaigns:

--**{campaign1}**

--**{campaign2}**

**…**

--**{campaignn}**

**"**

**NOTE: Do not use** "\r\n" **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 Engine, and the class implementing this interface should read the input, and when the program finishes, this class should print the output.

#### Input

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

* **RegisterInfluencer {typeName} {username} {followers}**
* **BeginCampaign {typeName} {brand}**
* **AttractInfluencer {brand} {username}**
* **FundCampaign {brand} {amount}**
* **CloseCampaign {brand}**
* **ConcludeAppContract {username}**
* **ApplicationReport**
* **Exit**

#### Output

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

#### Examples

|  |
| --- |
| **Input** |
| **RegisterInfluencer FashionInfluencer AlexFas\_33 20000**  **RegisterInfluencer BusinessInfluencer Tech#Guru 30000**  **RegisterInfluencer BloggerInfluencer TravelVirtu 25000**  **RegisterInfluencer BusinessInfluencer MarketPPMaven 50000**  **RegisterInfluencer FashionInfluencer StyleIcon 15000**  **RegisterInfluencer ReligionInfluencer NoahRivers 150000**  **RegisterInfluencer BusinessInfluencer TravelVirtu 35000**  **BeginCampaign ProductCampaign Nike**  **BeginCampaign ServiceCampaign FoodPanda**  **BeginCampaign ServiceCampaign Nike**  **BeginCampaign ProductCampaign Sony**  **BeginCampaign ServiceCampaign Uber**  **BeginCampaign HolidayCampaign Traveloka**  **BeginCampaign ServiceCampaign BookIt**  **AttractInfluencer Nike NotEX\_infl**  **AttractInfluencer TitanTech Tech#Guru**  **AttractInfluencer Nike TravelVirtu**  **AttractInfluencer Nike AlexFas\_33**  **AttractInfluencer Uber TravelVirtu**  **AttractInfluencer Uber MarketPPMaven**  **AttractInfluencer Uber Tech#Guru**  **AttractInfluencer BookIt MarketPPMaven**  **AttractInfluencer FoodPanda StyleIcon**  **FundCampaign Uber 20000**  **FundCampaign Traveloka 10000**  **FundCampaign BookIt -5000**  **CloseCampaign Traveloka**  **CloseCampaign Sony**  **CloseCampaign Nike**  **AttractInfluencer Uber MarketPPMaven**  **AttractInfluencer Uber MarketPPMaven**  **CloseCampaign Uber**  **ConcludeAppContract StyleIcon**  **ConcludeAppContract MarketPPMaven**  **ConcludeAppContract NoahRivers**  **ApplicationReport**  **Exit** |
| **Output** |
| **AlexFas\_33 registered successfully to the application.**  **Tech#Guru registered successfully to the application.**  **TravelVirtu registered successfully to the application.**  **MarketPPMaven registered successfully to the application.**  **StyleIcon registered successfully to the application.**  **ReligionInfluencer has not passed validation.**  **TravelVirtu is already registered in InfluencerRepository.**  **Nike started a ProductCampaign.**  **FoodPanda started a ServiceCampaign.**  **Nike campaign cannot be duplicated.**  **Sony started a ProductCampaign.**  **Uber started a ServiceCampaign.**  **HolidayCampaign is not a valid campaign in the application.**  **BookIt started a ServiceCampaign.**  **InfluencerRepository has no NotEX\_infl registered in the application.**  **There is no campaign from TitanTech in the application.**  **TravelVirtu is not eligible for the Nike campaign.**  **AlexFas\_33 has been successfully attracted to the Nike campaign.**  **TravelVirtu has been successfully attracted to the Uber campaign.**  **The budget for Uber is insufficient to engage MarketPPMaven.**  **Tech#Guru has been successfully attracted to the Uber campaign.**  **MarketPPMaven has been successfully attracted to the BookIt campaign.**  **StyleIcon is not eligible for the FoodPanda campaign.**  **Uber campaign has been successfully funded with 20000 $**  **Trying to fund an invalid campaign.**  **Funding amount must be greater than zero.**  **Trying to close an invalid campaign.**  **Sony campaign has reached its target.**  **Nike campaign has reached its target.**  **MarketPPMaven has been successfully attracted to the Uber campaign.**  **MarketPPMaven is already engaged for the Uber campaign.**  **Uber campaign cannot be closed as it has not met its financial targets.**  **StyleIcon concluded their contract.**  **MarketPPMaven cannot conclude the contract while enrolled in active campaigns.**  **NoahRivers has still not signed a contract.**  **MarketPPMaven - Followers: 50000, Total Income: 45000**  **Active Campaigns:**  **--ServiceCampaign - Brand: BookIt, Budget: 7500, Contributors: 1**  **--ServiceCampaign - Brand: Uber, Budget: 4000, Contributors: 3**  **Tech#Guru - Followers: 30000, Total Income: 13500**  **Active Campaigns:**  **--ServiceCampaign - Brand: Uber, Budget: 4000, Contributors: 3**  **TravelVirtu - Followers: 25000, Total Income: 10000**  **Active Campaigns:**  **--ServiceCampaign - Brand: Uber, Budget: 4000, Contributors: 3**  **AlexFas\_33 - Followers: 20000, Total Income: 10000** |

## Task 3: Unit Tests (100 points)

You will receive a **SocialMediaManager** skeleton with **Influencer** and **InfluencerRepository** classes inside. **InfluencerRepository** class will have some methods, fields, and constructors, that are working properly You are **NOT ALLOWED** to change any class. **Cover the whole class** (only **InfluencerRepository**) 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 CHANGE OR REMOVE ANY namespaces or usings.**
* **Do not use** "\r\n" **for a new line.**
* Do **NOT** use **Mocking** in your unit tests!
* In Judge, you upload **.zip** of the **unit test project** from the **skeleton.**