# C# OOP Retake Exam – 16 April 2024



1. **Overview**

*In the busy world of making content, where big ideas meet tight deadlines, a new challenge comes to life. "The Content Department" exam project invites you into the life of a team that makes cool and interesting content quickly. This journey asks for good planning, working well with others, and being creative. Everyone has special skills they use to complete tasks and solve problems. This project will not only test your skills in C# and how to design programs but also show you what it's like to work together in making content. Are you ready to guide your team to victory and make your mark online?*

## Setup

* Upload **only the TheContentDepartment** 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/Messages"** folder.
* Text document named **Expected** could be found in the **"Utilities/Results"** 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: **Resource** and **TeamMember**. There should also be a **ResourceRepository** and **MemberRepository**.

### Resource

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

#### Data

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

"Name cannot be null or whitespace."

* **Creator – string**
  + The name of the resource's creator.
* **Priority – int**
  + Exposes the resource's priority level.
* **IsTested – bool**
  + Boolean property that indicates the testing status of the resource.
  + Initial value should be set to **false**
* **IsApproved – bool**
  + Boolean property that indicates the approval status of the resource.
  + Initial value should be set to **false**

#### Behavior

##### void Test()

**Toggles** the **IsTested** status of the resource. This method **allows the testing status to be changed**, reflecting the resource's progression through the testing phase.

##### void Approve()

**Sets** the **IsApproved** flag to **true**, indicating that the resource has passed all necessary checks and is approved for use or publication.

#### 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:

**"{Name}** (**{objectTypeName}**), Created By: **{Creator}"**

#### Constructor

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

string name, string creator, int priority

#### Child Classes

There are three concrete types of **Resource**:

##### Exam

It has the highest **Priority** level**: 1**

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

stringname, string creator

##### Workshop

It has a middle **Priority** level**: 2**

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

stringname, string creator

##### Presentation

It has the lowest **Priority** level**: 3**

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

stringname, string creator

### TeamMember

TheTeamMember is a **base class** of **any** **type of team member,** and it **should not be able to be instantiated**.

#### Data

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

"Name cannot be null or whitespace."

* **Path – string**
  + This **property is set differently by the child classes**([**TeamLead**](#_TeamLead) and [**ContentMember**](#_ContentMember)).
  + Be careful with the **access modifier**!
* **InProgress – IReadOnlyCollection<string>**
  + A read-only collection maintaining a **list of resource names, that** **the team member is currently working on**.

#### Behavior

##### void WorkOnTask(string resourceName)

**Adds a resource name** in the **InProgress** collection.

##### void FinishTask(string resourceName)

**Removes** a resource name from the **InProgress** collection.

#### Constructor

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

string name, string path

#### Child Classes

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

##### TeamLead

Is **only allowed** to have a value of **Path** property: "Master"

* Check the second parameter path:
  + If the path value does not match the **TeamLead** validations, a new **ArgumentException** is thrown. The following message should be returned: "**{path}** path is not valid."

#### Constructor

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

stringname, string path

#### 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:

**"{Name}** (**{objectTypeName}**) – Currently working on **{**[**count of current tasks**](#InProgress)**}** tasks.**"**

##### ContentMember

Is **allowed** to have a value of **Path** property, among: "CSharp", "JavaScript", "Python", or "Java"

The Constructor of the **ContentMember takes the following actions:**

* Should take the following parameters upon initialization: stringname, string path
* Checks the second parameter path:
  + **ContentMember** should accept **only one of** the following values:   
    "CSharp", "JavaScript", "Python", or "Java".
  + If the path value does not match the **ContentMember** validations, a new **ArgumentException** is thrown. The following message should be returned: "**{path}** path is not valid."

#### 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:

**"{Name}** - **{Path}** path. Currently working on **{**[**count of current tasks**](#InProgress)**}** tasks.**"**

## ResourceRepository

The **ResourceRepository** is an **IRepository<IResource>. Collection** of the **resources**.

### Data

* **Models – IReadOnlyCollection<IResource>**
  + Returns a readonly **collection of all resources**, added to the repository.

### Behavior

**void Add(IResource model)**

* **Adds** a new **IResource** to the ResourceRepository.

**IResource TakeOne(string modelName)**

* Returns a resource with **a Name equal to the given modelName** from the collection, **if there is any**. Otherwise, it returns **null**.

## MemberRepository

The **MemberRepository** is an **IRepository<ITeamMember>. Collection** for the **members** of the team.

### Data

* **Models – IReadOnlyCollection<ITeamMember>**
  + Returns a readonly **collection of all members**, added to the repository.

### Behavior

**void Add(ITeamMember model)**

* **Adds** a new **ITeamMember** to the MemberRepository.

**ITeamMember TakeOne(string modelName)**

* Returns a member with **a Name equal to the given modelName** from the collection, **if there is any**. 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 the **IController**. You must create a **Controller** class, which implements the interface and implements all of its methods. The constructor of the **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 things, this is why you need some private fields in your controller class:

**Example:**

* **resources - ResourceRepository**
* **members - MemberRepository**

**Commands**

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

#### JoinTeam Command

##### Parameters

* **memberType - string**
* **memberName – string**
* **path – string**

##### Functionality

The method is designed to facilitate the **addition of new team members to a content development team**.

* If the given **memberType** is NOT presented as a valid **TeamMember's** child class (TeamLead, ContentMember), return the following message: "{memberType**}** is not a valid member type."
* A Content Team could only have one member for each position (CSharp, JavaScript, Python, or Java).
  + If the given path is equal to any of the members' Path property values, that means that the position is not vacant, and the following message should be returned: "Position is occupied."
* Verifies whether a team member with the same memberName already exists in the team (**MemberRepository**). If such a member is found, it prevents duplication by returning a message that the member is already part of the team: "{**memberName}** has already joined the team."
* If no duplication is found, carefully select the desired member class, according to the given path parameter. An ITeamMember from the correct class is created. Store the team member in the appropriate collection and return:   
  "{**memberName}** has joined the team. Welcome!"

#### CreateResource Command

##### Parameters

* **resourceType - string**
* **resourceName – string**
* **path - string**

##### Functionality

This method operates within the context of **ensuring that resources are not only created but also linked to the appropriate team members** who are **responsible for their development**. Here's an explanation of how this method functions:

* If the given **resourceType** is NOT presented as a valid **Resources's** child class (Exam, Workshop, or Presentation), return the following message: "{resourceType**}** type is not handled by Content Department."
* **Finding the responsible** ContentMember:
  + The method iterates through all **content members** trying to **find a ContentMember** matching the given **path**.
  + If **no such team member is found**, the following message is returned: **"**No content member is able to create the **{resourceName}** resource.**"**
  + If an appropriate ContentMember **is found**, check if their **InProgress collection contains the given resourceName**. If such exists, the following message is returned:   
    **"**The **{resourceName}** resource is being created.**"**
* **Resource creation**:
  + Depending on the **resourceType**, the method **creates a new instance** of the corresponding resource class (**Exam**, **Workshop**, or **Presentation**) with the specified **resourceName** and the **name of the responsible team member as the creator**.
  + After the **resource is created** and its **name is added to the InProgress collection of its creator**, it is **added** to the **repository of resources.**
* Finally, a **success message is returned**, indicating the resource's **successful creation**:  
  **"{creatorName}** created **{resourceType}** - **{resourceName}**.**"**

#### LogTesting Command

##### Parameters

* **memberName – string**

##### Functionality

The method is designed to **update the status of resources** **associated with a specific team member** in the content development team, specifically **marking a resource as having been tested**:

* The method attempts to **find a team member with the given memberName**. If **no matching team member is found**, it should **return a message** indicating the issue with the member name provided:  
  **"**Provide the correct member name.**"**
* **Find the highest priority NOT tested resource**:
  + Identify the **highest priority resource** (the one with the **lowest priority number**), that is **still not tested (where** the **IsTested** property value **is** equal to **False)**, and is **created by the located team member** from the **resources** collection (**where** the **Creator** property value **is equal to** the given **memberName** parameter).
  + If **no resources are found for the team member**, a **message should be returned**, indicating the absence of resources associated with the member:   
    **"{memberName}** has no resources for testing.**"**
* The method **identifies the TeamLead** from the repository of members.
* The **creator finishes working** on the resource and **passes it to the TeamLead.** The resource name is **excluded from the creator's InProgress** collection and **added to the TeamLead's InProgress** collection.
* **Resourcee is marked** **as tested**. **Returns a success message** indicating that the resource has been successfully tested: **"{resourceName}** is tested and ready for approval.**"**

#### ApproveResource Command

##### Parameters

* **resourceName – string**
* **isApprovedByTeamLead - bool**

##### Functionality

The method is dealing with the **approval process of resources** created within the team:

* The method starts by **finding the resource** in the respective repository. The parameters **resourceName** will **always return an existing model** within the collection.
* **Pre-Approval Testing Check:**
  + Before proceeding with approval or further actions, the method checks if the resource has been tested. A **not-tested resource cannot be approved**, reinforcing the importance of quality assurance.
  + If a resource hasn't been tested, a message is returned indicating the need for testing:  
    **"{resourceName}** cannot be approved without being tested.**"**
* The method **identifies the team lead from the collection of members**.
* **Approve or Re-Test resources**:
  + Based on the **isApproved** parameter, the method either approves the resource (setting its approval status to true) or marks it for re-testing.
  + If **isApprovedByTeamLead == true**, the resource's **IsApproved** **status is toggled to true**. The **team lead** is **noted to have finished their tasks** related to this resource. The following **message is returned:  
    "{teamLeadName}** approved **{resourceName}**.**"**
  + If **isApprovedByTeamLead == false**, the resource's **IsApproved** **status remains** and the resource's **IsTested** **status is toggled to false.** The following **message is returned:  
    "{teamLeadName}** returned **{resourceName}** for a re-test.**"**

#### DepartmentReport Command

##### Functionality

The method returns detailed information **for every resource that is approved in the ResourceRepository**. It also returns detailed information **for every member of the team, with their tasks, starting with the TeamLead.** To receive the correct output, use the ToString() method **of each model:**

"Finished Tasks:

--**{resource1}**

--**{resource2}**

**…**

--**{resourcen}**

Team Report:

--**{teamLeadName}** (TeamLead) - Currently working on {countOfTasks} tasks.

**{contentMember1} //{Name}** - **{Path} path.** Currently working on {countOfTasks} tasks.

**{contentMember2}**

**…**

**{contentMembern}**

**"**

**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:

* **JoinTeam** **{memberType} {memberName} {path}**
* **CreateResource** **{resourceType} {resourceName} {path}**
* **LogTesting {memberName}**
* **ApproveResource {resourceName} {isApprovedByTeamLead}**
* **DepartmentReport**
* **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** |
| **JoinTeam TeamLead YokoYong Master**  **JoinTeam ContentMember DaniDavis JavaScript**  **JoinTeam ContentMember DaniDavis Java**  **JoinTeam ContentMember PeterSully Java**  **JoinTeam ContentMember ValGendor Python**  **CreateResource Presentation Inheritance CSharp**  **JoinTeam ContentMember KrissThompson CSharp**  **JoinTeam TrainingMember DenisPeters CSharp**  **JoinTeam ContentMember ValGendor Java**  **CreateResource Lab LabDocument CSharp**  **CreateResource Exam JavaScriptOOP-Retake JavaScript**  **CreateResource Workshop TicTacToe Python**  **CreateResource Exam JavaScriptOOP-Regular JavaScript**  **CreateResource Exam JavaScriptOOP-Regular JavaScript**  **CreateResource Exam JavaOOP-Regular Java**  **CreateResource VideoContent WelcomeVideo Java**  **CreateResource Exam JavaOOP-Retake Java**  **CreateResource WorkShop Regex-Exercise Python**  **CreateResource Presentation Inheritance CSharp**  **CreateResource Exam C#OOP-Regular CSharp**  **LogTesting DenisPeters**  **LogTesting ValGendor**  **LogTesting ValGendor**  **LogTesting ValGendor**  **LogTesting DannyDavis**  **LogTesting PeterSully**  **LogTesting PeterSully**  **LogTesting KrissThompson**  **ApproveResource JavaOOP-Regular true**  **ApproveResource JavaOOP-Retake false**  **ApproveResource Inheritance true**  **ApproveResource C#OOP-Regular true**  **ApproveResource TicTacToe true**  **ApproveResource JavaScriptOOP-Retake true**  **DepartmentReport**  **Exit** |
| **Output** |
| **YokoYong has joined the team. Welcome!**  **DaniDavis has joined the team. Welcome!**  **DaniDavis has already joined the team.**  **PeterSully has joined the team. Welcome!**  **ValGendor has joined the team. Welcome!**  **No content member is able to create the Inheritance resource.**  **KrissThompson has joined the team. Welcome!**  **TrainingMember is not a valid member type.**  **Position is occupied.**  **Lab type is not handled by Content Department.**  **DaniDavis created Exam - JavaScriptOOP-Retake.**  **ValGendor created Workshop - TicTacToe.**  **DaniDavis created Exam - JavaScriptOOP-Regular.**  **The JavaScriptOOP-Regular resource is being created.**  **PeterSully created Exam - JavaOOP-Regular.**  **VideoContent type is not handled by Content Department.**  **PeterSully created Exam - JavaOOP-Retake.**  **WorkShop type is not handled by Content Department.**  **KrissThompson created Presentation - Inheritance.**  **KrissThompson created Exam - C#OOP-Regular.**  **Provide the correct member name.**  **TicTacToe is tested and ready for approval.**  **ValGendor has no resources for testing.**  **ValGendor has no resources for testing.**  **Provide the correct member name.**  **JavaOOP-Regular is tested and ready for approval.**  **JavaOOP-Retake is tested and ready for approval.**  **C#OOP-Regular is tested and ready for approval.**  **YokoYong approved JavaOOP-Regular.**  **YokoYong returned JavaOOP-Retake for a re-test.**  **Inheritance cannot be approved without being tested.**  **YokoYong approved C#OOP-Regular.**  **YokoYong approved TicTacToe.**  **JavaScriptOOP-Retake cannot be approved without being tested.**  **Finished Tasks:**  **--TicTacToe (Workshop), Created By: ValGendor**  **--JavaOOP-Regular (Exam), Created By: PeterSully**  **--C#OOP-Regular (Exam), Created By: KrissThompson**  **Team Report:**  **--YokoYong (TeamLead) - Currently working on 1 tasks.**  **DaniDavis - JavaScript path. Currently working on 2 tasks.**  **PeterSully - Java path. Currently working on 0 tasks.**  **ValGendor - Python path. Currently working on 0 tasks.**  **KrissThompson - CSharp path. Currently working on 1 tasks.** |

## Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside it. **DepartmentCloud** class will have some methods, fields, and constructors. **Cover the whole class** with the unit test to make sure that the class is working as intended.

* **Do NOT CHANGE OR REMOVE ANY namespaces or usings.**
* **Do not use** "\r\n" **for a new line.**
* In Judge, you upload **.zip** **(**with **ResourceCloud.Tests** inside**)** from the **skeleton.**