# C# OOP Exam Preparation II



**Highway To Peak**

Exam Preparation for the ["C# OOP" course @ SoftUni".](https://softuni.bg/trainings/4377/csharp-oop-february-2024)

You can submit your solutions in [Judge](https://judge.softuni.org/Contests/4501/CSharp-OOP-Retake-Exam-19-December-2023)

1. **Overview**

*Welcome to Gorakshep, a village town in the Himalayas. Every year, climbers come here for the Himalayan Heights Challenge. They aim to climb famous mountains like Mount Everest and K2.*

*These mountains are very high and hard to climb. Climbers need to pick their gear carefully and plan well. Everyone is excited as climbers get ready for their big climbs.*

## Setup

* Upload **only the HighwayToPeak** 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 **3** types of entities: **Peak, BaseCamp,** and **Climber**. There should also be a **PeakRepository** and **ClimberRepository**.

### Peak

#### Data

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

"Peak name cannot be null or whitespace."

* **Elevation -** **int**
  + Represents the **elevation of the specific peak in meters**.
  + Must be a **positive** **value**. If not, throw a new **ArgumentException** with the message: "Peak elevation must be a positive value."
* **DifficultyLevel –** **string**
  + Represents the level of difficulty to climb the specific peak. The property is allowed to accept the following options only: **"Extreme"**, **"Hard"** or **"Moderate".** This **validation** will occur in the **AddPeak** methodin the **Controller** class**.**

#### 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:

**"**Peak: **{Name}** -> Elevation: **{Elevation}**, Difficulty: **{DifficultyLevel}"**

#### Constructor

A Peak should take the following values upon initialization:

string name, int elevation, string difficultyLevel

### Climber

The Climber is a **base class** of any **type of climber,** 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:

"Climber's name cannot be null or whitespace."

* **Stamina – int**
  + The climber's **stamina,** in the mountains. Must be a value **from 0** **up to 10**, both inclusive.
  + If it exceeds **10** during any operation, it should be **reset to 10.**
  + If it drops **below zero** during any operation, it should be **reset to zero.**
* **ConqueredPeaks – IReadOnlyCollection<string>**
  + It will store a sequence of **names of peaks**, conquered by a climber.

#### Behavior

##### void Climb(IPeak peak)

The method will **add the given peakName** to the **collection** of **ConqueredPeaks**, if the **peak is already climbed** by the specific climber, **do NOT duplicate** the **peakName** in the collection of **ConqueredPeaks**, but climber can climb a peak unlimited times. The method will **always decrease** the climber’s **Stamina** by:

* If the peak’s DifficultyLevel is **"Extreme"** - 6 units ;
* If the peak’s DifficultyLevel is **"Hard"** - 4 units ;
* If the peak’s DifficultyLevel is **"Moderate"** - 2 units ;

##### void Rest(int daysCount)

It should be an **abstract method**. The climber's **Stamina** should be **recovered**. For every day of resting the different types of climbers will recover a different amount of Stamina. The exact recovery rate for each day of rest **will be defined in child classes**.

#### Override ToString() method:

Overrides the existing method ToString()and modifies it, so the returned string is in the following format:

**"{climberTypeName}** - Name: **{Name}**, Stamina: **{Stamina}**

Peaks conquered: no peaks conquered**/{peaksCount}"***HINT:* *The* ***ToString()*** *method should return output on* ***two*** *separate* ***lines***.

#### Constructor

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

string name, int stamina

#### Child Classes

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

##### OxygenClimber

The **OxygenClimber** class represents climbers who use supplemental oxygen. They may have higher stamina but recover more slowly due to reliance on external oxygen sources.

* **Will be allowed** to climb peaks with **extreme** difficulty level.
* **Will have an initial Stamina of 10 units.**
* **Will recover 1 unit of Stamina for every day of rest in the base camp.**

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

stringname

##### NaturalClimber

The **NaturalClimber** class is for climbers who climb without the aid of supplemental oxygen. They are acclimatized to high altitudes and can recover stamina more quickly, demonstrating greater natural adaptation to challenging environments.

* **Will NOT be allowed** to climb peaks with **extreme** difficulty level.
* **Will have an initial Stamina of 6 units.**
* **Will recover 2 units of Stamina for every day of rest in the base camp.**

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

stringname

### BaseCamp

#### Data

* **Residents – IReadOnlyCollection<string>**
  + An **alphabetically ordered collection** of the names of all climbers currently staying in the base camp.

#### Behavior

##### void ArriveAtCamp(string climberName)

A method to record the arrival of a climber at the base camp. It **adds the climber's name** to the **Residents** collection.

##### void LeaveCamp(string climberName)

A method to record the climber’s leaving from the base camp. It **removes the climber's name** from the **Residents** collection.

#### Constructor

A BaseCamp will take no parameters upon initialization.

## PeakRepository

The **PeakRepository** is an **IRepository<IPeak>. Collection** for the **peaks**.

### Data

* **All – IReadOnlyCollection<IPeak>**
  + Returns a readonly **collection of all peaks**, added to the repository.

### Behavior

**void Add(IPeak model)**

* **Adds** a new **IPeak** to the PeakRepository.

**IPeak Get(string name)**

* Returns a **peak with the given name** from the **collection**, if there is any. Otherwise, it returns **null**.

## ClimberRepository

The **ClimberRepository** is an **IRepository<IClimber>. Collection** for the **climbers**.

### Data

* **All – IReadOnlyCollection<IClimber>**
  + Returns a readonly **collection of all climbers**, added to the repository.

### Behavior

**void Add(IClimber model)**

* **Adds** a new **IClimber** to the ClimberRepository.

**IClimber Get(string name)**

Returns a **climber with the given name** 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 **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 things, this is why you need some private fields in your controller class:

**Example:**

* **peaks - PeakRepository**
* **climbers – ClimberRepository**
* **baseCamp - BaseCamp**

**Commands**

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

**AddPeak Command**

**Parameters**

* **name – string**
* **elevation – int**
* **difficultyLevel - string**

**Functionality**

The method should **create and add** a new **IPeak** to the **PeakRepository**.

* If a peak with the given name is already added to the PeakRepository, it should NOT be duplicated. Return the following message: "{name**}** is already added as a valid mountain destination."
* If the given difficulty level is NOT among one of the accepted values ("Extreme", "Hard" or "Moderate"), the peak should NOT be added and the following message should be returned: "{**difficultyLevel}** peaks are not allowed for international climbers."
* If none of the above cases is reached, the IPeak is successfully created. Store the peak in the appropriate collection and return: "{**name}** is allowed for international climbing. See details in {**correctRepositoryTypeName}**."

#### NewClimberAtCamp Command

##### Parameters

* **name - string**
* **isOxygenUsed - bool**

##### Functionality

The method should **create and add** a new **IClimber** to the **ClimberRepository** and the name of the new climber to the **base camp**.

* If a climber with the given **name** is already added to the ClimberRepository, it should NOT be duplicated. Return the following message: "{name**}** is a participant in {**correctRepositoryTypeName}** and cannot be duplicated."
* If the above case is not reached:
  + The isOxygenUsed used parameter, represents whether the climber uses external source of oxygen, when climbing peaks.
  + Pick the correct type of IClimber, depending on the isOxygenUsed parameter. Check the Climber child classes, and see which type uses supplemental oxygen and which does not.
  + Create the correct type of **IClimber** and add it to the ClimberRepository and the baseCamp. Return the following message: "{name**}** has arrived at the BaseCamp and will wait for the best conditions."

#### AttackPeak Command

##### Parameters

* **climberName - string**
* **peakName – string**

##### Functionality

The method is responsible for allowing the **specific climber** to attempt **to** **climb a specific peak**:

* **Validates if a climber with the given climberName is at the ClimberRepository**. If no climber with the provided name is found, return the following message: **"**Climber - **{climberName}**, has not arrived at the BaseCamp yet.**"**
* **Validates if a peak with the given peakName exists in the PeakRepository**. If no peak with the provided name is found, return the following message: **"{peakName}** is not allowed for international climbing.**"**
* **BaseCamp search** - If a climber with the given **climberName** is **NOT found** in the base camp, the attack of the peak is postponed. Return the following message: **"{climberName}** not found for gearing and instructions. The attack of **{peakName}** will be postponed.**"**
* If the peak’s **DifficultyLevel** is "Extreme", but the climber is **NaturalClimber**, the **climber will NOT be allowed** to **leave the** **baseCamp** and start an attack. Return the following message: **"{climberName}** does not cover the requirements for climbing **{peakName}**.**"**
* The climber leaves the **baseCamp** (*LeaveCamp() method*) and attacks the peak:
  + The climber will try to climb the peak, **executing the** **Climb()** **method**.
    - If the climber’s **Stamina** **drops to zero,** the attack will **NOT be successful**. The climber will **NOT return** to the **baseCamp**. Return the following message: **"{climberName}** did not return to BaseCamp.**"**
    - If the climber’s **Stamina** **doesn’t drop to zero,** the attack will **be successful**. The climber will **return** to the **basecamp** (*ArriveAtCamp() method*). Return the following message: **"{climberName}** successfully conquered **{peakName}** and returned to BaseCamp.**"**

#### CampRecovery Command

##### Parameters

* **climberName - string**
* **daysToRecover – int**

##### Functionality

The method allows a specified climber to recover his stamina over a given number of days. It focuses on climbers who are currently at the BaseCamp:

* Check if a climber with the given **climberName** is **presented at the base camp**. **If NOT**, return the following message: **"{climberName}** not found at the BaseCamp.**"**
* If the climber’s **Stamina** is **already at its maximum** (10 units), he does not need further recovery. Return the following message: **"{climberName}** has no need of recovery.**"**
* If the climber requires recovery (**Stamina** less than 10), **execute** the climber's **Rest()** method based on the **daysToRecover** parameter. Returnthe following message: **"{climberName}** has been recovering for **{count}** days and is ready to attack the mountain.**"**

#### BaseCampReport Command

##### Functionality

This method generates a report of the **climbers currently at the BaseCamp**, providing key details about each climber**.**

* For each climber in the **Residents** collection retrieve detailed information about the climber from the **ClimberRepository** (Name, Stamina, Count of ConqueredPeaks):

**"**BaseCamp residents:

Name: **{climber1Name}**, Stamina: **{Stamina}**, Count of Conquered Peaks: **{count}**

Name: **{climber2Name}**, Stamina: **{Stamina}**, Count of Conquered Peaks: **{count}**

Name: **{climber3Name}**, Stamina: **{Stamina}**, Count of Conquered Peaks: **{count}**

**…**

Name: **{climbernName}**, Stamina: **{Stamina}**, Count of Conquered Peaks: **{count}"**

* If there are **no climbers at the BaseCamp**, return: **"**BaseCamp is currently empty.**"**

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

#### OverallStatistics Command

##### Functionality

Returns information about each climber from the ClimberRepository. **Arrange** the climbers **by the count of conquered peaks in descending order**, then **by name - alphabetically**. **Arrange their peaks by Elevation** in **descending** order. To receive the correct output, use the ToString() method **of each climber and each climber’s conquered peak:**

"\*\*\*Highway-To-Peak\*\*\*

**{climber1}**

**{peak1}**

**{peak2}**

**…**

**{peakn}**

**{climber2}**

**{peak1}**

**{peak2}**

**…**

**{peakn}**

**...**

**{climbern}**

**{peak1}**

**{peak2}**

**…**

**{peakn}**

**"**

**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:

* **AddPeak** **{name} {elevation} {difficultyLevel}**
* **NewClimberAtCamp** **{name} {isOxygenUsed}**
* **AttackPeak** **{climberName} {peakName}**
* **CampRecovery {climberName} {daysToRecover}**
* **BaseCampReport**
* **OverallStatistics**
* **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** |
| **AddPeak Everest 8848 Extreme**  **AddPeak Kangchenjunga 8586 Hard**  **AddPeak Lhotse 8516 Hard**  **AddPeak Makalu 8463 Hard**  **AddPeak ChoOyu 8201 Moderate**  **AddPeak Dhaulagiri 8167 Moderate**  **AddPeak Manaslu 8156 Moderate**  **AddPeak NangaParbat 8126 Extreme**  **AddPeak Annapurna 8091 Extreme**  **AddPeak ShishaPangma 8012 Extreme**  **NewClimberAtCamp LiamSmith false**  **NewClimberAtCamp EthanLee true**  **NewClimberAtCamp MasonKim false**  **NewClimberAtCamp JacobMartinez true**  **NewClimberAtCamp JacobMartinez true**  **AttackPeak LiamSmith Lhotse**  **AttackPeak EthanLee Makalu**  **AttackPeak MasonKim UnlistedPeak**  **AttackPeak JacobMartinez Dhaulagiri**  **AttackPeak LiamSmith NangaParbat**  **AttackPeak EthanLee Everest**  **AttackPeak MasonKim ChoOyu**  **AttackPeak MasonKim Manaslu**  **AttackPeak JacobMartinez ShishaPangma**  **CampRecovery LiamSmith 6**  **CampRecovery LiamSmith 6**  **CampRecovery EthanLee 2**  **CampRecovery MasonKim 3**  **CampRecovery JacobMartinez 7**  **AttackPeak LiamSmith ChoOyu**  **AttackPeak LiamSmith Dhaulagiri**  **AttackPeak LiamSmith Manaslu**  **AttackPeak LiamSmith ChoOyu**  **AttackPeak EthanLee Annapurna**  **AttackPeak MasonKim Kangchenjunga**  **AttackPeak JacobMartinez NangaParbat**  **AttackPeak JacobMartinez Dhaulagiri**  **BaseCampReport**  **OverallStatistics**  **Exit** |
| **Output** |
| **Everest is allowed for international climbing. See details in PeakRepository.**  **Kangchenjunga is allowed for international climbing. See details in PeakRepository.**  **Lhotse is allowed for international climbing. See details in PeakRepository.**  **Makalu is allowed for international climbing. See details in PeakRepository.**  **ChoOyu is allowed for international climbing. See details in PeakRepository.**  **Dhaulagiri is allowed for international climbing. See details in PeakRepository.**  **Manaslu is allowed for international climbing. See details in PeakRepository.**  **NangaParbat is allowed for international climbing. See details in PeakRepository.**  **Annapurna is allowed for international climbing. See details in PeakRepository.**  **ShishaPangma is allowed for international climbing. See details in PeakRepository.**  **LiamSmith has arrived at the BaseCamp and will wait for the best conditions.**  **EthanLee has arrived at the BaseCamp and will wait for the best conditions.**  **MasonKim has arrived at the BaseCamp and will wait for the best conditions.**  **JacobMartinez has arrived at the BaseCamp and will wait for the best conditions.**  **JacobMartinez is a participant in ClimberRepository and cannot be duplicated.**  **LiamSmith successfully conquered Lhotse and returned to BaseCamp.**  **EthanLee successfully conquered Makalu and returned to BaseCamp.**  **UnlistedPeak is not allowed for international climbing.**  **JacobMartinez successfully conquered Dhaulagiri and returned to BaseCamp.**  **LiamSmith does not cover the requirements for climbing NangaParbat.**  **EthanLee did not return to BaseCamp.**  **MasonKim successfully conquered ChoOyu and returned to BaseCamp.**  **MasonKim successfully conquered Manaslu and returned to BaseCamp.**  **JacobMartinez successfully conquered ShishaPangma and returned to BaseCamp.**  **LiamSmith has been recovering for 6 days and is ready to attack the mountain.**  **LiamSmith has no need of recovery.**  **EthanLee not found at the BaseCamp.**  **MasonKim has been recovering for 3 days and is ready to attack the mountain.**  **JacobMartinez has been recovering for 7 days and is ready to attack the mountain.**  **LiamSmith successfully conquered ChoOyu and returned to BaseCamp.**  **LiamSmith successfully conquered Dhaulagiri and returned to BaseCamp.**  **LiamSmith successfully conquered Manaslu and returned to BaseCamp.**  **LiamSmith successfully conquered ChoOyu and returned to BaseCamp.**  **EthanLee not found for gearing and instructions. The attack of Annapurna will be postponed.**  **MasonKim successfully conquered Kangchenjunga and returned to BaseCamp.**  **JacobMartinez successfully conquered NangaParbat and returned to BaseCamp.**  **JacobMartinez successfully conquered Dhaulagiri and returned to BaseCamp.**  **BaseCamp residents:**  **Name: JacobMartinez, Stamina: 1, Count of Conquered Peaks: 3**  **Name: LiamSmith, Stamina: 2, Count of Conquered Peaks: 4**  **Name: MasonKim, Stamina: 4, Count of Conquered Peaks: 3**  **\*\*\*Highway-To-Peak\*\*\***  **NaturalClimber - Name: LiamSmith, Stamina: 2**  **Peaks conquered: 4**  **Peak: Lhotse -> Elevation: 8516, Difficulty: Hard**  **Peak: ChoOyu -> Elevation: 8201, Difficulty: Moderate**  **Peak: Dhaulagiri -> Elevation: 8167, Difficulty: Moderate**  **Peak: Manaslu -> Elevation: 8156, Difficulty: Moderate**  **OxygenClimber - Name: JacobMartinez, Stamina: 1**  **Peaks conquered: 3**  **Peak: Dhaulagiri -> Elevation: 8167, Difficulty: Moderate**  **Peak: NangaParbat -> Elevation: 8126, Difficulty: Extreme**  **Peak: ShishaPangma -> Elevation: 8012, Difficulty: Extreme**  **NaturalClimber - Name: MasonKim, Stamina: 4**  **Peaks conquered: 3**  **Peak: Kangchenjunga -> Elevation: 8586, Difficulty: Hard**  **Peak: ChoOyu -> Elevation: 8201, Difficulty: Moderate**  **Peak: Manaslu -> Elevation: 8156, Difficulty: Moderate**  **OxygenClimber - Name: EthanLee, Stamina: 0**  **Peaks conquered: 2**  **Peak: Everest -> Elevation: 8848, Difficulty: Extreme**  **Peak: Makalu -> Elevation: 8463, Difficulty: Hard** |

## Task 3: Unit Tests (100 points)

You will receive a skeleton with one class inside it. **TelevisionDevice** 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 **Television.Tests** inside**)** from the **skeleton.**