# Java OOP Retake Exam – 15 August 2023



1. **Overview**

*Handball is a team sport played with a ball, where two teams of seven players (including a goalkeeper) compete to score in the opponent's goal. The gameplay combines rapid attacks and strategic defense. We are developing a support application to enhance understanding and practice of the game, assisting teams with information and training for effective play.*

## Setup

* Upload **only the zip handball** package file in every task **except** **Unit Tests.**
* The **Controller** must be **implemented** for your task to **run**.
* **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.

## Task 1: Structure (50 points)

You are given **3** interfaces and must implement their functionalities in the **correct classes** The application has 3 types of entities: **Gameplay**, **Team**, **Equipment**. There should also be **Repository**.

### Equipment

BaseEquipment is a **base class** of any **type of equipment,** and it **should not be able to be instantiated**.

#### Data

* **protection** - **int**
* **price** - **double**
  + The price of the equipment.

#### Constructor

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

(int protection, double price)

#### Child Classes

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

##### Kneepad

It has **120 protection,** and its **price** is **15**.

Constructorsshould take no values upon initialization.

##### ElbowPad

It has **90 protection,** and its **price** is **25**.

Constructorsshould take no values upon initialization.

### Team

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

#### Data

* **name** - **String**
  + If the name **is null or whitespace,** throw a **NullPointerException** with a message:

"**Team name cannot be null or empty.**"

* + All names are unique.
* **country** - **String**
  + If the country **is null or whitespace,** throw a **NullPointerException** with а message:

"**Team country cannot be null or empty.**"

* **advantage** - **int**
  + The advantage of the **Team**.
  + If the advantage is below or equal to **0,** throw an **IllegalArgumentException** with а message:

"**Team avantage cannot be below or equal to 0.**"

#### Behavior

##### void play()

The **play()** method increases the **Team’s** advantage. Keep in mind that different types of **Team** can implement the method differently.

#### Constructor

A team should take the following values upon initialization:

(String name, String country, int advantage)

#### Child Classes

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

**Bulgaria**

**I can only play Outdoor!**

The constructorshould take the following values upon initialization:

**(String name, String country, int advantage)**

#### Behavior

##### void play()

* The method **increases** the team’s advantage by **115**.

##### Germany

**I can only play Indoor!**

The constructorshould take the following values upon initialization:

**(String name, String country, int advantage)**

#### Behavior

##### void play()

* The method **increases** the team’s advantage by **145**.

### Gameplay

BaseGameplay is a **base class** of any **type of Gameplay,** and it **should not be able to be instantiated**.

#### Data

* **name** - **String**
  + If the name **is null or whitespace,** throw a **NullPointerException** with a message:

"**Gameplay name cannot be null or empty.**"

* + All names are unique.
* **capacity** - **int**
  + The **numbers** of **Team** а **Gameplay** **can have.**
* **equipments** - **Collection<Equipment>**
* **teams** - **Collection<Team>**

#### Behavior

##### Constructor

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

**(String name, int capacity)**

##### int allProtection()

**Returns the sum** of **each equipment’s protection** in the **Gameplay.**

##### void addTeam(Team team)

**Adds** a Team on the **Team**. **No** **need** to check for empty space.

##### void removeTeam(Team team)

Removes a **Team** from the **Gameplay**.

##### void addEquipment(Equipment equipment)

Adds an **Equipment** in the **Gameplay**.

##### void teamsInGameplay()

The **teamsInGameplay()** method **calls all teams to play** in the gameplay.

##### String toString()

**Returns** a **String** with **information** about the **Gameplay** in the format below. If the **Gameplay doesn't have a team**, print **"none"** instead.

**"{gameplayName} {gameplayType}  
Team: {teamName1} {teamName2} (…) / Team: none  
Equipment: {equipmentsCount}, Protection: {allProtection}"**

#### Child Classes

There are 2 concrete types of **Gameplay**:

**Outdoor**

Has **150 capacity.**

The constructorshould take the following values upon initialization:

**(String name)**

##### Indoor

Has **250 capacity.**

The constructorshould take the following values upon initialization:

**(String name)**

### Repository

The **EquipmentRepository** is a **repository** for the **equipment** that is in the **gameplay**.

#### Data

* equipments - **Collection<Equipment>**

#### Behavior

**void аdd(Equipment equipment)**

* **Added** **equipment** to the **collection**.

**boolean remove(Equipment equipment)**

* **Removes** **equipment** from the **collection**. **Returns true** if the deletion was **successful**, **otherwise** - **false**.

**Equipment findByType(String type)**

* **Returns** the **first** **equipment** of the **given type**, if there is. **Otherwise**, 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 that you must implement in the correct classes.

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

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

### Data

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

* **equipment** - **Repository**
* **gameplays** - a **collection of Gameplay**

### Commands

There are several **commands** which control the **business** **logic** of the **application**. They are **stated** **below**. The **Gameplay** **name** passed to the methods will **always** be **valid**!

#### AddGameplay Command

##### Parameters

* gameplayType - String
* gameplayName - String

##### Functionality

**Adds** a Gameplay. **Valid** types are: "**Outdoor**" and "**Indoor**".

If the **Gameplay** **type** is **invalid**, you have to **throw a NullPointerException** with **the following message:**

* **"Invalid gameplay type."**

If the **Gameplay** is **added successfully**, the method should **return** the following **String**:

* **"Successfully added {gameplayType}."**

#### AddEquipment Command

##### Parameters

* **type** - **String**

##### Functionality

**Creates** **equipment** of the **given type** and **adds** it to the **Repository**. **Valid** types are "**Kneepad**" and "**ElbowPad**".

If the equipment **type** is **invalid**, throw an **IllegalArgumentException** with a message:

* **"Invalid equipment type."**

The **method** should **return** the following **String** if the **operation** is **successful**:

* **"Successfully added {equipmentType}."**

#### EquipmentRequirement Command

##### Parameters

* gameplayName - String
* equipmentType - String

##### Functionality

**Adds** the desired Equipment to the Gameplay with the **given name**. You have to remove the Equipment from Repository if the insert is **successful**.

If there is **no such equipment**, you have to **throw an IllegalArgumentException** with **the following message**:

* **"There isn't an equipment of type {equpmentType}."**

If **no exceptions** are **thrown return** the **String**:

* **"Successfully added {equpmentType} to {gameplayName}."**

#### AddTeam Command

##### Parameters

* gameplayName - String
* teamType - String
* teamName - String
* **country** - String
* advantage - int

##### Functionality

Check if the team typeis **valid. Valid** **Team** types are: "**Bulgaria**", "**Germany**". **Adds** the desired Team to the Gameplay with the **given name**.

If the **Team** **type** is **invalid**, you have to **throw an IllegalArgumentException** with **the following message**:

* **"Invalid team type."** - if the **Team** **type** is **invalid.**

If **no errors** are **thrown**, **return** one of the following strings:

* **"The coverage of the terrain is not suitable."** - if the **Team** **cannot play** in the **Gameplay**
* **"Successfully added {teamType} to {gameplayName}."** - if the **Team** is **added successfully** in the **Gameplay**

#### PlayInGameplay Command

##### Parameters

* **gameplayName** - **String**

##### Functionality

All **teams** with given name must be on the **Gameplay**.

**Returns** a **string** with information about **how many teams** were **successfully played** in the following **format**:

* **"Teams that have played: {playedCount}"**

#### PercentAdvantage Command

##### Parameters

* **gameplayName** - **String**

##### Functionality

Calculates the value of the **Gameplay** with the given name. It is calculated by the sum of all **Team’s** advantages in the **Gameplay**.

**Return** a **string** in the following **format**:

* **"The advantage of Gameplay {gameplayName} is {value}."**

#### GetStatistics Command

##### Functionality

Returns information about each gameplay. You can use the overridden **.toString() Gameplay** method.

**"{gameplayName} {gameplayType}  
Team: {teamName1} {teamName2} (…) / Team: none  
Equipment: {equipmentsCount}, Protection: {allProtection}**

**{gameplayName} {gameplayType}  
Team: {teamName1} {teamName2} (…) / Team: none  
Equipment: {equipmentsCount}, Protection: {allProtection}**

**(…)"**

**Note: Use \n or System.lineSeparator() 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:

* **AddGameplay** **{gameplayType} {gameplayName}**
* **AddEquipment** **{equipmentType}**
* **EquipmentRequirement** **{gameplayName} {equipmentType}**
* **AddTeam {gameplayName} {teamType} {teamName} {country} {advantage}**
* **PlayInGameplay {gameplayName}**
* **PercentAdvantage {gameplayName}**
* **GetStatistics**
* **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** |
| **AddGameplay Outdoor Handball2023**  **AddEquipment Kneepad**  **AddEquipment ElbowPad**  **EquipmentRequirement Handball2023 Kneepad**  **EquipmentRequirement Handball2023 ElbowPad**  **AddTeam Handball2023 Bulgaria HandballBulg Bulgaria 15**  **AddTeam Handball2023 Germany HandballGerm Germany 10**  **PlayInGameplay Handball2023**  **PercentAdvantage Handball2023**  **PlayInGameplay Handball2023**  **GetStatistics**  **Exit** |
| **Output** |
| **Successfully added Outdoor.**  **Successfully added Kneepad.**  **Successfully added ElbowPad.**  **Successfully added Kneepad to Handball2023.**  **Successfully added ElbowPad to Handball2023.**  **Successfully added Bulgaria to Handball2023.**  **The coverage of the terrain is not suitable.**  **Teams that have played: 1**  **The advantage of gameplay Handball2023 is 130.**  **Teams that have played: 1**  **Handball2023 Outdoor**  **Team: HandballBulg**  **Equipment: 2, Protection: 210** |

|  |
| --- |
| **Input** |
| **AddGameplay Outdoor** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France)  **AddTeam** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Bulgaria HandballSofia Bulgaria 10**  **AddTeam** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Bulgaria HandballVarna Bulgaria 12**  **AddGameplay Indoor Handball2021**  **AddTeam Handball2021 Germany HandballBerlin Germany 30**  **AddTeam Handball2021 Germany HandballMunich Germany 54**  **AddEquipment Kneepad**  **AddEquipment ElbowWPadD**  **EquipmentRequirement** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **ElboWwWPaDD**  **EquipmentRequirement Handball2021 Kneepad**  **AddEquipment ElbowPad**  **EquipmentRequirement** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **ElbowPad**  **PlayInGameplay** [**Handball2022**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France)  **PlayInGameplay Handball2021**  **AddTeam Handball2021 Germany Handball2021 Germany 76**  **GetStatistics**  **Exit** |
| **Output** |
| **Successfully added Outdoor.**  **Successfully added Bulgaria to Handball2022.**  **Successfully added Bulgaria to Handball2022.**  **Successfully added Indoor.**  **Successfully added Germany to Handball2021.**  **Successfully added Germany to Handball2021.**  **Successfully added Kneepad.**  **Invalid equipment type.**  **There isn't an equipment of type ElboWwWPaDD.**  **Successfully added Kneepad to Handball2021.**  **Successfully added ElbowPad.**  **Successfully added ElbowPad to Handball2022.**  **Teams that have played: 2**  **Teams that have played: 2**  **Successfully added Germany to Handball2021.**  **Handball2022 Outdoor**  **Team: HandballSofia HandballVarna**  **Equipment: 1, Protection: 90**  **Handball2021 Indoor**  **Team: HandballBerlin HandballMunich Handball2021**  **Equipment: 1, Protection: 120** |

## Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside – **Main**, **HandballPlayer,** and **Team**. **Team** 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. In Judge, you upload **.zip** to **handball (**with **TeamTests** inside**)** from the **skeleton**.