# Java OOP - Exam Preparation – 01



Link: <https://judge.softuni.org/Contests/4039/Java-OOP-Exam-Preparation-1-MLC>

1. **Overview**

Football has always been an interesting destination for young and old. You have to create a **football** project which keeps track of the players on the football stadium fields. The **Fields** have **Players** with different environmental requirements. Your task is to add, drag and take care of the players.

## Setup

* Upload **only the football** package 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.

## 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: **Field**, **Player**, **Supplement**. There should also be **SupplementRepository**.

### Supplement

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

#### Data

* **energy** - **int**
* **price** - **double**
  + The price of the supplement.

#### Constructor

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

(int energy, double price)

#### Child Classes

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

##### Powdered

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

Constructorsshould take no values upon initialization.

##### Liquid

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

Constructorsshould take no values upon initialization.

### Player

BasePlayer is a **base class** of any **type of player,** 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:

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

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

"**Player nationality cannot be null or empty.**"

* **kg** - **double**
  + The kilograms of the **Player**.
* **strength** - **int**
  + The strength of the **Player**.
  + If the strength is below or equal to **0,** throw an **IllegalArgumentException** with а message:

"**Players strength cannot be below or equal to 0.**"

#### Behavior

##### void stimulation()

The **stimulation()** method increases the **Player's** strength. Keep in mind that different type of **Player** can implement the method differently.

#### Constructor

A player should take the following values upon initialization:

(String name, String nationality, double kg, int strength)

#### Child Classes

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

**Women**

Has **initial kilograms of 60.00.**

**I can only play on ArtificialTurf!**

The constructorshould take the following values upon initialization:

**(String name, String nationality, int strength)**

#### Behavior

**void stimulation()**

* The method **increases** the player's strength by **115**.

##### Men

Has **initial kilograms of 85.50.**

**I can only play on NaturalGrass!**

The constructorshould take the following values upon initialization:

**(String name, String nationality, int strength)**

#### Behavior

**void stimulation()**

* The method **increases** the player's strength by **145**.

### Field

BaseField is a **base class** of any **type of Field,** 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:

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

* + All names are unique.
* **capacity** - **int**
  + The **numbers** of **Player** а **Field** **can have.**
* **supplements** - **Collection<Supplement>**
* **players** - **Collection<Player>**

#### Behavior

##### Constructor

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

**(String name, int capacity)**

##### int sumEnergy()

**Returns the sum** of **each supplement's energy** in the **Field.**

##### void addPlayer (Player Player)

**Adds** a Player on the **Field** if there is a **capacity** for it.

If there is **not enough capacity** to **add** the **Player** in the **Field** **throw an IllegalStateException** with **the following message:**

* **"Not enough capacity."**

##### void removePlayer (Player Player)

Removes a **Player** from the **Field**.

##### void addSupplement(Supplement supplement)

Adds a **Supplement** in the **Field**.

##### void drag()

The **drag()** method **stimulated all players** on the field.

##### String getInfo()

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

**"{fieldName} ({fieldType}):  
Player: {playerName1} {playerName2} {playerName3} (…) / Player: none  
Supplement: {supplementsCount}  
Energy: {sumEnergy}"**

#### Child Classes

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

**ArtificialTurf**

Has **150 capacity**

The constructorshould take the following values upon initialization:

**String name**

##### NaturalGrass

Has **250 capacity**

The constructorshould take the following values upon initialization:

**String name**

### SupplementRepository

The **SupplementRepositoryImpl** is a **repository** for the **supplements** that are in the **field**.

#### Data

* supplements - **Collection<Supplement>**

#### Behavior

**void add(Supplement supplement)**

* **Added** a **supplement** to the **collection**.

**boolean remove(Supplement supplement)**

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

**Supplement findByType(String type)**

* **Returns** the **first** **supplement** 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 instantied. 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:

* **supplement** - **SupplementRepository**
* **fields** - a **collection of Field**

### Commands

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

#### AddField Command

##### Parameters

* fieldType - String
* fieldName - String

##### Functionality

**Adds** a Field. **Valid** types are: "**ArtificialTurf**" and "**NaturalGrass**".

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

* **"Invalid field type."**

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

* **"Successfully added {fieldType}."**

#### DeliverySupplement Command

##### Parameters

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

##### Functionality

**Creates** a **supplement** of the **given type** and **adds** it to the **SupplementRepository**. **Valid** types are "**Powdered**" and "**Liquid**". If the supplement **type** is **invalid**, throw an **IllegalArgumentException** with a message:

* **"Invalid supplement type."**

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

* **"Successfully added {supplementType}."**

#### SupplementForField Command

##### Parameters

* fieldName - String
* supplementType - String

##### Functionality

**Adds** the desired Supplement to the Field with the **given name**. You have to remove the Supplement from the SupplementRepository if the insert is **successful**.

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

* **"There isn't a Supplement of type {**supplement**Type}."**

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

* **"Successfully added {**supplement**Type} to {fieldName}."**

#### AddPlayer Command

##### Parameters

* fieldName - String
* playerType - String
* playerName - String
* **nationality** - String
* strength - int

##### Functionality

Check if the player typeis **valid. Valid** **Player** types are: "**Men**", "**Women**". **Adds** the desired Player to the Field with the **given name**.

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

* **"Invalid player type."** - if the **Player** **type** is **invalid.**

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

* **"The pavement of the terrain is not suitable."** - if the **Player** **cannot play** in the **Field**
* **"Successfully added {playerType} to {fieldName}."** - if the **Player** is **added successfully** in the **Field**

#### DragPlayer Command

##### Parameters

* **fieldName** - **String**

##### Functionality

Drag all **Player** in the **Field** with the given name.

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

* **"Player drag: {dragCount}"**

#### CalculateStrength Command

##### Parameters

* **areaName** - **String**

##### Functionality

Calculates the value of the **Field** with the given name. It is calculated by the sum of all **Players**'s strengths in the **Field**.

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

* **"The strength of Field {fieldName} is {value}."**

#### GetStatistics Command

##### Functionality

Returns information about each field. You can use the overridden **.getInfo Field** method.

**"{fieldName} ({fieldType}):  
Player: {playerName1} {playerName2} {playerName3} (…) / Player: none  
Supplement: {supplementsCount}  
Energy: {sumEnergy}**

**{fieldName} ({fieldType}):  
Player: {playerName1} {playerName2} {playerName3} (…) / Player: none  
Supplement: {supplementsCount}  
Energy: {sumEnergy}**

**(…)"**

**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:

* **AddField** **{fieldType} {fieldName}**
* **DeliverySupplement** **{supplementType}**
* **SupplementForField** **{fieldName} {supplementType}**
* **AddPlayer {fieldName} {playerType} {playerName} {playerNationality} {strength}**
* **DragPlayer {fieldName}**
* **CalculateStrength {fieldName}**
* **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** |
| **AddField ArtificialTurf SantiagoBernabeu**  **DeliverySupplement Powdered**  **DeliverySupplement Powdered**  **DeliverySupplement Liquid**  **SupplementForField SantiagoBernabeu Powdered**  **SupplementForField SantiagoBernabeu Powdered**  **SupplementForField SantiagoBernabeu Liquid**  **AddPlayer SantiagoBernabeu Women Ivana Serbian 20**  **AddPlayer SantiagoBernabeu Women Petra Bulgarian 25**  **DragPlayer SantiagoBernabeu**  **CalculateStrength SantiagoBernabeu**  **DragPlayer SantiagoBernabeu**  **GetStatistics**  **Exit** |
| **Output** |
| **Successfully added ArtificialTurf.**  **Successfully added Powdered.**  **Successfully added Powdered.**  **Successfully added Liquid.**  **Successfully added Powdered to SantiagoBernabeu.**  **Successfully added Powdered to SantiagoBernabeu.**  **Successfully added Liquid to SantiagoBernabeu.**  **Successfully added Women to SantiagoBernabeu.**  **Successfully added Women to SantiagoBernabeu.**  **Player drag: 2**  **The strength of Field SantiagoBernabeu is 275.**  **Player drag: 2**  **SantiagoBernabeu (ArtificialTurf):**  **Player: Ivana Petra**  **Supplement: 3**  **Energy: 330** |

|  |
| --- |
| **Input** |
| **AddField ArtificialTurf** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France)  **AddPlayer** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Women Sara Argentinean 49**  **AddPlayer** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Women Mila Frenchwoman 51**  **AddField NaturalGrass SanPaolo**  **AddPlayer SanPaolo Men Messi Argentinean 34**  **AddPlayer SanPaolo Men Ronaldo Portuguese 38**  **DeliverySupplement Powdered**  **DeliverySupplement Protein**  **SupplementForField** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Vodka**  **SupplementForField SanPaolo Powdered**  **DeliverySupplement Liquid**  **SupplementForField** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France) **Liquid**  **DragPlayer** [**StadeDeFrance**](https://www.online-translator.com/translation/english-russian/Stade%20de%20France)  **DragPlayer SanPaolo**  **AddPlayer SanPaolo Men Ivan Macedonian 40**  **AddPlayer SanPaolo Indeterminate Shyman Chinese 16**  **GetStatistics**  **Exit** |
| **Output** |
| **Successfully added ArtificialTurf.**  **Successfully added Women to StadeDeFrance.**  **Successfully added Women to StadeDeFrance.**  **Successfully added NaturalGrass.**  **Successfully added Men to SanPaolo.**  **Successfully added Men to SanPaolo.**  **Successfully added Powdered.**  **Invalid supplement type.**  **There isn't a supplement of type Vodka.**  **Successfully added Powdered to SanPaolo.**  **Successfully added Liquid.**  **Successfully added Liquid to StadeDeFrance.**  **Player drag: 2**  **Player drag: 2**  **Successfully added Men to SanPaolo.**  **Invalid player type.**  **StadeDeFrance (ArtificialTurf):**  **Player: Sara Mila**  **Supplement: 1**  **Energy: 90**  **SanPaolo (NaturalGrass):**  **Player: Messi Ronaldo Ivan**  **Supplement: 1**  **Energy: 120** |

## Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside – **Main**, **Footballer,** and **FootballTeam**. **FootballTeam** 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 **football (**with **FootballTeamTests** inside**)** from the **skeleton**.