# Java OOP Exam – 9 December 2023



# Overview

There are many ways to fish, but spearfishing is undoubtedly one of the most exciting. You are part of a group of enthusiastic harpooners and enter the competition for the most interesting catch. Give it a shot, do your best.

## Setup

* Upload **only the** **harpoonDiver** package in every task **except Unit Tests**.
* **Do not modify the provided interfaces or their packages.**
* Use **strong cohesion** and **loose coupling.**
* **Use inheritance and the provided interfaces whenever 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 **4** interfaces and must implement their functionalities in the **correct classes**.

There are **4** types of entities in the application: **Diver, Diving, DivingSite, SeaCatch**. There are also **2** repositories: a **DiverRepository** and a **DivingSiteRepository**.

### Diver

Base**Diver** is a **base class** or any **type of diver** and **should not be instantiated**.

#### Data

* **name** – **String**
  + If the value of the name is either **null** or **empty** (containing only whitespaces), throw a **NullPointerException** with the following message: **"Diver name cannot be null or empty."**
  + The values of the names are **unique.**
* **oxygen** – **double**
  + The oxygen of а diver.
  + If the oxygen is a **negative** number, throw an **IllegalArgumentException** with the following message: **"Cannot create Diver with negative oxygen."**
* seaCatch – Sea**Catch**
  + A SeaCatch field type.

#### Behavior

##### void shoot()

The **shoot()** method decreases the diver's oxygen. Keep in mind that some Diver kinds can implement the method differently.

* The method **decreases** the diver's oxygen by **30 units**.
* The oxygen value **should** **not** drop **below** **zero**.
* Set the value to be zero if the oxygen value drops below zero.

##### boolean canDive()

The **canDive()** method returns a **boolean**. Tell us if the oxygen is more than zero.

#### Constructor

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

String name, double oxygen

#### Child Classes

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

**OpenWaterDiver**

Has **30 initial units of oxygen**.

The constructor should take the following values upon initialization:

Stri**ng** **name**

**DeepWaterDiver**

Has **90 initial units of oxygen**.

The constructor should take the following values upon initialization:

String name

**WreckDiver**

Has an **initial 150 units of oxygen.**

The constructor should take the following values upon initialization:

String name

### SeaCatch

The BaseSeaCatch class holds a **collection** of **sea creatures**. It should be **instantiated**.

#### Data

* **seaCreatures** – a collection of **Strings**

#### Constructor

The constructor should not take any values upon initialization.

### DivingSite

The **DivingSiteImpl** class holds information about the **sea creatures** that can be caught. It should be instantiated.

#### Data

* **name** – **String**
  + If the value of the **name** is either **null** or **empty** (containing only whitespaces), throw a **NullPointerException** with the following message: **"Invalid name!"**
* **seaCreatures** – a collection of Strings

#### Constructor

The constructor should take the following values upon initialization:

String name

### Diving

The **DivingImpl** class holds the main action, which is the **searching** method.

#### Behavior

##### void searching(DivingSite divingSite, Collection<Diver> divers)

Here is how the **searching** method works:

* Divers **cannot** dive on site if their **oxygen** is **equal to or below** 0.
* They dive into the water and **start searching the site for sea creatures** one by one.
* If they **find** a sea creature, they **shoot** it, and their **oxygen** is **decreased**.
* They add the **creature** to their **catch**. The **sea creature** should then be **removed** from the **diving** **site**.
* Divers **cannot** **continue** shooting if their **oxygen** **drops** to 0.
  + If their oxygen drops to 0, the next diver starts diving**.**

### DiverRepository

The **DiverRepository** class is a **repository** for divers.

#### Data

* divers – **a** **collection of divers**

#### Behavior

##### void add(Diver diver)

* Adds a diver to the collection.
* Every diver is unique in the collection.
  + It is guaranteed that there will not be a diver with the same name.

##### boolean remove(Diver diver)

* Removes a diver from the collection. Returns true if the deletion was successful.

##### Diver byName(String name)

* Returns a diver with that name.
* If the driver is not in the collection, return null.

##### Collection<Diver> getCollection()

* Returns an unmodifiable collection of divers.

### DivingSiteRepository

The **DivingSiteRepository** class is a **repository** for diving **sites**.

#### Data

* sites **– a collection of diving sites**

#### Behavior

##### void add(DivingSite divingSite)

* Adds a site for diving.
* Every site is unique in the collection.
  + It is guaranteed that there will not be a site with the same name.

##### boolean remove(DivingSite divingSite)

* Removes a site from the collection. Returns true if the deletion was successful.

##### DivingSite byName(String name)

* Returns a site with that name.
* If the site is not in the collection, return null.

##### Collection<DivingSite> getCollection()

* Returns an unmodifiable collection of sites.

## 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 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:

### Commands

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

#### AddDiver Command

##### Parameters

* **kind – String**
* **diverName – String**

##### Functionality

Creates a **diver** with the given **name** of the given **kind** and saves it in the repository. If the kind is invalid, throw an **IllegalArgumentException** with the following message:

**"Diver kind doesn't exist."**

Otherwise, the method should **return** the following message:

* **"Added {kind}: {diverName}."**

#### AddDivingSite Command

##### Parameters

* **siteName** - String
* seaCreatures – String... (seaCreatures)

##### Functionality

Create a **diving site** with the provided **sea creatures** and **name** and save it in the repository.

The method should **return** the following message:

* **"Added site: {siteName}."**

#### RemoveDiver Command

##### Parameters

* diverName – String

##### Functionality

Remove the diver from diving by removing them from the repository. If a diver with that name doesn’t exist, **throw IllegalArgumentException** with the following message:

* **"Diver {diverName} doesn't exist."**

##### If a diver is successfully excluded, remove them from the repository and return the following message:

* **"Diver {diverName} has removed!"**

#### StartDiving Command

##### Parameters

* **siteName - String**

##### Functionality

When the start diving command is called, the action happens. After picking only the divers that have oxygen **above 30 units**, the diving at the given site begins.

* If you **don't have any** **suitable** **divers**, throw an **IllegalArgumentException** with the following message: **"You must have at least one diver to start diving."**
* After a diving, you must **return the following message** with the **name of the diving site** and the **count** of the **diver/s** that **was/were removed** from the diving:

**"The dive took place at {siteName}. {removedDiverCount} diver/s was/were removed on this dive."**

#### GetStatistics Command

##### Functionality

Returns the information about the divers in the following format:

* If the divers don't have any catch, print **"None"** in their place.

**"The dive took place at {count} site/s.**

**Diver's statistics:**

**Name: {diverName}**

**Oxygen: {diverOxygen}**

**Diver's catch: {seaCreature1, seaCreature2, seaCreature3, …, seaCreaturen}**

**…**

**Name: {diverName}**

**Oxygen: {diverOxygen}**

**Diver's catch: {seaCreature1, seaCreature2, seaCreature3, …, seaCreaturen}"**

### Input / Output

You are provided with one interface, which will help you with the correct execution process of your program. The interface is called **Engine** and its **implementational** class should read the input. When the program finishes, the class should print the **output** to the **console**.

#### Input

These are the input commands:

* **AddDiver** **{diverKind} {diverName}**
* **AddDivingSite** **{siteName} {**String... (seaCreatures)}
* **RemoveDiver** **{diverName}**
* **StartDiving {siteName}**
* **GetStatistics**
* **Exit**

#### Output

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

#### Examples

|  |
| --- |
| **Input** |
| **AddDiver OpenWaterDiver Peter**  **AddDiver DeepWaterDiver George**  **AddDivingSite Sinemoretz Safrid Octopus Mullet Crab Mackerel Gobius Bonito**  **RemoveDiver Sotir**  **StartDiving Sinemoretz**  **AddDiver WreckDiver Asen**  **AddDiver OpenWaterDiver Diana**  **AddDivingSite Seychelles Tuna Scorpionfish**  **StartDiving Seychelles**  **AddDiver DeepWaterDiver Kiril**  **StartDiving Sinemoretz**  **GetStatistics**  **Exit** |
| **Output** |
| **Added OpenWaterDiver: Peter.**  **Added DeepWaterDiver: George.**  **Added site: Sinemoretz.**  **Diver Sotir doesn't exist.**  **The dive took place at Sinemoretz. 1 diver/s was/were removed on this dive.**  **Added WreckDiver: Asen.**  **Added OpenWaterDiver: Diana.**  **Added site: Seychelles.**  **The dive took place at Seychelles. 0 diver/s was/were removed on this dive.**  **Added DeepWaterDiver: Kiril.**  **The dive took place at Sinemoretz. 1 diver/s was/were removed on this dive.**  **The dive took place at 3 site/s.**  **Diver's statistics:**  **Name: Peter**  **Oxygen: 30**  **Diver's catch: None**  **Name: George**  **Oxygen: 0**  **Diver's catch: Safrid, Octopus, Mullet**  **Name: Asen**  **Oxygen: 0**  **Diver's catch: Tuna, Scorpionfish, Crab, Mackerel, Gobius**  **Name: Diana**  **Oxygen: 30**  **Diver's catch: None**  **Name: Kiril**  **Oxygen: 60**  **Diver's catch: Bonito** |

|  |
| --- |
| **Input** |
| **AddDiver OpenWaterDiver John**  **AddDiver DeepWaterDiver David**  **AddDiver OpenWaterDiver Lilia**  **RemoveDiver Lilia**  **AddDiver DeepWaterDiver Philip**  **AddDivingSite TampaBay MantaRay Lucian**  **AddDivingSite GreatBarrierReef RedSnapper**  **AddDiver WreckDiver Donald**  **StartDiving TampaBay**  **GetStatistics**  **AddDiver WreckDiver Maria**  **AddDiver WallDiver Daniel**  **StartDiving GreatBarrierReef**  **GetStatistics**  **Exit** |
| **Output** |
| **Added OpenWaterDiver: John.**  **Added DeepWaterDiver: David.**  **Added OpenWaterDiver: Lilia.**  **Diver Lilia has removed!**  **Added DeepWaterDiver: Philip.**  **Added site: TampaBay.**  **Added site: GreatBarrierReef.**  **Added WreckDiver: Donald.**  **The dive took place at TampaBay. 0 diver/s was/were removed on this dive.**  **The dive took place at 1 site/s.**  **Diver's statistics:**  **Name: John**  **Oxygen: 30**  **Diver's catch: None**  **Name: David**  **Oxygen: 30**  **Diver's catch: MantaRay, Lucian**  **Name: Philip**  **Oxygen: 90**  **Diver's catch: None**  **Name: Donald**  **Oxygen: 150**  **Diver's catch: None**  **Added WreckDiver: Maria.**  **Diver kind doesn't exist.**  **The dive took place at GreatBarrierReef. 0 diver/s was/were removed on this dive.**  **The dive took place at 2 site/s.**  **Diver's statistics:**  **Name: John**  **Oxygen: 30**  **Diver's catch: None**  **Name: David**  **Oxygen: 30**  **Diver's catch: MantaRay, Lucian**  **Name: Philip**  **Oxygen: 60**  **Diver's catch: RedSnapper**  **Name: Donald**  **Oxygen: 150**  **Diver's catch: None**  **Name: Maria**  **Oxygen: 150**  **Diver's catch: None** |

## Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside – **Main**, **DeepWaterDiver,** and **Diving**. **The Diving** 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 **scubaDiving (**with **DivingTests** inside**)** from the **skeleton**.