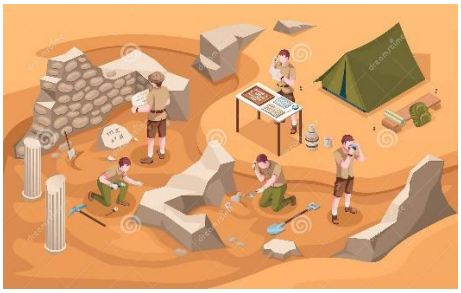
Java OOP Retake Exam – 22 August 2022



Overview

We are all curious about what lies beneath Earth'surface. We visit different museums that provide an insight into the history of humankind and our home – the Earth. You have been chosen to join an expedition aiming to reveal the hidden secrets "beneath our feet". Professionals from various fields are included, each of them collecting exhibits. Your mission is to help the expedition in the various activities and collect exhibits from each specialist. After collecting them, you must return information about the exhibits that have been found.

1. Setup

- Upload **only the goldDigger** 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.

2. Task 1: Structure (50 points)

You are given **4** interfaces and you must implement their functionalities in the **correct classes**.

There are **4** types of entities in the application: **`Discoverer`, `Museum`, `Operation`, `Spot.`** There are also 2 repositories: a **`DiscovererRepository`** and a **`SpotRepository`**.

Discoverer

`BaseDiscoverer` is a **base class** or any **type of discoverer** 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: **`"Discoverer name cannot be null or empty."`**
    - The values of the names are **unique.**
- **`energy`** – **`double`**
    - The energy of a discoverer.

- If the energy is a **negative** number, throw an **`IllegalArgumentException`** with the following message: **`"Cannot create Discoverer with negative energy."`**

- **`museum – Museum`**
  - **A `Museum` field type.**

Behavior

`void dig()`

The **`dig()`** method decreases the discoverer's energy. Keep in mind that some Discoverer types can implement the method differently.

- The method **decreases** the discoverer's energy by **15 units**.
- The energy value **should not** drop **below zero**.
- Set the value to be zero if the energy value goes below zero.

`boolean canDig()`

The **`canDig()`** method returns **boolean**. Tell us if the energy is more than zero.

Constructor

A **`BaseDiscoverer`** should take the following values upon initialization:

`String name, double energy`

Child Classes

There are several concrete types of **`BaseDiscoverer`**:

Archaeologist

Has **60 initial units of energy**.

The constructor should take the following values upon an initialization:

`String name`

Anthropologist

Has **40 initial units of energy**.

The constructor should take the following values upon an initialization:

`String name`

Geologist

Has **initial 100 units of energy.**

The constructor should take the following values upon an initialization:

`String name`

Museum

The **`BaseMuseum`** class holds a **collection** of **exhibits**. It should be **instantiated**.

Data

- **`exhibits`** – a collection of **Strings**

Constructor

The constructor should not take any values upon an initialization.

Spot

The **`SpotImpl`** class holds information about the **exhibits** that can be **found** and **inspected**. 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!"`**
- **`exhibits`** – a collection of Strings

Constructor

The constructor should take the following values upon initialization:

`String name`

Operation

The **`OperationImpl`** class holds the main action, which is the **`startOperation`** method.

Behavior

`void startOperation(Spot spot, Collection<Discoverer> discoverers)`

Here is how the **`startOperation`** method works:

- Discoverers **cannot** go on expeditions if their **energy** is **below** 0.
- They leave the base and **start collecting exhibits** one by one.
- If they **find** an exhibit, their **energy** is **decreased**.
- They add the **exhibit** to their **museum**. The **exhibit** should then be **removed** from the **state**.
- Discoverers **cannot continue** collecting exhibits if their **energy drops** to 0.
  - If their energy drops to 0, the next discoverer starts inspecting**.**

DiscovererRepository

The **DiscovererRepository** class is a **repository** for the **discoverers**.

Data

- **`discoverers`** – **a collection of discoverers**

Behavior

`void add(Discoverer discoverer)`

- Adds a discoverer to the base.
- Every discoverer is unique in the collection.
  - It is guaranteed that there will not be a discoverer with the same name.

`boolean remove(Discoverer discoverer)`

- Removes a discoverer from the collection. Returns true if the deletion was successful.

`Discoverer byName(String name)`

- Returns a discoverer with that name.
- If the discoverer is not in the collection, return null.

`Collection<Discoverer> getCollection()`

- Returns an unmodifiable collection of discoverers.

SpotRepository

The **SpotRepository** class is a **repository** for the overlook **places**.

Data

- **`spots` – a collection of spots**

Behavior

`void add(Spot spot)`

- Adds a spot for inspection.
- Every spot is unique in the collection.
    - It is guaranteed that there will not be a state with the same name.

`boolean remove(Spot spot)`

- Removes a spot from the collection. Returns true if the deletion was successful.

`Spot byName(String name)`

- Returns a spot with that name.
- If the spot is not in the collection, return null.

`Collection<Spot> getCollection()`

- Returns an unmodifiable collection of spots.

3. 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 **`ControllerImpl`** class, 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.

AddDiscoverer Command

Parameters

- **`kind - String`**

- **discovererName – String**

Functionality

Creates a **discoverer** 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:

"Discoverer kind doesn't exists."

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

- **"Added {kind}: {discovererName}."**

AddSpot Command

Parameters

- **spotName - String**
- **exhibits – String... (Varargs)**

Functionality

Creates a **spot** with the provided **exhibits** and **name** and save it in the repository.

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

- **"Added spot: {spotName}."**

ExcludeDiscoverer Command

Parameters

- **discovererName – String**

Functionality

Exclude the discoverer from diggings by removing them from the repository. If a discoverer with that name doesn't exist, **throw IllegalArgumentException** with the following message:

- **"Discoverer {discovererName} doesn't exists."**

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

- **"Discoverer {discovererName} has excluded!"**

InspectSpot Command

Parameters

- **spotName - String**

Functionality

When the inspect command is called, the action happens. You should start inspecting the given spot by sending the discoverers that are most suitable for the mission:

- You call each of the discoverers and pick only the ones that have energy above 45 units.
- If you **don't have any suitable discoverers**, throw an **IllegalArgumentException** with the following message: **"You must have at least one discoverer to inspect the spot."**
- After a mission, you must **return the following message** with the **name of the inspected spot** and the **count** of the **discoverers** that **had excluded** on the mission:
  "The spot {spotName} was inspected. {excludedDiscoverer} discoverers have been excluded on this operation."

GetStatistics Command

Functionality

Returns the information about the discoverers in the following format:

- If the discoverers don't have any museum exhibits, print **"None"** in their place.

"{inspectedSpotCount} spots were inspected.

Information for the discoverers:

Name: {discovererName}

Energy: {discovererName}

Museum exhibits: {museumExhibits$_1$, museumExhibits$_2$, museumExhibits$_3$, …, museumExhibits$_n$}"

…

Name: {discovererName}

Energy: {discovererEnergy}

Museum exhibits: museumExhibits$_1$, museumExhibits$_2$, museumExhibits$_3$, …, museumExhibits$_n$}"

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:

- **AddDiscoverer {discovererType} {discovererName}**
- **AddSpot {spotName} {String... (Varargs)}**
- **ExcludeDiscoverer {discovererName}**
- **InspectSpot {spotName}**
- **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
AddDiscoverer Archaeologist Ivan AddDiscoverer Anthropologist George AddDiscoverer Geologist Peter AddDiscoverer Anthropologist Kiril AddDiscoverer Geologist Mimmy AddDiscoverer Archaeologist Maria

```
AddDiscoverer Archaeologist Valery
AddDiscoverer NaturalExplorer Stam
AddSpot Perperikon
AddSpot BabiniVidiniKuli
ExcludeDiscoverer Maria
InspectSpot Perperikon
GetStatistics
Exit
```

Output

```
Added Archaeologist: Ivan.
Added Anthropologist: George.
Added Geologist: Peter.
Added Anthropologist: Kiril.
Added Geologist: Mimmy.
Added Archaeologist: Maria.
Added Archaeologist: Valery.
Discoverer kind doesn't exists.
Added spot: Perperikon.
Added spot: BabiniVidiniKuli.
Discoverer Maria has excluded!
The spot Perperikon was inspected. 0 discoverers have been excluded on this operation.
1 spots were inspected.
Information for the discoverers:
Name: Ivan
Energy: 60
Museum exhibits: None
Name: George
Energy: 40
Museum exhibits: None
Name: Peter
Energy: 100
Museum exhibits: None
Name: Kiril
Energy: 40
Museum exhibits: None
Name: Mimmy
Energy: 100
Museum exhibits: None
Name: Valery
Energy: 60
Museum exhibits: None
```

Input

```
AddDiscoverer Geologist Jolie
AddSpot MarianaTrench
InspectSpot MarianaTrench
```

```
GetStatistics
AddDiscoverer Archaeologist Lilia
AddDiscoverer Geologist Mia
AddDiscoverer Archaeologist Philip
AddDiscoverer Anthropologist Samantah
AddDiscoverer Anthropologist Petar
AddSpot Sahara
ExcludeDiscoverer David
ExcludeDiscoverer Philip
InspectSpot MarianaTrench
GetStatistics
Exit
```

Output

```
Added Geologist: Jolie.
Added spot: MarianaTrench.
The spot MarianaTrench was inspected. 0 discoverers have been excluded on this operation.
1 spots were inspected.
Information for the discoverers:
Name: Jolie
Energy: 100
Museum exhibits: None
Added Archaeologist: Lilia.
Added Geologist: Mia.
Added Archaeologist: Philip.
Added Anthropologist: Samantah.
Added Anthropologist: Petar.
Added spot: Sahara.
Discoverer David doesn't exists.
Discoverer Philip has excluded!
The spot MarianaTrench was inspected. 0 discoverers have been excluded on this operation.
2 spots were inspected.
Information for the discoverers:
Name: Jolie
Energy: 100
Museum exhibits: None
Name: Lilia
Energy: 60
Museum exhibits: None
Name: Mia
Energy: 100
Museum exhibits: None
Name: Samantah
Energy: 40
Museum exhibits: None
Name: Petar
Energy: 40
```

```
Museum exhibits: None
```

4. Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside – **Main**, **Archaeologist,** and **Excavation**. **Excavation** 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 **archaeologicalExcavations (**with **ExcavationTests** inside**)** from the **skeleton**.