Java OOP Exam – Online Shop

Overview

In this exam, you need to build an online shop project, which has **peripherals**, **components, and computers**. The project will consist of **model classes** and a **controller class**, which manages the **interaction** between the **peripherals**, **components**, and **computers**.

1. Setup

- Upload **only the `onlineShop`** 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 specific class than the interface has defined
- Make sure you have **no public fields** anywhere

Task 1: Structure (50 Points)

For this task's evaluation, logic, in the methods, isn't included.

You are given interfaces and you have to implement their functionality in the **correct classes**.

There are **4** types of entities in the application: **`Product`, `Component`, `Peripheral`, `Computer`**.

Product

The **`BaseProduct`** is a **base class** for **components**, **peripherals** and **computers** and it **should not be able to be instantiated**.

Data

- **id – int**
    - cannot be **less than or equal to 0**. In that case, throw **IllegalArgumentException** with message "**`Id can not be less or equal than 0.`**"
- **`manufacturer`** – **String**
    - cannot be **null or whitespace**. In that case, throw **IllegalArgumentException** with message "**`Manufacturer can not be empty.`**"
- **`model`** – **String**
    - cannot be **null or whitespace**. In that case, throw **IllegalArgumentException** with message "**`Model can not be empty.`**"
- **price – double**
    - cannot be **less than or equal to 0**. In that case, throw **IllegalArgumentException** with message "**`Price can not be less or equal than 0.`**"
- **overallPerformance – double**
    - cannot be **less than or equal to 0**. In that case, throw **IllegalArgumentException** with message "**`Overall Performance can not be less or equal than 0.`**"

Constructor

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

`(int id, String manufacturer, String model, double price, double overallPerformance)`

Override **`toString()`** method in the format:

```
"Overall Performance: {overall performance}. Price: {price} - {product type}:
            {manufacturer} {model} (Id: {id})"
```

Child Classes

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

- **`Component`**
- **`Peripheral`**
- **`Computer`**

Component

The **`BaseComponent`** is a derived class from **`BaseProduct`** and a **base class** for any **components** and it **should not be able to be instantiated**.

Data

- **`generation – int`**

Constructor

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

`(int id, String manufacturer, String model, double price, double overallPerformance, int generation)`

Override **`toString()`** method in the format:

```
"Overall Performance: {overall performance}. Price: {price} - {product type}:
            {manufacturer} {model} (Id: {id}) Generation: {generation}"
```

Child Classes

There are several specific types of **components**, where the **overall performance** has a **different multiplier**:

- **`CentralProcessingUnit`** - multiplier is **1.25**
- **`Motherboard`** – multiplier is **1.25**
- **`PowerSupply`** – multiplier is **1.05**
- **`RandomAccessMemory`** – multiplier is **1.20**
- **`SolidStateDrive`** – multiplier is **1.20**
- **`VideoCard`** – multiplier is **1.15**

Example: If we create the **`CentralProcessingUnit`** with overallPerformance – 50 from the constructor, and multiplier **1.25**, the overallPerformance should be 62.50.

Peripheral

The **`BasePeripheral`** is a derived class from **`BaseProduct`** and a **base class** for any **peripherals** and it **should not be able to be instantiated**.

Data

- **`connectionType`** – **String**

Constructor

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

```
(int id, String manufacturer, String model, double price, double overallPerformance,
String connectionType)
```

Override **`toString()`** method in the format:

```
"Overall Performance: {overall performance}. Price: {price} - {product type}:
            {manufacturer} {model} (Id: {id}) Connection Type: {connection type}"
```

Child Classes

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

- **`Headset`**
- **`Keyboard`**
- **`Monitor`**
- **`Mouse`**

Computer

The **`BaseComputer`** is a derived class from **`BaseProduct`** and a **base class** for any **computers** and it **should not be able to be instantiated**.

Data

- **`components`** – **`List`**
- **`peripherals`** – **`List`**

Constructor

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

```
(int id, String manufacturer, String model, double price, double overallPerformance)
```

Override **`toString()`** method in the format:

```
"Overall Performance: {overall performance}. Price: {price} - {product type}:
            {manufacturer} {model} (Id: {id})"
" Components ({components count}):"
"  {component one}"
"  {component two}"
"  {component n}"
" Peripherals ({peripherals count}); Average Overall Performance ({average overall
performance peripherals}):"
"  {peripheral one}"
"  {peripheral two}"
"  {peripheral n}"
```

Note: Be careful, some of the rows have one or two whitespaces at the beginning of the sentences!

Behavior

`double getOverallPerformance()`

Override the base functionality (if the components collection is empty, it should return only the **computer overall performance**, otherwise return the sum of the **computer overall performance** and the average overall performance from all components)

`double getPrice()`

Override the base functionality (The price is equal to the **total sum** of the **computer price** with the **sum of all component prices** and the **sum of all peripheral prices**)

`void addComponent(Component component)`

If the components collection contains a component with the same component type, throw an **`IllegalArgumentException`** with the message **`"Component {component type} already exists in {computer type} with Id {id}."`**

Otherwise add the component in the components collection.

`Component removeComponent(String componentType)`

If the components collection is empty or does not have a component of that type, throw an **`IllegalArgumentException`** with the message **`"Component {component type} does not exist in {computer type} with Id {id}."`**

Otherwise remove the component of that type and return it.

`void addPeripheral(Peripheral peripheral)`

If the peripherals collection contains a peripheral with the same peripheral type, throw an **`IllegalArgumentException`** with the message **`"Peripheral {peripheral type} already exists in {computer type} with Id {id}."`**

Otherwise add the peripheral in peripherals collection.

`Peripheral removePeripheral(String peripheralType)`

If the peripherals collection is empty or does not have a peripheral of that type, throw an **`IllegalArgumentException`** with the message **`"Peripheral {peripheral type} does not exist in {computer type} with Id {id}."`**

Otherwise remove the peripheral of that type and return it.

Child Classes

There are several specific types of **computers,** where the **overall performance** has a **different value**:

- **`DesktopComputer`** – overall performance is **15**
- **`Laptop`** – overall performance is **10**

Child classes should **not** receive an overall performance as a parameter from the constructor.

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

The first interface is the **`Controller`**. You must create a **`ControllerImpl`** class, which implements the interface and implements all of its methods. The constructor, of the **`ControllerImpl`**, does not take any arguments. The given methods should have the logic, described for each in the **Commands** section.

Commands

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

NOTE: For each command, except for **`"addComputer"`** and **`"buyBest"`**, you must check if a computer, with that id, exists in the computers collection. If it doesn't, throw an **`IllegalArgumentException`** with the message **`"Computer with this id does not exist."`**.

addComputer Command

Parameters

- **`computerType`** – **`String`**
- **`id`** – **`int`**
- **`manufacturer`** – **`String`**
- **`model`** – **`String`**
- **`price`** – **`double`**

Functionality

Creates a computer with the correct type and **adds it to the collection of computers**.

If a computer, with the **same id,** already exists in **the computers collection,** throw an **`IllegalArgumentException`** with the message **`"Computer with this id already exists."`**

If the computer type is invalid, throw an **`IllegalArgumentException`** with the message **`"Computer type is invalid."`**

If it's successful, returns **`"Computer with id {id} added successfully."`**.

AddComponent Command

Parameters

- **`computerId`** – **`int`**
- **`id`** – **`int`**
- **`componentType`** – **`String`**
- **`manufacturer`** – **`String`**
- **`model`** – **`String`**
- **`price`** – **`double`**
- **`overallPerformance`** – **`double`**
- **`generation`** – **`int`**

Functionality

Creates a component with the correct type and **adds it to the computer with that id**, **then adds it to the collection of components in the controller**.

If a component, with the **same id**, already exists in **the components collection**, throws an **`IllegalArgumentException`** with the message **`"Component with this id already exists."`**

If the component type is invalid, throws an **`IllegalArgumentException`** with the message **`"Component type is invalid."`**

If it's successful, returns **`"Component {component type} with id {component id} added successfully in computer with id {computer id}."`**.

RemoveComponent Command

Parameters

- **`componentType`** – **`String`**
- **`computerId`** – **`int`**

Functionality

Removes a component, with the given **type from the computer** with that **id,** then removes component from the **collection of components**.

If it's successful, it returns **`"Successfully removed {component type} with id {component id}."`**.

AddPeripheral Command

Parameters

- **`computerId`** – **`int`**
- **`id`** – **`int`**
- **`peripheralType`** – **`String`**
- **`manufacturer`** – **`String`**
- **`model`** – **`String`**
- **`price`** – **`double`**
- **`overallPerformance`** – **`double`**
- **`connectionType`** –**`String`**

Functionality

Creates a peripheral, with the correct type, and **adds it to the computer with that id**, **then adds it to the collection of peripherals in the controller**.

If a peripheral, with the **same id**, already exists in **the peripherals collection**, it throws an **`IllegalArgumentException`** with the message **`"Peripheral with this id already exists."`**

If the peripheral type is invalid, throws an **`IllegalArgumentException`** with the message **`"Peripheral type is invalid."`**

If it's successful, it returns **`"Peripheral {peripheral type} with id {peripheral id} added successfully in computer with id {computer id}."`**.

RemovePeripheral Command

Parameters

- **`peripheralType`** – **`String`**
- **`computerId`** – **`int`**

Functionality

Removes a peripheral, with the given **type from the computer** with that **id,** then removes the peripheral from the **collection of peripherals**.

If it's successful, it returns **`"Successfully removed {peripheral type} with id { peripheral id}."`**.

BuyComputer Command

Parameters

- **`id`** – **`int`**

Functionality

Removes a computer, with the given **id**, from the **collection of computers**.

If it's successful, it returns **`toString`** **method on the removed computer**.

BuyBest Command

Parameters

- **`budget`** – **`double`**

Functionality

Removes the computer with the highest overall performance and with a price, less or equal to the budget, from the **collection of computers**.

If there are not any computers in the collection or the budget is insufficient for any computer, throws an **`IllegalArgumentException`** with the message **`"Can't buy a computer with a budget of ${budget}."`**

If it's successful, it returns **`toString`** **method on the removed computer**.

GetComputerData Command

Parameters

- **`id`** – **`int`**

Functionality

If it's successful, it returns **`toString`** **method on the computer with the given id**.

Close Command

Functionality

Ends the program.

2. 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.

You are given the **`EngineImpl`** class with written logic in it. In order the code to be **compiled**, some parts are **commented**, **don't forget to comment them out**. The **try-catch block** is also **commented** in order for the program to **throw exceptions and for you to see them**, **comment it out** when you are **ready** with this too.

Input

Below, you can see the **format** in which **each command** will be given in the input:

- **`AddComputer {computer type} {id} {manufacturer} {model} {price}`**

- **AddComponent {computer id} {component id} {component type} {manufacturer} {model} {price} {overall performance} {generation}**
- **RemoveComponent {component type} {computer id}**
- **AddPeripheral {computer id} {peripheral id} { peripheral type} {manufacturer} {model} {price} {overall performance} {connection type}**
- **RemovePeripheral {peripheral type} {computer id}**
- **BuyComputer {id}**
- **BuyBestComputer {budget}**
- **GetComputerData {id}**
- **Close**

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
AddComputer Laptop 4 Asus ROG 700 AddComponent 4 3 CentralProcessingUnit Intel Xeon 1600 82 9 AddComponent 4 6 Motherboard Asus ROG 1250 70 8 AddComponent 4 7 PowerSupply Fortron FSP 700 70 2 AddComponent 4 10 RandomAccessMemory Kingston HyperX 900 80 4 AddComponent 4 13 SolidStateDrive Samsung Evo 800 85 7 AddComponent 4 17 VideoCard Nvidia GeForce 2000 97 9 AddPeripheral 4 3 Headset Razer Thresher 300 70 AUX GetComputerData 4 RemovePeripheral Headset 4 BuyComputer 4 Close
Output
Computer with id 4 added successfully. Component CentralProcessingUnit with id 3 added successfully in computer with id 4. Component Motherboard with id 6 added successfully in computer with id 4. Component PowerSupply with id 7 added successfully in computer with id 4. Component RandomAccessMemory with id 10 added successfully in computer with id 4. Component SolidStateDrive with id 13 added successfully in computer with id 4. Component VideoCard with id 17 added successfully in computer with id 4. Peripheral Headset with id 3 added successfully in computer with id 4. Overall Performance: 105.51. Price: 8250.00 - Laptop: Asus ROG (Id: 4) Components (6): Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3) Generation: 9 Overall Performance: 87.50. Price: 1250.00 - Motherboard: Asus ROG (Id: 6) Generation: 8 Overall Performance: 73.50. Price: 700.00 - PowerSupply: Fortron FSP (Id: 7) Generation: 2 Overall Performance: 96.00. Price: 900.00 - RandomAccessMemory: Kingston HyperX (Id: 10) Generation: 4 Overall Performance: 102.00. Price: 800.00 - SolidStateDrive: Samsung Evo (Id: 13) Generation: 7 Overall Performance: 111.55. Price: 2000.00 - VideoCard: Nvidia GeForce (Id: 17) Generation: 9 Peripherals (1); Average Overall Performance (70.00):

```
  Overall Performance: 70.00. Price: 300.00 - Headset: Razer Thresher (Id: 3) Connection
Type: AUX
Successfully removed Headset with id 3.
Overall Performance: 105.51. Price: 7950.00 - Laptop: Asus ROG (Id: 4)
 Components (6):
  Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3)
Generation: 9
  Overall Performance: 87.50. Price: 1250.00 - Motherboard: Asus ROG (Id: 6) Generation: 8
  Overall Performance: 73.50. Price: 700.00 - PowerSupply: Fortron FSP (Id: 7) Generation: 2
  Overall Performance: 96.00. Price: 900.00 - RandomAccessMemory: Kingston HyperX (Id: 10)
Generation: 4
  Overall Performance: 102.00. Price: 800.00 - SolidStateDrive: Samsung Evo (Id: 13)
Generation: 7
  Overall Performance: 111.55. Price: 2000.00 - VideoCard: Nvidia GeForce (Id: 17)
Generation: 9
 Peripherals (0); Average Overall Performance (0.00):
```

Input

```
AddComputer Laptop 4 Asus ROG 700
AddComputer Tablet 5 Asus ROG 700
AddComputer Laptop 0 Asus ROG 700
AddComputer Laptop 4 Asus ROG 700
AddComputer Laptop 7 Asus ROG 0
AddComponent 4 3 CentralProcessingUnit Intel Xeon 1600 82 10
AddComponent 55 33 CentralProcessingUnit Intel Xeon 1600 82 10
AddComponent 4 3 CentralProcessingUnit Intel Xeon 1600 82 10
AddComponent 4 30 InvalidComponent Intel Xeon 1600 82 10
AddComponent 4 0 CentralProcessingUnit Intel Xeon 0 82 10
AddComponent 4 -1 CentralProcessingUnit Intel Xeon 0 82 10
AddComponent 4 30 CentralProcessingUnit Intel Xeon 0 82 10
AddComponent 4 30 CentralProcessingUnit Intel Xeon 1600 0 10
AddComponent 4 13 SolidStateDrive Samsung Evo 800 85 8
RemoveComponent Motherboard 4
RemoveComponent SolidStateDrive 1
RemoveComponent SolidStateDrive 4
GetComputerData 100
GetComputerData 4
BuyComputer 4
BuyComputer 4
Close
```

Output

```
Computer with id 4 added successfully.
Computer type is invalid.
Id can not be less or equal than 0.
Computer with this id already exists.
Price can not be less or equal than 0.
Component CentralProcessingUnit with id 3 added successfully in computer with id 4.
Computer with this id does not exist.
Component with this id already exists.
Component type is invalid.
Id can not be less or equal than 0.
Id can not be less or equal than 0.
Price can not be less or equal than 0.
Overall Performance can not be less or equal than 0.
Component SolidStateDrive with id 13 added successfully in computer with id 4.
Component Motherboard does not exist in Laptop with Id 4.
Computer with this id does not exist.
Successfully removed SolidStateDrive with id 13.
```

```
Computer with this id does not exist.
Overall Performance: 112.50. Price: 2300.00 - Laptop: Asus ROG (Id: 4)
 Components (1):
  Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3)
Generation: 10
 Peripherals (0); Average Overall Performance (0.00):
Overall Performance: 112.50. Price: 2300.00 - Laptop: Asus ROG (Id: 4)
 Components (1):
  Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3)
Generation: 10
 Peripherals (0); Average Overall Performance (0.00):
Computer with this id does not exist.
```

Input

```
AddComputer DesktopComputer 1 Asus Huracan 500
AddComponent 1 1 CentralProcessingUnit Ryzen 3950 1700 80 10
AddComponent 1 4 Motherboard MSI MEG 1700 80 7
AddComponent 1 16 VideoCard Nvidia Quadro 4000 90 6
AddPeripheral 1 2 Monitor Dell S27 800 60 HDMI
AddComputer Laptop 4 Asus ROG 700
AddComponent 4 3 CentralProcessingUnit Intel Xeon 1600 82 11
AddComponent 4 6 Motherboard Asus ROG 1250 70 7
AddComponent 4 17 VideoCard Nvidia GeForce 2000 97 8
AddPeripheral 4 3 Headset Razer Thresher 300 70 Bluetooth
AddComputer DesktopComputer 2 Acer GX 490
AddComponent 2 9 PowerSupply Corsair Hydro 200 40 8
AddComponent 2 14 SolidStateDrive Samsung Evo 800 85 8
AddPeripheral 2 5 Monitor Dell S27 800 60 HDMI
GetComputerData 1
GetComputerData 4
GetComputerData 2
BuyBestComputer 6000
GetComputerData 4
Close
```

Output

```
Computer with id 1 added successfully.
Component CentralProcessingUnit with id 1 added successfully in computer with id 1.
Component Motherboard with id 4 added successfully in computer with id 1.
Component VideoCard with id 16 added successfully in computer with id 1.
Peripheral Monitor with id 2 added successfully in computer with id 1.
Computer with id 4 added successfully.
Component CentralProcessingUnit with id 3 added successfully in computer with id 4.
Component Motherboard with id 6 added successfully in computer with id 4.
Component VideoCard with id 17 added successfully in computer with id 4.
Peripheral Headset with id 3 added successfully in computer with id 4.
Computer with id 2 added successfully.
Component PowerSupply with id 9 added successfully in computer with id 2.
Component SolidStateDrive with id 14 added successfully in computer with id 2.
Peripheral Monitor with id 5 added successfully in computer with id 2.
Overall Performance: 116.17. Price: 8700.00 - DesktopComputer: Asus Huracan (Id: 1)
 Components (3):
  Overall Performance: 100.00. Price: 1700.00 - CentralProcessingUnit: Ryzen 3950 (Id: 1)
Generation: 10
  Overall Performance: 100.00. Price: 1700.00 - Motherboard: MSI MEG (Id: 4) Generation: 7
  Overall Performance: 103.50. Price: 4000.00 - VideoCard: Nvidia Quadro (Id: 16) Generation:
6
 Peripherals (1); Average Overall Performance (60.00):
  Overall Performance: 60.00. Price: 800.00 - Monitor: Dell S27 (Id: 2) Connection Type: HDMI
```

```
Overall Performance: 110.52. Price: 5850.00 - Laptop: Asus ROG (Id: 4)
 Components (3):
  Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3)
Generation: 11
  Overall Performance: 87.50. Price: 1250.00 - Motherboard: Asus ROG (Id: 6) Generation: 7
  Overall Performance: 111.55. Price: 2000.00 - VideoCard: Nvidia GeForce (Id: 17)
Generation: 8
 Peripherals (1); Average Overall Performance (70.00):
  Overall Performance: 70.00. Price: 300.00 - Headset: Razer Thresher (Id: 3) Connection
Type: Bluetooth
Overall Performance: 87.00. Price: 2290.00 - DesktopComputer: Acer GX (Id: 2)
 Components (2):
  Overall Performance: 42.00. Price: 200.00 - PowerSupply: Corsair Hydro (Id: 9) Generation:
8
  Overall Performance: 102.00. Price: 800.00 - SolidStateDrive: Samsung Evo (Id: 14)
Generation: 8
 Peripherals (1); Average Overall Performance (60.00):
  Overall Performance: 60.00. Price: 800.00 - Monitor: Dell S27 (Id: 5) Connection Type: HDMI
Overall Performance: 110.52. Price: 5850.00 - Laptop: Asus ROG (Id: 4)
 Components (3):
  Overall Performance: 102.50. Price: 1600.00 - CentralProcessingUnit: Intel Xeon (Id: 3)
Generation: 11
  Overall Performance: 87.50. Price: 1250.00 - Motherboard: Asus ROG (Id: 6) Generation: 7
  Overall Performance: 111.55. Price: 2000.00 - VideoCard: Nvidia GeForce (Id: 17)
Generation: 8
 Peripherals (1); Average Overall Performance (70.00):
  Overall Performance: 70.00. Price: 300.00 - Headset: Razer Thresher (Id: 3) Connection
Type: Bluetooth
Computer with this id does not exist.
```

Task 3: Unit Testing (100 Points)

You will receive a skeleton with one class inside. The class will have some methods, fields and constructors. Cover the whole class with unit test to make sure that the class is working as intended.