Java OOP Exam – 11 December 2021

1. Overview

Cat houses are a modern convenience for any home that has a cat. You need to create a catHouse project to monitor the cat's habits. Every House has a Cat that requires different care. Your job is to add, feed, and care for the cats.

2. Setup

- Upload only the catHouse 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.

3. Task 1: Structure (50 points)

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

There are **3** types of entities in the application: **House, Cat, Toy**.

There should also be **ToyRepository**.

BaseToy

BaseToy is a base class of any type of toy and it should not be able to be instantiated.

Data

- softness int
- price double
 - o The price of the toy.

Constructor

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

(int softness, double price)

Child Classes

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

Ball

The ball has a softness of 1 and a price of 10.

Note: The Constructor **should take no values** upon initialization.



















Mouse

The mouse has a **softness of 5** and a **price of 15**.

Note: The Constructor should take no values upon initialization.

BaseCat

BaseCat is a base class of any type of cat and it should not be able to be instantiated.

Data

- name String
 - o If the name is null or whitespace, throw a NullPointerException with a message:
 - "Cat name cannot be null or empty."
 - o All names are unique.
- breed String
 - o If the breed is null or whitespace, throw a NullPointerException with a message:
 - "Cat breed cannot be null or empty."
- kilograms int
 - The kilograms of the Cat.
- price double
 - The price of the Cat.
 - If the price is below or equal to 0, throw an IllegalArgumentException with a message:
 - "Cat price cannot be below or equal to 0."

Behavior

void eating()

The eating() method increases the Cat's kilograms. Keep in mind that some breeds of Cat can implement the method differently.

Constructor

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

(String name, String breed, double price)

Child Classes

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

ShorthairCat

Has initial kilograms of 7.

Can only live in ShortHouse!

The constructor should take the following values upon initialization:

(String name, String breed, double price)

Behavior

void eating()

• The method increases the cat's kilograms by 1.



















LonghairCat

Has initial kilograms of 9.

Can only live in LongHouse!

The constructor should take the following values upon initialization:

(String name, String breed, double price)

Behavior

void eating()

• The method increases the cat's kilograms by 3.

BaseHouse

BaseHouse is a base class of any type of house and it should not be able to be instantiated.

Data

- name String
 - o If the name is null or whitespace, throw a NullPointerException with a message: "House name cannot be null or empty."
 - All names are unique.
- capacity int
 - o The number of Cat an House can have.
- toys Collection<Toy>
- cats Collection<Cat>

Behavior

int sumSoftness()

Returns the sum of each toy's softness in the House.

```
void addCat(Cat cat)
```

Adds a Cat in the House if there is a capacity for it.

If there is not enough capacity to add the Cat in the House, throw an IllegalStateException with the following message:

"Not enough capacity for this cat."

```
void removeCat(Cat cat)
```

Removes a Cat from the House.

void buyToy(Toy toy)

Buy (adds) a Toy in the House.

void feeding()

The **feeding()** method **feeds all cats** in the **House**.

String getStatistics()

Returns a **String** with **information** about the **House** in the format below.

If the House doesn't have a cat, print "none" instead.



















```
"{houseName} {houseType}:
```

Cats: {catName1} {catName2} {catName3} ... / Cats: none

Toys: {toysCount} Softness: {sumSoftness}"

Constructor

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

(String name, int capacity)

Child Classes

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

ShortHouse

Has 15 capacity.

The constructor should take the following values upon initialization:

(String name)

LongHouse

Has 30 capacity.

The constructor should take the following values upon initialization:

(String name)

ToyRepository

The toy repository is a repository for the toys that are in the house.

Data

toys - Collection<Toy>

Behavior

void buyToy(Toy toy)

Adds a toy to the collection.

boolean removeToy(Toy toy)

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

Toy findFirst(String type)

Returns the first toy 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, 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 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**. The given methods should have the following logic:

Data

You need some private fields in your controller class:

- toys ToyRepository
- houses Collection<House>

Commands

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

Note: The House name passed to the methods will always be valid!

AddHouse Command

Parameters

- type String
- name String

Functionality

Creates and adds a House to the houses' collection. Valid types are: "ShortHouse" and "LongHouse".

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

"Invalid house type."

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

"{houseType} is successfully added."

BuyToy Command

Parameters

type-String

Functionality

Creates a toy of the given type and adds it to the ToyRepository. Valid types are: "Ball" and "Mouse". If the toy type is invalid, throw an IllegalArgumentException with a message:

"Invalid toy type."

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

"{toyType} is successfully added."

ToyForHouse Command

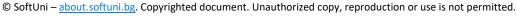
Parameters

- houseName String
- toyType String

Functionality

Adds (buys) the desired Toy to the House with the given name. You have to remove the Toy from the **ToyRepository** if the insert is **successful**.

















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

"Toy of type {toyType} is missing."

If no exceptions are thrown, return the String:

"Successfully added {toyType} to {houseName}."

AddCat Command

Parameters

- houseName String
- catType-String
- catName String
- catBreed String
- price double

Functionality

Creates and adds the desired Cat to the House with the given name. Valid Cat types are: "ShorthairCat", "LonghairCat".

Note: The method must first check whether the cat type is valid.

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

"Invalid cat type."

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

- "Unsuitable house." if the given Cat cannot live in the House.
- "Successfully added {catType} to {houseName}." if the Cat is added successfully in the House.

FeedingCat Command

Parameters

houseName - String

Functionality

Feeds all **Cat** in the **House** with the given name.

Returns a string with information about how many cats were successfully fed, in the following format:

"Feeding a cat: {fedCount}"

SumOfAll Command

Parameters

• houseName - String

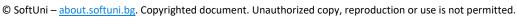
Functionality

Calculates the value of the **House** with the given name. It is calculated by the sum of all **Cat's** and **Toy's** prices in the House.

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

- "The value of House {houseName} is {value}."
 - o The value should be formatted to the 2nd decimal place!



















Statistics Command

Functionality

Returns information about each house. You can use House's getStatistics method to implement the current functionality.

```
"{houseName} {houseType}:
Cats: {catName1} {catName2} {catName3} ... / Cats: none
Toys: {toysCount} Softness: {sumSoftness}"
"{houseName} {houseType}:
Cats: {catName1} {catName2} {catName3} ... / Cats: none
Toys: {toysCount} Softness: {sumSoftness}"
. . . "
```

End Command

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:

- AddHouse {type} {name}
- BuyToy {type}
- ToyForHouse {houseName} {toyType}
- AddCat {houseName} {catType} {catName} {catBreed} {price}
- FeedingCat {houseName}
- SumOfAll {houseName}
- **Statistics**
- End

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
AddHouse ShortHouse HouseForPersian
AddHouse LongHouse HouseForSphynx
BuyToy Ball
BuyToy Ball
BuyToy Mouse
ToyForHouse HouseForPersian Ball
ToyForHouse HouseForSphynx Ball
ToyForHouse HouseForPersian Mouse
```

















AddCat HouseForPersian ShorthairCat Matsa Persian 124.26 AddCat HouseForPersian ShorthairCat Pinko Persian 111.11 AddCat HouseForPersian ShorthairCat Pinko Sphynx 111.11 FeedingCat HouseForPersian

FeedingCat HouseForSphynx SumOfAll HouseForPersian FeedingCat HouseForPersian

Statistics

End

Output

ShortHouse is successfully added.

LongHouse is successfully added.

Ball is successfully added.

Ball is successfully added.

Mouse is successfully added.

Successfully added Ball to HouseForPersian.

Successfully added Ball to HouseForSphynx.

Successfully added Mouse to HouseForPersian.

Successfully added ShorthairCat to HouseForPersian. Successfully added ShorthairCat to HouseForPersian.

Successfully added ShorthairCat to HouseForPersian.

Feeding a cat: 3 Feeding a cat: 0

The value of House HouseForPersian is 371,48.

Feeding a cat: 3

HouseForPersian ShortHouse: Cats: Matsa Pinko Pinko Toys: 2 Softness: 6 HouseForSphynx LongHouse:

Cats: none

Toys: 1 Softness: 1

Input

AddHouse ShortHouse ForBritan

AddCat ForBritan LonghairCat Sisi Britan 453.40

AddCat ForBritan ShorthairCat Mani Britan1 111.40

AddHouse LongHouse ForPersian

AddCat ForPersian ShorthairCat Esmeralda Persian 100.20

AddCat ForPersian LonghairCat Diamond Persian1 231.40

AddCat ForBritan InvalidCat Chico Radgol 86.42

AddCat ForPersian InvalidCat Bully GreySiam 186.43

BuyToy Mouse

ToyForHouse ForPersian Mouse

ToyForHouse ForBritan Mouse

BuyToy Ball

ToyForHouse ForPersian Mouse

FeedingCat ForBritan

FeedingCat ForBritan

AddCat ForBritan LonghairCat JustName Breed -6

Statistics

End



















Output

ShortHouse is successfully added.

Unsuitable house.

Successfully added ShorthairCat to ForBritan.

LongHouse is successfully added.

Unsuitable house.

Successfully added LonghairCat to ForPersian.

Invalid cat type. Invalid cat type.

Mouse is successfully added.

Successfully added Mouse to ForPersian.

Toy of type Mouse is missing. Ball is successfully added. Toy of type Mouse is missing.

Feeding a cat: 1 Feeding a cat: 1

Cat price cannot be below or equal to 0.

ForBritan ShortHouse:

Cats: Mani

Toys: 0 Softness: 0 ForPersian LongHouse:

Cats: Diamond

Toys: 1 Softness: 5

Task 3: Unit Tests (100 points)

You will receive a skeleton with three classes inside - Main, Cat and House. House 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 cats (with HouseTests inside) from the skeleton.

















