受験者(Examinee): Anh Vu Dang Sign Out (/signout)

Software Engineering Mid-term exam (2020-11-17)

【日本語はこちら】 (?lang=ja)

For all problems, submit a Java program that describes the entire class, including import statements.

- Programs that cannot be compiled or executed as Java programs are not subject to scoring.
- Although an operation example is shown, please note that satisfying the operation example does not necessarily give a perfect score.

Question 1

Declare a public class `One` that meets the following requirements.

- If you give a sequence of integers as an argument of the java command, the product of them is displayed.
- Display 0 if a non-integer string is included.

Make the following operation example work.

```
$ java One 1 2 3 4
24
$ java One This is a pen
0
```

```
public class One {
    public static void main(String[] args) {
        int res = 1;
        for (String arg : args) {
            int intArg;
```

Question 2

Declare a public class `Two` that meets the following requirements.

- When executed, it accepts the input from the standard input and displays the character string that concatenates the first characters of each line that was input before a blank line was input.

Make the following operation example work.

```
$ java Two
INIAD
TOYO

IT
$ java Two
This
is
a
pen

Tiap
```

```
import java.util.Scanner;

public class Two {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer();
```

Question 3

3.1

Declare a public class `Machine` that meets the following requirements.

- Has a String instance field `model` that represents the model number.
- Has an int type instance field `price` that represents the price.
- Has a constructor `Machine (String model, int price)` that sets its arguments to the corresponding instance fields.
- The instance method `String getModel()` returns the model number.
- The instance method `int getPrice()` returns the price.
- All instance fields can only be set from the constructor and can be accessed by subclasses.
- Constructors and instance methods can be accessed from any class.

When executed with the following Main class, it should be displayed as shown in the operation example.

```
public class Main {
    public static void main(String[] args) {
        Machine m = new Machine("M367", 1000);
        System.out.println(m.getModel());
        System.out.println(m.getPrice());
    }
}
```

```
$ java Main
M367
1000
```

```
public class Machine {
    protected String model;
    protected int price;

    public Machine(String model, int price) {
```

3.2

Declare a public class `AdvancedMachine` that inherits from the `Machine` class and meets the following requirements.

- The static method `int getTotalPrice()` returns the sum of the prices of all `AdvancedMachine` instances.
- The instance method `int getPrice()` returns the price including tax. (The tax rate is 10%.)
- You cannot declare a subclass of this class.
- Fields declared in this class cannot be accessed by other classes.
- Constructors and instance methods declared in this class can be accessed by any class.

When executed with the following Main class, it should be displayed as shown in the operation example.

```
public class Main {
    public static void main(String[] args) {
        AdvancedMachine m1 = new AdvancedMachine("M367", 1000);
        AdvancedMachine m2 = new AdvancedMachine("RX231", 2000);
        System.out.println(m1.getModel());
        System.out.println(m1.getPrice());
        System.out.println(AdvancedMachine.getTotalPrice());
    }
}
```

```
$ java Main
M367
1100
3000
```

```
public class AdvancedMachine extends Machine {
    private static int totalPrice = 0;

    AdvancedMachine(String model, int price) {
        super(model, price);
```

Question 4

4.1

Declare a public abstract class `Plant` in the package `four` that represents plants to meet the following requirements.

- Has an int type instance field `price` that represents the price.
- Has a constructor `Plant(int price)` that sets its arguments to the corresponding instance fields.
- The instance method `int getPrice()` returns the price.
- Has an abstract method `String getColor()` that is supposed to return the color.
- Instance fields cannot be accessed by other classes.
- Constructors and instance methods can be accessed from any class.

```
package four;

abstract public class Plant {
    private int price;
```

4.2

Declare an interface `Edible` in the package `four` to meet the following requirements.

- Has a method `boolean isSweet()` that returns whether it is sweet or not.
- Has a method `boolean isSour()` that returns whether it is sour or not.

```
package four;

public interface Edible {
    public boolean isSweet();
```

4.3

Declare a public class `Citrus` in the package `four` that represents citrus fruits to meet the following requirements.

- Inherits the `Plant` class and implements the `Edible` interface.
- It has a constructor `Citrus(int price, String color, boolean sweet)` , which indicates the price, color, and whether or not it is sweet. — Be sure to make it sour.
- Fields declared in this class cannot be accessed by other classes.
- Constructors and instance methods declared in this class can be accessed by any class.

When executed with the following Main class, it should be displayed as shown in the operation example.

```
package four;

public class Main {
    public static void main(String[] args) {
        Citrus lemon = new Citrus(100, "Yello", true);
        System.out.println(lemon.getPrice());
        System.out.println(lemon.getColor());
        System.out.println(lemon.isSweet());
        System.out.println(lemon.isSour());
    }
}
```

```
$ java four.Main
100
Yello
true
true
```

```
package four;

public class Citrus extends Plant implements Edible {
    private String color;
    private boolean sweet;
```

Question 5

5.1

Declare a public class `Item` in the package `five` that represents a product that meets the following requirements.

- Has a String instance field `name` that represents a name.
- Has a constructor `Item(String name)` that sets its arguments to the corresponding instance fields.
- The instance method `String getName()` returns the name.
- Implements the `Comparable` interface to compare with other `Item` instances using the `name` field.
- Fields declared in this class cannot be accessed by other classes.
- Constructors and instance methods declared in this class can be accessed by any class.

```
package five;

public class Item implements Comparable<Item> {
    private String name;
```

5.2

Declare a public class `Shelf` in the package `five` that represents a shelf that meets the following requirements.

- The instance method `void push(Item item)` adds the product given as an argument to the shelf.
- The instance method `void print()` sorts the products on the shelves in ascending order by name and displays them separated by line breaks.
- Fields declared in this class cannot be accessed by other classes.
- The instance methods declared in this class can be accessed by any class.

When executed with the following Main class, it should be displayed as shown in the operation example.

```
package five;

public class Main {
    public static void main(String[] args) {
        Shelf shelf = new Shelf();
        shelf.push(new Item("INIAD"));
        shelf.push(new Item("TOYO"));
        shelf.push(new Item("AKABNE"));
        shelf.print();
    }
}
```

```
$ java five.Main
AKABNE
INIAD
TOYO
```

```
package five;

import java.util.*;

public class Shelf {
    private ArrayList<Item> items = new ArrayList<Item>();

    public void push(Item item) {
        items.add(item);
    }

    public void print() {
        Collections.sort(items);
        for (Item item : items) {
```

* Your answer will be automatically sent, but please press "SUBMIT" button to ensure your answers were perfectly sent.

SUBMIT