**Computer Language 2023**

**Assignment #5**

**Due: 15/May 23:59:59**

**Student ID:\_21102054\_ Name: \_Lim Saeyeon\_**

**1. Carefully read the following descriptions.**

**Find CalcEx.java file and add implementations to meet the following requirements:**

**도표이(가) 표시된 사진

자동 생성된 설명**

**1) Add *Bootable* interface: this interface should have “public abstract void boot()” method and “default void shutdown()” method. The shutdown method must provide an implementation for printing out a shutdown message (“---Shutdown---“) to the console.**

**2) Add *Refundable* interface: this interface should have “public abstract void refund()” method.**

**3) Add *Device* abstract class: this class must have “private String vendor” field which will be set by Device(String name) constructor. Also, a getter method for this field (i.e., vendor) must be implemented here.   
Device class should have “abstract void printlLogo()” method and “public void turnOn()” which prints out “[vendor field] is on now…” to the console.**

**4) Add *Calculator* abstract class: this class extends Device class and implements Bootable and Refundable interfaces.   
The boot() method of Bootable interface must be implemented in this class. The boot() method prints out “<<< Calculator Bootup >>>” to the console.   
The refund() method of Refundable interface must be implemented in this class. The refund() method prints out “[vendor name](use getter method) + Refunded!” to the console.  
Also, this class provide the following public abstract methods: int add (int num1, int num2), int subtract(int num1, int num2), double average(int[] nums) that will be implemented by NormalCalculator and AbsCalculator.**

**5) *NormalCalculator* and *AbsCalculator* extend Calculator class, and must implement all abstract methods defined in Calculator, Device, and Bootable/Refundable interfaces. The difference between NormalCalculator and AbsCalculator is as follows:**

**- Add method takes two integer numbers and then prints the result of addition**

**- Subtract method takes two integer numbers and then prints the result of subtraction**

**- Average method takes an array of integer numbers and then prints the average**

**- AbsCalculator does the same thing except that all the input parameters are converted to positive integer numbers (e.g,. myCalc.add(2, -3) will be treated like myCalc.add(2, 3))**

**- printLogo() method must be implemented by these classes. The printLogo() method of AbsCalculator and NormalCalculator prints “\*ITM\*" and “!I!T!M!”, respectively.**

**- Constructor which takes a single String parameter must be supported. This constructor should invoke a superclass’s constructor method to set the vendor field of Device class.**

**6) Line 1 ~23 cannot be modified.**

**Output)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**Your code:**

**0) My All Files**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**1) Bootable (Interface)**

public interface Bootable {  
 public abstract void boot();  
 default void shutdown() {  
 System.*out*.println("---Shutdown---");  
 }  
}

**2) Refundable (Interface)**

public interface Refundable {  
 public abstract void refund();  
}

**3) Device (Abstract Class)**

abstract class Device {  
 private String vendor;  
  
 public Device(String name) {  
 this.vendor = name;  
 }  
 public String getVendor() {  
 return vendor;  
 }  
 abstract void printLogo();  
 public void turnOn() {  
 System.*out*.println(vendor + " is on now...");  
 }  
}

**4) Calculator (Abstract Class)**

abstract class Calculator extends Device implements Bootable, Refundable {  
 public Calculator(String name) {  
 super(name);  
 }  
 @Override  
 public void boot() {  
 System.*out*.println("<<< Question1.Calculator Bootup >>>");  
 }  
 @Override  
 public void refund() {  
 System.*out*.println(getVendor() + " Refunded!");  
 }  
 public abstract int add(int num1, int num2);  
 public abstract int subtract(int num1, int num2);  
 public abstract double average(int[] nums);  
}

**5) AbsCalculator**

class AbsCalculator extends Calculator {  
  
 public AbsCalculator(String name) {  
 super(name);  
 }  
  
 @Override  
 public void printLogo() {  
 System.*out*.println("\*ITM\*");  
 }  
 @Override  
 public int add(int num1, int num2) {  
 return Math.*abs*(num1) + Math.*abs*(num2) ;  
 }  
 @Override  
 public int subtract(int num1, int num2) {  
 return Math.*abs*(num1) - Math.*abs*(num2);  
 }  
 @Override  
 public double average(int[] nums) {  
 double sum = 0;  
 for (int i : nums) sum += Math.*abs*(i);  
 return sum / nums.length ;  
 }  
}

**6) NormalCalculator**

class NormalCalculator extends Calculator {  
 public NormalCalculator(String name) {  
 super(name);  
 }  
  
 @Override  
 public void printLogo() {  
 System.*out*.println("I!T!M!");  
 }  
 @Override  
 public int add(int num1, int num2) {  
 return num1 + num2 ;  
 }  
 @Override  
 public int subtract(int num1, int num2) {  
 return num1 - num2;  
 }  
 @Override  
 public double average(int[] nums) {  
 double sum = 0;  
 for (int i : nums) sum += i;  
 return sum / nums.length;  
 }  
}

**Your result (screenshot):**

**텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

**1) Bootable (Interface)**

The Bootable interface declares abstract method boot() and also provides a default method shutdown(). The shutdown() method provides a default implementation that simply prints "---Shutdown---".

**2) Refundable (Interface)**

The Refundable interface declares abstract method refund().

**3) Device (Abstract Class)**

The Device class has a private member variable vendor to store the name.

The constructor Device is used to initialize the vendor variable with the provided name.

The getVendor method is a public getter method that allows access to the vendor variable.

The printLogo method is declared as abstract so that another concrete subclass must provide its implementation.

The turnOn method is a non-abstract method that simply prints a message, “[vendor] on now…”, indicating that the device is turned on by vendor.

**4) Calculator (Abstract Class)**

The Calculator class is an abstract class that extends Device and implements the Bootable and Refundable interfaces. It represents a calculator device.

Constructor Calculator calls the constructor of the Device class and pass the provided name.

The boot() method prints a message indicating the boot-up process of the calculator.

The refund method prints a message indicating that the calculator has been refunded.

The add method is an abstract method that takes two integers as input and returns their sum.

The subtract method is an abstract method that takes two integers as input and returns the result of subtracting num2 from num1.

The average method is an abstract method that takes an array of integers as input and calculates and returns the average of the numbers in the array.

**5) AbsCalculator**

The AbsCalculator class extends the functionality of the Calculator class by providing implementations for the abstract methods. It performs mathematical operations using the absolute values of the numbers.

The constructor AbsCalculator calls the constructor of the Calculator class and pass the provided name.

The printLogo prints the logo of the calculator, which in this case is "ITM".

The add method calculates the sum of the absolute values of num1 and num2.

The subtract method calculates the difference between the absolute values of num1 and num2.

The average method calculates the average of the absolute values of the numbers in the nums array.

The absolute values of num1 and num2 in the operation are Math.It is obtained through the Math.abs().

**6) NormalCalculator**

Both the code structure and function are the same as abscalculator except that the value itself is calculated, not the absolute value operation of num1 and num2.

**2. Carefully read the following code and descriptions.**

|  |
| --- |
| import java.util.Scanner;  public class InputException {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  System.*out*.println("Input three integer numbers!");  int sum=0, n=0;  for(int i = 0; i<3; i++) {  System.*out*.print(i+">>");  n = scanner.nextInt(); *// Exception occurs when invalid input given* sum += n;  }  System.*out*.println("Sum is " + sum);  scanner.close();  } } |

**- The above program is crashed when a user types invalid type input (e.g., “jinwoo”, 130.130, etc). Use try/catch clause to avoid such exception/crash and finally get the result.**

**- When invalid type data given, your program must print "Not integer number. Please input again!" and keep waiting for a new input number from the user.**

**- When your program takes 3 valid integer input data from the user, the result should be computed and printed out.**

텍스트이(가) 표시된 사진

자동 생성된 설명**Example Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

import java.util.Scanner;  
public class InputException {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Input three integer numbers!");  
 int sum = 0, n = 0, count = 0;  
  
 while (count < 3) {  
 try {  
 System.*out*.print(count + ">>");  
 n = scanner.nextInt();  
 sum += n;  
 count++;  
 } catch (Exception notInt) {  
 System.*out*.println("Not an integer number. Please input again!");  
 scanner.nextLine();  
 }  
 }  
  
 System.*out*.println("Sum is " + sum);  
 scanner.close();  
 }  
 }

**Your result (screenshot):**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

I used a while loop with a try-catch block to handle input exceptions. Inside the try block, the program prompts the user for input and tries to read an integer using scanner.nextInt(). If the input is a valid integer, the program adds it to the sum and increments the count. If an InputMismatchException occurs, the catch block is executed. It displays an error message and clears the input buffer using scanner.nextLine() to discard the invalid input. The program continues to prompt the user until three valid integer inputs are received.

**3. Carefully read the following code and descriptions.**

public class EqualsEx {  
 public static void main(String[] args) {  
 Rect a = new Rect(2,3);  
 Rect b = new Rect(3,2);  
 Rect c = new Rect(3,4);  
 if(a.equals(b))  
 System.*out*.println("a is equal to b");  
 if(a.equals(c))  
 System.*out*.println("a is equal to c");  
 if(b.equals(c))  
 System.*out*.println("b is equal to c");  
 }  
}  
  
class Rect {  
 private int width;  
 private int height;  
 public Rect(int width, int height) {  
 this.width = width;  
 this.height = height;  
 }  
}

**- The above program prints out nothing because Rect instances point different addresses.**

**- Improve Rect class to compare the instance identity based on the area. In other words, Rect with the width of 2 and the height of 3 (here, its area is 6) should be identical to Rect with the width of 3 and the height of 2 (area of this rect is also 6!).**

**Example Output)**

텍스트이(가) 표시된 사진

자동 생성된 설명

**Your code:**

public class EqualsEx {  
 public static void main(String[] args) {  
 Rect a = new Rect(2,3);  
 Rect b = new Rect(3,2);  
 Rect c = new Rect(3,4);  
 if (a.equals(b))  
 System.*out*.println("a is equal to b");  
 if (a.equals(c))  
 System.*out*.println("a is equal to c");  
 if (b.equals(c))  
 System.*out*.println("b is equal to c");  
 }  
}  
  
class Rect {  
 private int width;  
 private int height;  
 public Rect(int width, int height) {  
 this.width = width;  
 this.height = height;  
 }  
 private int getArea() { return width \* height; }

@Override  
 public boolean equals(Object obj) {  
 return getArea() == ((Rect)obj).getArea();  
 }  
}

**Your result (screenshot):**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**Your explanation on the code:**

I added overridden equals method in the Rect class. The equals method compares the areas of two Rect objects to determine their equality. It calculates the area by multiplying the width and height of the Rect object. If the areas of two Rect objects are equal, the equals method returns true, otherwise, it returns false. By modified this code, I can compare sizes of Rect objects.