Exam DIT042 / DIT043: Object-Oriented Programming

2022-10-24 - PM (Lindholmen)

Examiner:	Francisco Gomes de Oliveira Neto
Questions:	+46 31 772 69 51
	Francisco will visit the exam hall at ca 15:00 and ca 16:30.
Results:	Will be posted within 15 working days.
Grades:	00–49 points: U (Fail)
	50–69 points: 3 (Pass)
	70–84 points: 4 (Pass with credit)
	85–100 points: 5 (Pass with distinction)
Allowed aids:	No aids (books or calculators) are allowed.
Reviewing:	2022-11-08 13:00–14:00,
	Jupiter building, 4th Floor, Room 400, Lindholmen

Please observe carefully the instructions below. Not following these instructions will result in the deduction of points.

- Write clearly and in legible English (illegible translates to "no points"!). Answers that require code must be written using the Java programming language.
- Motivate your answers, and clearly state any assumptions made. Your own contribution is required.
- Before handing in your exam, **number and sort the sheets in task order!** Write your anonymous code and page number on every page! The exam is anonymous, **do not** leave any information that would reveal your name on your exam sheets.
- When answering what is being printed by the program, make sure to write as if the corresponding answer were being printed in the console.
- You can assume that all classes in the code presented in this exam are in the **same package**. Similarly, **you can assume** that all classes used in the code (Exceptions, Lists, Streams, etc.) are properly imported in their respective file.
- For **all class diagrams** in your exam, write all methods and attributes relevant to your code, including **constructors, getters and setters**.

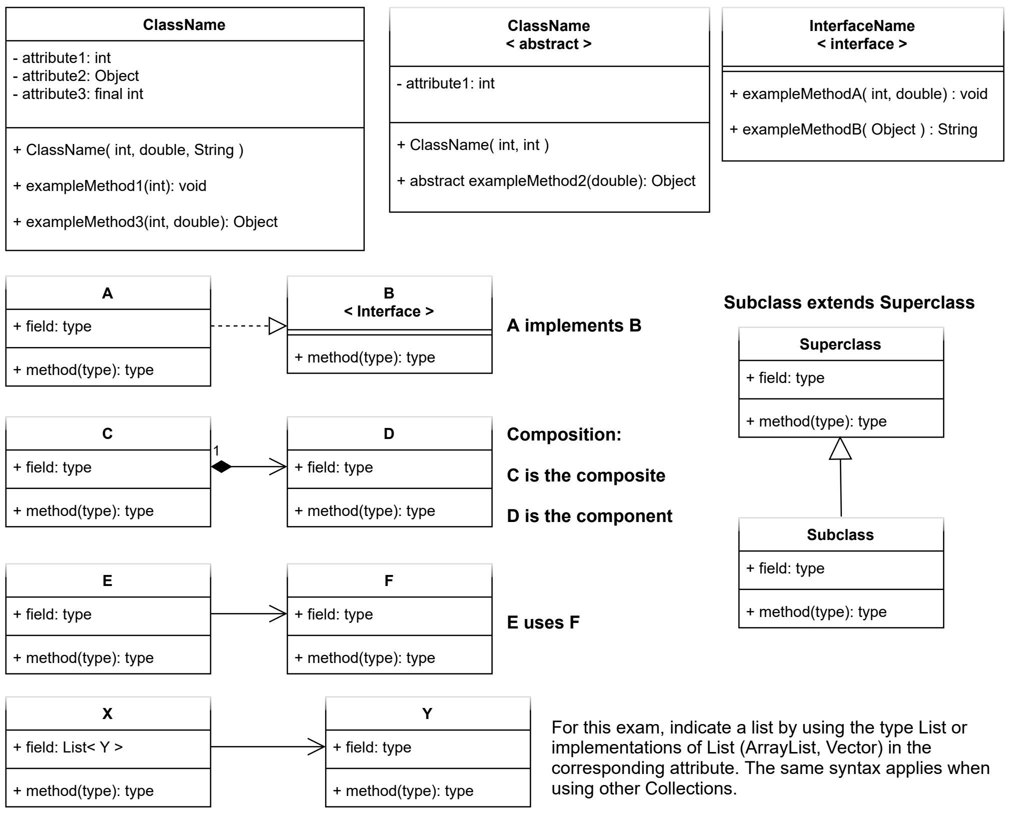

Figure 1: The elements of a class diagram. For this exam, you do not need to represent Exceptions in your diagrams.

Table 1: List of useful methods from the List interface and String class.

Class	Method specification	Description
List	add(E e): boolean	Appends the specified element to the end of the collection.
List	contains(Object o): boolean	Returns true if this collection contains the element.
List	get(int index): Object	Returns the element at the specified index.
List	isEmpty(): boolean	Returns true if this collection contains no elements.
List	size(): int	Returns the number of elements in this collection.
String	length(): int	Returns the number of characters in the string.
String	isEmpty(): boolean	Returns a boolean value indicating whether the string is empty.
String	contains(String s): boolean	Returns a boolean indicating whether the specified string 's' is a substring of the current string.
String	startsWith(String s): boolean	Returns a boolean indicating whether the current string starts with the specified string 's'.
String	endsWith(String s): boolean	Returns a boolean indicating whether the current string starts with the specified string 's'.
String	substring(int start, int end): String	Returns a subset from the string with characters from the start index until the end index (not included).

Question 1 [Total: 10 pts]: Write **exactly** what is printed when running the Java program below. **Important:** The **ordering and the formatting is important**. If you want to skip printing lines, you should number your printed lines to indicate what you skipped.

```java
public class Point {
    int x;
    int y;
    static int total = 0;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
        updateTotal();
    }
    public void updateTotal(){ total = total + this.x + this.y; }
    public String toString() {
        return "x= " + x + " y= " + y + " total = " + total;
    }
}
// ----------------------------------------------------
public class PointMain {
    public static void mixContent(Point p1, Point p2) {
        int tempX = p1.x;
        int tempY = p1.y;
        p1.x = p2.y + tempX;
        p1.y = p2.x + tempY;
        p2.x = tempY;
        p2.y = tempX;
    }
    public static void mixContent(int p1, int p2) {
        p1 = p1 % 2 == 0 ? p1 + 5 : p1 + 10;
        p2 = p1 + p2;
        System.out.println("p1 = " + p1 +  " p2 = " + p2);
    }
    public static void main(String[] args) {
        Point p1 = new Point(-2, 3);
        Point p2 = new Point(7, 8);
        Point p3 = new Point(6, 1);

        mixContent(p2, p1);
        mixContent(p1, p3);
        System.out.println(p1);
        System.out.println(p2);
        System.out.println(p3);

        mixContent(p3.x, p2.x);
        p2.updateTotal();
        p2 = p3;
        p3 = p1;
        System.out.println(p1);
        System.out.println(p2);
        System.out.println(p3);

        mixContent(p2, p3);
        p1 = new Point(p2.x, p3.y);
        System.out.println(p1);
        System.out.println(p2);
        System.out.println(p3);
    }
}
```

Question 2 [Total: 40 pts]: The code below is for an application that monitors diets and calorie (kcal) intake. The diet has foods and a maximum caloric limit, specified when creating the diet. We can register food in this diet. Each food has a name, portion, kcal and protein; all of them can be retrieved but cannot not be changed once the food is created. Based on your knowledge of OOP concepts, answer:

Q2.1 [7 pts]: The code below has a problem (bug). Explain: i) What is the problem in the code?; ii) What are the consequences of this problem in the execution of the program? You can use the main method to illustrate your explanation; iii) How to fix the problem?

```java
public class Food {
  static final String name;
  static final double portion;
  static final double kcal;
  static final double protein;

  public Food(String name, double portion,
              double kcal, double protein) throws Exception {
    if(name.isEmpty()) {
      throw new Exception("Food name cannot be empty.");
    }
    if(portion <= 0) {
      throw new Exception("Portion must be greater than or equal to zero.");
    }
    this.name = name;
    this.portion = portion;
    this.kcal = kcal;
    this.protein = protein;
  }

  public String getName()    { return name; }
  public double getPortion() { return portion; }
  public double getKcal()    { return kcal; }
  public double getProtein() { return protein; }

  public String toString() {
    return this.portion +" of " + this.name + " has: " +
           this.kcal + "kcal "+ this.protein + "g protein";
  }
}
```

```java
// ----------------------------------------------------------
public class Diet {
  private List<Food> dietMenu;
  private double maxKcal;

  public Diet(double kcalLimit) throws Exception {
    if(kcalLimit <= 0) {
      throw new Exception("Max kcalorie must be greater than zero.");
    }
    maxKcal = kcalLimit;
    this.dietMenu = new ArrayList<>();
  }

  public boolean addFood(Food food) {
    // implement this method.
    return false;
  }
  public double getTotalDietKcals(){
    // implement this method.
    return 0.0;
  }
}
// ----------------------------------------------------------
public class FoodMain {
  public static void main(String[] args) {
    try{
      Diet diet = new Diet(1800);
      Food chicken = new Food("Chicken Fillet", 100.0, 100.0, 21.0);
      Food chocolate = new Food("Chocolate", 100.0, 550.0, 4.7);
      Food cheese  = new Food("Gouda Cheese", 200, 708.0, 50.0);
      Food nutella = new Food("Nutella", 100.0, 546.0, 6.0);
      Food bacon   = new Food("Bacon", 100.0, 350.0, 13.0);

      System.out.println(diet.addFood(chicken));   // true
      System.out.println(diet.addFood(chocolate)); // true
      System.out.println(diet.addFood(cheese));    // true
      System.out.println(diet.addFood(nutella));   // false - would exceed
      System.out.println(diet.addFood(bacon));     // true

    }catch(Exception exception){
      System.out.println(exception.getMessage());
    }
  }
}
```

Important: The questions below assume that you fixed the problem in Q2.1.

Q2.2 [8 pts]: Implement the functionalities: (i)`getTotalDietKcals()`: retrieve the current kcal registered in the diet (i.e., sum of all food items' kcal); and (ii) `addFood(Food food)` method: adds a food to the diet if the food does not cause the diet to exceed the kcal limit (returns a boolean indicating whether the addition was scessful).

Q2.3 [10 pts]: Implement the method `void printTopProteinDiff()` in the Diet class. This method should **print** the name of the two registered foods that have the biggest protein difference. You should use the absolute difference since $|a - b| == |b - a|$. Your algorithm will be evaluated based on correctness, readability and efficiency:

- Print: "`Top protein difference: <food-name1> and <food-name2>`"
- In case there are less than two food items registered, your method should throw an exception with the message: "Not enough food items. At least two food items must be registered".
- In case of a tie (i.e., two pairs of food have the same protein difference), you should keep the first pair found.
- In Java, use `Math.abs(a,b);` to calculate the absolute difference between two numbers `a` and `b`.
- Using the main method above as an example, the biggest difference would be between Chocolate and Gouda Cheese: ($|4.7 - 50.0| = 45.3$).

Q2.4 [8 pts]: Explain at least one common and one distinguishing aspect between encapsulation and immutable objects.

Q2.5 [7 pts]: Your boss wants your program to save the diet information in files but you cannot modify the Food class. Your colleague Ash, Misty and Brock have suggestions. Ash suggests you to use an Object output stream to save the list of foods, Misty suggests you to use a text stream for saving foods as strings, and Brock suggests you to use a Scanner. For each suggestion, **explain and justify** if they are appropriate solutions.

Question 3 [Total: 40 pts]: You started at a team that wrote an encoder for strings. An Encoder is a program that transforms a string into another string by operating on its content (e.g., reversing, shuffling, swapping characters, etc). The code for the Encoder class is below.

There are two ways to encode a string. The first way uses a split point (Splitter) that, based on an arbitrary point in the string, we split and swap its content. Using the string `"Deep breath"` with a split point of 2 we get: `"ep breathDe"`. The split point is only used in the Splitter. The second way is a Reverser that simply reverses the characters of a String using a StringBuilder for that.[1] For the same example, the Reverser would result in `"htaerb peeD"`.

[1] Do not worry about understanding the StringBuilder, it is a class that comes with Java.

```java
public class Encoder {
    private String content;
    private int splitPoint;
    private String type;

    public Encoder(String content, String type) {
        this.content = content;
        this.splitPoint = 0;
        this.type = type;
    }
    public Encoder(String content, int splitPoint, String type) {
        this.content = content;
        this.splitPoint = splitPoint;
        this.type = type;
    }
    public String encodeMessage(){
        String encodedMessage = "";
        if(type.equals("Splitter")) {
            String part1 = content.substring(0, splitPoint);
            String part2 = content.substring(splitPoint, content.length());
            encodedMessage = part2 + part1;

        } else if(type.equals("Reverser")) {
            // class from Java to reverse the string.
            StringBuilder builder = new StringBuilder(content);
            encodedMessage = builder.reverse().toString();
        }
        return encodedMessage;
    }

    // getters and setters
    public String getContent(){return this.content;}
    public void setContent(String content) {this.content = content;}
    public int getSplitPoint(){return this.splitPoint;}
    public void setSplitPoint(int newSplit){this.splitPoint = newSplit;}
}
```

Currently, there are no external classes using your Encoder. However, your boss warns you about the following upcoming modifications:

- You must prevent the creation of Splitter encoders when the split point is negative or greater than the length of the string.
- You **can assume** that the strings given to the Encoder are never null or empty.
- You need to add a new Encoder named `"Quantum"` with the following encoding algorithm: i) if the length of the string is an odd number, then the result will be a concatenation of the string to itself; ii) if the string length is an even number, then the result should be only the first half of the string. Examples:
  - `"Deep" (length 4): "De".`
  - `"Deep breath" (length 11): "Deep breathDeep breath".`

Q3.1 [10 pts]: The SOLID design pattern is composed of five principles: Single-responsibility principle, Open closed principle, Liskov substitution principle, Interface segregation principle, and Dependency inversion principle. Which of the principles are particularly important for improving the design of the Encoder class? In your answer you must: (i) explain the **definition** your chosen principle(s), (ii) explain **how and why** you can improve the Encoder using that principle (refer to parts of the class, such as methods, lines of code, etc.).

Q3.2 [8 pts]: Based on your suggestions, write the class diagram for your improved design. You should also include the Quantum encoder in your solution.

Q3.3 [5 pts]: Considering your new design, i) how do you intend to prevent the creation of splitter encoders with invalid split points? ii) Will your cause any changes in other classes in your design? Justify your answer.

Q3.4 [7 pts]: Write the complete code for your Quantum encoder.

Q3.5 [10 pts]: Explain at least three differences between Inheritance and Composition.

Question 4 [10 pts]: The code below represents a hierarchy of Calculators. Using your knowledge of Inheritance and overriding methods, write what is printed in the marked lines below.

```java
public class Calculator {
    public int sum(int x, int y) { return x + y;}
    public int subtract(int x, int y) { return x - y;}
    public int multiply(int x, int y) { return x * y;}
}
public class EchoCalculator extends Calculator {
    public int sum(int x, int y) {
        int sumResult = x + y;
        return this.multiply(sumResult, 2);
    }
    public int subtract(int x, int y) {
        int subResult = x - y;
        return this.multiply(subResult, 2);
    }
}
public class CrypticCalculator extends EchoCalculator {
    public int sum(int x, int y) {
        return super.sum(x, y) * subtract(x, y);
    }
    public int multiply(int x, int y) {
        return (x * x) + (y * y);
    }
}
// ------------------------------------------------------
public class MainCalculator {
    public static void main(String[] args) {
        Calculator calc1 = new EchoCalculator();
        System.out.println(calc1.sum(6, 7));      //a
        System.out.println(calc1.subtract(6, 7)); //b
        System.out.println(calc1.multiply(6, 7)); //c

        Calculator calc2 = new CrypticCalculator();
        System.out.println(calc2.sum(6, 7));      //d
        System.out.println(calc2.subtract(6, 7)); //e
        System.out.println(calc2.multiply(6, 7)); //f
    }
}
```