# Exam DIT042 / DIT043:
# Object-Oriented Programming

2021-10-28 - PM (Lindholmen)

| | |
|---|---|
| **Examiner:** | Francisco Gomes de Oliveira Neto |
| **Questions:** | +46 31 772 69 51<br>Francisco will visit the exam hall.<br>at ca 15:00 and ca 16:30. |
| **Results:** | Will be posted within 15 working days. |
| **Grades:** | 00–49 points: U (Fail)<br>50–69 points: 3 (Pass)<br>70–89 points: 4 (Pass with credit)<br>90–100 points: 5 (Pass with distinction) |
| **Allowed aids:** | No aids (books or calculators) are allowed. |
| **Reviewing:** | 2021-11-19 13:00–14:30,<br>Jupiter building, 4th Floor, Room 424, Lindholmen |

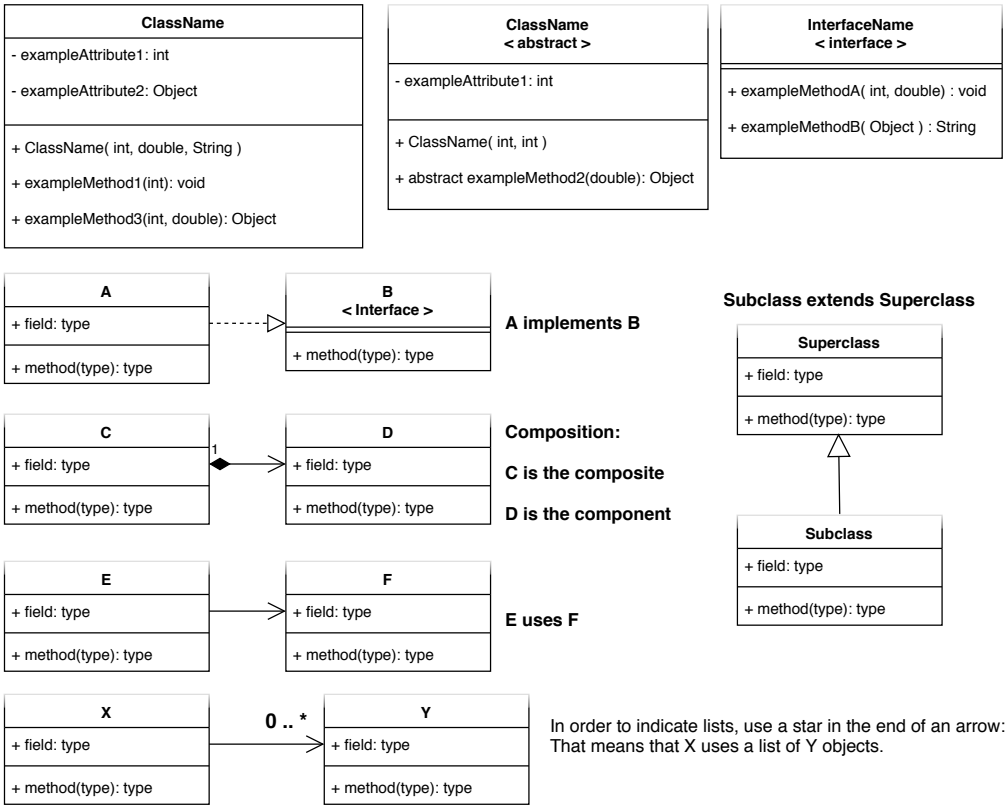



Figure 1: The elements of a class diagram.

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

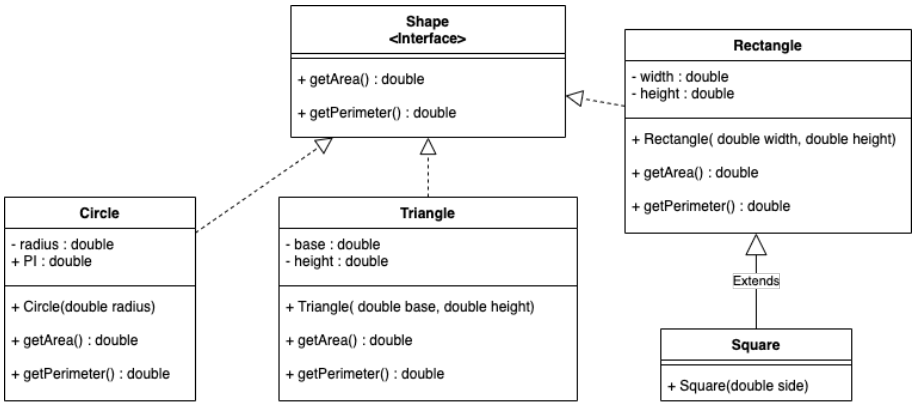
- Write clearly and in legible English (illegible translates to "no points"!). Answers that require code must be written using the Java programming language.
- Motivate your answers, and clearly state any assumptions made. Your own contribution is required.
- Before handing in your exam, **number and 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**.

**Q1 [15 pts]:** Considering the Java program below, what will be printed to the console when running the main method? **Important:** Your answers must be the same as it would be shown in the console when running the code below.

```
1  public class Mixer {
2
3      private int num1;
4      private int num2;
5      static int total = 0;
6
7      public Mixer(int num1, int num2) {
8          this.num1 = num1;
9          this.num2 = num2;
10
11         total = total + num1;
12         System.out.println("Total in the constructor is " + total);
13     }
14
15     public String toString() {
16         return "num1: " + num1 + ", num2: " + num2;
17     }
18
19     public static void updateTotal(int num1, int num2) {
20         total = total + num1 + num2;
21         System.out.println("Total after update is " + total);
22     }
23
24     public static void magicCalculation(int num1, int num2) {
25         num1 = num1 + num2;
26         num2 = num1 - num2;
27
28         System.out.println("method: num1 = " + num1 + " and num2 = " + num2);
29     }
30
31     public static void mixContent(Mixer mA, Mixer mB) {
32         int tempNum1 = mA.num1;
33         int tempNum2 = mA.num2;
34
35         mA.num1 = mB.num2;
36         mA.num2 = mB.num1;
37         mB.num1 = tempNum2;
38         mB.num2 = tempNum1;
39     }
40
41     public static void main(String[] args) {
42
43         Mixer mA = new Mixer(-5, -7);
44         Mixer mB = new Mixer(3, 5);
45         Mixer mC = new Mixer(4, 2);
46
47         mixContent(mA, mB);
48         updateTotal(mA.num1, mB.num1);
49
```

```
50          System.out.println("mA is " + mA);
51          System.out.println("mB is " + mB);
52          System.out.println("mC is " + mC);
53
54          int num1 = 8;
55          int num2 = 3;
56          magicCalculation(num1, num2);
57          System.out.println("main: num1 = " + num1+ " and num2 = " + num2);
58
59          mixContent(mA, mC);
60          System.out.println("mA is " + mA);
61
62          mB = mC;
63          mB.num1 = 10;
64          System.out.println("mB is " + mB);
65          System.out.println("mC is " + mC);
66
67          mC.num1 = num1;
68          mC.num2 = num2;
69          System.out.println("mA: "+mA);
70          System.out.println("mB: "+mB);
71          System.out.println("mC: "+mC);
72      }
73  }
```

**Q2 [25 pts]:** The class diagram below represents a software to create different shapes. Using your knowledge about object-oriented design, answer the questions below.




**Q2.1 [10 pts]:** Your colleague Jane states that using an Abstract Class for Shape is a better design solution than using an Interface. Do you agree or disagree with Jane? Justify your answer.

**Q2.2 [15 pts]:** List at least two advantages and two disadvantages of inheriting from abstract classes instead of normal classes (i.e., non-abstract classes).

**Q3 [45 pts]:** You were hired to write the software for a University system and you were responsible for working on the Student class. You were asked to implement the following change requests.

```
1  public class Student {
2
3      private String fullName;
4      private String id;
5
6      public Student(String fullName, String id) {
7          this.fullName = fullName;
8          this.id = id;
9      }
10
11     public double getGPA(){
12         // You must create this method.
13     }
14
15     public String toString() {
16         return this.id + ": " + this.fullName;
17     }
18     public String getFullName() {
19         return fullName;
20     }
21     public void setFullName(String fullName) {
22         this.fullName = fullName;
23     }
24     public String getId() {
25         return id;
26     }
27 }
```

**Request 1:** Students must now store their list of course results. This list should start empty when creating the student.

Each course result has a course name, an amount of credit that can include decimal numbers and a course grade that is an integer between 0–100. Once created, the information regarding a course result cannot be changed. Two course results are equal if they have the same course name and course credits.

You must prevent the creation of course results with grades less than zero, or greater than 100. When those cases occur, the message `"Invalid course result data."` should, eventually, be shown to the user.

**Q3.1 [10 pts]:** Write the code for the course result class.

**Q3.2 [15 pts]:** Describe the relationship between the student class and your course grade class. You must: (i) name and justify your choice of relationship, (ii) describe the code modifications needed to connect those two classes.

**Request 2:** The student object should calculate its GPA (grade average point). The GPA is a weighted average of the course grades based on the course credits. Considering that the student has i = 0...N course results:

$$GPA = \frac{\sum_{i=0}^{N} credits[i] * grade[i]}{\sum_{i=0}^{N} credits[i]} \tag{1}$$

**Q3.3 [10 pts]:** Considering your modifications to the Student class in **Q3.2**, write the code for the method `getGPA()`.

**Request 3:** Add a new type of student, the exchange student. The exchange student has the same state and behaviour as the regular student, except for a new way to calculate the GPA. The GPA for the exchange student object is the arithmetic mean (regular mean) of all grades in its course results.

**Q3.4 [10 pts]:** Write the class diagram for all your classes, including (i) the modifications in the Student class, (ii) your course result class and (iii) the exchange student. If applicable, you must include the constructors, getters and setters for all classes.

**Q4 [15 pts]:** The code below represents different types of membership. Each membership is associated with a user name and includes their corresponding points (that start at zero) and a credit system for purchases at a store.

All membership types can add points, add store credit and, lastly, convert those membership points into store credit. Those operations differ depending on the specific type of membership, as described in each class below. Using your knowledge of Inheritance and overriding methods, answer the questions below:

```
1  public class Membership {
2
3      private String userName;
4      private double storeCredit;
5      private int memberPoints;
6
7      public Membership(String userName, double initialCredit) {
8          storeCredit = initialCredit;
9          userName = userName;
10         memberPoints = 0;
11     }
12
13     public void addCredit(double addedCredit) {
14         this.storeCredit = this.storeCredit + addedCredit;
15     }
16
17     public void addMemberPoints(int addedPoints) {
18         this.memberPoints += addedPoints;
19     }
20
21     public void convertPoints(int numOfPoints) throws Exception {
22         if(this.memberPoints < numOfPoints || numOfPoints < 50) {
23             throw new Exception("Not enough points to convert.");
24         }
25         // Deduct points, convert to the corresponding credit
26         // value and add that credit to the membership.
27         this.memberPoints = this.memberPoints - numOfPoints;
28         double convertedCredits = numOfPoints * 0.20;
29         this.addCredit(convertedCredits);
30     }
31
32     public String toString(){
33         String result = userName + ": Points = " + memberPoints + ". ";
34         result += "Credits: " + storeCredit;
35         return result;
36     }
37
38     // getters.
39     public String getUsername() { return this.userName; }
40     public double getStoreCredit() { return storeCredit; }
41     public int    getMemberPoints() { return memberPoints; }
42 }
```

```java
public class SilverMembership extends Membership{
    public SilverMembership(String username, double initialCredit){
        super(username, initialCredit);
    }

    public void addMemberPoints(int numOfPoints){
        final int BONUS_POINTS = 10;
        super.addMemberPoints(numOfPoints + BONUS_POINTS);
    }
}
public class GoldMembership extends SilverMembership {
    public GoldMembership(String userName, double initialCredit){
        super(userName, initialCredit);
    }

    public void addCredit(double addedCredit){
        double bonusCredit = addedCredit + (addedCredit * 0.1); // +10%
        super.addCredit(bonusCredit);
    }
}
public class MembershipMain {
    public static void main(String[] args) {
        try{
            Membership m1 = new Membership("Alan Turing", 5000);
            m1.addCredit(1000);
            m1.addMemberPoints(200);
            m1.convertPoints(100);
            System.out.println(m1);

            Membership m2 = new SilverMembership("Ada Lovelace", 100);
            m2.addCredit(1000);
            m2.addMemberPoints(200);
            m2.convertPoints(80);
            System.out.println(m2);

            Membership m3 = new GoldMembership("Mary Keller", 500);
            m3.addCredit(5000);
            m3.addMemberPoints(400);
            m3.convertPoints(100);
            System.out.println(m3);

            Membership m4 = new GoldMembership("Grace Hopper", 100);
            m4.addCredit(1000);
            m4.addMemberPoints(200);
            m4.convertPoints(40);
            System.out.println(m4);

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

**Q4.1 [5 pts]:** The code above does not have a compilation problem. However, the code has a functionality problem (bug) that will lead to unexpected behaviour of the program. You must: (i) identify the line of code of the bug, (ii) explain what the bug will cause and (iii) explain how to fix the bug.

**Q4.2 [10 pts]: Assuming that you fixed the bug** above and the object behaves as expected; what is printed when executing the main method in the MembershipMain class.