Exam DIT042 / DIT043:
Object-Oriented Programming

2022-01-03 - 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) 70–89 points: 4 (Pass with credit) 90–100 points: 5 (Pass with distinction)
Allowed aids:	No aids (books or calculators) are allowed.
Reviewing:	2022-01-21 10:00–11:30, 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**.

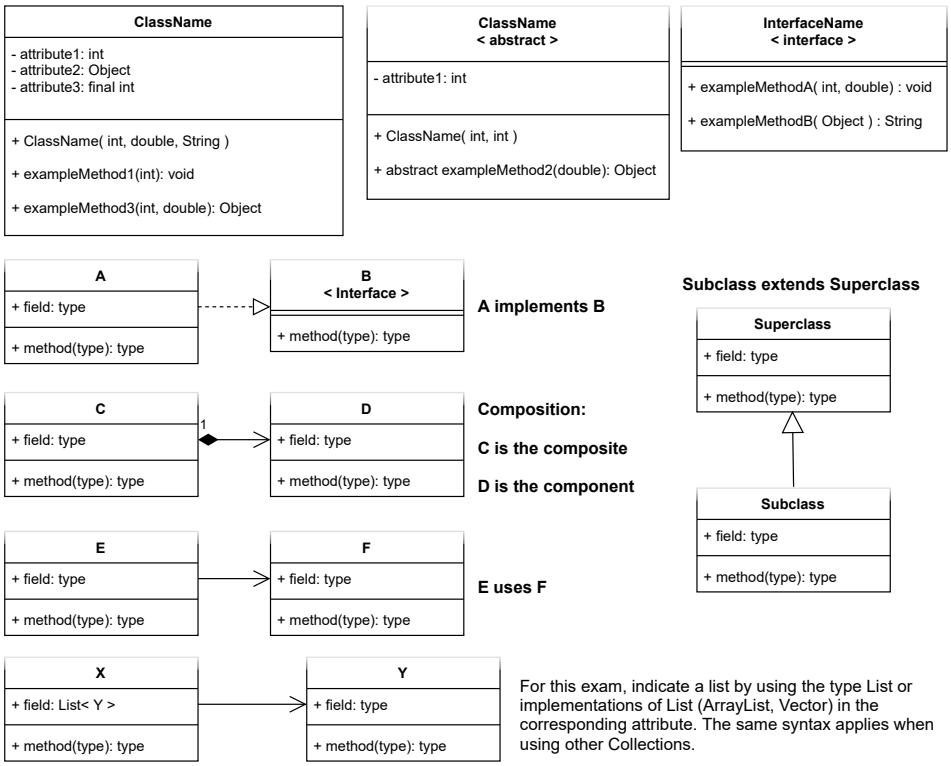

Figure 1: The elements of a class diagram.

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	remove(int index): Object	Removes the element at the specified position. The collection readjusts itself automatically shifting to the left.
List	remove(Object o): boolean	Removes the first occurrence of the specified element.
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	split(String separator): String[]	Returns an array of strings by splitting the current string in parts that match the specified separator.

Question 1 [Total: 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. So, both the ordering and the formatting is important. If you want to skip printing lines, you should number your printed specific lines to indicate what you skipped.

```
1  public class Mixer {
2    private int num1;
3    private int num2;
4    static int total = 0;
5
6    public Mixer(int num1, int num2) {
7      this.num1 = num1;
8      this.num2 = num2;
9      total = total + num2;
10     System.out.println("contructor: total: " + total);
11   }
12
13   public String toString() {
14     return "num1 = " + num1 + " num2 = " + num2;
15   }
16
17   public static void updateTotal(int num1, int num2) {
18     num1 = num1 + num2;
19     num2 = num1 - num2;
20     total = total + num1 + num2;
21     System.out.println("method: num1 = " + num1 + " num2 = " + num2 +
22                        " total = " + total);
23   }
24
25   public static void mixContent(Mixer mA, Mixer mC) {
26     int temp1 = mA.num1;
27     int temp2 = mA.num2;
28
29     mA.num1 = mC.num1;
30     mA.num2 = mC.num2;
31     mC.num1 = temp1;
32     mC.num2 = temp2;
33   }
34
35   public static void main(String[] args) {
36     Mixer mA = new Mixer(1, 2);
37     Mixer mB = new Mixer(-1, -2);
38     Mixer mC = new Mixer(4, -4);
39
40     mixContent(mA, mB);
41     mixContent(mC, mB);
42     System.out.println("mA has " + mA);
43     System.out.println("mB has " + mB);
44     System.out.println("mC has " + mC);
45
```

```
46
47      updateTotal(mA.num2, mB.num2);
48
49      int num1 = 8;
50      int num2 = 3;
51      updateTotal(num1, num2);
52      System.out.println("main: num1 = " + num1 + " num2 = " + num2 +
53                         " total = " + total);
54
55      mixContent(mA, mC);
56      System.out.println("mA has " + mA);
57
58      mC = mA;
59      mC.num1 = 10;
60      System.out.println("mA has " + mA);
61      System.out.println("mC has " + mC);
62
63      mA = new Mixer(100, 50);
64      mC.num1 = num1;
65      mC.num2 = num2;
66      System.out.println("mA has " + mA);
67      System.out.println("mC has " + mC);
68    }
69  }
```

Question 2 [Total: 30 pts]: You are hired by a social network company similar to Twitter. In this platform, users can create posts. Each post has a content and a number of likes. When creating a Post object, the content must be specified, whereas the number of likes must always start at zero. The senior developers created a diagram for the Post class.

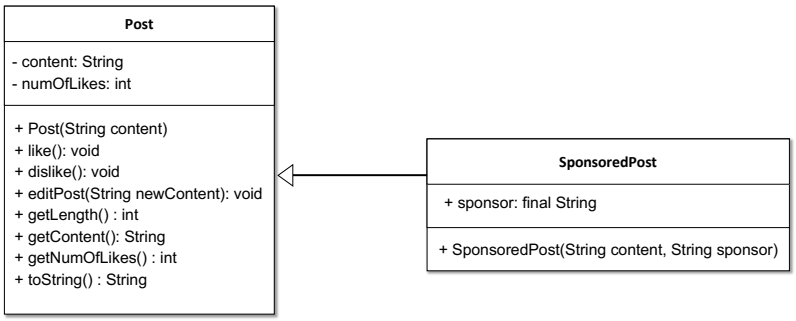

Figure 2: Class diagram for the Post and SponsoredPost class.

Q2.1 [10 pts]: Write the Java code for the class Post. The methods listed in the diagram should have the functionalities described in Table 2.

Table 2: Description of the methods of the class Post.

`like()`	Increases the number of likes by 1.
`dislike()`	Decreases the number of likes by 1. The number of likes should never be negative and should stop at zero when trying to decrease it below zero.
`getLength()`	Returns the number of characters in the content string.
`editPost(String newContent)`	Should update the value of the content attribute to the specified `newContent`. Works as a setter for the class.
`getContent()`	Should return the value of the `content` attribute. Works as a getter for `content`.
`getNumOfLikes()`	Should return the current number of likes for the post object. Works as a getter for `numOfLikes`
`toString()`	Returns the following string representing the post object: `"<content>: <numOfLikes> likes."`

Q2.2 [5 pts]: Note that SponsoredPost is a subclass via Inheritance of the Post class. Your colleague Ann says that the SponsoredPost object is an immutable object. Do you agree or disagree with Ann? Justify your answer.

Q2.3 [15 pts]: In Java, Inheritance and Interfaces are two ways of using polymorphism. You should answer: (i) one advantage of using polymorphism in object-oriented programming, (ii) one advantage of using Interfaces over Inheritance, and (iii) one advantage of using Inheritance over Interfaces.

Question 3 [Total: 40 pts]: Your senior colleague wrote the code below for a User of the social network. The User class uses the Post class specification (from Question 2). Each User has a username, birth year, a list with the name of their friends and a list of Post objects. You are asked to complete the code in the questions below.

```
public class User {

    private int birthYear;
    private String username;
    private List<String> friends;
    private List<Post> posts;

    public void addFriend(String friendName){
        friends.add(friendName);
    }

    public void removeFriend(String friendName){
        friends.remove(friendName);
    }

    public void createPost(String content){
        posts.add(new Post(content));
    }

    public void createSponsoredPost(String content, String sponsor){
        posts.add(new SponsoredPost(content,sponsor));
    }

    //You can assume that there are getters and setters for all attributes.
}
```

Q3.1 [10 pts]: Write code to allow the creation of User objects. There are two ways to create Users. The first way is to specify the username, the birth year, whereas the list of posts and the list of friends should begin empty. The second way to create users is equal to the first for all attributes, but with a specified list of friends.

In both ways of creating User objects, the system should prevent the creation of users with a birth year less than 2004 or users with an empty username. The error message for those respective cases should be `"Invalid user information. The user must be born after 2004."` and `"Invalid user information. User's name cannot be empty."`.

Q3.2 [7 pts]: Write a method that returns the content of the longest post of a user. The longest post is the content of the Post object with the highest number of characters. If there are several posts tied with the highest number of characters, you should return the one with higher index in the list. If the user has no posts, you should return an exception with the message below. For instance:

- `posts = ["Short post.", "very long post."] -> "very long post." (has 15 characters).`
- `posts = ["short", "long post A", "long post B"] -> "long post B." (has 11 characters)`
- `posts = [] -> Exception:  "User did not create any posts yet.")`

Q3.3 [8 pts]: Write a method that returns a list with the post content that have the highest number of likes. So if there are posts with the same highest number of likes, they should all be in the same list. If the user has no posts, you should return an exception with the message below. For example, consider each post object below as: `"(<content>,<numOfLikes>)"`:

- `posts = [("Hello World",1), ("Equality!",10)] -> ["Equality!"]`
- `posts = [("SEM is the best!",10), ("Best day ever!",2), ("I love OOP",10)] -> ["SEM is the best", "I love OOP"]`
- `posts = [] -> Exception:  "User did not create any posts yet.")`

Q3.4 [15 pts]: Describe the relationship between the User and the Post class. You must: (i) name and justify the relationship, (ii) describe how this relationship is implemented based on the code of both classes.

Question 4 [Total: 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, what is printed when executing the main method?**

```
1  public class Membership {
2
3    private String userName;
4    private double storeCredit;
5    private int memberPoints;
6
7    public Membership(String userName, double storeCredit) {
8      this.storeCredit = storeCredit;
9      this.userName = userName;
10     this.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 membership points, convert to the corresponding credit value
26     // 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(3000);
      m1.addMemberPoints(250);
      m1.convertPoints(60);
      System.out.println(m1);

      Membership m2 = new SilverMembership("Ada Lovelace", 100);
      m2.addCredit(500);
      m2.addMemberPoints(200);
      m2.convertPoints(50);
      System.out.println(m2);

      Membership m3 = new GoldMembership("Mary Keller", 500);
      m3.addCredit(3500);
      m3.addMemberPoints(400);
      m3.convertPoints(200);
      System.out.println(m3);

      Membership m4 = new SilverMembership("Grace Hopper", 1000);
      m4.addCredit(1000);
      m4.addMemberPoints(110);
      m4.convertPoints(70);
      System.out.println(m4);

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