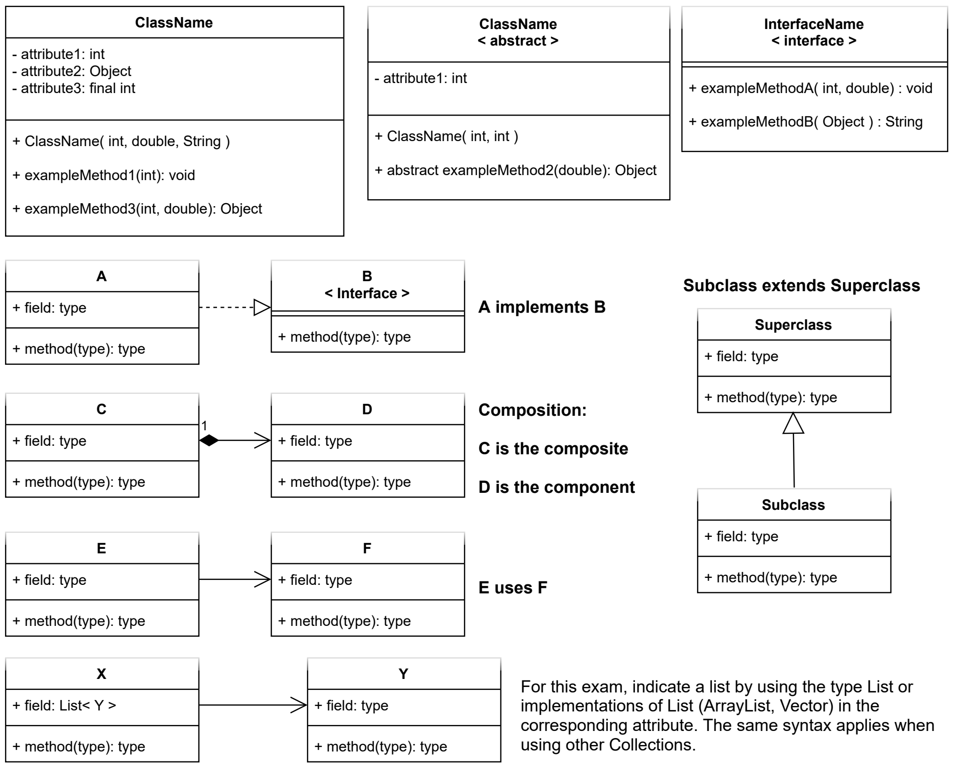
Exam DIT042 / DIT043: Object-Oriented Programming

2022-08-16 - AM (Lindholmen)

Examiner:	Francisco Gomes de Oliveira Neto
Questions:	+46 31 772 69 51
	Francisco will visit the exam hall at ca 09:30 and ca 11:30.
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:	2022-09-06 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**.



For this exam, indicate a list by using the type List or implementations of List (ArrayList, Vector) in the corresponding attribute. The same syntax applies when using other Collections.

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]: 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 Mixer {

  private int n1;
  private int n2;
  static int total = 10;

  public Mixer(int n1, int n2) {
    this.n1 = n1;
    this.n2 = n2;
    total = total + n2;
    System.out.println("constructor: total: " + total);
  }

  public String toString() {
    return "n1 = " + n1 + " n2 = " + n2;
  }

  public static void updateTotal(int n1, int n2) {
    n1 = n1 + n2;
    n2 = n1 - n2;
    total = total + n1 + n2;
    System.out.println("method: n1 = " + n1 + " n2 = " + n2 +
        " total = " + total);
  }

  public static void mixContent(Mixer mA, Mixer mC) {
    int temp1 = mA.n1;
    int temp2 = mA.n2;

    mA.n1 = mC.n2;
    mA.n2 = mC.n1;
    mC.n1 = temp2;
    mC.n2 = temp1;
  }

  public static void main(String[] args) {
    Mixer mA = new Mixer(5, 10);
    Mixer mB = new Mixer(2, 4);
    Mixer mC = new Mixer(-3, 3);

    mixContent(mA, mC);
    mixContent(mC, mB);
    System.out.println("mA has " + mA);
    System.out.println("mB has " + mB);
    System.out.println("mC has " + mC);

    updateTotal(mA.n1, mB.n2);


    int n1 = 5;
    int n2 = 10;
    updateTotal(n1, n2);
    System.out.println("main: n1 = " + n1 + " n2 = " + n2 +
        " total = " + total);

    mixContent(mA, mC);
    System.out.println("mA has " + mA);

    mC = mA;
    mC.n1 = 10;
    System.out.println("mC has " + mC);
    System.out.println("mA has " + mA);

    mA = new Mixer(-10, -5);
    mC.n1 = n1;
    mC.n2 = n2;
    System.out.println("mA has " + mA);
    System.out.println("mC has " + mC);
  }
}
```

Question 2 [45 pts]: You were hired to write the software for a University system and you are responsible for working on the Student class. Answer the questions below.

```java
public class Student {

    private String fullName;
    private String id;

    public Student(String fullName, String id) {
        this.fullName = fullName;
        this.id = id;
    }

    public void printTopTwoResults(){
        // You must create this method.
    }

    public String toString() {
        return this.id + ": " + this.fullName;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getId() {
        return id;
    }
}
```

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

You must prevent the creation of course results with the following cases: 1) grades less than zero, or greater than 100; 2) credits less than or equal to zero; 3) Empty course names. When any of those cases occur, the message "`Invalid course result data.`" should, eventually, be shown to the user.

The string representation of the course result should be formatted as: `(<course name>, <credits>, <grade>)`

Q2.1 [10 pts]: Write the code for the course result class described above.

Request 2: The method `printTopTwoResults()` must **print the names of the two courses** with highest grades among the student's course results. Use the examples below to format your printed string.
In case there is a tie among the highest grades, you should keep the first occurrence. If the student has less than two course results registered, then the method should throw an exception with the message illustrated below.

- **Course results:** [(″DIT043″, 7.5, 100), (″DIT843″, 7.5, 99), (″DIT355″, 7.5, 80), (″DIT113″, 7.5, 99)];
  Output: ″Best grades: DIT043 and DIT843″
- **Course results:** [(″DIT043″, 7.5, 99), (″DIT843″, 7.5, 100), (″DIT355″, 7.5, 50), (″DIT113″, 7.5, 99)];
  Output: ″Best grades: DIT843 and DIT043″
- **Course results:** [(″DIT043″, 7.5, 99), (″DIT843″, 7.5, 100), (″DIT355″, 7.5, 100)];
  Output: ″Best grades: DIT843 and DIT355″
- **Course results:** [(″DIT043″, 7.5, 100)];
  Output (Exception): ″Cannot print. Student has less than two courses results.″

Q2.2 [12 pts]: Considering your modifications to the Student class in **Q2.1**, write the code for the method `printTopTwoResults()`.

Q2.3 [8 pts]: Explain one advantage and one disadvantage of creating or using immutable objects.

Q2.4 [8 pts]: The Student class above is well encapsulated. From an Object-oriented programming perspective, explain at least two benefits that encapsulation brings to the Student class.

Q2.5 [7 pts]: Your colleague Alex is writing a method that uses input and output streams to save the Student information into a file. She wants to store the student objects, including their respective course results. What modifications are needed in the Student class to enable saving those objects in a file?

Question 3 [Total: 40 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, polymorphism and overriding methods, answer the questions below:

```java
public class Membership {

  protected String userName;
  protected double storeCredit;
  protected int memberPoints;

  public Membership(String userName, double storeCredit) {
    this.storeCredit = storeCredit;
    this.userName = userName;
    this.memberPoints = 0;
  }

  public void addCredit(double addedCredit) {
    this.storeCredit = this.storeCredit + addedCredit;
  }

  public void addMemberPoints(int addedPoints) {
    this.memberPoints += addedPoints;
  }

  public void convertPoints(int numOfPoints) throws Exception {
    if(this.memberPoints < numOfPoints || numOfPoints < 50) {
      throw new Exception("Not enough points to convert.");
    }
    // Deduct membership points, convert to the corresponding credit value
    // and add that credit to the membership.
    this.memberPoints = this.memberPoints - numOfPoints;
    double convertedCredits = numOfPoints * 0.20;
    this.addCredit(convertedCredits);
  }

  public String toString(){
    String result = userName + ": Points = " + memberPoints + ". ";
    result += "Credits: " + storeCredit;
    return result;
  }

  // getters.
  public String getUsername() { return this.userName; }
  public double getStoreCredit() { return storeCredit; }
  public int  getMemberPoints() { return memberPoints; }
}
```

```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("Hedy Lamar", 500);
      m1.addCredit(2000);
      m1.addMemberPoints(100);
      m1.convertPoints(60);
      System.out.println(m1);

      Membership m2 = new SilverMembership("Mary Keller", 500);
      m2.addCredit(1000);
      m2.addMemberPoints(80);
      m2.convertPoints(90);
      System.out.println(m2);

      Membership m3 = new GoldMembership("Ada Lovelace", 500);
      m3.addCredit(1500);
      m3.addMemberPoints(400);
      m3.convertPoints(200);
      System.out.println(m3);

      Membership m4 = new GoldMembership("Dorothy Vaughan", 300);
      m4.addCredit(1000);
      m4.addMemberPoints(110);
      m4.convertPoints(70);
      System.out.println(m4);

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

Q3.1 [15 pts]: What is printed when executing the main method above?

Q3.2 [12 pts]: Explain three benefits provided by polymorphism in the code above. You must also indicate whether each benefit affects the design or the execution of the code above.

Q3.3 [8 pts]: Your colleagues Sam suggests to change the Membership class to an Abstract Class. Do you agree or disagree with Sam? Justify your answer.

Q3.4 [5 pts]: Your colleagues Alex and Charlie discuss the visibility of the attributes in Membership. Alex suggests changing the visibility of the attributes above from protected to private to improve encapsulation. Charlie states that changing the attributes to private will introduce bugs in the code. Do you agree with Charlie or Alex? Justify your answer.