Step 12 – Recap - exam

The task you were given in step 11 was great. Feel free to work with it until the exam. But if you want something more to work with towards the exam, then you can try last year's exam?

Warning: SO, this exercise is from a couple of years back. Much has changed. But it's a good bit of practice, so you can use it as that.

Last year they learned a little about UML. We have not had that this year. In the intro video, I therefore give a short review so that the task becomes easier to understand.

Goals for this step:

- I have an overall understanding of package structures in Java.

Relevant chapters in the book:

- Nothing new, but feel free to review chapter 3 (Visibility modifiers).

Here is the questionnaire for step 12 where you can tell how it went.

Task – exam from a previous year

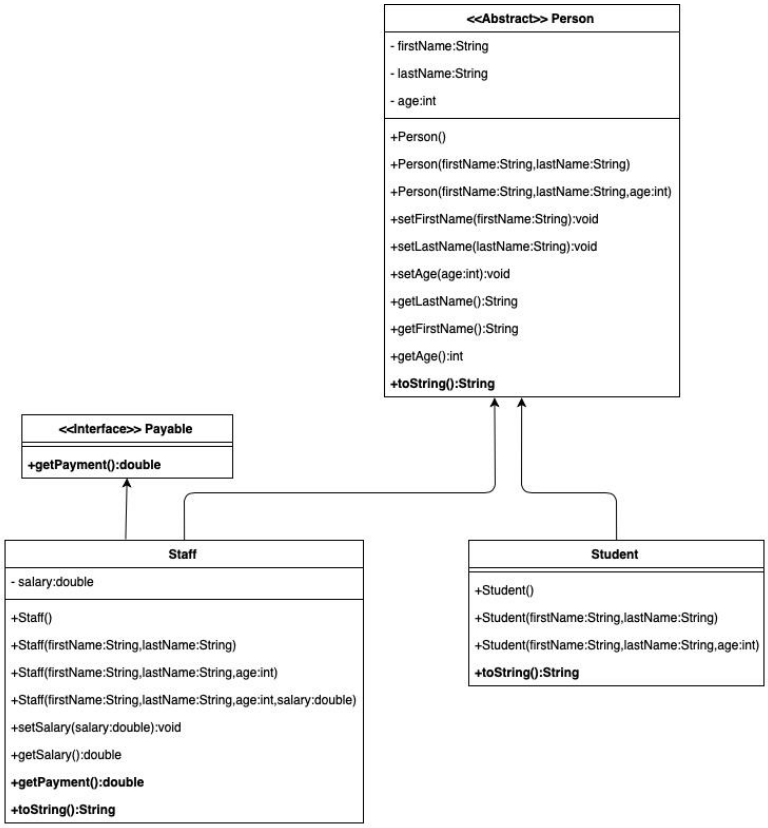
Important information:

The assignments in the exam cover a wide range of learning objectives in the course. It is therefore important that you try to answer all the questions. This means that you must avoid investing too much time in a simple sub-task. If you are very stuck, then you should leave it and go over to another. Rather return to the sub-task later when you have solved others. You can use pseudo-code to describe how you try to solve a sub-task, but it gives limited calculation in relation to working code.

Task 1

- Create an abstract class Person with three private attributes; firstName, lastName and age. Include getter and setter methods for each attribute. It also has an abstract method toString (). (3 points)
- Create a Payable interface that contains a double getPayment () method. (2 points)
- Create a Class and Student class using guesser and setter methods for all attributes (see UML diagram (Figure 1) below). (5 points)
- Implement an additional attribute salary for the Staff class that represents the monthly salary (1 point).
- Make the Staff class implement the Payable interface. (2 points)
- A call to the getPayment () method on a Staff object should return pay for an entire year. (2 points)
- Implement a twoString () method for Staff and Student. This method should return the class attributes as follows (3 points):
    - For the Student class, toString () must return the following string (String): Student: first name, last name, age.

- For the Staff class, toString () must return the following string (String): Staff: first name, last name, age, annual salary.



Figur 1: UML for første del av oppgaven

- Create a class called Address that represents the address of a person in the Person class. The class contains the following:
  - Three private instance variables: street (String), postcode (String), and country (String). (2 points).
  - A designer who initializes street, postcode and country with a given value. (2 points)
    - A toString () method that prints the address as follows: (2 points)

Street: gate

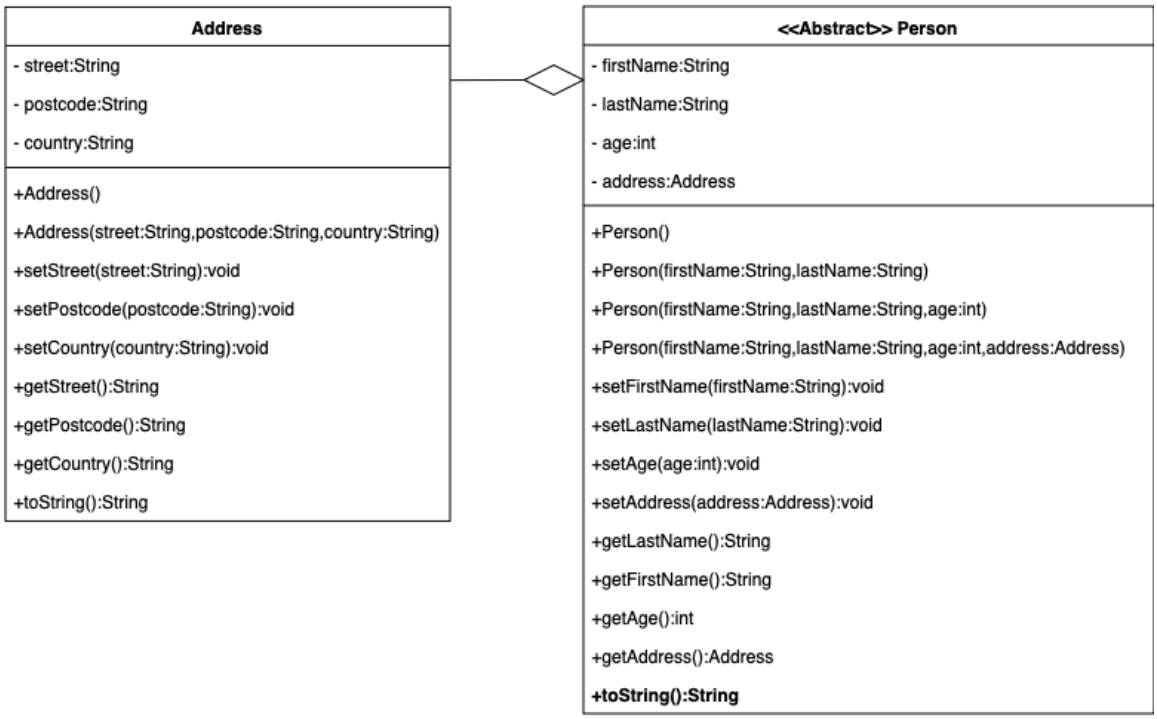
Postal code: postal code

Country: land

- Modify the Person class from earlier in the problem so that each person has an address. Include methods for retrieving and setting the address of a person (see UML diagram (Figure 2 below)). (5 points).
- Implement a test class named TestPerson. Write the beginning of the program through public static void main (). In the test class, create two references of type Person. One should refer to a created object of type Staff and the other to a created object of type Student. Only

the Staff object must have an address filled in. Both the created objects must have all values set through the use of a constructor. Both the Staff and Student objects should print the toString () method. Print the salary for the Staff object. Then change the salary to the Staff object to a higher value and print the new annual salary. (6 points).

Explain polymorphism in object-oriented programming. Use Exercise 1 in this exam to exemplify. (5 points).



Figur 2: UML for forholdet mellom Address og Person

Task 2 –

- Create a Class Main. The class' responsibility is to start the program for task 2. (2 points)
- Create a class Program that receives input from a user. A user should be able to enter names (and press Enter) several times until the user no longer wants to enter more names (8 points). When the user does not want to enter more names, the program should print information about:
    - All the names entered by the user. (4 points). If you manage to print them in alphabetical order, you get 4 extra points <feel free to try this, but we have not been in sorting strings this year…>.
    - Average number of letters in the names. (4 points)
    - The longest name (if several names were the same length, it is okay for only one of these to be printed). (4 points)

- The program should only allow names that are at least 2 letters long and the name should only contain letters (6 points). Tips & Warnings The java.lang.Character class has a static method boolean isLetter (char ch) ...

You can see an example of running such a program below:

```
Velkommen! Skriv inn navn separerte med <Enter>. Skriv "avslutt" for å avslutte.
Jesper
Cam2illa
Navn må være på minst to bokstaver og ikke inneholde tall. Prøv igjen!
Camilla
John
avslutt
Her er resultatet:
Camilla
Jesper
John
Gjennomsnittlig lengde på navnene:5
Lengste navn:Camilla

Process finished with exit code 0
```

Task 3 –

Below you see a silly program that does not do much of any use, but which you can use to show that you understand Java code.

Choose your own string of 10 random lowercase letters where no two letters are alike. Use your string as the value for the yourString variable in the program. Explain what the program does and what is printed when the value of yourString is your random string. (15 points).

```java
public class Main {
    public static void main(String[] args) {
        String yourString = "";//Your string value
        method1(yourString);
    }
    private static void method1(String s) {
        char[] chars = s.toCharArray();
        char[] earlyLetters = new char[]{'a', 'b', 'c', 'd'};
        char[] lateLetters = new char[]{'w', 'x', 'y', 'z'};
        for (int i = 0; i<chars.length; i++){
            if(charInArray(chars[i], earlyLetters)){
                System.out.println("early letter found");
            } else if (charInArray(chars[i], lateLetters)){
                try{
                    method2(chars[i]);
                } catch (RuntimeException re){
                    System.out.println(re.getMessage());
                }
            } else {
                System.out.println("Found " + chars[i]);
            }
        }
    }
    private static boolean charInArray(char c, char[] chars) {
        for (int i = 0; i<chars.length; i++){
            if (c==chars[i]) {
                return true;
            }
        }
        return false;
    }
    private static void method2(char c) {
        throw new RuntimeException("This is a " + c);
    }
}
```

Programmet for oppgave 3

Lykke til!