Eksamen PGR103 vår 2020
Hjemme-eksamen
(English version below)

Viktig informasjon:
Oppgavene i eksamen dekker et bredt spekter av læringsmålene i emnet. Der er derfor viktig at du forsøker å svare på alle oppgavene. Det betyr at du må unngå å investere for mye tid i en enkel del-oppgave. Hvis du står veldig fast, så bør du la den ligge og gå over til en annen. Gå heller tilbake til del-oppgaven senere når du har løst andre. Du *kan* benytte pseudo-kode for å beskrive hvordan du forsøker å løse en del-oppgave, men det gir begrenset med uttelling i forhold til fungerende kode.

Oppgave 1: 40 poeng

- Opprett en abstrakt klasse `Person` med tre private attributter; `firstName`, `lastName` og `age`. Inkluder getter- og setter-metoder for hvert attributt. Den har også en abstrakt metode `toString()`**.** (3 poeng)

- Opprett et grensesnitt `Payable` som inneholder en metode `double getPayment()`. (2 poeng)

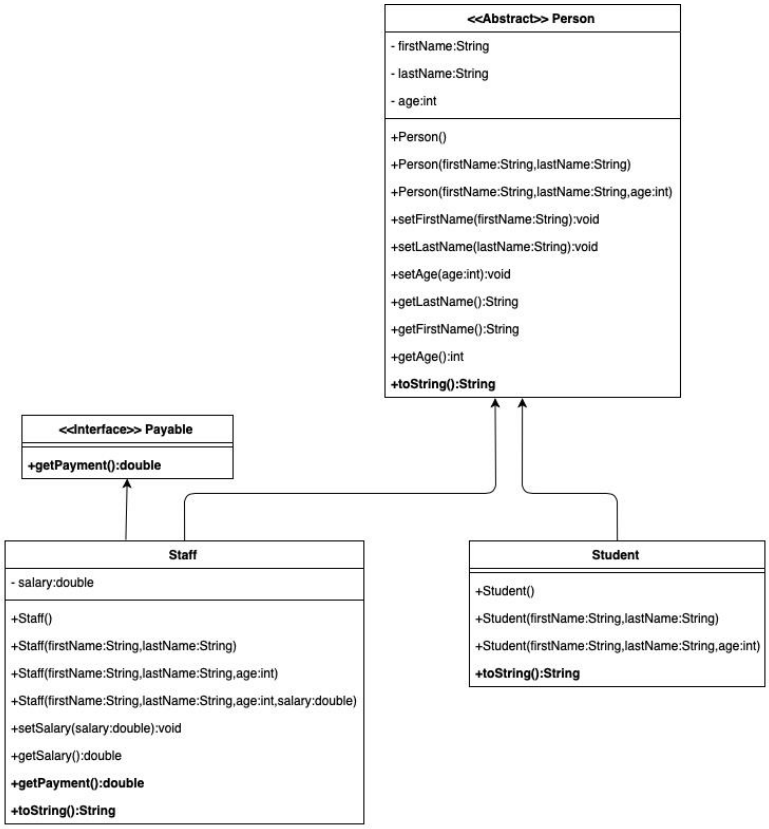
- Opprett en klasse `Staff` og `Student` med gettter- og setter-metoder for alle attributter (se UML-diagram (Figur 1) nedenfor). (5 poeng)

- Implementere et ekstra attributt `salary` for klassen `Staff` som representerer månedslønnen (1 poeng).

- Gjør slik at klassen `Staff` implementerer grensesnittet `Payable`. (2 poeng)

- Et kall på metoden `getPayment()` på et `Staff` objekt skal returnere lønn for et helt år. (2 poeng)

- Implementer en `toString()` metode for `Staff` og `Student`. Denne metoden skal returnere klassens attributter på følgende måte (3 poeng):
    - For `Student` klassen, skal `toString()` returnere følgende streng (`String`): Student: fornavn, etternavn, alder.
    - For `Staff` klassen, skal `toString()` returnere følgende streng (`String`): Staff: fornavn, etternavn, alder, årlig lønn.



Figur 1: UML for første del av oppgaven

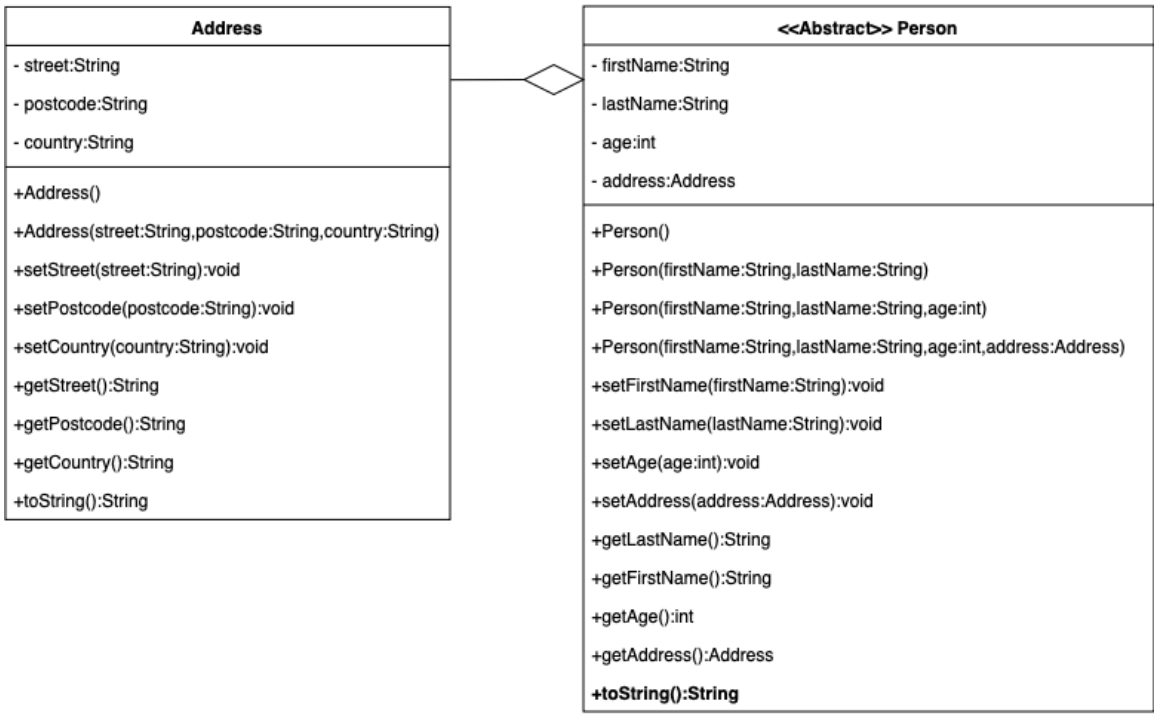
- Lag en klasse kalt `Address` som representerer adressen til en person av klassen `Person`. Klassen inneholder følgende:
  - Tre private instansvariabler (instance variables): `street` (`String`), `postcode` (`String`), og `country` (`String`). (2 poeng).
  - En konstruktør som initialiserer `street`, `postcode` og `country` med en gitt verdi. (2 poeng)
  - En `toString()` metode som skriver ut adressen på følgende måte: (2 poeng)
    Street: gate
    Postcode: postnummer
    Country: land

- Modifiser klassen `Person` fra tidligere i oppgaven slik at hver person har en adresse. Inkluder metoder for å hente og sette adressen til en person (se UML-diagram (Figur 2 nedenfor)). (5 poeng).

- Implementer en test klasse med navn `TestPerson`. Skriv begynnelsen av programmet gjennom `public static void main()`. I test klassen, opprett to referanser av type `Person`. Den ene skal referere til et opprettet objekt av type `Staff` og den andre til et opprettet objekt av type `Student`. Det er kun `Staff`-objektet som skal ha adresse fylt ut. Begge de opprettede objektene skal få alle verdier satt gjennom bruk av konstruktør. Både `Staff` og `Student` objektene skal

skrive ut `toString()` metoden. Skriv ut `salary` for `Staff` objektet. Forandre deretter `salary` til `Staff` objektet til en høyere verdi og skriv ut den nye årslønnen. (6 poeng).

- Forklar polymorfisme i objektorientert programmering. Bruk Oppgave 1 i denne eksamen til å eksemplifisere. (5 poeng).



Figur 2: UML for forholdet mellom Address og Person

Oppgave 2 – 32 poeng

- Lag en klasse `Main`. Klassens ansvar er å starte programmet for oppgave 2. (2 poeng)

- Lag en klasse `Program` som tar imot input fra en bruker. En bruker skal kunne skrive inn navn (og trykke Enter) flere ganger inntil brukeren ikke lenger ønsker å skrive inn flere navn (8 poeng). Når brukeren ikke ønsker å skrive inn flere navn skal programmet skrive ut informasjon om:
    - Alle navnene som brukeren skrev inn. (4 poeng). Hvis du klarer å skrive de ut i alfabetisk rekkefølge får du 4 ekstrapoeng.
    - Gjennomsnittlig antall bokstaver i navnene. (4 poeng)
    - Det lengste navnet (hvis flere navn var like lange er det greit at kun ett av disse skrives ut). (4 poeng)

- Programmet skal kun tillate navn som er minimum 2 bokstaver langt og navnet skal bare inneholde bokstaver (6 poeng). Hint: Klassen `java.lang.Character` har en statisk metode `boolean isLetter(char ch)`...

Du kan se et eksempel på kjøring av et slikt program nedenfor:

```
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
```

Oppgave 3 – 15 poeng

Nedenfor ser du et tullete program som ikke gjør mye hensiktsmessig, men som du kan benytte til å vise at du forstår Java-kode.

Velg deg en egen streng på 10 tilfeldige små bokstaver der ingen bokstaver er like. Strengen din skal du bruke som verdi for variabelen `yourString` i programmet. Forklar hva programmet gjør, og hva som blir skrevet ut når verdien av `yourString` er din tilfeldige streng. (15 poeng).

```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!

English version

Important information:
The exams 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 have to avoid investing too much time on a simple task. If you are stuck, you should leave it and move on to another task. Return to the task later when you have solved others. You *may* use pseudo-code to describe how you try to solve a task, but it will give you a limited score compared to working code.

Assignment 1: 40 points

- Create an abstract class `Person` with three private attributes; `firstName`, `lastName` and `age`. Include getter and set methods for each attribute. It also has an abstract method `toString()`. (3 points)

- Create a `Payable` interface that contains a method `double getPayment()`. (2 points)

- Create a `Staff` and `Student` class with getter and setter methods for all attributes (see UML chart (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 a full year's salary. (2 points)

- Implement a `toString()` method for `Staff` and `Student`. This method should return the class attributes as follows (3 points):
    - For the `Student` class, `toString()` should return the following string (`String`): Student: first name, last name, age.
    - For the `Staff` class, `toString()` should return the following string (`String`): Staff: first name, last name, age, annual salary. (3 points)

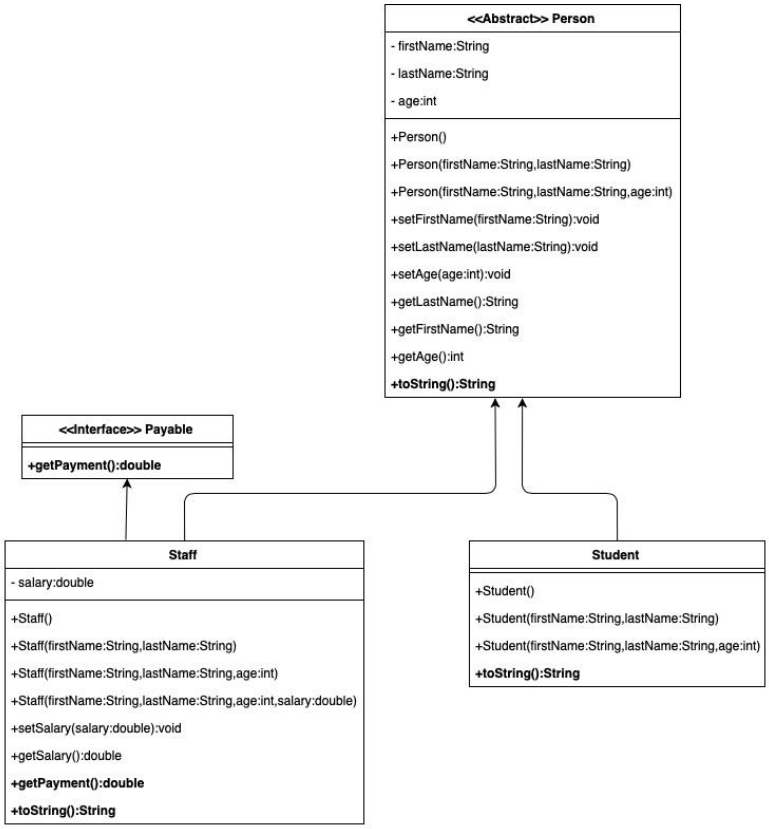


Figure 1: UML for the first part of the assignment

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

  Street: street
  ZIP code: ZIP code
  Country: country

- Modify the `Person` class from earlier in the assignment so that each person has an address. Include methods for retrieving and setting a person's address (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 should have an address filled in. Both created objects should have all

values set by using a constructor. Both the `Staff` and `Student` objects should print the result from the `toString()` method. Print the `salary` for the `Staff` object. Then change the salary of the `Staff` object to a higher value and print the new annual salary. (6 points).

- Explain polymorphism within object-oriented programming. Use assignment 1 in this exam to examplify. (5 points)

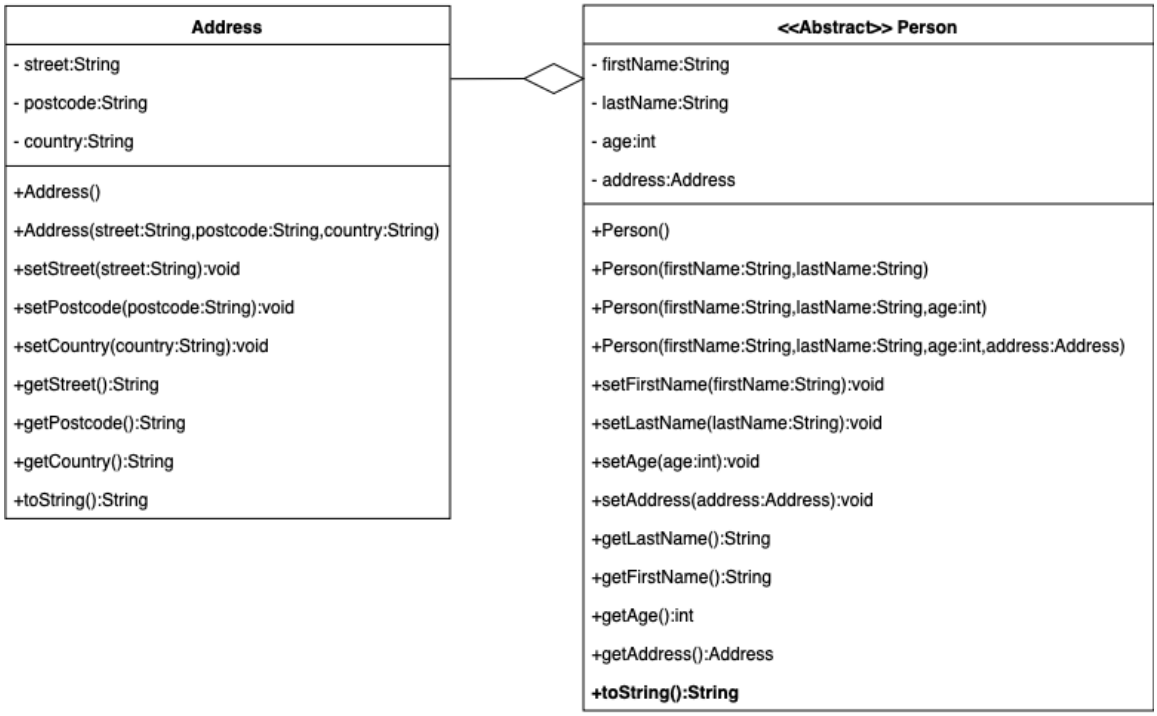


Figure 2: UML describing the relationship between Address and Person

Assignment 2 – 32 points

- Create a class `Main`. The class's responsibility is to start the program in assignment 2. (2 points)

- Create a class `Program` that receives input from a user. A user should be able to enter names (and press Enter) repeatedly until the user no longer wants to enter names (8 points). When the user does not want to enter more names, the program should print information on:
  - All the names that the user entered. (4 points). If you manage to print them in alphabetical order, you get 4 extra points.
  - Average number of letters in the names. (4 points)

    - The longest name (if several names had the same length, it is OK that only one of these is printed). (4 points)

- The program should only allow names that are at least 2 letters long and the name should contain only letters (6 points). Hint: The class `java.lang.Character` 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
```

Assignment 3 – 15 points

Below is a silly program that does not do much good, but which you can use to show that you are able to understand Java code.

Choose your own string of 10 random lowercase letters where no letters are the same. Use your string as value for the `yourString` variable in the program. Explain what the program does and what gets printed when the value of `yourString` is your random string.

```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);
    }
}
```

The program for assignment 3

Good luck!