Trinn 12 – Repetisjon - eksamen

Siste trinn! Oppgaven dere fikk i trinn 11 var stor. Jobb gjerne med den fram mot eksamen. Men hvis dere vil ha noe mer å jobbe med fram mot eksamen, så kan dere jo forsøke dere på fjorårets eksamen? **Advarsel**: Jeg synes ikke fjorårets eksamen var så god når det gjelder å teste måloppnåelse. Det forklarer jeg nærmere i ukens introvideo. Ta derfor gjerne en titt på den. Årets eksamen kommer derfor ikke til å likne så mye på fjorårets. Men mengdetrening er bra uansett!

I fjor lærte de litt om UML. Det har vi ikke hatt i år. I introvideoen gir jeg derfor en kort gjennomgang slik at oppgaven blir enklere å forstå.

Mål for dette trinnet:

- Jeg har en overordnet forståelse av pakkestrukturer i Java.

Relevante kapitler i pensumboka:

- Ingen noen nye, men hvis du ikke har lest kapittel 3 før (Visibility modifiers), så kan du jo gjøre det nå😊

Her er spørreskjemaet for trinn 12 der du kan fortelle hvordan det gikk.

Oppgaven – fjorårets eksamen

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)

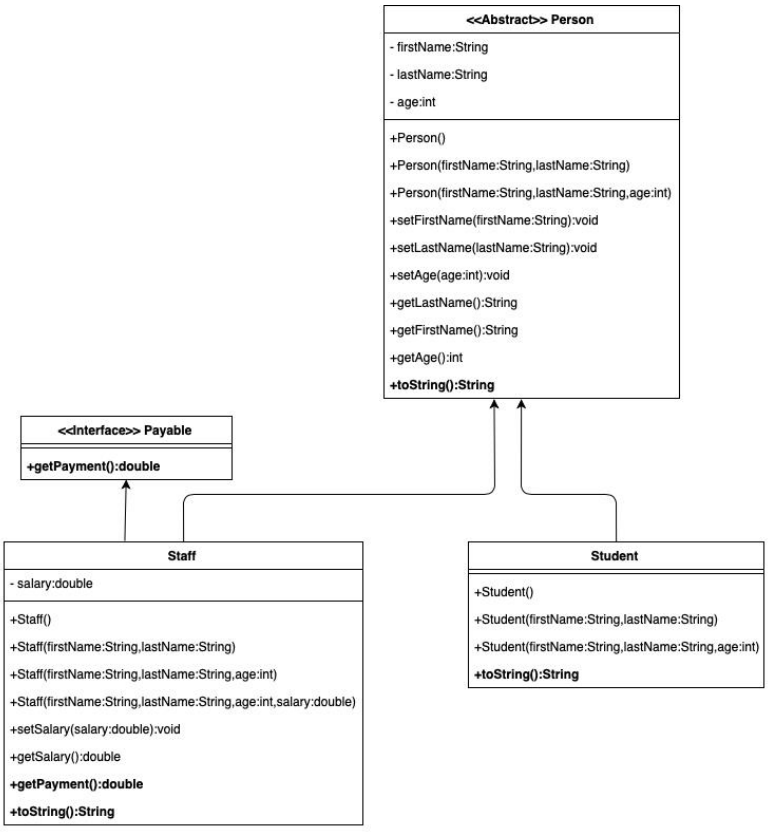
- Opprett en klasse `Staff` og `Student` med gettter- og setter-metoder for alle attributter (se UML-diagram (Figur 1) nedenfor). (5 poeng)

- 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

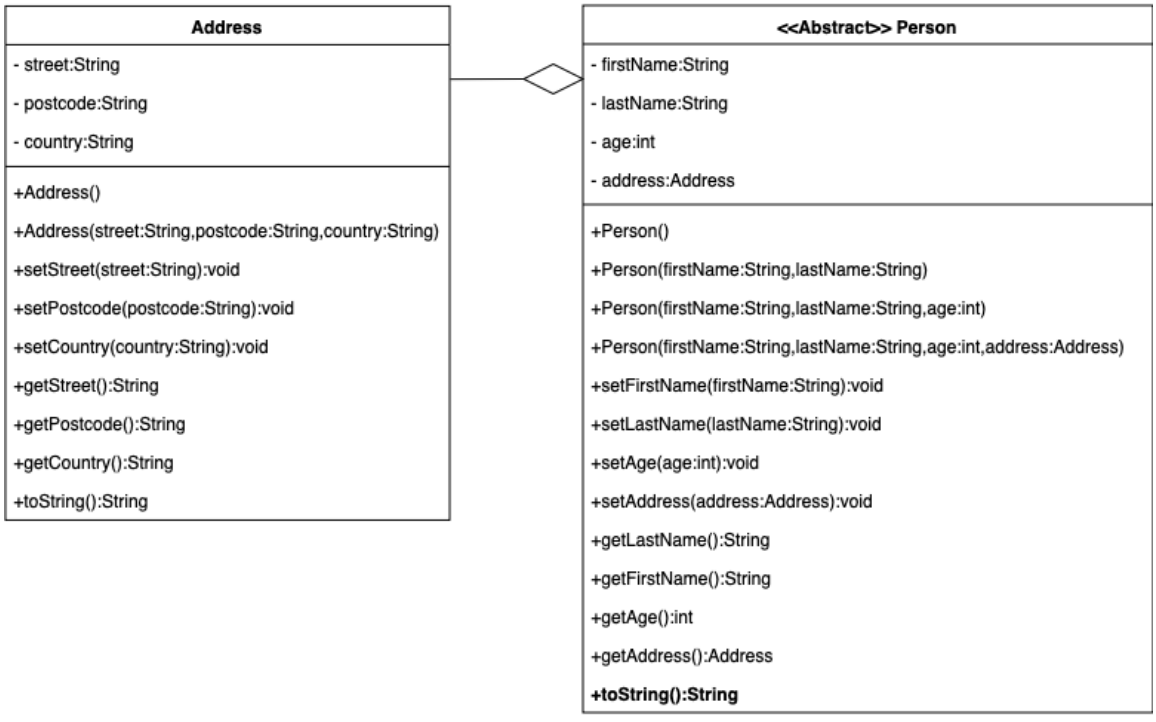
  Postcode: postnummer

  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 <forsøk deg gjerne på denne, men vi har ikke vært inne på sortering av strenger i år…>.
    - 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).

```
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!