Oblig 4 (INF1000 - Høst 2011)

Filmregister

[*Nye presiseringer er markert med **grønn tekst**.* Oppdaterte datafiler kommer...]

> **Mål:** *I denne oppgaven vil du lære hvordan man kan behandle en større mengde data på en objektorientert måte, og få trening i hele pensum. Oppgaven kombinerer alle programmerings-elementer vi har sett tidligere i kurset, i tillegg til HashMap.*

Leveringsfrist

Fredag 11. nov kl. 16.00. Leveres i Joly. ***Viktig!** Det blir svært begrensede muligheter for å levere nye forsøk i Oblig 4 eller å få mye utsetttelse. Kandidatlista til eksamen bør være klar 14 dager før eksamen, dvs. 18. nov (bare én uke etter den vanlige fristen). Alle som skal ta eksamen bør få obligen sin godkjent før dette.* ***Hvis det du leverer innen fristen ikke er ferdig må du fortsette å jobbe med obligen og levere ferdig utgave så fort som mulig.***

Leveringskrav

Som for tidligere oppgaver gjelder det at Oblig 4 skal leveres individuelt - to studenter kan altså **ikke** levere felles eller kopier av samme løsning. Kravene til innleverte oppgaver ved Ifi finner du her: www.mn.uio.no/ifi/studier/admin/obliger/. Retningslinjer for obligatoriske oppgaver finner du her: www.mn.uio.no/ifi/studier/admin/obliger/oblig-retningslinjer.html

Løsningen skal være objektorientert. Det betyr at programmet skal bruke objekter av klasser du selv definerer, og interaksjon mellom disse for å løse de forskjellige deloppgavene. Besvarelsen må bruke minst 3 klasser og 2 HashMap-er, men du kan godt utvide dette med flere klasser, HashMap-er, og andre datastrukturer etter ønske. Det skal leveres to ting:

- **Oblig4.java:** Denne filen skal inneholde det ferdige Java-programmet ditt (all Java-kode skal ligge i én fil). Før du leverer skal du *lage en kommentar øverst* i .java-filen der du skriver litt info om besvarelsen din, f.eks. hvilken deloppgave du har spesielle spørsmål til, eller om du fullførte alle deloppgaver og ekstraoppgaver. Skriv også hvilken leveringsmåte du valgte for UML-klassediagrammet (se neste punkt). Husk å kommentere koden din (se tips fra INF1100).
- **UML-klassediagram:** Du skal levere et UML-*klassediagram* over systemet ditt. Diagrammet skal inneholde navnene på klassene, og de viktigste koblingene mellom disse (særlig pekerne som er deklarert øverst i klassene, før metodene), illustrert som forhold (linjer) mellom klassene. Gi navn på forholdene, og angi antall på begge sider av disse. Du skal velge en av følgende 3 måter å levere diagrammet på. Skriv hvilken du valgte øverst i Oblig4.java: **(a)** elektronisk innlevert *sammen med din Oblig4.java*, **(b)** på *papir*, eller **(c)** via *mail*.

  Se hint 2 nedenfor for tips om UML-klassediagrammer. For elektronisk levering (a) bruk en av disse filtypene: .pdf, .jpg, .png, .gif, (eller .txt), og pass på å levere den i samme levering som programfilen (.java-filen) men lagt inn som tilleggsfil rett etter at du har lagt inn .java-filen i Joly (se hint 10). For å levere på papir går du på gruppetimen din og leverer personlig til din gruppelærer.

Hvis du har spørsmål om obligen kan du skrive disse på kurs-bloggen. Der vil det bli lagt ut svar og evt. andre tips og presiseringer til obligen. Du kan også få hjelp på gruppen din, under orakeltjenesten (8. - 10. nov), på ekstragruppen (rom Modula, fredager kl. 14.15 - 18), eller fra Studieoraklene (rom 3102 i 3. etg. OJD). Hvis du fortsatt savner opplysninger for noen deloppgaver kan du legge til grunn egne forutsetninger så lenge de ikke bryter med oppgavens "ånd". Skriv i så fall disse i kommentaren din øverst i programmet.

Oppgave

I denne obligen skal du lage et kommandostyrt system som behandler informasjon om filmer. Programmet skal lese inn data fra følgende to datafiler, og gi brukeren mulighet til å utføre kommandoer som finner og viser frem forskjellig type informasjon om filmene. Programmet trenger ikke skrive til fil eller endre datafilene, og *du skal ikke levere disse filene.*
***NB!** Datafilene vil bli oppdatert. Utgavene som er lagt ut kan brukes, men de mangler filmer fra det siste året.*

- **filmdata.txt**: Tekstbasert datafil med informasjon om filmer. (Filstørrelse: 870 KB)
- **persondata.txt**: Tekstbasert datafil med informasjon om personer (regissører og skuespillere). (Filstørrelse: 620 KB)

Filen filmdata.txt

Denne filen inneholder informasjon om filmer, med én linje for hver film. Hver linje har 6 eller flere felt, som er adskilt vha. et tabulator-tegn (dette skilletegnet angis som "`\t`" i Java):

```
kode  tittel  år  regissør  skuespillere;...  sjangre  [evt.tillegg]...
```

Noen filmer mangler data i felt 3-6 (regissør, skuespillere, eller sjangre), da står det "-" i feltet.

- **kode** er en unik kode for hver film, og består av tre bokstaver (én stor og to små bokstaver) etterfulgt av et tall (1 og oppover).
- **tittel** og **år** er navn på filmen og året den kom.
- **regissør** er en kode som angir regissøren til filmen, og består av tre små bokstaver etterfulgt av et tall (som går fra 1 og oppover). Noen filmer har flere regissører, da står kodene til disse adskilt med semikolon (";").
- **skuespillere** er en rekke med koder som angir skuespillere, disse angis på samme måte som regissør. Oftest er det mer enn én skuespiller, og da står kodene til disse adskilt med semikolon (";").
- **sjangre** er en sammenhengende tekststreng som angir sjangre for filmen, i form av enkelt-bokstav-koder eller enkelt-sifre. Tallsifrene angir om filmen forekommer i forskjellige lister. Vi skal bare bruke noen få av disse kodene i oppgaven, men her er alle:

  a=action A=animation b=biographical c=comedy C=children d=drama D=documentary e=epic E=adventure f=fantasy F=family g=computer_animation(datagrafikk) h=horror H=superhero i=history k=crime m=musical M=music n=film_noir o=Eastern(Østen) r=romantic_comedy R=romance s=science_fiction S=sports t=thriller w=war W=western x=disaster

  1=Står i listen Films considered the greatest ever fra Wikipedia
  2=Står i listen List of Academy Award-winning films (filmer som fikk Oscar)
  3=Står i listen Palme d'Or (filmer som vant i Cannes)
  4=Står i listen Sight & Sound (liste omtalt i link 1= ovenfor).
  5=Står i listen AFI's 100 Years... 100 Movies (10th Anniversary Edition)

- [***evt. tillegg***]: For noen av filmene er det mer enn 6 felt i linjen. Vi skal bare bruke `s=Serienavn` i felt 7, dette står på filmer som er del av en serie, f.eks. i James Bond-filmene står det "s=James Bond". [De andre tilleggsfelt trengs ikke i deloppgavene, men du kan bruke de hvis du ønsker det, de er tatt fra TMDb og kan inneholde: **r=*rating*** er en karakter for filmen, fra 0 til 10.0. **i=*imdb*** er IMDb-id til filmen. **c=*land1*;*land2*;...** er landene filmen kommer fra. **o=*original_name*** er navnet til filmen på originalspråket.]

Filen persondata.txt

Denne filen har data på to felt, adskilt med tabulator-tegn: **kode**, og fullt **navn** til en person. Her står navnene til alle regissører og skuespillere som var angitt i den andre datafilen ved hjelp av kode.

Tips: Disse kodene hopper ikke over noen tall, f.eks. hvis det finnes ole9, så vil også ole1 til ole8 finnes. Dette kan du bruke til å kunne søke raskere etter personer (eller filmkoder, som følger samme prinsipp) når bruker taster 3 bokstaver og ønsker en liste over matchende personer eller filmer.

Kommandoene

Oppgaven din blir å lage et program som kan lese inn disse to filene, og svare på følgende typer kommandoer som bruker kan gi. Du kan velge hvordan du vil sette opp menyen, men det anbefales følgende meny:

```
Eksempler på kommandoer du kan taste inn:
m      = Vis denne menyen
s      = Vis statistikk
Aaa1   = Vis info om en film
Aaa    = Finn film
tom1   = Vis info om en person
tom    = Finn person
2000s  = Vis info om et tiår
2010   = [ Ekstraoppgave: Vis info om et år ]
:a     = [ Ekstraoppgave: Vis info om en sjanger ]
q      = Avslutt

Kommando ('m' = meny): _
```

Idéen her er at bruker kan taste inn en hvilken som helst av disse typer kommandoer, og programmet finner hva slags kommando ble gitt ut fra antall tegn i kommandoen og om disse tegn er sifre, bokstaver, eller andre tegn.

1. **Vis statistikk:** Skal skrive ut totalt antall filmer, og antall filmer i hvert av tiårene 1990-1999, 2000-2009, og 2010-2019.
2. **Vis info om en film:** Skal la brukeren angi en film, og skriver deretter ut følgende informasjon om filmen: tittel, år, fullt navn til regissør, og fullt navn til skuespillerne. Du kan også gi enda mer informasjon om filmen, f.eks. sjangre eller evt. serie som filmen tilhører. Til denne kommandoen skal brukeren angi film ved koden til en av filmene (f.eks. "`Ava1`" for Avatar).
3. **Finn film:** Skriver ut informasjon om en film på samme måte som kommendoen over (**Vis info om en film**), men lar brukeren søke etter en film:
   - **(a)** *søk:* Hvis bruker bare tastet inn tre bokstaver, og den første er en stor bokstav, så skal programmet skrive ut en liste med filmene hvor film-koden (og dermed også tittelen) begynner med disse bokstavene, og brukeren skal kunne velge ønsket film fra listen. Du kan bruke de unike film-kodene i datafilen til dette, disse ser bort fra "`The `" og "`A `" i begynnelsen av filmnavn.
   - **(b) [*Ekstraoppgave: navn*]:** Hvis du ønsker det kan du også implementere andre måter å angi filmer på, f.eks. med full tittel til filmen, eller med noen bokstaver fra begynnelsen av filmtittelen (f.eks. "Avat" eller "Avata" for Avatar).
4. **Vis info om en person,** og **Finn person:** Skal fungere omtrent som kommandoene ovenfor, med de samme måtene å angi ønsket person på (men med små bokstaver i stedet). Informasjonen som vises skal inneholde filmene som personen regisserte og de som hun spilte i, med tittel og år.
5. **Vis info om et tiår:** Hvis brukeren taster inn fire tallsifre etterfulgt av bokstaven s (f.eks. 1990s), og siste tallsiffer er 0, så skal programmet vise følgende: **(a)** «*Beste filmer*»: Filmene som står i 2 eller flere av filmlistene 1-5 i det tiåret. Filmlistene er angitt med kodene 1-5 i felt nr. 6 for hver film, f.eks. kode «2» står på filmer som har fått Oscar. **(b) [*Ekstraoppgave:* «*Mest aktiv regissør*»:]** Skriv ut regissøren som lagde flest filmer det tiåret.
6. **[*Ekstraoppgave:* Vis info om et år:]** Hvis brukeren taster inn et årstall mellom 1900 og 2010 så skal programmet vise følgende to ting om det året: **(a)** «*Årets film*»: Hvilken film står i flest æreslister (1-5) det året; og **(b)** «*Årets sjanger*» blant comedy, fantasy, horror, eller science-fiction: her skal programmet finne hvilken av disse 4 sjangrene forekommer i flest filmer det året, basert på sjanger-kodene c, f, h, s. Skriv også ut antall filmer resultatet er baserert på.
7. **[*Ekstraoppgave:* Vis info om en sjanger:]** Taster bruker kolon etterfulgt av en av bokstavene a, E, H, x (som står for a=action, E=eventyrfilm, H=superhero, x=disaster), så skal programmet vise følgende to ting om valgt sjanger: **(a)** *Skuespilleren* som spilte i flest filmer i sjangeren; og **(b)** Navnene på *filmseriene* som har minst en film innen sjangeren.
8. **[*Ekstra-ekstraoppgave:* Filmliste:]** ***NB!*** *Dette er en oppgave som gir deg mulighet til å lage tilleggsfunksjonalitet som gjør programmet mer nyttig for eget bruk - dersom du ønsker for eksempel å holde rede på hvilke filmer du selv har, eller hvilke filmer du har sett, eller dine vurderinger av filmer du har sett.* Legg til mulighet for brukeren til å lage en liste av filmer. Du velger hvilke muligheter programmet gir brukeren, og hvordan du programmerer det, men legg til et menyvalg som forklarer funksjonalitet og fremgangsmåte til bruker. Listen som lages bør lagres på en egen datafil, f.eks. når programmet avsluttes, og hentes inn igjen når programmet starter.

Hint

Disse hint er ment for de som ønsker litt ekstra-hjelp. Du trenger ikke lese dette avsnittet for å løse obligen, og det anbefales sterkt at du lager ditt eget forslag til datastruktur og gjerne skisserer dette i UML *før* du ser på forslaget til programskall nedenfor. Da får du samtidig gjort deg godt kjent med oppgaveteksten. Dette vil du ha mye igjen for senere, selv om du skulle ende med å bruke den ferdiglagede koden. Du kan også spørre om og finne flere hint i bloggen. Det kan være lurt å løse litt enklere oppgaver før du går løs på deloppgavene i denne obligen, særlig Ukeoppgaver 9 om HashMap (som er hovedtemaet i obligen). Oppgavesettet har løsningsforslag.

1. **Programskall:** Her er et eksempel på et mulig skall for programmet, men du vil lære mer med obligen hvis du setter opp programstrukturen din på egen hånd *før du studerer dette eksemplet!*

```
/* Skriv en kommentar om din besvarelse her.
 * ...
 *
 * Leveringsmåte for UML-klassediagram: ...
 */
import easyIO.*;
import java.util.HashMap;

class Oblig4 {
    public static void main(String[] args) {
        new Filmregister().ordrelokke();
    }
}

class Person {
    String kode;
    String navn;

    String filmerRegissert; // Filmkoder adskilt f.eks. med semikolon.
    String filmerSpilt; // Filmkoder adskilt f.eks. med semikolon.
```

```
    // filmerRegissert og filmerSpilt kan også overføres til arrayer av
    // Film[]-pekere når alle filmer er lagt inn; eller du kan lagre de
    // i små HashMap-er.  Se hint 3 for flere tips.

    // Evt. metoder for å behandle en person.
}

class Film {
    // Variabler for dataene som gjelder for en film.
    // ...

    // Evt. metoder for å behandle en film.
}

class TiAar {
    // Variabler for dataene som gjelder for et tiår.
    // ...

    // Evt. metoder for å behandle et tiår.
}

class Filmregister {
    In tast = new In();
    Out skjerm = new Out();

    HashMap<String,Person> personer = new HashMap<String,Person>();
    HashMap<String,Film> filmer = new HashMap<String,Film>();
    TiAar[] tiaar = new TiAar[14]; // [0]=1880-1889, ..., [13]=2010-2019

    /**
      * Konstruktør: Leser datafilene, lagrer innholdet i objekter av
      * klassene Person, Film, (og evt. TiAar), og putter Person- og
      * Film-objektene i HashMap-ene «personer» og «filmer».
      */
    Filmregister() {
        // Leser datafilen "persondata.txt":
        In fil = new In("persondata.txt");
        fil.inLine(); // Hopp over første linje, som ikke har data.
        while (! fil.endOfFile()) {
            // Les en linje fra datafilen:
            String kode = fil.inWord();
            String navn = fil.inLine();

            // Opprett Person-objekt, og lagre det i HashMap-en «personer».
            // ...

            //skjerm.out(kode.charAt(0)); // Testutskrift.
        }
        fil.close();

        // Leser datafilen "filmdata.txt":
        fil = new In("filmdata.txt");
        fil.inLine(); // Hopp over første linje, som ikke har data.
        while (! fil.endOfFile()) {
            // Følgende setning leser inn en hel linje fra datafilen og
            // oppretter en array med de forskjellige feltene i linjen.
            // felt[0] vil da inneholde filmkoden, felt[1] tittel, osv.
            String linje = fil.inLine();
            String[] felt = linje.split("\t");

            // Opprett Film-objekt med de innleste felt-datane, og evt.
            // TiAar-objekt hvis det ikke finnes allerede, og lagre
            // filmobjektet i HashMap-en «filmer».
            // ...

            // For å teste om det var flere enn 6 felt i linjen kan
            // du bruke if-setningen: if (felt.length > 6).

            skjerm.out(felt[0].charAt(0)); // Testutskrift.
        }
        fil.close();
    }

    /** Ordreløkke: */
    void ordrelokke() {
        String ordre = ""; // Kommandoen som bruker taster inn.
        char char0 = 'm'; // Første tegn i kommandoen.

        visMeny();

        while (! ordre.equals("q")) {

            // Skriv ut ledetekst og les inn en ordre fra tastatur.
            skjerm.out("Kommando ('m' = meny): ");
            ordre = tast.readLine();

            int ordreLengde = ordre.trim().length();
            if (ordreLengde > 0) {
                char0 = ordre.charAt(0);
            }

            if (ordreLengde == 1 && char0 == 'm') {
                visMeny();

            } else if (ordreLengde >= 3 && char0 >= 'A' && char0 <= 'Z') {
                visInfoOmFilm(ordre);

            } // else if ...osv...

            // Skriv en else-if gren for hver ordretype.
        }
    }

    void visMeny() {
        // Legg til de andre linjene av menyen her.
        skjerm.outln("\nMeny:");
        skjerm.outln("m      = Vis meny");
        skjerm.outln("q      = Avslutt");
    }

    void visInfoOmFilm(String kode) {
        // Vis info om filmen som har angitt kode (inn-parameter).
    }

    // Lag en metode for hver ordre her.  Disse metodene kan
    // kalle på metoder i de andre klassene.
}
```

2. **UML-klassediagram:** Se eksempel på side 236 i læreboka. Du kan bruke nesten et hvilket som helst tegneprogram for å lage diagrammet på datamaskin (hvis du vil levere det via innleveringssystemet eller mail), eller du kan scanne inn en papir-tegning. Det er scanner ved ekspedisjonen i 4. etg. OJD og på

Abel-stua (kjelleren i Matematikk-bygningen). Hvis du velger elektronisk levering, skal du bruke en av disse filtypene: .pdf, .jpg, .png, .gif, (eller .txt). UML-klassediagrammer ble gjennomgått i Ukeoppgaver 7.

3. **String contains og split:** For å teste om en film tilhører en sjanger kan du bruke den forhåndsdefinerte metoden «contains» for tekster, f.eks. hvis «sjangre» er en String-variabel med det som sto i felt 6 for filmen «film», så vil følgende if-setning teste om filmen er en action-film:

```
if (film.sjangre.contains("a")) { // ...
```

Tilsvarende kan du teste for koder på mer enn ett tegn, men det fungerer best hvis du legger til et skilletegn på slutten av teksten man skal lete i, f.eks. hvis du har valgt å lagre skuespillerlista til en film i en String-variabel «stars», og plusset på skilletegn bak (`stars = stars + ";";`), så kan du teste om Sandra Bullock spiller i filmen slik:

```
if (film.stars.contains("san1;")) { // ...
```

Dette gjelder hvis du lagrer skuespillerlisten til en film i en String-variabel. To andre måter å lagre slike små lister på er som en array (som kan opprettes vha. `stars.split(";")`) eller i en liten HashMap. Du kan se et eksempel på split i koden ovenfor. *NB!* Tekst-verdien du bruker split på skal ikke inneholde skilletegnet helt foran, så hvis teksten har ";" som første tegn bør dette fjernes (f.eks. vha. `substring`) før man utfører `split(";")`, hvis ikke kan man få en tom streng som første resultat av `split`.

4. **keySet og substring:** Du vil få bruk for metodene for manipulasjon av HashMap-er og String-er, disse kan du lese mer om i kapittel 6 og 9 i læreboka, les bl.a. om substring, indexOf, startsWith, parseInt, og split på side 105-114 om tekster; og om put, get, keySet, values, size, og containsKey på side 181-192 om HashMap-er.

5. **Store og små bokstaver:** Det er mange måter å teste om et tegn i ordren som brukeren tastet inn er stor eller liten bokstav. Her er tre mulige måter, velg en av dem:

```
if (tegn >= 'A' && tegn <= 'Z') { // ...
if (Character.isUpperCase(tegn)) { // ...
if (ordre.toLowerCase().equals(ordre)) { // ...
```

Den første måten er også vist i siste else-if i koden ovenfor og går ut på å teste om et gitt tegn er mellom `'A'` og `'Z'`, i så fall vet vi at det er en stor bokstav. Bruk `'a'` og `'z'` for å teste for små bokstaver. Neste if-setning viser bruk av den forhåndsdefinerte metoden `isUpperCase(tegn)` til klassen `Character`, denne klassen er alltid tilgjengelig i Java, på samme måte som `String`. Bruk `isLowerCase(tegn)` for å teste for små bokstaver. Siste alternativ viser hvordan du kan teste om alle bokstaver i en ordre er små bokstaver.

6. **Tall eller bokstav:** Det er også mange måter å finne ut om et gitt tegn er et tallsiffer:

```
if (tegn >= '0' && tegn <= '9') { // ...
if (Character.isDigit(tegn)) { // ...
if (ordre.matches("[0-9]+")) { // ...
```

Siste alternativet tester om hele ordren består av bare tallsifre (`"[0-9]"` matcher et hvilket som helst tall, og "+"-tegnet betyr «en eller flere forekomster»). Det finns også `Character.isLetter(`*tegn*`)` for å teste om et tegn er en bokstav.

7. **Er ordren et tiår?** (*NB: Følgende er ikke pensum*). Du kan også bruke følgende til å teste om bruker ga ordren om tiår:

```
if (ordre.matches("[0-9][0-9][0-9]0s")) { // ...
```

Spesielt interesserte kan lese mer om dette i Java API-en under String.matches(*regex*) og regular expression (se også side 114 i læreboka).

8. **Lesing av 6 eller flere felt:** Linjene i datafilen "`filmdata.txt`" kan inneholde 6 eller flere felt adskilt med tabulator-tegnet. En enkel måte å lese datafilen på som tar høyde for dette er vist i koden ovenfor der det står `split("\t")`. Og for å sjekke om 7. felt er serienavn kan du bruke `if (felt[6].startsWith("s="))`.

9. **String til int og tilbake:** Man kan konvertere fra **String til int** ved hjelp av `parseInt` slik:

```
int tall = Integer.parseInt(tekstVerdi);
```

Her skal *tekstVerdi* være en String som bare inneholder tallsifre. Bruk *`tekstVerdi`*`.substring(`*`f_o_m, til`*`)` for å plukke ut ønsket del-streng hvis tallsifrene er blandet med andre tegn i *tekstVerdi*. Du kan lese mer om `substring` på side 105-106 i læreboka, eller i Java API-en under String.

Det omvendte, konvertering fra **int til String**, kan ordnes ved å plusse på den tomme strengen:

```
String tekst = "" + intVerdi;
```

10. **Info om innlevering:** Joly krever at første fil som legges inn i en innlevering er en .java-fil, derfor kan du ikke levere UML-diagrammet alene i Joly. Hvis du valgte leveringsmåte (a) for diagrammet (dvs. sammen med .java-filen), så legger du inn begge filene i samme levering i Joly: Først legger du inn Oblig4.java, og så legger du inn UML-diagrammet ditt som tilleggsfil.

Du kan *diskutere* med andre studenter hvordan dere skal løse oppgaven, men det er ikke lov å *kopiere* noe Java-kode fra dem, selv om du endrer på koden etterpå; og det er heller ikke lov å hente programbiter fra andre besvarelser, for eksempel fra Internet. Hvis du diskuterer mye med en annen student skriv hvem det gjelder i kommentaren din øverst i programmet (da unngår du mistanke om at du prøvde å skjule dette, når vi oppdager det i likhets-kontrollen vi foretar med innleveringene).

Kilder

Alle dataene som står i datafilene er tatt fra Wikipedia og TMDb (disse er valgt fordi de er åpne og krever ikke lisens for bruk). Filmlisten er tatt fra de alfabetiske listene over "filmer som har egen Wikipedia-side".

Hvis du har spørsmål, kommentarer, eller rettelser til obligen kan du skrive de i kurs-bloggen eller maile de til meg, josek [at] ifi.uio.no..