# Oblig 3

Nytt forsøk

**Forfall** 17. feb. av 23.59 **Poeng** 1 **Må leveres** en filopplasting

**Alle oppgavene skal være forsøkt løst etter beste evne (du *må* ikke løse bonusoppgavene hvis det er noen). Hvis du har en oppgave som du har forsøkt på, men ikke fått helt til, er det OK om denne kommenteres ut slik at vi kan se hvordan du har tenkt/hva du har forsøkt på. Resterende kode bør kunne kompileres og kjøres uten at det kræsjer. Oppgaver som er gjennomført og fungerer burde *ikke* kommenteres ut.**

Les: Obligatoriske oppgaver - Informasjon (https://hiof.instructure.com/courses/7058/pages/obligatoriske-oppgaver-informasjon)

Obligen er strukturert opp i oppgaver, dere skal levere alle filene samlet.

**Teorispørsmålene:** Leveres som docx/pdf/txt navngitt: Oblig3_FILM_<DittNavn>.<filtype>
Eks: Oblig2_TV_LarsEmilKnudsen.pdf

**Programeringsoppgavene:** Du skal levere **.java** filene som en .zip fil navngitt: Oblig3_FILM_<DittNavn>.zip
Eks: Oblig2_TV_LarsEmilKnudsen.zip

**I TILLEGG:** I denne obligen er det fint hvis dere laster opp **alle .java filene en og en** direkte, i *tillegg* til en .zip fil med alle.

## Teori

### Oppgave 1.1 - Teori

Lag deg en oversikt over hva følgende ord/begreper betyr:

- Refaktorere
- Static (metode, variabel)
- Final (klasse, metode, variabel)
- Abstract (klasse, metode)
- Interface

### Oppgave 1.2 - Sammenligning

I denne oppgaven skal dere gå sammen to og to (en du ikke har samarbeidet med). Ta for dere forrige oblig og forklar deres implementasjon.
Fokuser på følgende metoder (kan gjerne se på mer om dere ønsker også):

- leggTilEpisode(Episode episoden)
- oppdaterGjennomsnittligSpilletid()

- hentEpisoderISesong(int sesongNummer)

Hva har dere gjort? Hvorfor har dere gjort det slik? Hva er forskjellig? Skriv et lite avsnitt om refleksjoner og funn.

Skriv hvem dere har gått sammen med, men skriv hver deres tekst.

Til de som *ikke* finner noen å gå sammen med, skriv opp navn og e-post i skjemaet her (https://docs.google.com/spreadsheets/d/1geuysHJD5oX3bFdUMF9OwOK_MAmRFe8UYEbJf2fNj7Q/edit?usp=sharing). Den neste som skriver seg på, skriver seg opp i kolonne to, og tar kontakt for å avtale tid med den første.

**Oppgave 1.3 - Klassediagram**

Gå gjennom oppgavene 2.1-2.7 under programmering. Lag et klassediagram over alle klassene med variabler og metoder, samt relasjonene mellom disse klassene.
Hvis du gjør noe mer enn det som er definert i programmeringsoppgaven underveis, løser noe med ekstra metoder eller lignende, lag et modifisert klassediagram i tillegg som reflekterer dette.
Legg med klassediagrammet(ene) i teoridokumentet.

# Programmering

Vi skal fortsette med å utvide oppgaven vi lagde i Oblig 2. Du kan fortsette på din egen implementasjon, eller du kan starte fra løsningsforslaget her: Oblig2_ProposedSolution.zip (https://hiof.instructure.com/courses/7058/files/1276857?wrap=1) (https://hiof.instructure.com/courses/7058/files/1276857/download?download_frd=1).

Vi lagde oss en oversikt over TVSerie og tilhørende episoder, vi ønsker nå å utvide prosjektet slik at det det kan håndtere filmer og roller/skuespillere.

**Oppgave 2.1 - Produksjon**

Vi ønsker i utgangspunktet å lage klassen Film. Når vi begynner på det ser vi at så og si alle instansvariablene stemmer overens med de som eksisterer i Episode-klassen vi allerede har laget. Nå kan vi ikke si at Film "er en" Episode, så det blir ikke så naturlig for oss å sette opp arv på den måten.

Det vi KAN si er at Episode "*er en*" Produksjon. Vi kan også si at Film "*er en*" Produksjon.

Vi ønsker da å abstrahere instansvariabler fra Episode, til en ny klasse Produksjon.

1. Abstraher (flytt) instansvariablene: tittel og spilletid til den nye klassen Produksjon
2. Lag så get- og set-metoder, samt konstruktør i Produksjon
3. Sett så Episode til å extende Produksjon
4. Pass på å kalle "superkonstruktøren" i konstruktøren(e) til Episode
5. Verifiser at all "testkode" vi har laget tidligere fortsatt fungerer som den skal

**Oppgave 2.2 - Film**

Nå som vi har klassen Produksjon, kan vi lage klassen Film. Sett så denne Film-klassen til å arve fra klassen Produksjon.

Lag en konstruktør i Film som tar de nødvendige parameterne, og "sender de videre" til superkontruktøren i Produksjon.

Lag et par filmobjekter i Main.java og skriv ut tittelen.

### Oppgave 2.3 - Utvide klasser

Vi ser at utgivelsesdato og beskrivelse er aktuelt å ha med som en del av film, dette kan også være aktuelt å ha med i Episode.

Legg derfor til instansvariabelen utgivelsesdato og beskrivelse i klassen Produksjon, og lag relaterte get- og set-metoder. Gjør nødvendige endringer i konstruktørene i Produksjon, Episode og Film for å få med denne dataen.

Dere står her, som tidligere fritt til å velge datatype, men LocalDate-klassen er fortsatt et greit alternativ for utgivelsesdato.

### Oppgave 2.4 - Regissør

Vi er interessert i å vite litt om hvem som har laget en film eller episode. I første omgang tar vi bare for oss regissøren. Regissøren er en person (og det er også andre vi ønsker informasjon om senere), lag derfor en klasse Person. Velg selv hvilke data du ønsker å lagre om en Person. Lag en metode som lar deg hente en persons fulle navn.

Lag deretter instansvariabelen regissor av typen Person, i klassen Produksjon. Lag relaterte get- og set-metoder.

Test at du får laget et par regissører i Main.java og lagt han/henne til i både en Film og en Episode. Hent og skriv ut hvem som er regissør der du har lagt en til.

### Oppgave 2.5 - Roller

Nå ønsker vi også en oversikt over roller (og de som spiller dem). Lag derfor en klasse Rolle. Denne bør minst inneholde: rolleFornavn, rolleEtternavn og skuespiller (av typen Person, hvem som spiller rollen).

Eksempel- Iron Man, Robert Downey. Jr

Det er relevant å anta at en Produksjon (Episode eller Film), kan inneholde mange roller. Lag derfor en instansvariabel som inneholder en liste med roller i Produksjon. Lag en relatert get-metode. Lag deretter to metoder:

leggTilEnRolle(Rolle enRolle)

leggTilMangeRoller(ArrayList<Rolle> flereRoller)

Test dette i Main.java ved å lage noen relaterte roller og legg de til i film/episode(r) ved hjelp av disse metodene. Hent så ut alle rollene til en film og skriv de ut.

### Oppgave 2.6 - toString()

@Override toString() i alle klasser som ikke har det allerede.

### Oppgave 2.7 - Hente alle roller

Vi ønsker å vite om alle roller (og da også skuespillere) som har spilt i en TVSerie. Lag en metode i TVSerie som heter hentRollebesetning(), som returnerer en liste med roller. Du må altså hente alle roller fra alle episoder som tilhørere TVSerien.

Skriv så ut navnet på rollene, og skuespillerne som spiller disse.

(Du vil mest sannsynlig få samme skuespiller og rolle flere ganger, siden de spiller i flere episoder, dette er OK)

**BONUSOPPGAVER:**

### Oppgave 3.1 - Bedre uthenting av alle roller

Vi ønsker i utgangspunktet at hentRollebesetning(), ikke skal kunne ha duplikater. Gjør endringer i metoden for å få til dette.

### Oppgave 3.2 - Hvor mange episoder?

Hent ut alle roller/skuespillere som har spilt i en TVSerie, og finn ut hvor mange episoder hver enkelt skuespiller har spilt i.

Det kan her kanskje være nødvendig å lage en ny metode?

Skriv ut skuespiller og antall episoder den er med i.

### Oppgave 3.4 - It's over 9000!

Generer 14,430 episoder av Days of our Lives. Første episode starter 8. November 1965.

Sett det opp slik at det blir laget en episode for hver hverdag i uka. Når man kommer til et nytt år, skal det lages en ny sesong.

Hent så ut alle episoder i sesong 42.