# Oblig 5 OOP Martin Røsok

1)

Nestede klasser er klasser som er definert inni en annen klasse. Det er da bare den ytre klassen som har tilgang til den nestede klassen. Dette gjør vi for å øke innkapsulering og potensielt bidra til mer lesbar kode. Hvis en klasse bare skal brukes av en annen klasse er det en god ide å neste de.

Nestede klasser er delt inn i to kategorier: statiske og ikke-statiske. Nestede klasser som deklareres statisk er kalt statiske nestede klasser. Ikke-statiske nestede klasser kaller vi indre klasser

Tråder er en måte å separere del-prosesser i en større applikasjon og la de kjøres parallelt. En eller flere mindre prosesser (kodesekvenser) kan få sin egen tråd som kan kjøres uavhengig av hovedapplikasjonen (main thread). Dette gjør at vi kan utføre logikk parallelt (f.eks strømme en fil samtidig som vi håndterer brukerinput etc). Vi kan også utnytte CPU bedre og sikre en bedre opplevelse for brukeren (som f.eks ikke opplever at hele applikasjonen "fryser" mens vi prøver å laste inn et bilde fra internett)

2)  
Jeg gikk sammen med Jorunn Schøyen for å drøfte Oblig 4. Jorunn har ikke blitt helt ferdig med å implementere all funksjonaliteten så vi så på hvordan jeg hadde løst deler hun ikke hadde fått til – og hva vi hadde gjort annerledes ellers.

Første bit med statiske lister og implementering av Comparable har vi gjort likt. Jorunn gjorde sine lister public, mens jeg satt de til private og tror det er mer riktig.

Oppsettet for brukergrensesnittet løste vi nokså likt ved å følge oppsettet som Lars Emil har demonstrert på forelesning. Vi instansierte main-metoden vår for å få tilgang på filmlisten fra de andre klassene, og satt opp en Scene og FXMLLoader på samme måte. Jorunn har ikke håndtert exceptions med try/catch men satt throws Exception på alle metodene. Jeg er usikker på hvilke problemer dette vil medføre.

Hovedforskjellen i hvordan grensesnittet er satt opp mtp på kode ligger i hvordan vi håndterte nye visninger. Jorunn har satt opp metoder for å gå til andre visninger (rediger/ny) i main metoden. Hun har også brukt nye Scener for dette. Jeg har brukt klassen Dialog for å lage dialogbokser for disse funksjonene og kalt de i kontrolleren. Begge er legitime løsninger men et eksempel på forskjellig strategier.

Jorunn hadde noe problemer med å få funksjonaliteten i rediger/slett/ny implementert. Hun løste dette etter å ha snakket med Lars Emil etter innlevering, men vi har delt versjonen ved innlevering og sett på denne. Her har hun brukt pseudokode som viser hvordan hun har tenkt. Hun har foreslått separate kontrollere og views for NyFilm og RedigerFilm – noe jeg tror er overflødig. Jeg brukte bare en kontroller/view for dialogboksen, da det å lage en ny film og å redigere en film kan bruke det samme grensesnittet. Sletting av filmen gjorde jeg med en knapp i hovedvinduet.

BONUS TIME

Noen ord om innleveringen av oblig 5: Jeg har brukt en lokal database for håndtering av persistent lagring. Dette gjør at når jeg leverer inn prosjektet har den som retter ikke tilgang til samme data og vil ikke kunne kjøre programmet og se funksjonaliteten i praksis. Jeg har vært på studasstime og fått demonstrert programmet og fått oppgaven godkjent der. Jeg har også lagt ved en sql-dump av databasen med tilhørende tabell og tabelldata. Om retteren har lyst vil det være nokså enkelt å importere disse dataene inn i en egen lokal database. Brukernavn, passord og sannsynligvis localhost adresse må da endres i MovieQuery-klassen, men det burde være det hele (med mindre det også trengs å installere mySQL driveren i IntelliJ….)