

Code Review – Filmsamlingsprojektet v2

Dette er et skema til code review for et projekt – hvor der både bliver fokuseret på overordnet objektorienteret struktur, navngivning af metoder og variabler, brug af kommentarer, håndtering af exceptions og andre detaljer. Der er ikke fokus på brugergrænseflade og brugeroplevelsen.

Reviewet foregår lidt anderledes i denne omgang, da i som gruppe agerer reviewer og reviewee. Reviewet skal dog stadig forgå under et møde, hvor et gruppemedlem fra reviewee gruppen agere programmør for reviewer gruppen (de som altså ikke selv skal agere programmør i et andet review). Begge parter udfylder dette dokument sammen, udskriver til PDF og overdrager efterfølgende til programmøren. Dette gøres af to omgange (se grupperne under Materialer), så hver gruppe står tilbage med to reviews, som de tager afsæt i til efterfølgende refactorering.

Start med at én reviewer cloner projektet fra Github og åbner det i IntelliJ – en anden reviewer redigerer dette dokument i Word. Både reviewere og programmør følger med på skærmen med projekt-koden.

Programmør:	Gruppe 2	
GitHub repository	https://github.com/nial0003/FilmProjekt.git	Commit hash (7 cifre) ?
Reviewer1:	Muzaffer Kaan Celik	

Spørgsmål markeret med  ikonet er dårlige – de skal helst kunne besvares med nej.

Klik i checkboxes (□) hvis der kan svares "ja" til et spørgsmål. Hvis der er grund til at give eksempler eller uddybende kommentarer, så skriv dem med *blå skrift* på linjen umiddelbart under spørgsmålet. (*Hvis man trykker enter i slutningen af en linje, burde den automatisk skifte til blå skrift på linjen under.*)

Nogle spørgsmål er måske ikke relevante – for eksempel er der ingen grund til at skrive om exception-håndtering for en klasse der ikke har exceptions. I de tilfælde, så **slet** blot spørgsmålet!

GitHub

Før der kigges på kode, tag et kig på hvad der ellers ligger i GitHub

- Er der en .gitignore fil?
-  Er der uvedkommende filer i Github (for eksempel class filer, eller lignende)?[?](#)
- Ligger dokumentation og andet ikke-kode i en docs-mappe?

Packages

Der bør være tre packages i jeres projekt under src/main/java. De burde hedde datasource, domainmodel og ui (eller lignende). Følgende spørgsmål vil omhandle den overordnede package/lag-opbygning af projektet.

- Forefindes de tre packages under src/main/java i projektet?[?](#)
- Er hver package godt navngivet?[?](#)
- Er det let at regne ud hvilken rolle hver package har?[?](#)

- Er filerne i hver package repræsentative for det lag som packagen råder over?

Klasser

Der bør være mindst seks klasser, som ikke implementere Comparator interfacet, i hele projektet: Main, Movie, MovieCollection, Controller, UserInterface og Filehandler. Hvis nogle af dem er navngivet anderledes, så diskutér om navngivning er værre, bedre eller det samme. Derudover, burde der også være flere klasser, der implementerer Comparator interfacet og som dermed muliggør sortering på flere Movie attributter. Disse burde have navne hvor Comparator indgår, samt hvilket attribut, der sammenlignes på (ex. AgeComparator).

De følgende spørgsmål (indtil linjen --) er præcis de samme, men gentaget for Movie, MovieCollection, Controller, UserInterface, FileHandler og Comparator-klassen (som reviewes samlet).

Hvis der er hele sektioner, der ikke giver mening at svare på – hvis der for eksempel ikke er brugt exceptions i en klasse, så må I gerne slette det afsnit.

Movie-klassen: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?
- Er der brugt tydelig ental/flertal for at vise om klassen indeholder ét eller flere underobjekter

Attributter/fields

- Er der oprettet attributter der burde være lokale variable i stedet?
- Er der erklæret ubenyttede attributter?

Er attributter navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er navnene tydeligt og korrekt ental eller flertal?
- Er navnene alle på samme sprog (engelsk/dansk)?

Har attributter korrekt access?

- Er der brugt private?
- Er der getter og setter (hvis nødvendigt)?
- Er attributter gjort public uden getter og setter?
- Er der hverken brugt private eller public, men bare 'ingenting'?

Getters/Setters

- Er der ubenyttede get og set metoder?
- Er der metoder kaldet get eller set, som ikke er en getter/setter?

- Eller omvendt, er der gettere eller setttere der hedder noget andet end get eller set?

Metoder

Kig på hver metode, først "udefra", altså uden at læse koden inde i metoderne – det hjælper måske at "folde" alle metoderne sammen (ctrl+shift+minus (cmd+shift+minus på Mac)).

Er metoder navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er det tydeligt hvad hver metode gør, ud fra sit navn?
- Er det tydeligt nok ud fra navnet, om metoder returnerer eller modtager værdier?
- Hvis metoden modtager parametre, er det så umiddelbart til at regne ud hvad de er til, ud fra navngivningen?
- Hvis metoden returnerer en værdi, giver datatypen så umiddelbar mening?
- Er der en fornuftig ensartethed/struktur i navngivningen af metoderne?
- Er metodenavnene navngivet med korrekt brug af camelCase?
- Er parameternavne navngivet med korrekt brug af camelCase?
- Er navnene alle på samme sprog (engelsk/dansk)?

Kode

Kig derefter ind i hver metode, og vurdér koden inde i dem – I skal ikke besvare en sektion særligt for hver eneste metode, men kigge på metoderne som helhed, og blot notere hvis der er et sted i en metode, der afviger.

Lokale variabler inde i metoder

- Er variabler navngivet korrekt camelCase?
- Er variablernes navne i tilstrækkelig grad fra "problemdomænet"?
- Er variablerne erklæret, hvor de skal bruges?
- Har variablerne den korrekte datatype?
- Er navnene alle på samme sprog (engelsk/dansk)?

Generel kodestil

- Er koden pænt (og korrekt) formateret med indrykninger etc.?
- Er koden "skimbar", og umiddelbart let at overskue, uden at skulle nærlæse hver linje?
- Er der smarte "ninja-tricks" der gør koden kompakt, men måske sværere for nogle at læse?

Exceptions

- Bliver exceptions fanget (altså er der en catch fremfor blot en throws)?
- Bliver exceptions blot håndteret med en catch med printStackTrace?
- Bliver exceptions håndteret, der hvor det giver mening – kan programmet fortsætte efter en exception?
- Er der lavet custom exceptions (dvs. egne exception typer) – og giver de bedre mening end de oprindelige?

Kommentarer

- Er der kommentarer, der f.eks. viser hvor en fancy løsning er fundet på nettet?
 - Er der kommentarer, der beskriver hvordan/hvorfor en metode er opbygget anderledes end forventet?
 - Er der kommentarer til udvikleren selv, om ting der skal rettes/ændres i fremtiden?
 - Er der kommentarer der ikke længere giver mening, og måske burde have været slettet?
 - Er der gode kommentarer i programmet?
 - Er der overflødige kommentarer, kunne f.eks. være `i++; // lægger en til i` ?
-

MovieCollection-klassen: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?
- Er der brugt tydelig ental/flertal for at vise om klassen indeholder ét eller flere underobjekter

Attributter/fields

- Er der oprettet attributter der burde være lokale variable i stedet?
- Er der erklæret ubenyttede attributter?

Er attributter navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er navnene tydeligt og korrekt ental eller flertal?
- Er navnene alle på samme sprog (engelsk/dansk)?

Har attributter korrekt access?

- Er der brugt private?
- Er der getter og setter (hvis nødvendigt)?
- Er attributter gjort public uden getter og setter?
- Er der hverken brugt private eller public, men bare 'ingenting'?

Getters/Setters

- Er der ubenyttede get og set metoder?
- Er der metoder kaldet get eller set, som ikke er en getter/setter?
- Eller omvendt, er der gettere eller setttere der hedder noget andet end get eller set?

Metoder

Kig på hver metode, først "udefra", altså uden at læse koden inde i metoderne – det hjælper måske at "folde" alle metoderne sammen (ctrl+shift+minus (cmd+shift+minus på Mac)).

Er metoder navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er det tydeligt hvad hver metode gør, ud fra sit navn?
- Er det tydeligt nok ud fra navnet, om metoder returnerer eller modtager værdier?
- Hvis metoden modtager parametre, er det så umiddelbart til at regne ud hvad de er til, ud fra navngivningen?
- Hvis metoden returnerer en værdi, giver datatypen så umiddelbar mening?
- Er der en fornuftig ensartethed/struktur i navngivningen af metoderne?
- Er metodenavnene navngivet med korrekt brug af camelCase?
- Er parameterne navngivet med korrekt brug af camelCase?
- Er navnene alle på samme sprog (engelsk/dansk)?

Kode

Kig derefter ind i hver metode, og vurdér koden inde i dem – I skal ikke besvare en sektion særskilt for hver eneste metode, men kigge på metoderne som helhed, og blot notere hvis der er et sted i en metode, der afviger.

Lokale variabler inde i metoder

- Er variabler navngivet korrekt camelCase?
- Er variablene navne i tilstrækkelig grad fra "problemdomænet"?
- Er variablerne erklæret, hvor de skal bruges?
- Har variablerne den korrekte datatype?
- Er navnene alle på samme sprog (engelsk/dansk)?

Generel kodestil

- Er koden pænt (og korrekt) formateret med indrykninger etc.?
- Er koden "skimbar", og umiddelbart let at overskue, uden at skulle nærlæse hver linje?
- Er der smarte "ninja-tricks" der gør koden kompakt, men måske sværere for nogle at læse?

Exceptions

- Bliver exceptions fanget (altså er der en catch fremfor blot en throws)?
- Bliver exceptions blot håndteret med en catch med printStackTrace?
- Bliver exceptions håndteret, der hvor det giver mening – kan programmet fortsætte efter en exception?
- Er der lavet custom exceptions (dvs. egne exception typer) – og giver de bedre mening end de oprindelige?

Kommentarer

- Er der kommentarer, der f.eks. viser hvor en fancy løsning er fundet på nettet?
 - Er der kommentarer, der beskriver hvordan/hvorfor en metode er opbygget anderledes end forventet?
 - Er der kommentarer til udvikleren selv, om ting der skal rettes/ændres i fremtiden?
 -  Er der kommentarer der ikke længere giver mening, og måske burde have været slettet?
 -  Er der gode kommentarer i programmet?
 -  Er der overflødige kommentarer, kunne f.eks. være `i++; // lægger en til i` ?
-

User Interface-klassen: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?
- Er der brugt tydelig ental/flertal for at vise om klassen indeholder ét eller flere underobjekter

Attributter/fields

-  Er der oprettet attributter der burde være lokale variable i stedet?
-  Er der erklæret ubenyttede attributter?

Er attributter navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er navnene tydeligt og korrekt ental eller flertal?
- Er navnene alle på samme sprog (engelsk/dansk)?

Har attributter korrekt access?

- Er der brugt private?
- Er der getter og setter (hvis nødvendigt)?
-  Er attributter gjort public uden getter og setter?
-  Er der hverken brugt private eller public, men bare 'ingenting'?

Getters/Setters

-  Er der ubenyttede get og set metoder?
-  Er der metoder kaldet get eller set, som ikke er en getter/setter?
-  Eller omvendt, er der gettere eller setttere der hedder noget andet end get eller set?

Metoder

Kig på hver metode, først "udefra", altså uden at læse koden inde i metoderne – det hjælper måske at "folde" alle metoderne sammen (ctrl+shift+minus (cmd+shift+minus på Mac)).

Er metoder navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er det tydeligt hvad hver metode gør, ud fra sit navn?
- Er det tydeligt nok ud fra navnet, om metoder returnerer eller modtager værdier?
- Hvis metoden modtager parametre, er det så umiddelbart til at regne ud hvad de er til, ud fra navngivningen?
- Hvis metoden returnerer en værdi, giver datatypen så umiddelbar mening?
- Er der en fornuftig ensartethed/struktur i navngivningen af metoderne?
- Er metodenavnene navngivet med korrekt brug af camelCase?
- Er parameternavne navngivet med korrekt brug af camelCase?
- Er navnene alle på samme sprog (engelsk/dansk)?

Kode

Kig derefter ind i hver metode, og vurdér koden inde i dem – I skal ikke besvare en sektion særskilt for hver eneste metode, men kigge på metoderne som helhed, og blot notere hvis der er et sted i en metode, der afviger.

Lokale variabler inde i metoder

- Er variabler navngivet korrekt camelCase?
- Er variablers navne i tilstrækkelig grad fra "problemdomænet"?
- Er variablerne erklæret, hvor de skal bruges?
- Har variablerne den korrekte datatype?
- Er navnene alle på samme sprog (engelsk/dansk)?

Generel kodestil

- Er koden pænt (og korrekt) formateret med indrykninger etc.?
- Er koden "skimbar", og umiddelbart let at overskue, uden at skulle nærlæse hver linje?
- Er der smarte "ninja-tricks" der gør koden kompakt, men måske sværere for nogle at læse?

Exceptions

- Bliver exceptions fanget (altså er der en catch fremfor blot en throws)?
- Bliver exceptions blot håndteret med en catch med printStackTrace?
- Bliver exceptions håndteret, der hvor det giver mening – kan programmet fortsætte efter en exception?
- Er der lavet custom exceptions (dvs. egne exception typer) – og giver de bedre mening end de oprindelige?

Kommentarer

- Er der kommentarer, der f.eks. viser hvor en fancy løsning er fundet på nettet?
- Er der kommentarer, der beskriver hvordan/hvorfor en metode er opbygget anderledes end forventet?
- Er der kommentarer til udvikleren selv, om ting der skal rettes/ændres i fremtiden?

- Er der kommentarer der ikke længere giver mening, og måske burde have været slettet?
 - Er der gode kommentarer i programmet?
 - Er der overflødige kommentarer, kunne f.eks. være `i++; // lægger en til i` ?
-

Controller-klassen: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?
- Er der brugt tydelig ental/flertal for at vise om klassen indeholder ét eller flere underobjekter

Attributter/fields

- Er der oprettet attributter der burde være lokale variable i stedet?
- Er der erklæret ubenyttede attributter?

Er attributter navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er navnene tydeligt og korrekt ental eller flertal?
- Er navnene alle på samme sprog (engelsk/dansk)?

Har attributter korrekt access?

- Er der brugt private?
- Er der getter og setter (hvis nødvendigt)?
- Er attributter gjort public uden getter og setter?
- Er der hverken brugt private eller public, men bare 'ingenting'?

Getters/Setters

- Er der ubenyttede get og set metoder?
- Er der metoder kaldet get eller set, som ikke er en getter/setter?
- Eller omvendt, er der gettere eller setttere der hedder noget andet end get eller set?

Metoder

Kig på hver metode, først "udefra", altså uden at læse koden inde i metoderne – det hjælper måske at "folde" alle metoderne sammen (ctrl+shift+minus (cmd+shift+minus på Mac)).

Er metoder navngivet godt?

- Er navnene selvforklarende?
- Er navnene fra "problemdomænet"?
- Er det tydeligt hvad hver metode gør, ud fra sit navn?
- Er det tydeligt nok ud fra navnet, om metoder returnerer eller modtager værdier?

- Hvis metoden modtager parametre, er det så umiddelbart til at regne ud hvad de er til, ud fra navngivningen?[?](#)
- Hvis metoden returnerer en værdi, giver datatypen så umiddelbar mening?[?](#)
- Er der en fornuftig ensartethed/struktur i navngivningen af metoderne?[?](#)
- Er metodenavnene navngivet med korrekt brug af camelCase?[?](#)
- Er parameternavne navngivet med korrekt brug af camelCase?[?](#)
- Er navnene alle på samme sprog (engelsk/dansk)?[?](#)

Kode

Kig derefter ind i hver metode, og vurdér koden inde i dem – I skal ikke besvare en sektion særskilt for hver eneste metode, men kigge på metoderne som helhed, og blot notere hvis der er et sted i en metode, der afviger.

Lokale variabler inde i metoder

- Er variabler navngivet korrekt camelCase?[?](#)
- Er variabernes navne i tilstrækkelig grad fra "problemdomænet"?[?](#)
- Er variablerne erklæret, hvor de skal bruges?[?](#)
- Har variablerne den korrekte datatype?[?](#)
- Er navnene alle på samme sprog (engelsk/dansk)?[?](#)

Generel kodestil

- Er koden pænt (og korrekt) formateret med indrykninger etc.[?](#)
- Er koden "skimbar", og umiddelbart let at overskue, uden at skulle nærlæse hver linje?[?](#)
- Er der smarte "ninja-tricks" der gør koden kompakt, men måske sværere for nogle at læse?[?](#)

Exceptions

- Bliver exceptions fanget (altså er der en catch fremfor blot en throws)?[?](#)
- Bliver exceptions blot håndteret med en catch med printStackTrace?[?](#)
- Bliver exceptions håndteret, der hvor det giver mening – kan programmet fortsætte efter en exception?[?](#)
- Er der lavet custom exceptions (dvs. egne exception typer) – og giver de bedre mening end de oprindelige?[?](#)

Kommentarer

- Er der kommentarer, der f.eks. viser hvor en fancy løsning er fundet på nettet?[?](#)
- Er der kommentarer, der beskriver hvordan/hvorfor en metode er opbygget anderledes end forventet?[?](#)
- Er der kommentarer til udvikleren selv, om ting der skal rettes/ændres i fremtiden?[?](#)
- Er der kommentarer der ikke længere giver mening, og måske burde have været slettet?[?](#)
- Er der gode kommentarer i programmet?[?](#)
- Er der overflødige kommentarer, kunne f.eks. være `i++; // lægger en til i`?[?](#)

FileHandler-klassen: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?
- Er der brugt tydelig ental/flertal for at vise om klassen indeholder ét eller flere underobjekter

Attributter/fields

-  Er der oprettet attributter der burde være lokale variable i stedet?
-  Er der erklæret ubenyttede attributter?

Er attributter navngivet godt?

- Er navnene selvførligende?
- Er navnene fra "problemdomænet"?
- Er navnene tydeligt og korrekt ental eller flertal?
- Er navnene alle på samme sprog (engelsk/dansk)?

Har attributter korrekt access?

- Er der brugt private?
- Er der getter og setter (hvis nødvendigt)?
-  Er attributter gjort public uden getter og setter?
-  Er der hverken brugt private eller public, men bare 'ingenting'?

Getters/Setters

-  Er der ubenyttede get og set metoder?
-  Er der metoder kaldet get eller set, som ikke er en getter/setter?
-  Eller omvendt, er der gettere eller setttere der hedder noget andet end get eller set?

Metoder

Kig på hver metode, først "udefra", altså uden at læse koden inde i metoderne – det hjælper måske at "folde" alle metoderne sammen (ctrl+shift+minus (cmd+shift+minus på Mac)).

Er metoder navngivet godt?

- Er navnene selvførligende?
- Er navnene fra "problemdomænet"?
- Er det tydeligt hvad hver metode gør, ud fra sit navn?
- Er det tydeligt nok ud fra navnet, om metoder returnerer eller modtager værdier?
- Hvis metoden modtager parametre, er det så umiddelbart til at regne ud hvad de er til, ud fra navngivningen?
- Hvis metoden returnerer en værdi, giver datatypen så umiddelbar mening?
- Er der en fornuftig ensartethed/struktur i navngivningen af metoderne?

- Er metodenavnene navngivet med korrekt brug af camelCase?
- Er parameternavne navngivet med korrekt brug af camelCase?
- Er navnene alle på samme sprog (engelsk/dansk)?

Kode

Kig derefter ind i hver metode, og vurdér koden inde i dem – I skal ikke besvare en sektion særskilt for hver eneste metode, men kigge på metoderne som helhed, og blot notere hvis der er et sted i en metode, der afviger.

Lokale variabler inde i metoder

- Er variabler navngivet korrekt camelCase?
- Er variablene navne i tilstrækkelig grad fra "problemdomænet"?
- Er variablerne erklæret, hvor de skal bruges?
- Har variablerne den korrekte datatype?
- Er navnene alle på samme sprog (engelsk/dansk)?

Generel kodestil

- Er koden pænt (og korrekt) formateret med indrykninger etc.?
- Er koden "skimbar", og umiddelbart let at overskue, uden at skulle nærlæse hver linje?
- Er der smarte "ninja-tricks" der gør koden kompakt, men måske sværere for nogle at læse?

Exceptions

- Bliver exceptions fanget (altså er der en catch fremfor blot en throws)?
- Bliver exceptions blot håndteret med en catch med printStackTrace?
- Bliver exceptions håndteret, der hvor det giver mening – kan programmet fortsætte efter en exception?
- Er der lavet custom exceptions (dvs. egne exception typer) – og giver de bedre mening end de oprindelige?

Kommentarer

- Er der kommentarer, der f.eks. viser hvor en fancy løsning er fundet på nettet?
 - Er der kommentarer, der beskriver hvordan/hvorfor en metode er opbygget anderledes end forventet?
 - Er der kommentarer til udvikleren selv, om ting der skal rettes/ændres i fremtiden?
 - Er der kommentarer der ikke længere giver mening, og måske burde have været slettet?
 - Er der gode kommentarer i programmet?
 - Er der overflødige kommentarer, kunne f.eks. være `i++; // lægger en til i` ?
-

- Er der brugt override-notation?
- Er koden til metoden implementeret ved brug at compareTo-metoden eller ud fra at compare-metoden returnerer 0, 1 eller -1?
- Kan koden til metoden simplificeres?

Comparator-klasserne: -navn-

- Er klassen godt navngivet?
- Er det let at regne ud hvilken rolle klassen har?

compare-metoder

Er metoden godt implementeret?

Ja

Yderligere kommentarer til koden fra reviewerne

Her kan I skrive yderligere noter, som I reviewere måtte have – ting der ikke lige passer ind i skemaerne ovenfor – ros til særlig elegant kode, spørgsmål til hvordan programmøren fandt på en eller anden løsning!