Projekt 2019

En streamingtjeneste

Kursets læringsmål

Efter kurset skal I kunne

- ANALYSERE en problemformulering (mhp at KONSTRUERE brugbare, mindre systemer på op til ca. 1000 linjers Java kode); dvs:
- DESIGNE et system (på baggrund af analysen);
- IMPLEMENTERE systemet (på baggrund af design);
- TESTE systemet (inkl. REDEGØRE for om det virker som ønsket samt VURDERE i hvilken grad afprøvningen understøtter en sådan konklusion);
- PRÆSENTERE systemets formål, opbygning og virkemåde både skriftligt og mundtligt for en relevant målgruppe;
- ANVENDE basale Java-konstruktioner (jf. kursusindhold); samt
- FORKLARE basale Java-konstruktioner (jf. kursusindhold) for en IT-professionel målgruppe

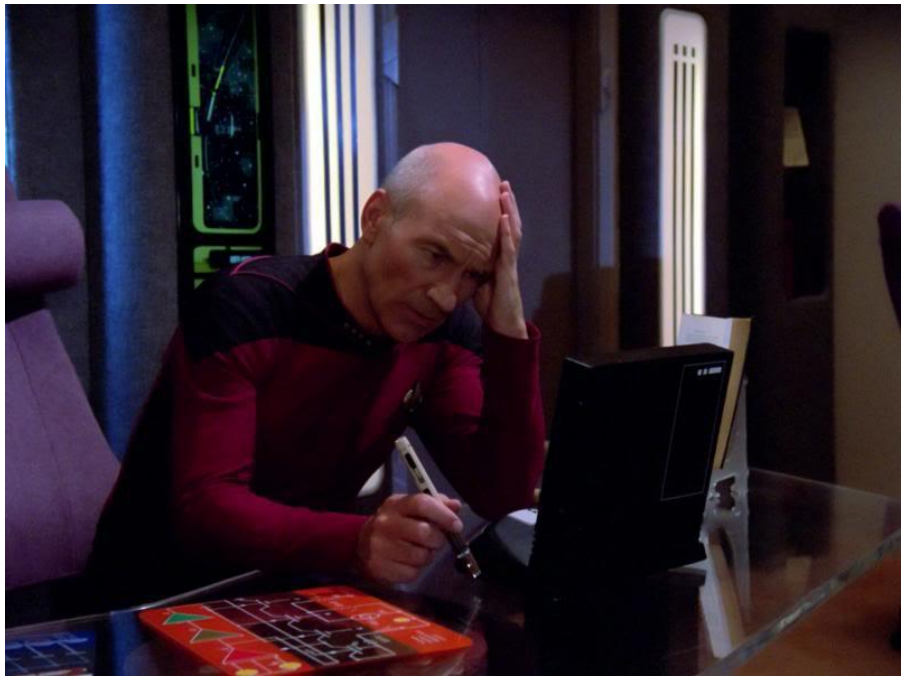
Hvad skal I lave?

- I projektet skal I udvikle et softwaresystem til brug for streamingtjenester som Netflix, HBO, Viaplay etc.
- I systemet er der ”medier”, der kan afspilles. Medier kan være film eller serie-episoder. Både film og serier er inddelt i kategorier som fx crime, war, drama, family, romance og sci-fi. Serier består desuden af sæsoner og episoder.
- Løsningen skal i en grafisk brugergrænseflade vise oversigt over film, serier, sæsoner og episoder, samt give mulighed for at man klikker sig ind på den enkelte film, serie, sæson eller episode og læser om den/sætter den i gang/tilføjer den til sin egen liste. Løsningen skal ikke kunne afspille film ”rigtigt” – det er fint blot at lave en play-knap, som ikke gør noget andet end fx at skifte farve når man klikker på den.

Hvad skal det kunne?

Løsningen skal understøtte almindelige opgaver, som I kender det fra streamingtjenester, fx

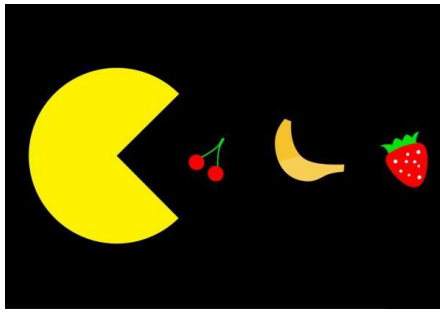
- Brugeren vil gerne se en oversigt over alle krigsfilm
- Brugeren vil gerne se en oversigt over alle dramaserier
- Brugeren vil gerne gemme en film i ”min liste”
- Brugeren vil gerne se sin liste
- Brugeren vil gerne slette en film fra sin liste
- Brugeren vil gerne skifte til en anden bruger
- Brugeren vil søge efter en bestemt film eller serie
- Der skal være billeder knyttet til film og serier



Må I lave mere?

Der kan tænkes en masse ekstra features ind, som fx (ikke prioriteret rækkefølge)

- En bruger vil gerne søge efter film, der har rating større end 8,5
- En bruger vil gerne søge efter film fra 80'erne
- Det skal være muligt for en bruger at se hvor langt han eller hun er kommet i en given serie og se næste episode
- Der skal kunne tilføjes og fjernes kategorier på en nem måde, fx "Christmas" eller "Halloween".
- Der skal kunne tilføjes film, serier, sæsoner og episoder hvis brugeren er administrator
- Der skal kunne tilføjes og fjernes brugere
- En bruger kan være klassificeret som "barn" og må således kun se film i kategorien "Family"
- Der skal kunne tilføjes nye typer media, som fx e-bøger, lydbøger eller spil



Godt råd #1

Analyser -> design -> programmer -> test -> dokumenter -> gentag

- Det er bedre at lave et velstruktureret og fungerende program med få features, end et stort, rodet og halvfærdigt program.
- Start med en simpel prototype, som I kan bygge videre på.
- Lad være med at lave en indviklet brugergrænseflade eller at udtænke overkomplicerede opgaver, som brugeren skal kunne udføre.

Formål med projektet

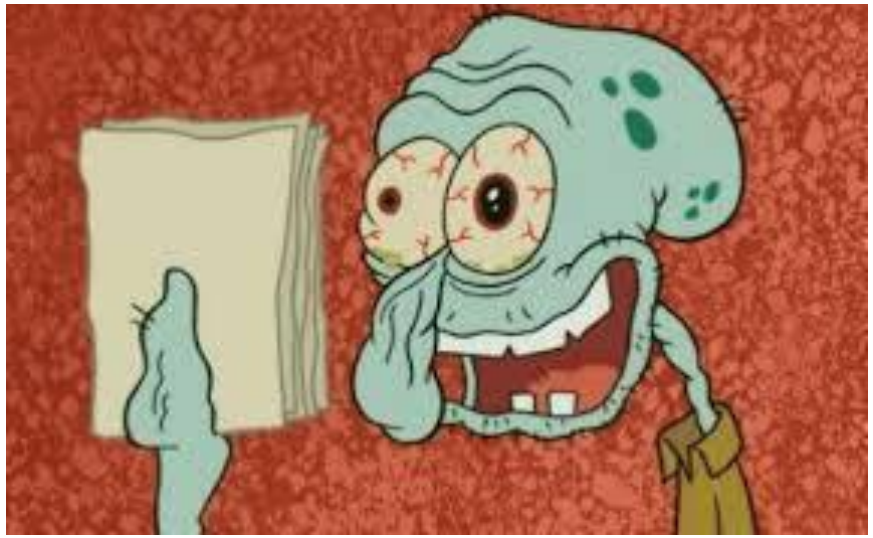
- I projektet skal I *analysere* et givet problem (*domæne-analyse*), *designe* samt *implementere* et softwaresystem til at løse det. I jeres løsning skal I *anvende* basale Javakonstruktioner som klasser, metoder, interfaces, løkker, Collections mv.
- Brug jeres faglighed og gå struktureret til opgaven.
- Hvad har I lært i kurset, der understøtter analyse og design?
- Hvad har I lært om implementering af et program?
- Hvilke basale Javakonstruktioner kan I med fordel anvende?

Godt råd #2

Rapporten er vigtig. Skriv den løbende

- Husk hvad jeres opgave er; et fungerende program OG en rapport
- I forventes at aflevere en velskrevet, velstruktureret, akademisk rapport. Den kan I ikke skrive natten før aflevering.

Hovedvægten i evalueringen lægges på *projektrapporten*. Rapportens formål er at forklare hvordan løsningen er struktureret, hvordan de enkelte delproblemer er løst og dokumentere at programmets funktionalitet er testet i form af tests. Det er vigtigt, at I begrunder jeres designvalg og vurderer funktionalitet og færdighedsgrad af jeres produkt.



Hvad er en akademisk rapport?

I skal udfærdige en rapport, hvori I *præsenterer* systemets *formål*, *opbygning* og *virkemåde*. I skal *forklare* jeres løsning for en målgruppe, der er på samme IT-niveau som jer selv. I skal i rapporten *begrunde* jeres designbeslutninger og fravalg, samt *overveje* og *diskutere* alternative løsningsmuligheder.

Som hjælp får I

- ”Udformning af rapporter” af Peter Sestoft
- ”ClickTicket: Et biograf reserveringssystem” af tre kvikke studerende fra forrige år

- Begge dele ligger på learnIT.

Men hvad ER en akademisk rapport?

Vi kommer til at tale mere om rapporten næste gang vi ses. Dog kan jeg løfte sløret lidt her.

En akademisk rapport indeholder ikke blot en beskrivelse af jeres system men også

Refleksion, overvejelser, afvejning af fordele og ulemper, præsentation af alternativer, vurdering, sammenligning, etc etc.

Fx:

I vores overvejelser om programmets interne struktur har vi valgt at benytte os af Model-View-Controller (*MVC*) strukturmønstret, for at adskille datamodellen (Model) fra brugergrænsefladen (*View*) og programlogikken (*Controller*). Således afkobles store dele af programmet fra hinanden, og gør det nemt for de enkelte programmører at udvikle på programmet parallelt. I *MVC*

I analysefasen af et projekt er målet at forstå programmets domæne ved at identificere fænomener (konkrete ting fra virkeligheden), som man kan abstrahere til koncepter (generaliseret idé fra en kollektion af fænomener med fælles karakteristika) (Brabrand, 2017, s. 3). Resultatet af dette

Den type data kollektion vi benytter oftest og har valgt til at holde data i programmet er *ArrayLists*. Fx valgte vi at benytte en *ArrayList* til at holde reservationer, da antallet af reservationer ændrer sig som nye reservationer oprettes. I forhold til listerne for *theaters*, *films* og

Godt råd #3

Test er vigtigt. Test løbende

- Ligesom rapportskrivning, er test og dokumentation typisk ikke programmørers og studerendes yndlingsbeskæftigelse. Gør det derfor løbende.
- Det gør det faktisk også lettere at teste, fordi I gør det når I har allermest kendskab til koden, nemlig lige efter I har skrevet den.

Hvordan tester man?

- I skal *systematisk afprøve* systemet og *dokumentere* jeres afprøvning. I rapporten skal I *redegøre* for om det virker som ønsket, og *vurdere* i hvilken grad jeres afprøvning understøtteren en sådan konklusion.
- Her skal jeres faglighed igen på banen.
- Hvad har I lært om at afprøve og dokumentere?

HJÆÆÆLPPP

- Spørgetime d. 6.12 i Aud 1 kl. 10-12 v. Signe
- Study lab og øvelsestimer som der plejer de næste to uger v. TA'er
- I er altid velkomne til at skrive til signek@itu.dk, hvis I er i tvivl om noget

Spørgsmål ?