Grundlæggende Programmering Projekt 2019

– en streamingtjeneste

1 Formål

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 Java-konstruktioner som klasser, metoder, interfaces, løkker, Collections mv.
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.

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.

Projektet involverer altså *udvikling*, *test* og *dokumentation* af et Java-program med en grafisk brugergrænseflade, samt en *akademisk* rapport, der beskriver programmet, samt jeres overvejelser og beslutninger.

2 Emne

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 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 – udleveres i .zip-fil.

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 medier, som fx e-bøger, lydbøger eller spil

Løsningen skal i en grafisk brugergrænseflade vise oversigt over film, serier og 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.

I skal i løsningen hente data om film og serier fra .txt-filer, som I får udleveret. Jeg foreslår, at I starter med at lave en funktion, som kan fylde nogle Collections med film, serier, episoder, etc. Når jeres program starter op, kører I som det første "fillCollection"-metoden, så I er klar til at arbejde med objekterne.

I skal nøje overveje hvordan forholdet mellem film, serier, sæsoner og episoder er og hvilken klassestruktur I vil bruge for at afspejle dette forhold. Tænk over, at det skal være muligt at tilføje og fjerne film uden at hele jeres program skal skrives om.

Brugergrænsefladen skal være simpel. I behøver ikke lave et lækkert design – fokuser hellere på noget simpelt, som ikke er for indviklet at lave eller forstå for brugeren. Tag gerne udgangspunkt i streamingtjenester, som I kender.

Løsningen skal have en fornuftig fejlhåndtering. Hvis der opstår fejl i programmet skal de håndteres passende steder og brugeren skal informeres, hvis det er relevant. I skal i jeres løsning lave mindst én hjemmelavet checked Exception og én hjemmelavet unchecked Exception.

3 Regler

Aflevering

Opgaven gennemgås ved forelæsningen fredag d. 22. november kl. 10-12 og projektrapporten samt koden skal afleveres senest **d. 20. december kl 14:00** gennem LearnIt.

Det er jeres ansvar at aflevere til tiden. Hvis I afleverer for sent, kan I **ikke** gå til eksamen! Der er **ingen** fleksibilitet fra ITU's side her.

I bør afsætte rigeligt tid til at gennemgå og uploade rapport samt kodefiler mv, for at sikre korrekt og rettidig aflevering.

Gruppestørrelse

Projektet udføres i **grupper à 3 personer**.

Grupper

I skal selv danne grupper og meld jer til grupper på LearnIT **senest mandag d. 25. november**.

Programmeringssprog

Systemet skal programmeres i Java. I må bruge Swing eller JavaFX til brugergrænsefladen.

Husk at sørge for et modulært design; specielt bør I sørge for at separere **brugergrænseflade** (view), **programlogik** (controller) og **data** (model – her tænkes ikke på database, men på de data, I har liggende om film, serier, brugere etc.). Tænk **Low coupling, High cohesion.**

Rapport

Projektrapporten skal have et omfang på **maksimalt 15 normalsider** (excl. figurer og bilag).

Forside: Rapportens forside skal oplyse: kursus, projekttitel, navne, ITU-email og dato. *Alle afleveringer på ITU skal benytte ITU' s standard-forside, som kan findes på ITU' s intranet*.

Om rapportens form og indhold, se vejledningen "*Udformning af rapporter*", der findes på kursets LearnIt side.

Snyd

I må **ikke** benytte andres tekst, illustrationer eller programkode uden udtrykkelig eksplicit kildeangivelse.
Plagiat kan resultere i udelukkelse fra eksamen (endog bortvisning fra universitetet!)!

4 Evaluering

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.

Rapporten bør indeholde

- En præsentation, præcisering og afgrænsning af krav til systemet, fx i form en liste af de arbejdsopgaver (tasks) og de data som systemet kan håndtere.
- Begrundede designbeslutninger vedrørende brugergrænseflade, datahåndtering og programmets interne struktur (klasser, interfaces, objekter og relationer mellem disse).
- Teknisk orienteret beskrivelse af programmets interne struktur.
- Begrundet afprøvning af det resulterende system, og vurdering af i hvilken grad systemet opfylder sit formål. Afprøvningen kan vedrøre brugergrænsefladens funktionalitet og unit test af klasserne.
- Kort konklusion om i hvilken grad produktet opfylder de opstillede krav, herunder en mangelliste, der også indeholder kendte fejl.
- Reflektioner over arbejdsprocessen, fx hvordan I har samarbejdet, hvordan det evt kunne have været forbedret, samt i hvilket omfang I mener, at I har nået projektets læringsmål.

Bilag bør indeholde

- Et appendix med en kort brugervejledning til jeres streamingtjeneste
- Et appendix med en kondenseret projektdagbog
- Et appendix med samtlige test-cases

Kode

- Derudover forventes en ZIP-pakke med Java-kildeteksten til det udarbejdede softwaresystem
- En .jar, der kan køre
- Husk at fjerne irrelevante filer (IDE-projektfiler, kompilerede .class-filer, mv.)

Der lægges vægt på at brugergrænsefladen understøtter hyppigt forekommende opgaver godt og at den programmeringsmæssige løsning er modulær, velstruktureret, vedligeholdsvenlig, veldokumenteret og ikke indeholder overflødige dele. Læg mærke til at projektrapporten både omhandler *produkt* (softwaresystemet og dets opbygning,

begrundede valg etc) samt *proces* (hvordan I har samarbejdet), dog med mest vægt på omtalen af produktet.

5 Gode Råd

Prioritering

Løs hellere nogle delproblemer godt end alle overfladisk. Det er *meget* bedre udtrykkeligt at beslutte, at et specielt delproblem (fx. "brugeren skal kunne se hvor han eller hun er kommet til i en serie") slet ikke håndteres i systemet (og skrive det på en mangelliste!), end at programmere en uigennemtænkt, uafprøvet og udokumenteret løsning til delproblemet. Sådan en del-løsning kan komplicere den samlede softwareløsning, reducere dens brugbarhed og vanskeliggøre videreudvikling.

Divide and Conquer

Lad være med at lave alt på én gang (Waterfall + Big Bang).

Forsøg at se på hvert delproblem for sig, og lav arbejdsblade til hvert delproblem, så I kan huske hvad I har overvejet og besluttet og hvorfor:

- Eksempel: Lav funktionalitet, som kan fylde Collections med film, serier og brugere.
- Eksempel: Undersøg om jeres brugergrænsefladedesign kan understøtte alle opgaverne.
- Eksempel: Gennemtænk hvordan klasser og interfaces er struktureret og tænk over hvad der sker, hvis der kommer nye typer enheder eller kategorier.

Start simpelt

Hvis noget virker meget uoverskueligt, så *Solve a Simpler Problem First*. Lav en løsning der er åbenlyst utilstrækkelig, men dog et skridt i den rigtige retning. Derefter er I meget klogere, og kan tage et nyt skridt.

Fx kan man starte med kun at have film og kun at have én bruger. I kan også undlade billeder til de enkelte film i første omgang.

Behandl ikke rapporten som en eftertanke

Husk at arbejde på rapporten undervejs i projektet, og at indberegne rapportskrivning i jeres tidsestimater.

Arbejdsbladene udfærdiget undervejs i projektet kan med fordel benyttes i rapport-skrivningsprocessen, således alle begrundelser for designbeslutninger og tests bliver reflekteret i den endelige rapport. Husk at bruge tegninger og tabeller – ikke kun tekst – når I skriver rapport. Rapporten skal kunne give et fyldestgørende overblik over jeres arbejde og tanker herom.

Behandl ikke tests som en eftertanke

Husk at designe jeres program således at I kan teste dets funktion. Programmet kan nemmere testes såfremt I deler det op i logiske, velafgrænsede enheder – for

eksempel ved at separere brugerfladen (view) fra programlogikken (controller) og den underliggende data (model).

Test inkrementelt

Det er en fordel at teste koden, efterhånden som I skriver den - på den måde er I bedst stillede til at identificere de interessante corner-cases som skal testes og I får samtidig sikret jer, at færre fejl kommer til at påvirke andre moduler som kodes sidenhen.

Lad bruger-opgaverne diktere jeres GUI

En god GUI er optimeret til at de hyppigste bruger-opgaver er hurtige og lette at klare. Desuden er den konsistent, dvs. prøv altid at vise den samme data, fx film, på samme måde, for ikke at forvirre brugere.

Overvej hvorledes I kan lave så få vinduer som muligt. Tænk længe over, hvordan disse bedst kan vise den gemte data og hjælpe med at visualisere opgaverne; f.eks. afspille enhed, læse om enhed, gemme enhed i egen liste.

6 Inden aflevering

Inden aflevering bør I sikre at alle krav, som er beskrevet i dette dokument, er opfyldt (gennemgå gerne dokumentet flere gange).
I kan evt. benytte følgende checkliste:
_ En *velskrevet* rapport i .pdf-format
_ Rapporten indeholder en generel introduktion til programmet
_ Der er logisk sammenhæng med overskrifter i alle afsnit
_ Der bliver ikke blandet mange forskellige emner sammen i de enkelte afsnit
_ Der beskrives ikke ting, der først introduceres senere i rapporten
_ Der laves ikke urimelige antagelser af læseren (fx at vedkommende kan læse tanker, kender jeres kode minutiøst eller ikke behøver begrundelser for påstande for at tro på dem)
_ Rapporten indeholder en reflekteret beskrivelse af arbejdsprocessen
_ Rapporten indeholder en sektion, der beskriver den overordnede teststrategi for hvert komponent i systemet
_ Samtlige test-cases er vedlagt i appendix
_ Brugermanual er vedlagt i appendix. Manualen er målrettet alle brugere af jeres streamingtjeneste og omhandler derfor ikke Java-kode eller andre nørderier
_ Rapporten imødekommer vurderingspunkterne beskrevet i "Evaluering" ovenfor
_ I har vedlagt en beskrivelse af hvordan programmet kan køres af andre. Det er jeres opgave at sikre, at programmet kan køres af undervisere og censor. Det skal være muligt for eksaminator og censor at køre projektet som en del af bedømmelsen af projektet. Hvis det er nødvendigt at have login og password skal dette vedlægges
_ Der er brugt ITU standardforside
_ En ZIP-pakke med alle Java kodefiler samt andre filer/komponenter der indgår i jeres program. ...uden IDE-relaterede filer, kompilerede .class filer, mv.
_ JAR-fil til jeres program.