SENSORVEILEDNING FOR DATA1600 PROG. UTV.

Emnekode	DATA1600
Emnenavn	Programutvikling
Studieår semester	Vår 2019
Studiepoeng	10
Emneansvarlig	Henrik Lieng
Eksamenstype	Prosjektarbeid i gruppe

Forventninger til besvarelsen.
Informasjon om hva som kjennetegner en god, middels og under middels besvarelse.

Intro. Sensuren for denne eksamen bruker et poengsystem, 0-100, med 10 delpunkter med lik poengfordeling (dvs. maksimalt 10 poeng for hvert delpunkt). Poengsystemet er designet etter UHR sine generelle beskrivelser for A-E karakterene, der blant annet helhetsinntrykk er bakt inn i poengfordelingen. Det anbefales dermed å følge poengsystemet slavisk. Spesifikke forventninger til hver av de 10 delpunktene er beskrevet under.

Generelle beskrivelser
God besvarelse: mellom 80 og 100 poeng. Karakter B: 80-89 poeng; karakter A: 90-100 poeng
Middels besvarelse (karakter C): mellom 60 og 79 poeng
Under middels besvarelse: mellom 40 og 59 poeng. Karakter D: 50-59 poeng; karakter E: 40-49 poeng.

Spesifikke beskrivelser
Punkt 1, helhetsinntrykk: Hvorvidt programmet som er utviklet oppnår den generelle beskrivelsen av oppgaveteksten.
Punkt 2, OOP kvalitet: Kvalitet av klasser etter generelle prinsipper for objekt-orientert programmering. Gjelder da oppbygning av klasser (datafelt, metoder etc.) og synergien mellom klasser. Merk at registreringselementene i oppgaveteksten beskriver programmets generelle data, og er ikke en opplisting av de klassene som utelukkende skal implementeres. Det forventes at kandidatene har tatt utgangspunkt i opplistingen i oppgaveteksten og etter dette har produsert Java-klasser med høy kvalitet (etter low-coupling, high cohesion og andre prinsipper).

Punkt 3, GUI: Brukbarhet og design av det grafiske brukergrensesnittet. Her forventes det at GUI kan enkelt navigeres i og at de forskjellige funksjonalitetene er enkle å gjennomføre. Utseendemessig, så er det et pluss at GUIet har et pent design (krav for 9-10 poeng); dvs. at CSS-markering har blitt brukt naturlig for det grafiske designet. Utover det forventes det ikke et avansert GUI, med for eksempel animasjoner og egen-designede grafiske bilder/elementer.

- Variasjon på antall gruppemedlemmer tas hensyn til under dette punktet. Grupper med 1-2 kandidater har blitt anbefalt å ikke prioritere det grafiske designet fremfor kodingen og er klar over bonuspoengene beskrevet under. Det skal fortsatt gis maksimalt 10 poeng for dette punktet, så hvis en gruppe på 2 studenter får 9 poeng for GUIet, skal de få 10 og ikke 11 poeng.
    - 2 studenter: 2 bonuspoeng
    - 1 student: 4 bonuspoeng

Punkt 4, funksjonaliteter: Funksjonalitetene i programmet skal fungere etter oppgavebeskrivelsen. Programmet burde testes kritisk, som å sjekke om kandidatene har tatt hensyn til håndtering av feil som ugyldig data.

Punkt 5, filhåndtering-lagring: Lagring av data til fil med både csv og jobj skal fungere. Filtypen jobj relaterer til filer lagret med Java serialization. Det forventes at csv filen(e) kan åpnes i Excel. Krav til teknisk løsning:

- Abstrakt grensesnitt (interface/abstrakt klasse) som representerer metoden(e) for lagring. To konkrete klasser som hver representerer lagring til en av de to filformatene.
- Korrekt feilhåndtering med avvikshåndtering. IO-feil skal kastes videre fra klassene som håndterer lagringen og fanges i klassen som er koplet til GUI (controller eller en hjelpeklasse til controller). Feil skal håndteres ved å på en naturlig måte gi beskjed til bruker (noe som ikke inkluderer feilmelding til konsollen).

Punkt 6, filhåndtering-innlasting: Innlasting av data fra fil med både csv og jobj skal fungere. Krav til teknisk løsning reflekterer teknisk løsning fra forrige punkt. Løsningene skal ikke kombineres, dvs. vi forventer en abstrakt klasse/interface for lagring og en annen abstrakt klasse/interface for innlasting.

- Abstrakt grensesnitt (interface/abstrakt klasse) som representerer metoden(e) for innlasting. To konkrete klasser som hver representerer innlasting fra en av de to filformatene.
- Korrekt feilhåndtering med avvikshåndtering. Se Punkt 5. I denne oppgaven er det to typer feil: IO-feil og feil format. For csv relaterer feil format til feil i tekstformatet, som at feil separator brukes. Det forventes at egen-definerte avviksklasser har blitt brukt for feil tekstformat. For jobj relaterer feil format til feil byte-code format, som for eksempel feil objekt (ClassCastException etc.) eller ugyldig byte data (ObjectStreamException etc.). Det anbefales å teste disse løsningene ved å forsøke å laste inn ugyldige filer.

Punkt 7, tråder: Innlastingsfunksjonaliteten skal kjøres i en egen tråd separat fra hoved-(fx)-tråden. Det er anbefalt for studentene at det inkluderes en metode for å teste denne løsningen (enten ved å laste inn en stor fil eller å emulere en større operasjon ved å sette innlesningstråden på vent i noen sekunder). Hvis det ikke inkluderes former for testing, så burde det fortsatt være greit å evaluere løsningen ut ifra koden. I den tekniske løsningen

skal det fokuseres på koordineringen mellom hovedtråden og innlesingstråden. Løsningen skal korrekt løse problemstillingen med at når innlesningen er ferdig skal innlest data lastes inn i GUI. Bruker skal ellers kunne navigere som relativt normalt, dvs. programmet skal ikke fryse mens data lastes inn. Alternativt kan ProgressBar brukes. Funksjonaliteter som fil-innlasting og manipulering av data burde være slått av mens innlesningen arbeider. Den korrekte løsningen for håndtering av tråder i dette tilfellet er å lage en klasse som utvider Task i javafx.concurrent. Denne klassen inkluderer en utility-succeeded metode som kjøres etter at tråden er ferdigkjørt og den naturlige løsningen er å bruke denne etter at innlesningsmetoden (call()) er ferdigkjørt. Løsningen for tråder og løsningene for innlastingen skal separeres, dvs. innlastingsklassene skal ikke utvide Task eller liknende.
Punkt 8, MVC: Arkitekturmønsteret model-view-controller skal brukes til å tydelig separere view-controller og programlogikk. Controllerklasser skal ikke inneholde programlogikk som generelt hører til andre klasser. View skal være deklarert i egne fxml filer. FXML bygget med SceneBuilder er tillatt. Videre, forventes det at kodestrukturen er naturlig inndelt i pakker/mapper. Programmet skal til slutt defineres som en modul, kompatibelt med Maven konfigurasjonsfilen til prosjektet. Programmet skal dermed kunne kjøres fra Maven uavhengig av IDE. For studenter som ikke har fått til Maven oppsett er det tillatt å levere koden (uten moduler) som et Java 8 IntelliJ eller NetBeans prosjekt.
Punkt 9, lesbarhet: Koden skal følge standard regler for formatering og lesbarhet. Det forventes at navn (klasser, metoder, variabler, pakker etc.) tydelig beskriver ansvaret til kode-elementet. Aktivt brukt av hjelpemetoder og liknende for å unngå store metoder og duplikat kode er et pluss, og forventes fra de beste besvarelsene.
Punkt 10, rapporter: Rapportene skal inneholde alle elementer som er listet opp i oppgaveteksten. Det forventes ikke noe konkret rundt de generelle diskusjonene, men de må være inkludert og det må være fullstendige diskusjoner (f.eks. generell refleksjon betyr ikke «Det gikk braPUNKTUM»). De individuelle rapportene må bekrefte opplisting av arbeid for hvert medlem i grupperapporten, både ut ifra oppsummering av eget bidrag og evaluering mellom gruppemedlemmene. Det forventes at de gode kandidatene tydelig får frem hvorfor de mener utvalgt kode er av høy kvalitet. En generell opplisting av kode uten diskusjon skal ikke gi noe uttelling. Hvis det kommer tydelig frem at kandidaten ikke har bidratt med signifikante kodebidrag (FXML og CSS teller ikke), skal kandidaten få 0 poeng for dette punktet.

Hvis det kommer frem, uten tvil, at kandidaten ikke har bidratt til prosjektet, skal kandidaten få strykkarakter.