Semesteroppgave for Programutvikling, V19

Hovedmålene i denne semesteroppgaven er å lære hvordan å arbeide i team, hvordan å utarbeide et større program, og lære om planlegging og gjennomføring av et større programmeringsprosjekt i gruppe. Det forventes at du arbeider 50-60 timer med semesteroppgaven, fordelt over 5 uker.

Denne oppgaven representerer eksamen i Programutvikling og karakteren du får i emnet er utelukkende basert på ditt bidrag i denne semesteroppgaven. Karakter blir satt etter følgende generelle mål:

- At du har arbeidet effektivt med medlemmene i din gruppe
- At dere har implementert et program i samsvar med oppgavebeskrivelsen
- At tekniske løsninger er utarbeidet med høy kvalitet

Hver gruppe har en unik komposisjon av personligheter, erfaringer, og motivasjonsnivå. Din gruppe må planlegge bidraget til hvert medlem av gruppen, der alle medlemmene har blitt tildelt et konkret ansvarsområde. Hvert medlem skal kunne påvirke planleggingen av deres egne bidrag og dette burde være basert på erfaring og motivasjon. Dere vil i løpet av prosjektet møte ulike administrative utfordringer, noen som kan håndteres og andre som ikke kan håndteres. Hvis dere møter utfordringer eller konflikter dere ikke løser sammen, må du eller dere i gruppen kontakte deres kontaktperson. I verste fall, kan eneste naturlige løsning være å gjøre endringer i gruppen, som å dele gruppen i to.

Obligatoriske arbeidskrav

For å kunne levere semesteroppgaven, må du ha bestått de tre obligatoriske arbeidskravene i emnet. Hvis du allerede har bestått disse de siste 2 årene, er det ikke nødvendig å få godkjent arbeidskravene på nytt. Merk at hvis du har godkjente arbeidskrav fra tidligere år, må alle arbeidskravene ha blitt godkjent. Hvis du for eksempel har fått en prøve godkjent, men du har ikke bestått en oppgave, må du få alle arbeidskrav godkjent på nytt inkludert prøven.

Oppgavebeskrivelsene for de obligatoriske oppgavene gir fullstendig informasjon angående krav for innlevering og godkjenning, samt tidsfrister for innlevering. Disse finner du på Canvas.

Innlevering

Besvarelse av semesteroppgaven skal bestå av følgende elementer:

- Kildekode. Programmet må implementeres med programmeringsspråket Java, med bruk av JavaFX biblioteket for det grafiske brukergrensesnittet. Kodeprosjektet skal være modulbasert konfigurert med Maven. Se Canvas for hvordan å sette opp prosjektet med Java 11/JavaFX/Maven.
- Grupperapport.
- Individuell rapport. Denne rapporten skal leveres i en egen innlevering og skal ikke inkluderes i gruppebesvarelsen.

Gruppeinnleveringen skal leveres i Inspera som en pakket ZIP fil. Besvarelsen skal ikke overstige 50 MB.

Tekniske krav og krav til funksjonalitet

Dere skal implementere et registreringssystem. Vedlegget på slutten av dette dokumentet beskriver tre alternativer dere skal velge mellom. Dere skal bare implementere en av disse tre alternativene.

Teknisk sett er de tre alternativene like utfordrende. Dere har derfor samme utgangspunkt for å oppnå en god karakter uansett hvilket alternativ dere velger.

Programmet dere skal implementere skal inneholde funksjonalitetene beskrevet under. Dere blir utelukkende evaluert basert på disse funksjonalitetene. Det vil si, dere kan gjerne legge til andre funksjonaliteter til programmet, men slik ekstra funksjonalitet blir ikke evaluert.

Registrering av elementer

Bruker skal ha muligheten til å registrere individuelle elementer fra et grafisk brukergrensesnitt. Hvis brukeren taster inn ugyldig data, skal brukeren få beskjed om dette. Det skal dermed ikke være mulig å legge til elementer med ugyldig data.

Det grafiske brukergrensesnittet for å legge til elementer skal være designet slik at det er enkelt for brukeren å forstå hvordan å legge til elementer. Det blir lagt noe vekt på det grafiske designet, men ikke mye. Det vil si, det skal se bra ut, men det forventes ikke noe ekstraordinært her som animasjoner eller andre dynamiske GUI funksjonaliteter.

Visualisering av eksisterende elementer

De elementene som er lagt til fra bruker og lest inn fra fil skal listes opp i det grafiske brukergrensesnittet. Det skal være mulig for bruker å kunne sortere elementene etter hver datakolonne. I tillegg forventes det at elementer kan filtreres ut ifra filtreringsmuligheter som passer for den spesifikke applikasjonen.

Bruker skal kunne velge et individuelt element, gjøre endringer på elementets data, og slette elementet. Det grafiske grensesnittet skal være designet slik at det kommer tydelig frem hvilket element som er valgt (for eksempel, ved å markere valgt rad med mørkere bakgrunn).

Lagring av data til fil

Programmet skal støtte lagring av programmets data til fil. Videre, skal programmet støtte to typer formater for lagring som brukeren kan velge mellom:

- Lagring til .csv fil kompatibelt som er kompatibelt med regnearksprogrammer som Microsoft Excel. Denne filtypen lagrer data som tekst, der hvert element er skrevet ut for hver linje i tekstfilen. Hver datakolonne separert med et tegn, som for eksempel semikolon.
- Lagring til .jobj fil med Javas støtte for serialisering. I denne filtypen lagres data i et binært dataformat som er kompatibelt med Java sine serialiseringsklasser.

Den tekniske løsningen for filhåndtering skal implementeres med en abstrakt klasse som representerer lagring til fil og to konkrete klasser som representerer lagring med de to filformatene beskrevet over. Dette er et eksempel på Strategy designmønsteret, der vi har to strategier på hvordan å lagre data til fil.

Applikasjonen skal dynamisk velge mellom de to løsningene for lagring av fil. Dette betyr at programmet må automatisk velge hvilken av de to løsningene som skal brukes ut ifra det brukeren gir som input til programmet. Her anbefales det at dere bruker JavaFX sin FileChooser for å la bruker velge fil. Da kan programmet ut ifra filtypen (csv eller jobj) velge hvilken løsning som skal brukes. Se dokumentasjonen for FileChooser for mer informasjon:

https://docs.oracle.com/javase/8/javafx/api/javafx/stage/FileChooser.html

De ulike eksterne feilene som kan oppstå ved lagring av data til fil må korrekt håndteres. Det betyr at avvik skal kastes fra klassen som lagrer dataene til fil. Slike avvik skal deretter fanges i controller-klassen, eller tilsvarende, som er koplet opp til brukergrensesnittet. Hvis feil oppstår skal controller klassen påse at feilinformasjonen blir på en naturlig måte fremstilt til bruker.

Innlesning av data fra fil

Programmet skal støtte muligheten for å laste data inn fra fil. Den tekniske løsningen for dette reflekterer løsningen fra fillagringen. Det betyr, innlasting fra to filformater skal støttes: csv og jobj. Strategy designmønsteret skal igjen brukes til å støtte innlesning fra disse to formatene. Det vil si, en

abstrakt klasse skal representere metoden for å lese data fra fil. To konkrete klasser som arver fra den abstrakte klassen vil dermed representere innlesning fra de to forskjellige filformatene.

Applikasjonen skal automatisk velge hvilken løsning som skal brukes basert på filformatet til filen brukeren velger. JavaFX sin FileChooser kan brukes til å velge en fil.

Håndtering av feil for innlesning av data er noe mer komplisert enn for lagring av data. Her skal programmet ta hensyn til ugyldig formatert data. For csv, kan dette være at data er separert med komma istedenfor semikolon. For jobj, kan data ha blitt lagret fra en utdatert versjon av dataklassen. For slik ugyldig data, skal egendefinerte avvik kastes fra filbehandlingsklassen og håndteres av en klasse der det er naturlig å sette opp feilmeldinger til bruker. Slike egendefinerte avvik skal ha navn som godt beskriver typen feil (husk at avviksklasser skal avsluttes med Exception, som for eksempel InvalidStudentException).

Funksjonaliteten for innlesning av data skal ikke «fryse» programmet. For å unngå dette, skal metoden som leser data fra fil kjøres i en egen tråd. Bruker skal kunne navigere i brukergrensesnittet samtidig som programmet laster inn data fra fil, men skal ikke kunne legge til elementer eller endre på eksisterende elementer. Det anbefales at dere lager en test-fil som er såpass stor at det tar litt tid å lese inn dataene, slik at dere får testet dette. Eventuelt kan dette emuleres ved å sette tråden for innlesning på vent i noen sekunder (med Thread.sleep).

Innlesningsmetoden kan kjøres i en tråd opprettet med en klasse som arver fra Task i javafx.concurrent. Klassene som implementerer filhåndteringen (den abstrakte klassen og de to konkrete klassene for csv og jobj) skal ikke utvide Task. Det vil si, løsningen for innlasting og løsningen for tråder skal separeres fra hverandre i kodestrukturen.

Bruk av OOP og Model-View-Controller

Programmeringsstilen objekt-orientert programmering skal brukes som den primære metoden for å håndtere kompleksitet. Et sentralt mål innen programutvikling er å utvikle kode som kan vedlikeholdes og videreutvikles. Oppgavebeskrivelsen beskriver data som skal modelleres for hvert konsept, og det forventes at dere tar utgangspunkt i denne beskrivelsen.

Klassestrukturen skal følge Model-View-Controller (MVC) mønsteret. Se Canvas for et eksempel på hvordan dette settes opp. Merk at det å sette opp en MVC struktur er relativt enkelt. Det som er vanskelig, er å beholde MVC etterhvert som mer funksjonalitet blir lagt til applikasjonen. Husk at Controller delen skal bare fungere som et bindeledd mellom GUI og back-end kode. Flytt derfor logikk-kode fra Controller-klasser til andre back-end klasser der det er mulig. Hvis dere ender opp med komplisert eller mye kode relatert til hendelser, kan det være smart å separere dette i egne klasser også (f.eks klasser som KeyEventHandler, RegisterElement etc.).

Videre, vurderes lesbarheten og den generelle kvaliteten til koden. Prinsipper fra Clean Code (bok, se Canvas) antas som gode prinsipper som burde følges.

Rapporter

Grupperapport

Gruppen må sammen utarbeide en grupperapport for semesteroppgaven. Den skal inneholde følgende elementer:

- Belyse de delene av oppgaven dere er spesielt fornøyd med.
- Diskusjon rundt elementer som dere mener kunne ha blitt gjennomført bedre og/eller aspekter i besvarelsen dere ikke er helt fornøyd med.
- Utdyp hva dere har lært i henhold til prosjektarbeid og hva dere vil gjøre annerledes for deres neste store gruppeprosjekt.
- Beskrivelse av arbeidet som har blitt uført for hvert medlem.

Individuell rapport

Den individuelle rapporten skal inneholde følgende elementer:

- Generell refleksjon av din erfaring med semesteroppgaven.
- En oppsummering av dine individuelle bidrag til gruppearbeidet.
- Evaluering av bidragene gjort av hver av de andre medlemmene i gruppen.
- Et utvalg av kildekode med forklaring som belyser et av dine bidrag. Koden skal demonstrere god kvalitet og lesbarhet. Beskriv hvorfor du mener koden holder høy kvalitet og lesbarhet.

Evaluering

Evalueringen vil være basert på punktene under. Alle punktene teller like mye i karaktersettingen.

- Oppnåelse av et fungerende program i henhold til oppgavebeskrivelsen.
- Kvalitet av klassene som representerer elementene i registeret
- Brukbarheten til det grafiske brukergrensesnittet
- Interaksjon for å legge til, modifisere, og slette elementer. Registrerte elementer skal være grafisk opplistet og skal kunne sorteres og filtreres. Feil som ugyldig data fra bruker skal håndteres på en naturlig måte.
- Lagring til fil fra GUI, med støtte for både csv og jobj. Bruk av Strategy designmønster for løsning i kode. Feilhåndtering med avvik.
- Innlesning av data fra fil, med støtte for både csv og jobj. Bruk av Strategy designmønster for løsning i kode. Feilhåndtering med egen-definerte avvik.
- Bruk av tråder for innlesning av data.
- Bruk av MVC, der back-end kode er tydelig separert fra front-end kode. Controller delen fungerer som et bindeledd mellom front og back-end, men skal ikke ha andre ansvarsområder.
- Kvalitet og lesbarhet av kode.
- Evaluering av grupperapporten og den individuelle rapporten.

Code of conduct

Kopiering av kode fra Internett eller fra ferdigstilte løsninger, samt kopiering av kode mellom grupper, tilsvarer juks. Dette vil ikke bare føre til strykkarakter, men vil også føre til utvisning fra alle norske universiteter og høgskoler.

Gruppearbeid kan være krevende, spesielt for et større prosjekt som denne semesteroppgaven. Vi tolererer imidlertid ikke uakseptabel oppførsel. Dette inkluderer mobbing eller trakassering basert på kjønn, etnisitet, hudfarge, uførhet, seksualitet, religion eller tro, eller alder.

Mobbing er en form for psykologisk trakassering. Mobbing kan være trusler som undergraver selvtillit, kompetanse, og integriteten til offeret. Psykologisk trakassering kan også inkludere kontinuerlig ufortjent kritikk, nedsettende bemerkninger, høylytt kjefting, banning og støtende ordbruk, og påtrengende oppførsel. Om du er selv et offer for trakassering eller observerer trakassering, må du rapportere dette til emneansvarlig.

Vedlegg

Dette vedlegget beskriver tre alternativer for hvilken type registreringssystem som kan utvikles i semesteroppgaven. Dere skal bare velge en av disse alternativene.

Det forventes at dere deler inn dataene til programmet med klasser av god kvalitet. Ta utgangspunkt i datarepresentasjonene og utform klasser som er korte, konkrete, og med godt definerte ansvarsområder.

Alternativ 1: Forsikring

Det skal utvikles et Java-program for et forsikringsselskap. Programmet skal kunne registrere selskapets kunder, opprette ulike typer forsikringer, registrere skader, og ubetalte erstatninger.

Registreringsdata:

Kunde

- Dato for opprettet kundeforhold
- Navn
- Fakturaadresse
- Forsikringsnummer
- Alle kundens forsikringer
- Skademeldinger
- Ubetalte erstatninger

Data for generelle forsikringer

- Årlig forsikringspremie
- Dato for opprettet forsikring
- Forsikringsbeløp
- Forsikringsbetingelser (informasjon om hva forsikringen dekker)

Båtforsikring

- Samme som forsikring, pluss:
- Eier
- Registreringsnummer
- Båttype og modell
- Lengde i antall fot
- Årsmodell
- Motortype og motorstyrke

Hus- og innboforsikring

- Samme som forsikring, pluss:
- Boligens adresse
- Byggeår
- Boligtype
- Byggemateriale
- Standard
- Antall kvadratmeter
- Forsikringsbeløp for bygning
- Forsikringsbeløp for innbo

Fritidsboligforsikring

- Samme som forsikring, pluss:
- Adresse (forskjellig fra fakturaadresse)
- Byggeår
- Boligtype
- Byggemateriale
- Standard
- Antall kvadratmeter
- Forsikringsbeløp for bygning
- Forsikringsbeløp for innbo

Reiseforsikring

- Samme som forsikring, pluss:
- Forsikringsområde (hvor forsikringen gjelder)
- Forsikringssum

Skademelding

- Dato for skade
- Skadenummer
- Type skade
- Beskrivelse av skaden
- Kontaktinformasjon til eventuelle vitner
- Takseringsbeløp av skaden
- Utbetalt erstatningsbeløp (kan være lavere enn taksert beløp)

Alternativ 2: Vikarbyrå

Det skal lages et Java-program til et vikarbyrå. Vikarbyråets oppgave er å formidle kontakt mellom personer som ønsker seg et midlertidig engasjement og virksomheter som trenger vikarer. Programmet som skal utvikles skal gjøre det lettere å få rett vikar til rett virksomhet.

Registreringsdata:

Arbeidsgiver

- Offentlig eller privat sektor
- Adresse
- Bransje
- Kontaktinformasjon
- Liste av ledige vikariater

Ledige vikariater

- Offentlig eller privat sektor
- Arbeidssted
- Arbeidsgiver
- Jobbkategori
- Engasjementets varighet
- Arbeidstid
- Stillingstype

- Krav til kvalifikasjoner
- Lønnsbetingelser
- Arbeidsvilkår
- Kontaktinformasjon
- Stillingsbeskrivelse
- Liste over søkere

Jobbsøker

- Kontaktinformasjon
- Personlig informasjon som alder
- Ønsket jobbkategori
- Utdannelse
- Jobberfaring
- Lønnskrav
- Referanser

Arbeidsforhold

- Vikaren (tidligere jobbsøker)
- Vikariatet (som nå ikke er ledig lenger)

Alternativ 3: Kulturhus

En av landets små kommuner har fått et nytt kulturhus med kinosal, teatersal og forsamlingssal. Kjernevirksomheten er å arrangere forskjellige typer arrangementer, som teater, barneforestillinger, foredrag, debattkvelder og politiske møter m.m. i tillegg til å vise filmer i kinosalen.

Det skal lages Java-program som skal kunne registrere alle arrangementer, billettsalg, opplysningsvirksomhet mot publikum, samt en oversikt/promotering av arrangementer.

Registreringsdata:

Lokale

- Lokalets navn
- Type (kino, konsertsal, foredragssal, teatersal osv.)
- Antall plasser

Kontaktperson

- Navn
- Telefonnummer
- Epost adresse
- Nettside (hvis den eksisterer)
- Eventuelt tilknyttet firma eller virksomhet
- Tekst med andre opplysninger

Arrangement

- Kontaktperson
- Type (kino, konsertsal, foredragssal, teatersal osv.)
- Navn på arrangement

- Artister/personer som skal delta i arrangementet
- Program
- Sted (hvilket lokale)
- Tidspunkt
- Billettpris
- Billettsalg (solgte/utsolgte billetter)

Billett

- Eventuelt plassnummer
- Navn på lokale
- Dato og klokkeslett
- Pris
- Telefonnummer til kjøper