Semesteroppgave: planlegging og oppsett

DATS1600 oppgavesett 1. Obligatorisk oppgave inkludert.

Dette oppgavesettet representerer planleggingsfasen av prosjektet deres. Denne fasen er har en lengde på fire uker (inkludert introduksjonsuken). I løpet av disse fire ukene skal du:

- ha blitt plassert i en gruppe
- satt deg inn i semesteroppgaven
- planlagt kodestrukturen til programmet dere skal implementere (UML klassediagram)
- satt opp et Git repository for hele gruppen
- implementert et brukergrensesnitt i FXML uten funksjonalitet
- levert inn Obligatorisk Innlevering 1 på Fronter. Denne innleveringen inkluderer del 1 av semesteroppgaven (planleggingsfasen).

Arbeidsmetoder

Ressurser:

https://en.wikipedia.org/wiki/Pair_programming

https://en.wikipedia.org/wiki/Agile_software_development

https://en.wikipedia.org/wiki/Extreme_programming

Oppgavene i dette faget er designet slik at dere får både erfaring med å programmere sammen (pair-programming) og programmere individuelt mot forskjellige deler av programmet. Oppgavene i starten av implementeringsfasen representerer kjernen av programmet. Denne kjernen skal implementeres sammen. Det er snakk om oppgavene fra 'datastruktur og grafikk' til 'filbehandling og exceptions'. I disse oppgavene, er det en rekke utvidelsesoppgaver som imidlertid kan implementeres individuelt, selv om hovedoppgavene må løses sammen. Hvis dere utvikler kjernen sammen, så vil alle gruppemedlemmene være i en posisjon der de kan bidra individuelt senere i prosjektet. Hvis imidlertid bare en student utvikler kjernen, vil det være vanskelig for de andre å bidra noe som blant annet ikke vil være positivt for karakteren (programmet vil ha mindre funksjonalitet).

Utvidelsesoppgavene er designet slik at de kan (og bør) implementeres individuelt. Her vil en student ta 'ansvaret' for den gitte delen av programmet (som f.eks. manipuleringseditoren). Det vil imidlertid være metoder og klasser som kan brukes flere steder i programmet, noe som tilsier at alle medlemmene burde ha en god oversikt over hva de andre i gruppen har implementert, slik at dere ikke implementerer samme metode eller klasse to ganger.

Karaktermessig, ligger fokuset på kvaliteten på koden og programmet. Et program med høy kvalitet og lite funksjonalitet vil få bedre karakter enn et program med lav kvalitet men med mye funksjonalitet. Det beste er så klart et program med både høy kvalitet og mye funksjonalitet der en stor grad av selvstendighet demonstreres.

På slutten av faget er meningen at dere skal gå sammen igjen og arbeide med oppgavene 'dynamisk brett' og 'tråd-programmering'. Pair-programming på dette tidspunkt vil ha stor nytteverdi fordi dere vil ha opparbeidet forskjellige erfaringer og programmeringsmetoder, noe som kan deles mellom dere studenter. Disse to oppgavene dekker også to hovedemner i faget som er ment som forberedelse til videre fag i utdannelsen.

Dette prosjektet er et myk versjon av metodikken 'extreme programming', med korte implementasjonssykluser (iterativ utvikling), tungt bruk av testing og kodegjennomgang, og antatte endringer i kodestruktur. Selv om arbeidsmetoder og prosjektutførelse i programutvikling blir presentert teoretisk i et senere fag i dataingeniørutdannelsen (Systemutvikling), vil det være nyttig å ha praktisk erfaring med iterativ utvikling. Tips:

- Sett av et tidspunkt hver uke der dere utfører 'planning game', noe som inkluderer diskusjon av følgende spørsmål:
    - Hvilke funksjonaliteter skal implementeres? (Oppdater en realistisk liste over arbeid som skal fokuseres på.)
    - Hva skal implementeres i neste iterasjon/uke, og av hvem? (Deleger oppgaver til hele gruppen eller til individuelle gruppemedlemmer som skal utføres før neste møte.)
    - Trenger vi justeringer? (Diskuter om ting går som det skal og juster deres plan deretter, sammenlikn utført arbeid med planlagt arbeid.)
- Bruk et digitalt verktøy for prosjektutførelse, helst sky-basert slik at alle medlemmer har tilgang. Jeg anbefaler å bruke et Kanban Board.
- Bruk kontaktpersonen deres (noe som kan tilsvare en 'Scrum master'). Studentassistenten som er kontaktperson for deres gruppe vil sannsynligvis ha bedre erfaring i programutvikling enn alle dere i gruppen. Vær åpen for kritikk og diskuter ikke bare faglige utfordringer, men også prosjektutførelse.

Intro til semesteroppgave

I dette kurset skal dere implementere et program som representerer det matematiske spillet 'Game of Life' (GoL), utviklet av matematikeren John Conway. Dette spillet er en passende øvelse for dette kurset fordi alle emnene i faget kan inkluderes som øvelser i implementeringsprosessen. Spillet er også visuelt attraktivt, noe som hjelper mtp. motivasjon. Før dere starter å arbeide med oppgavene under, sett deg inn i hvordan dette spillet fungerer.

Gruppene blir sammensatt etter kartleggingsprøven. I starten av semesteret vil det dermed lønne seg å individuelt sette seg inn i oppgavene og komme opp med kodestruktur som kan diskuteres i deres første møte. Det anbefales ikke at du starter med selve implementasjonen, siden dette er noe dere burde gjøre sammen i gruppen.

Programmeringsmiljø med Git

Git er presentert i en tutorialtime. En bra ressursside for Git:
https://www.sbf5.com/~cduan/technical/git

Utvikling av programmer gjøres via DRC miljøer (Distributed Revision Control). Ved korrekt bruk av DRC, vil man enkelt kunne gå tilbake til tidligere versjoner av programmet (hvis man f.eks. har gjort en feil i programmeringen) og smelte sammen kode hvis flere utviklere har arbeidet i samme kodefil. Dere kan lese om versjonskontrollsystem her:
https://en.wikipedia.org/wiki/Revision_control

Dere skal bruke Git i utviklingen av deres program, som er verdens mest brukte DRC. Bruk Internett til laste ned (og deretter installere) korrekt versjon av Git for ditt operativsystem.

Det anbefales at dere, i denne oppgaven, bruker Git skallet for å utføre Git-relaterte operasjoner, som opplasting av kode etc. Det vil være en fordel å kjenne til de underliggende kommandoene til Git før dere går over til et grafisk verktøy. Les om de forskjellige Git kommandoene her: http://gitref.org/

IDEen dere bruker skal ha integrert støtte for Git. Etter at dere har satt opp Git prosjektet deres, burde IDEen automatisk oppdage Git prosjektet. Prosjektet kan da håndteres fra IDEen istedenfor fra Git konsollen.

Dere trenger et Git repository for at hele gruppen får tilgang til dataene relatert til prosjektet. Det finnes flere skytjenester der ute som tilbyr gratis repository for mindre grupper og for studenter. Jeg har hatt god erfaring med https://bitbucket.org/ og https://education.github.com/pack (husk å sett opp private repositories).

Korrekt oppsett av prosjektet er viktig av flere grunner. For eksempel, burde dere påse at .class filer (resultatet av Java kompilering) og andre 'midlertidige' filer ikke blir delt for ikke å blåse opp størrelsen til repositoriet. Filtyper som skal ignoreres av Git angis i .ignore filen i prosjektet. Se http://git-scm.com/docs/gitignore.

Denne oppgaven er utført når alle medlemmene i gruppen har innlastet ('clone' i git) gjeldene versjon av Git repoen på sin egen datamaskin. Til slutt vil jeg anbefale at dere på gruppen bruker samme IDE slik at eventuelle forskjeller mellom IDEer ikke skaper unødvendige problemer.

Modellering (design av objekter og klassser)

Design- og arkitekturmønstre er presentert i forelesning. Konseptene presentert i forelesning er relevant for denne oppgaven.

For de fleste av dere, vil utførelse av semesteroppgaven være den første gangen dere implementerer et større program. Planlegging av dette programmet vil naturligvis være utfordrende siden dere mangler erfaring. Det er imidlertid ønskelig at dere prøver, så godt dere kan, å gjennomføre 'litt' planlegging. Mer spesifikt, skal dere lage et UML diagram som representerer arkitekturen til programmets kodestruktur. Andre typer planlegging, som prosjektdokumentering og risikoanalyse, må ikke gjennomføres. Mer formell studie av prosjektplanlegging skal gjennomføres i senere fag (som DAFE2200). Merk at strukturen dere kommer opp på nå i starten av prosjektet kan (og sannsynligvis bør) endres på underveis i prosjektutførelsen (se arbeidsmetoder).

Skum gjennom alle oppgavene i faget. Identifiser problemer som relaterer til objekt-orientert programmering. Det vil si: hvordan skal data (objekter) representeres i programmet og hvordan skal funksjonalitet (metoder) relateres til data? For hvert problem, diskuter og vurder om kjente designmønstre kan brukes til å løse problemet. Dere kan referere til design mønstrene som ble gjennomgått under forelesning. Andre designmønstre kan naturligvis også vurderes:

https://en.wikipedia.org/wiki/Software_design_pattern

Husk: målet for semesteroppgaven er ikke utelukkende at dere skal implementere et program som utfører spillet Game of Life. I tillegg, skal programstrukturen være av høy kvalitet, noe som betyr at strukturen er mottakelig for senere endringer og utvidelser. Selv om dere vet at prosjektet ender på slutten av semesteret, så skal dere planlegge programmet som om implementasjon og vedlikehold hadde foregått over mange år (minst 10 år).

Lag et UML klassediagram som representerer deres planlagte kodestruktur. Bruk et digitalt modelleringsverktøy.

Oversikt over UML programmer:
https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Ut ifra min erfaring, støtter programmet UMLet all funksjonalitet som dere trenger. Det anbefales imidlertid at dere prøver flere verktøy og velger det verktøyet dere er mest komfortable med.

Brukergrensesnitt med FXML

FXML er presentert i forelesning (uke 5). Bra ressurser for FXML:

JavaFX 8 dokumentasjon og den litt eldre JavaFX 2.2 dokumentasjonen.

Dere skal bruke FXML for å programmere det grafiske brukergrensesnittet. Synskomponenter til programmer og nettsider utvikles mest naturlig via deklarativ programmering, som HTML og XML. FXML, en variant av XML, tilbyr en slik deklarativ programmeringsfremgangsmåte.

I DAPE1400, programmerte dere GUI layout direkte via JavaFX. I dette semesteret skal dere istedenfor bruke en deklarativ fremgangsmåte med FXML, som er en abstraksjon av JavaFX biblioteket. Mer spesifikt skal dere bruke en tolk i JavaFX biblioteket som konverterer FXML kode til JavaFX objekter.

Sett dere inn i FXML ved å utføre tidligere JavaFX eksamensoppgaver fra DAPE1400. Det vil si, implementer layoutene angitt i skjermbildene under i FXML. Inkluder enkel funksjonalitet som å endre å en knapp sin tekst når bruker klikker på knappen. Hendelser skal håndteres i FXML controller klasser, noe dere må sette dere inn i.

Etter at dere har satt dere inn i FXML, programmer brukergrensesnittet til GoL. Dere kan selv velge om brukergrensesnittet implementeres direkte med FXML eller via et verktøy som JavaFX Scene Builder. På dette tidspunktet skal dere utelukkende fokusere på GUI-komponentene, og ikke funksjonalitet. Controller klassen som håndterer logikk vil derfor være tom. Hvis dere er usikre på hvilken type GUI-element som skal brukes, er det OK med å bruke knapper (Button) som bladnoder i scenegrafen.

Til slutt, opprett resten av klassene, samt metoder og interfacer, dere har angitt i deres UML klassediagram. Last opp (push) endringene til Git repositoriet. Påse at alle medlemmer får lastet ned (pull) endringene og får kjørt Java programmet.

Obligatorisk oppgave 1

Den første obligatoriske oppgaven i dette faget omhandler planlegging og opprettelse av programmeringsmiljø for GoL implementasjon. Innleveringen skal inneholde følgende:

- Tekstdokument (medlemmer.txt) som angir medlemmene i gruppen (navn + studentnr.).
- UML klassediagram for semesteroppgaven. Akseptert filformat er enten et bildeformat (som PNG) eller PDF.
- Skjermbilde av programmet, som tydelig viser de planlagte GUI-komponentene til GoL. Annoter bildet med forklaringer (av f.eks. underliggende JavaFX layout).
- Bevis for at dere har satt opp et Git prosjekt (for eksempel, skjermbildet av GitHub/Bitbucket).
- FXML dokumentet for brukergrensesnittet til semesteroppgaven (dvs. bevis på at dere bruker FXML til å representere brukergrensesnittet).

Merk at de tre første elementene er en del av semesteroppgaven (se under) og er ikke en del av obligatorisk oppgave 1. De er av praktiske årsaker inkludert i denne listen slik at dere ikke trenger å sende inn to innleveringer. Alle filer skal pakkes inn i en zip fil og leveres på Fronter.

NB: bruk ZIP, ikke lever individuelle filer på Fronter og ikke bruk RAR eller andre formater. Hvis ZIP ikke brukes, vil ikke innleveringen bli tolket som 'levert' (noe som kan føre til for sen innlevering og minuspoeng).

http://www.wikihow.com/Zip-Files-Together

Frist: 5. Februar

Innlevering, del 1 av semesteroppgaven

Planleggingsfasen av programmet teller 10 % på karakteren. For sen innlevering vil resultere i minuspoeng. Følgende elementer blir automatisk inkludert i deres besvarelse av semesteroppgaven etter at dere har levert obligatorisk oppgave 1:

- medlemmer.txt.
- UML klassediagram.
- Skjermbilde av programmet.

Frist: 5. Februar (leveres sammen med obligatorisk oppgave 1).