Generel projektbeskrivelse

1 Introduktion

Formålet med programmeringsprojektet er at give deltagerne erfaring med at designe og konstruere et simpelt produkt i form af et Java-program med en grafisk brugergrænseflade (GUI) og tilhørende teknisk dokumentation.

Projektet går ud på at implementere et mindre programværktøj, der kan hjælpe med at behandle en problemstilling fra efterårets forløb. Projektet udføres i grupper á 2-3 personer. Grupper med 3 personer forventes at lave visse udvidelser beskrevet under de enkelte opgaver.

Der er mulighed for at vælge mellem følgende opgaver:

Opgave I: Fordelingsanalyse

Opgave II: Protokolsimulering

Detaljerede beskrivelser af opgaverne kan findes via *projektsiden*:

`www.imm.dtu.dk/courses/02121/proj/projekt.html`

2 Generelle opgavekrav

Der skal konstrueres et GUI-baseret Java-program, der implementerer et værktøj med den krævede funktionalitet. Programmets konstruktion skal dokumenteres i en rapport som beskrevet i de følgende afsnit.

For hver opgave er der stillet en række basale krav til funktion og udseende. Det er op til deltagerne at uddybe og detaljere disse krav og tage stilling til eventuelle uklarheder.

Værktøjet kan eventuelt udbygges med nogle af de udvidelsesforslag, der er nævnt i hver opgave. Det anbefales kraftigt at vente med dette, til de basale funktioner er implementeret.

Programmet skal implementeres i Java (version 1.6).

3 Programdesign

Programmet bør designes efter *Model-View-Control* princippet som beskrevet i GUI-noterne. Dette princip giver en god programopdeling, tillader systematisk test af indre dele og kan evt. benyttes som grundlag for arbejdsdeling.

Opgaven kan med fordel løses i en række trin:

1. Model. Fastlæggelse af datastrukturer til modellering af systemets tilstand og mulige operationer på denne.

2. Visning. Fastlæggelse af grafisk repræsentation af tilstanden.
3. Styring (eng. *control*). Fastlæggelse af, hvordan operationerne aktiveres/vælges fra brugergrænsefladen.
4. Ekstra funktionalitet (gemning/hentning, automatisering osv.).

4 Afprøvning og demonstration

Det forventes ikke, at der laves en udtømmede test af alle programmets dele inden for kursets tidsramme, men som mininum kræves:

- At der fastlægges en *strategi* for afprøvningen. En sådan forventes at udvælge en eller flere centrale klasser i modellen til grundig komponenttest (eng. unit test) og derudover lave en funktionel test af modellen og til sidst hele systemet.
- At der forberedes en **10-minutters programdemonstration** der viser programmets væsentligste funktionalitet. Hertil skal der udarbejdes en *demonstrationsplan*, der beskriver, hvordan demonstrationen udføres. Planen bør specificere en række interaktionstrin (testcases), hvor der for hvert trin anføres:

  1. Formålet med trinnet, dvs. hvilken overordnet funktionalitet det skal vise (fx *flytning af objekt*).
  2. Den tilstand GUI-en skal starte i. [Udelades, hvis der blot fortsættes fra forrige trin.]
  3. De handlinger der skal foretages (fx *klik på objekt A, så på 'flyt'-knappen og til sidst et frit punkt B*).
  4. De forventede observerbare reaktioner i GUI-en (fx *objekt A rykker til position B*).

  Demonstrationsplanen skal inkluderes som et 1-2 siders bilag til rapporten og skal endvidere gives til hjælpelæreren **før** demonstrationen.

  [Det ses at demonstrationsplanen udgør en minimal funktionel test på GUI-niveau. Derudover bør der defineres og gennemføres mere systematiske funktionelle test i det omfang tiden tillader det.]
- At der laves en opsummering de gennemførte tests af programmet og de konklusioner der kan drages heraf.

Nogle noter om GUI-test kan findes på materialesiden.

5 Nogle råd

- I bør afholde jer fra at bruge Java-mekanismer, der ikke er gennemgået i kurset (fx. træk-og-slip), medmindre I har sat jer godt ind i dem.
- Undlad at bruge tid og kræfter på at gøre grafikken "lækker", fx ved at bruge 3D grafik, animation eller lydeffekter. En sådan produktmodning kunne fx tænkes overladt til kvalificerede multimedie-designere.

- Der lægges vægt på, at koden er læselig og forståelig. Gør god brug af metodeopdeling (og husk også at dokumentere hjælpemetoder). I forventes kun at benytte "naive" algoritmer, fx. algoritmer der eksplicit gennemløber alle muligheder. I kursus 02105 vil I senere lære at gøre algoritmerne mere effektive.
- Det anbefales at benytte Eclipse som udviklingsværktøj.
- Programkoden bør organiseres ved brug af Javas pakke-begreb. Se eksempler i GUI-noten.
- Programdokumentationen bør benytte JavaDoc-standarden. Links hertil kan findes på materialesiden.
- Ved gemning af systemets tilstand i en fil kan man enten benytte et selvvalgt tekstuelt format eller Javas automatiske *serialisering* af objekter (se materialesiden).

6 Rapportkrav

Programmet skal dokumenteres i en kort *teknisk rapport*. Formålet med rapporten er at beskrive jeres arbejde på en sådan måde, at andre hurtigt kan sætte sig ind i det, fx. med henblik på tilrettelser eller videreudvikling. Rapporten skal have følgende indhold:

- **Forside.** Skal indeholde information om hvad, hvem, hvor (institut og universitet) og hvornår. Underskrifter kan placeres her.
- **Arbejdsdeling.** Rapporten skal indeholde en overordnet beskrivelse af, hvem der har bidraget til hvilke dele af programudviklingen og rapporten. [DTU-krav.]
- **Introduktion.** Kort, med afgrænsning af hvad rapporten omhandler, dvs. angivelse af den valgte opgave og eventuelle udvidelser.
- **Kravanalyse.** Formålet med dette afsnit er at fastlægge *funktionaliteten* af jeres program. Altså, **hvad** jeres program skal tilbyde brugeren og **ikke hvordan** det kan programmeres. Spørgsmål, der bør være besvaret[1] af dette afsnit er:

  Hvilke åbne delproblemer er der i opgavebeskrivelsen? Hvilke løsningmuligheder er der for disse? Hvilke løsninger har I valgt, og med hvilke overvejelser? Er der i øvrigt uklarheder eller uspecificerede tilfælde i opgavebeskrivelsen, og hvordan har I afklaret dem? Har I indført nye begreber? Hvordan skal bruger-grænsefladen være bygget op? [Skitsér lay-out] Hvilke overvejelser har I gjort jer om hvordan man bruger programmet?
- **Programdesign og -implementering.** Formålet med dette afsnit er at give læseren et klart billede af, hvordan programmet er organiseret og hvordan dets forskellige dele arbejder sammen. Spørgsmål, der bør være besvaret af dette afsnit er:

  Hvad er den overordnede programstruktur? [Gerne illustreret, fx. med klassediagrammer.] Hvilke klasser er de vigtigste, hvad er deres ansvarsområder og centrale metoder? Hvilke klasser hører til Model, View og Control?

[1] Det vil være dårlig stil at besvare de opstillede spørgsmål slavisk. Meningen er, at det ud fra jeres beskrivelser skal være muligt at finde svarene.

Desuden for de forskellige aspekter:

Model-aspekter. Hvordan er opgavens forskellige begreber repræsenteret? Hvilke datastrukturer er valgt til at repræsentere systemets tilstand og hvorfor? Hvilke ikke-trivielle algoritmer har I konstrueret? Hvilket format bruges til at gemme og hente tilstanden?

Visnings-aspekter. Hvilke grafik-komponenter er brugt til realisering af GUI-en? Hvordan vises de forskellige dele af tilstanden? Hvordan er grafikken og modellen knyttet sammen?

Styrings-aspekter. Hvordan udføres de forskellige operationer via GUI-en? Hvornår og hvordan opdateres grafikken? Hvordan reagerer programmet på uventede hændelser? Hvordan hentes og gemmes tilstanden?

- **Afprøvning.** Hvordan de forskellige dele af programmet blevet afprøvet? Hvilke typer test er der foretaget? Hvor godt virker programmet? Hvor grundigt er det blevet afprøvet?
- **Konklusion.** Hvad er blevet opnået (og hvad ikke)? Hvordan kunne programmet forbedres eller udbygges?
- **Bilag.** Se nedenfor.

Rapporten skal være *kort*: Den forventes at være på 7-10 sider og bør **ikke overstige 12 sider** (eksklusiv forside og bilag). Detaljer kan gives i bilag med behørig henvisning. Rapporten skrives på *dansk* eller evt. *engelsk*.

En **brugervejledning** (eng. *user's guide*) skal være inkluderet som bilag. Denne skal fortælle, hvad programmet kan, hvordan programmet køres og hvordan forskellige opgaver udføres. De underliggende begreber for værktøjet kan antages kendt. Brugervejledningen forventes at fylde 1-2 sider.

Al **programkode** skal inkluderes som bilag på en sådan måde, at det er nemt at finde rundt i. Programteksten skal selvfølgelig være velkommenteret. Som nævnt forventes JavaDoc standarden benyttet, men den dermed genererbare dokumentation ønskes ikke vedlagt.

Endvidere skal alle **tests og resultater** såvel som **demonstrationsplanen** optræde som bilag.

OBS: *Hver gruppe forventes at udarbejde program og dokumentation selvstændigt.* ***Enhver brug*** *af materiale fra andre kilder (andre grupper, www, ...) skal eksplicit angives i introduktionen med detaljer om samarbejde/oprindelse. I modsat fald vil sådan brug blive betragtet som forsøg på snyd og blive indberettet til studieadministrationen.*

7 Projektaflevering

Hver gruppe skal aflevere rapporten i ét underskrevet papireksemplar senest **fredag d. 22. januar kl. 9.00** i postboksene i bygning 322's vestvendte trappeopgang. Samtidigt skal hver gruppe aflevere både rapporten og Java-programmet elektronisk via CampusNet.

Samme dag, mellem kl. 9 og 12, skal hver gruppe demonstrere deres program over for en hjælpelærer.

Det vil også være muligt at aflevere og demonstrere torsdag d. 21. januar i stedet (aflevering senest **kl. 12.00**). Dette angives ved gruppetilmeldingen.

Instruktion omkring gruppetilmelding, elektronisk aflevering samt demonstration vil fremgå af siden om *praktiske detaljer*. Rettelser, uddybninger og FAQ til opgaverne kan findes på *projekt-siden*. Væsentlige ændringer vil blive annonceret via *CampusNet*.

8 Evalueringskriterier

Ved bedømmelse af opgaven vil der blive lagt vægt på følgende:

- At kravanalysen beskriver hvilke overvejelser der er gjort i forbindelse med fastlæggelse af opgaven og valg mellem forskellige løsningsmuligheder.
- At designbeskrivelsen giver et klart billede af programmets opbygning og virkemåde.
- Den tekniske kvalitet af programmet (korrekthed, robusthed, læselighed, klasseopdeling, metodeopdeling, kodedokumentation, afprøvning)
- At brugergrænsefladen tilbyder den ønskede funktionalitet på en rimelig praktisk måde.
- At rapporten er velskrevet og læselig (struktur, sproglig kvalitet, illustrationer, lay-out)
- Om der herudover er implementeret væsentlig ekstra funktionalitet.