UltimateChessPro

Prosjekthåndbok

PO100D-A 12V

25. april 2012

Bjørn Tore Gjerde

Jonas Bo Grimsgaard

Jørgen Dalheim Olsen

Malin Livoll Schei

Forord:

Dette prosjektet er gjennomført i sammenheng med faget Programmeringsprosjekt PO100D-A 12V ved HiST, der vi fikk i oppgave å lage et sjakkspill ved å bruke Java som programmeringsspråk.

Selve hensikten med dette prosjektet var å utfordre gruppemedlemmene til å lage som nevnt over, et sjakkspill hvor det ble lagt ca. like mye vekt på både GUI og logikk. Dette innebar at vi måtte finne en måte å koble sammen logikken med GUI-en for å få hele spillet til å få et brukergrensesnitt slik at det ble brukervennlig, samt det å gi gruppen en utfordring. Dokumentering er også en del av prosjektet, der vi skal skrive en prosjekthåndbok som inneholder arbeidskontrakt, programkoden, klassediagram, et gantt-diagram som forklarer hvordan vi planla at prosjektet skulle bli utført på forhånd, møteinnkallinger og referat, timelister, statusrapporter og testing av koden.

Innholdsliste:

1.0 Oppgavebeskrivelse

Oppgaven går ut på å programme et sjakkspill der to spillere kan spille mot hverandre.

Krav til oppgaven:

- Få ett sjakkbrett med startoppstilling

- Flytte brikker og slå vekk brikker fra motstanderen. Datamaskinen skal kontrollere at det er et lovlig flytt og nekte å flytte brikken hvis det ikke er lovlig. Det er mange forskjellige regler for hvordan brikker kan flyttes. I første inkrement skal dere bare implementere de ”vanlige” reglene, ikke de mer spesielle, så som rokade, en passent og bondeforvandling.

2.0 Hvordan oppgaven ble løst

2.1 Generelt

Denne gruppeoppgaven er noe anderledes enn hva vi vanligvis har jobbet som ellers ved HiST. Gruppen består av fire personer, to fra tredjeklasse, en fra andreklasse og den siste fra førsteklasse. Dette har ført til en utfordring for gruppen som en helhet da vi har avstatt forskjellige tidsskjemaer for jobbing av denne oppgaven. For å kunne veie opp for dette har vi vært avhengig av god digital kommunikasjon mellom gruppemedlemmene. Plattform for denne kommunikasjonen har hovedsakelig vært Skype og Facebook.

Tidlig delte vi gruppen inn i to team, første team består av tredjeklassingene og det andre av de to resterende. Før en oppgave angripes, har disse to teamene samarbeidet om en tenkt løsning, for så å delegere deloppgaver innad i gruppen. Når de ulike delene av oppgaven så er løst, blir den flettet sammen og vurdert av hvert enkelt medlem.

Grunnet denne gruppeinndelingen har vi ikke hatt noen definert leder, men heller valgt en flat gruppestruktur. Årsaken til at vi valgte å gå bort ifra en leder er at det i denne gruppesammensetningen ville ført til mer arbeid, da vi er såpass spredt med tanke på tidsskjema, mulighet til å arbeide og klassetrinn. Da teamet har tidligere erfaringer ved å jobbe sammen, var vi klar over alles styrker og svakheter, og visste at vi ikke ville komme opp i en situasjon der en leder var nødvendig. Om det skulle oppstå konflikter var vi forberedt på

å følge en demokratisk fremgangsmetode, der flertallet har avgjørende bestemmelsesrett.

2.2 Bruk av litteratur

Av litteratur har vi holdt oss til

- *Programmering i Java, Else Lervik og Vegard B. Havdal, ISBN: 82-05-39050-8*.

Vi har også hatt god nytte av JavaDoc og Tutorials som Java tilbyr.

2.3 Uventede problemer

Vi har ikke støtt på noen uventede problemer, men heller forventede. Siden vi har vært klar over dette har vi tilrettelagt for at disse hendelsene ikke har påvirket fremgangen i prosjektet.

Ett av gruppemedlemmene deltok på IP Valencia og var derfor to uker i Spania under innspurten av prosjektet. Grunnet dårlig internettdekning der, førte dette til en komplisert situasjon for gruppen da alle våre kommunikasjonskanaler foregår over internett. Bruk av Subversion har også gjort det komplisert for å integrere denne personen i gruppearbeidet under oppholdet i Spania. Dette ble løst ved at dette medlemmet heller fokuserte på dokumenter og filer som kan jobbes med i frakoblet modus.

3.0 Gjennomføring av prosjektet

Vi fikk ikke velge oppgave selv i dette prosjektet, men vi hadde en del valgmuligheter med tanke på hvor komplisert resultatet skulle bli. Vi startet med å fordele litt oppgaver mellom oss på Facebook, hvor vi skulle forberede oss til det første oppstartsmøte for gruppemedlemmene. Det ble laget brukergrensesnitt-skisser, arbeidskontrakt, klassediagram og gantt-diagram som ble presentert her.

Etter dette, arbeidet vi med å lage et mer detaljert klassediagram, hvor alle gruppemedlemmene hadde kommet med tilbakemelding til utkastet. Deretter ble modell-klassene utarbeidet, slik at vi kunne ta fatt på logikken. Vi bestemte oss på dette stadiet for å ha to klasser som håndterte spillreglene, det vil si hvor en brikke kunne bevege seg. Vi gikk senere bort fra dette, og hadde en klasse (GameRules), som håndterte dette. Tanken i starten var å først finne ut, på et tomt brett, hvor en brikke kunne flytte. For så å sjekke hvor den kunne flytte på det aktuelle brettet med alle de andre brikkene. Men etter litt om og men, ble vi enige om at dette ble bare unødvendig.

Det ble også besluttet, på samme stadie i prosessen, at lovlige trekk for en brikke skulle

representeres i en todimensjonal tabell, med verdi 1 for lovlig trekk, og null for ulovlig.

Vi hadde som sagt et oppstartsmøte der vi fikk planlagt hvordan oppgaven skulle løses. Hele gruppen møttes onsdag eller torsdag rundt 09:30, der det ble opplyst om hva som var blitt gjort, og vi diskuterte hva som skulle gjøres fremover. Vi har også hatt to møter med veileder, henholdsvis 15. mars og 11. april.

Det meste av tiden ble selvfølgelig brukt til koding. Vi fulgte "Model View Control"-prinsippet da vi designet systemet. Prinsippet bygger på at man deler opp systemet i tre deler; brukergrensesnitt(View), underliggende data(Model) og sammenflettingslag(Control). Tanken er at endringer i brukergrensesnitt eller underliggende data, ikke skal ha så stor innvirking på de andre delene. Vi endte opp med å ha en kombinasjonsklasse(BoardPanel), som var både litt kontroller og litt brukergrensesnitt. Dette gjorde det enkelt å fordele programmeringsoppgaver innad i gruppen. Noen kunne jobbe med brukergrensesnittet, uten å tenke så mye på hva de andre gjorde med logikk-klassene. Brukergrensesnittet er så og si helt fritt for logikk, og det vil dermed være uproblematisk å skrive et nytt brukergrensesnitt i ettertid.

Modell-klassene ble programmert ganske tidlig i prosessen, siden disse ikke var avhengig av andre deler av systemet.

Videre ble brukergrensesnittet og logikk-klassene programmert parallelt. Like etter at vi hadde skrevet JUnit tester for logikken, fikk vi testet dem ved å faktisk flytte brikkene på brettet med regelklassen GameRules.java aktivert.

Hva gikk bra:
Vi er svært fornøyd med oppdelingen av koden. Det skal ikke være noe problem å bruke et annet grensesnitt enn det vi har per dags dato. Og vi føler derfor vi har fulgt MVC-prinsippet, som vi satte oss som et mål å gjøre.

Vi er også fornøyd med sammarbeidet når det kommer til skriving av kode. Flere teammedlemmer programmerte samtidig, og de forskjellige modulene ble sydd sammen i ettertid. Versjonskontrollsystemet Subversion fungerte stort sett bra, med unntak av et par rare feilmeldinger.

Hva gikk dårlig:
Gruppesammensetningen var ikke optimal. Den besto av personer fra tre forskjellige klassetrinn, med tre forskjellige timeplaner. Det bidro til at det ble vanskelig å jobbe sammen. De to teammedlemmene fra tredjeklasse, kunne bare jobbe om ettermiddagene og i helgene. Det ene teammedlemmet måtte være hjemme med barn på disse tidspunktene. Problemet ble til dels løst med digital kommunikasjon.

Hva kunne vært gjort annerledes:
Sånn som det er lagt opp nå, er så godt som all logikk plassert i klassen GameRules.java. Vi ser i ettertid at det kunne vært et alternativ å sette noe av logikken inn i brikken den gjelder.

Samtidig fungerer det godt slik vi gjorde det.

Erfaringer med samarbeidet:
Vi har lært at det kunne vært lurt med litt mer planlegging, når ingen av oss hadde særlig tid til å møtes. Skype og Facebook har vært flittig brukt, for å holde hverandre oppdatert på status underveis. Tatt i betraktning at vi nesten aldri var alle gruppemedlemmene samlet for å arbeide, er vi veldig godt fornøyde med hvordan samarbeidet fungerte

Resultatet:
Modellene var lette å sette sammen og inneholdt få metoder, med unntak av noen brikke-typer som hadde spesialregler i seg(bonde, tårn, konge).

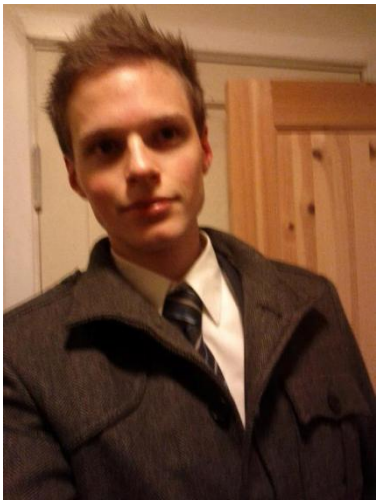
Logikken var en del verre, men det var også forventet. Metoden som sjekker om en av spillerne er satt i sjakk, viste seg å være en vanskelig nøtt å knekke.

Brukergrensesnittet er ganske enkelt laget, noe som gjør det lett for brukere å anvende spillet. På menylinjen er det en nedtrekksliste som heter “Edit”, der fargene på sjakkrutene kan endres til andre farget for å skape litt forandring.

Ekstrafunksjonalitet som ble lagt til i prosjektet, utenom vanlige trekk, og spilling av sjakk har vært; angring, lagring/henting av spill, skifte bakgrunnsfarger, bonde transformasjon, rokkade trekk samt et sidepanel som viser utslåtte brikker for hver spiller.

Vi er jevnt over fornøy med resultatet, men vi fikk noe tidsnød mot slutten. Vi skulle ønske å få implementert noe ekstra funksjonalitet, som tidtaking for hver spiller, lagring av score og replay av et spill. Det sistnevnte hadde vært enkelt å implementere, siden vi allerede lagrer alle trekk vi gjør.

3.1 Presentasjon av teamets medlemmer

	**Jonas Bo Grimsgaard:** Far til to. Samboer med ei blondine de siste 5 årene. Har vært student så lenge han kan huske. Er aktiv innen friluft og fotografering. Elsker å tilbringe tid sammen med barna sine. Er til tider sjefete, og har en ulempe for å fokusere for mye på smådeltaljer istedetfor å fokusere på helheten. Dette er også en styrke da ingenting blir overlatt til tilfeldigheter.
	**Jørgen Dalheim Olsen:** Jørgen bor sammen med to gode venner et leilighetskollektiv mellom Bakklandet og Singsaker. På fritiden bruker Jørgen mest tid på venner, spill, film, musikk, se og spille fotball og av og til hender det faktisk han slår til med litt trening.
	**Bjørn Tore Gjerde:** Lever for tiden studielivet i Trondheim, og stortrives med det. Venner, musikk, trening og data står sentralt i hverdagen, og han liker å regne seg selv som en positiv fyr.

Malin Schei: Født 26 mai 1992 kommer fra Surnadal der hun har gått på skole helt til våren 2011. Malin elsker å stå på ski, sparke fotball og spille håndball. Ser på seg selv som ei relativt aktiv jente. Har en stor tendens til å bli veldig opptatt av hvordan ting skal se ut.

3.2 Arbeidskontrakt

Følger som vedlegg.

3.3 UML Diagrammer

3.3.1 Klassediagram ved oppstart

Følger som vedlegg.

3.3.2 Klassediagram ved slutt

Følger som vedlegg.

3.4 Framdriftsplan

Dette er et ganttdiagram vi satte opp ved starten av prosjektet. Dette hjalp oss underveis til å estimere hvordan vi lå ann i forhold til tidsskjemaet vi hadde. Men som alltid viste det seg å bli litt mer jobbing mot slutten en det som var planlagt.

3.4.1 Ganttdiagram

Følger som vedlegg.

3.5 Møteinnkallinger og møtereferat

Her er alle møteinkallingene, og følgende møtereferat for de møtene, vi har hatt gjennom prosjektet.

3.5.1 Møteinnkalling 15. Mars 2012

Følger som vedlegg.

3.5.2 Møtereferat 15. Mars 2012

Følger som vedlegg.

3.5.3 Møteinnkalling 11. April 2012

Følger som vedlegg.

3.5.4 Møtereferat 11. April 2012

Følger som vedlegg.

3.6 Timelister med statusrapport

Timelister med statusrapport for hver uke følger vedlagt.

3.7 Testrapport

Vedlagt følger en testrapport.

3.8 Dokumentasjon av kildekoden

På vedlagt CD følger dokumentasjon av kildekoden som javadoc.

Vedleggsliste:

01 - Arbeidskontrakt
02 - UML - klassediagram
03 - Gantt-diagram
04 - Møteinnkalling 15.03.12
05 - Møtereferat 15.03.12
06 - Møteinnkalling 11.04.12
07 - Møtereferat 11.04.12
08 - Statusrapporter
    - Uke 11
    - Uke 12
    - Uke 13
    - Uke 14
    - Uke 15
    - Uke 16
    - Uke 17
09 - Testrapport
10 - Dokumentasjon av kildekode – følger digitalt på vedlagt CD.