UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i: IN1010 – Objektorientert programmering
Eksamenssemester: Vår 2020
Tid for eksamen: Torsdag 4. juni kl 9:00 til torsdag 11. juni kl 9:00
Oppgavesettet er på 5 sider

Spill

I dette oppgavesettet skal du utforme og skrive ditt eget program. Mange valg overlates til deg som programmerer. Det er imidlertid ikke nok at programmene fungerer, det vil bli lagt vekt på at du programmerer objektorientert slik du har lært i IN1010. Det er også noen få krav til grensesnitt, klasser og subklasser som skal være med.

Det er viktig at alle programmene dine kan kompileres og kjøres. Det er selvfølgelig best om du klarer å lage programmer som virker perfekt, men om du ikke klarer det, er det allikevel viktig at du lager programmer som kompilerer og kjører, selv om det ikke løser oppgaven helt på riktig måte. Om du har kode som bare nesten virker, kommenter den ut og skriv en kommentar med en begrunnelse for hvorfor du har denne koden der og hvilke problemer du opplever som gjør at koden ikke virker. Programmer som leveres men som ikke kompilerer og kjører, vil ikke gi særlig uttelling.

Husk at det ikke er lov å dele kode og det er ikke lov å bruke kode du ikke har skrevet selv.
Ikke bruk Æ. Ø eller Å (små og store) noe sted i programmene dine.

Alle oppgaven er gitt en vekt i poeng som er den maksimale uttellingen du kan få på denne oppgaven. Du må til sammen ha minst 40 poeng for å få karakteren bestått. For å bestå må du også ha besvart oppgave 1a) slik at resten av oppgavene kan kommunisere med bruker.

I de påfølgende oppgavene skal du programmere et enkelt spill der en (eller flere) spillere beveger seg fra sted til sted og samler gjenstander fra skattkister på hvert sted. Spilleren har et bestemt antall trekk før spillet avsluttes. I hvert trekk kan spilleren ta med seg og/eller legge igjen (selge) en gjenstand, og deretter bevege seg til et nytt sted. Målet med spillet er å samle størst mulig formue gjennom å plukke opp og selge gjenstander. Er det noe tvil om hvordan spillet du skal skrive skal virke, så gjør dine egne fornuftige antagelser og bruk fantasien slik at spillet blir morsomt. Du kan legge ved en «lesmeg.txt»-fil i enhver mappe hvis du ønsker å gi en lenger forklaring.

I oppgave 1 programmerer du klasser som gjør at spilleren kan kommunisere med brukere, inkludert en robot. I oppgave 2 skriver du en enkel versjon av spillet, uten veivalg – hvert sted har kun én utgang. I oppgave 3 skal du lage et lite GUI-vindu der resultatet vises, og i oppgave 4 skal du videreutvikle spillet slik at hvert sted har 3 utganger spilleren kan velge mellom. I oppgavene 5 og 6 skal du igjen utvide spillet, slik at flere spillere kan spille sammen.
En spiller i programmet (et objekt av klassen Spiller) styres av en bruker som bestemmer om spilleren skal ta med seg og/eller selge noe på hvert sted og hvilken utgang som skal velges videre. Brukerne kan være av to forskjellige slag: Menneskelige brukere som kommuniserer gjennom et

terminalbasert brukergrensesnitt (i kommandovinduet) eller roboter som gjør valg ved å trekke tilfeldige tall.
Når flere spillere deltar i samme spill, beveger spillerne seg fra sted til sted etter tur. Men oppgavesettet inneholder også en mulighet for at spillerne er tråder, og da beveger spillerne seg uavhengig av hverandre (og utfører ikke nødvendigvis trekkene sine i tur og orden). Når alle spillerne har utført sine trekk avsluttes spillet. Det skrives ut en resultatliste med formue for hver spiller, sortert med vinneren først.
I en kjøring av spillet er det noen parametere som du i denne oppgaven kan sette som konstanter: Filnavn for steder og gjenstander, og antall trekk. Før selve spillet starter skal programmet ditt bygge opp et terreng med steder, og plassere en skattkiste med gjenstander på hvert sted. Deretter opprettes en eller flere spillere - hver spiller har et navn og en referanse til et objekt som tilbyr metoder for å kommunisere med brukeren. Spillerne gjør så ett og ett trekk, inntil alle trekk er gjennomført. Før kjøringen avsluttes beregnes resultatene og skrives ut.

Oppgave 1. Brukergrensesnitt (10 poeng)

Spiller-objekter i spillet ditt skal kommunisere med brukere (inkludert roboter) gjennom et felles interface Brukergrensesnitt med metodene

- void giStatus(String status) som gir brukeren informasjon om det som skjer i spillet (for eksempel om det stedet spilleren er kommet til).
- int beOmKommando(String spoersmaal, String[] alternativer) som gir brukeren et valg (parameteren spoersmaal) med noen alternativer (parameteren alternativer). Metoden returnerer et heltall som angir indeksen for det valgte alternativet.

Du skal skrive to klasser som implementerer interface Brukergrensnitt for ulike brukere. Det kan lønne seg å lese oppgave 2 først, slik at du ser hvordan klassene skal brukes.

a) Klassen Terminal (6 poeng, MÅ SKRIVES) som kommuniserer med en menneskelig bruker med et kommandobasert grensesnitt i terminalvinduet ved hjelp av klassen Scanner fra Javas bibliotek (dette er slik du alltid har kommunisert med brukere før GUI ble introdusert). Klassen skal ha et Scanner-objekt som parameter til konstruktøren.
b) Klassen Robot (4 poeng) som simulerer en robot ved å trekke et tilfeldig valg blant alternativene. Du *kan* skrive ut status og valg om du ønsker, for å gjøre det morsommere å senere spille mot roboter (dette gir ikke ekstra poeng).

Lever svaret på oppgave 1 sammen med svaret på oppgave 2 i Inspera.

Oppgave 2. Et forenklet spill (40 poeng)

a) Les og forstå først hele spillet slik det er beskrevet i introduksjonen og i oppgave 1 og 2. Tegn deretter objekter som viser en typisk datastruktur mens spillet er i gang. I oppgave 2a) skal du tegne **et forenklet spill med kun én spiller, og kun én utgang fra hvert sted**: Ta med et objekt av klassen Spill, et objekt av klassen Terreng og et objekt av klassen Spiller. Tegn også noen Sted-objekter, der hvert sted har en Skattkiste med Gjenstander. Vis hvordan hvert Sted har en utgang som referer til et annet Sted. Hver spiller reiser rundt med en ryggsekk til å ha gjenstander i. Et Spiller-objekt vet til enhver tid hvor det er. Lag en pdf-versjon av tegningen din eller ta et bilde av den og lever dette som en del av besvarelsen av oppgave 2.

Du skal nå skrive noen klasser, og noen av disse skal du gjenbruke (lage subklasser av) i oppgave 4. Det kan lønne seg å lese oppgave 4 før du skriver disse klassene, slik at det blir lettere for deg å lage subklasser når du kommer til oppgave 4.

b) Skriv klassen Gjenstand med en konstruktør som gir verdier til alle instansvariablene, og med metoder for å lese dem av. Du må selv velge hvor i programmet ditt verdiene til gjenstandene skal leses inn fra filen «gjenstander.txt». Denne filen inneholder et navn og en verdi for en gjenstand på hver linje. Om det skulle være for få linjer på filen til å lage alle gjenstandene du skal bruke, kan du gjerne gjenbruke noen navn og verdier.

c) Skriv klassen Skattkiste som inneholder gjenstander. Du må selv velge hvordan skattkistene skal fylles opp ved starten av spillet og hvor mange gjenstander det skal være i hver kiste. En skattkiste skal ha metoder som gjør at en gjenstand kan tas ut av den og en gjenstand kan legges ned i den. Skattkisten er magisk, derfor velges gjenstanden som tas ut tilfeldig blant de som er i kisten. Når en gjenstand legges ned av en spiller (selges) returneres en pris (som spilleren får for gjenstanden), og siden skattkisten er magisk skal denne prisen variere litt tilfeldig i forhold til den verdien som ligger i gjenstanden.

d) Skriv klassen Sted. Konstruktøren til Sted skal ta en parameter String med beskrivelse av stedet (som i filen «steder.txt»). Klassen trenger en metode som plasserer en skattkiste med gjenstander på stedet og en metode som returnerer en referanse til skattkisten til en spiller (slik at spilleren kan bruke skattkisten). Objekter av klassen Sted har bare en utgang i oppgave 2, men det må finnes en metode for å gå videre, som returnerer en referanse til neste sted.

e) Skriv klassen Terreng for et spill der hvert sted kun har én utgang. Konstruktøren til klassen Terreng leser filen «steder.txt» som inneholder beskrivelser av steder, og bygger opp et terreng ved å opprette Sted-objekter og lenke dem sammen. Husk at hvert sted også skal ha en skattkiste med gjenstander. Klassen Terreng skal inneholde en metode hentStart() som returnerer en referanse til det stedet som er starten på spillet. Om du ønsker mer variasjon i spillet kan du lage en syklisk vei gjennom stedene, og returnere et tilfeldig valgt startsted.

f) Skriv klassen Spiller. Klassen skal ha en konstruktør som tar (minst) to parametere: Startsted og referanse til et objekt av en av klassene som implementerer Brukergrensesnitt. En spiller skal også ha en ryggsekk til å ta med gjenstander i. Som ryggsekk skal du bruke et objekt av klassen Skattkiste.
Klassen Spiller skal ha en metode nyttTrekk() som gjennomfører alle operasjonene i ett trekk. Først skal spilleren kommunisere med sitt brukergrensesnitt for å bestemme om en gjenstand skal legges igjen (selges) til kisten på stedet. Når en gjenstand selges må spillerens formue oppdateres. Hvis ryggsekken til spilleren har plass, kan deretter brukeren bestemme om en gjenstand (tilfeldig trukket fra skattkisten) skal tas med videre. Til slutt velger spilleren utgang og går til neste sted, (i oppgave 2 er neste sted gitt siden hvert sted bare har én utgang).

g) Skriv klassen Spill. I oppgave 2 er det kun én spiller i et spill. Spill skal ha en main-metode som oppretter terreng og spiller, og deretter går i løkke for hvert trekk til spillet er ferdig og resultatet (navn på spilleren og formuen) skrevet ut. Om du ønsker kan du dele opp dette i en klasse SpillKontroll (med main-metode som setter alle parametere og bestemmer typen spill når du utvider mulighetene i senere oppgaver) og en klasse Spill.

h) Kompiler og kjør programmet med filene steder.txt og gjenstander.txt (som du finner i Inspera). Bruk brukergrensesnittet Terminal. Ta bilder av skjermen som viser en kjøring. Ta også ett eller flere bilder av skjermen som viser en kjøring med brukergrensesnittet Robot, hvis du har programmert Robot-klassen.

Legg alle filene fra oppgave 1 og 2 sammen i en mappe (husk tegningen i oppgave 2a, bildene av kjøringene og de to data-filene) slik at alt som trengs for å kjøre programmet ligger i denne mappen. Komprimer denne mappen til en zip-fil. Last opp denne zip-filen under oppgave 2 i Inspera.

Oppgave 3. Grafisk brukergrensesnitt for resultatet (10 poeng)

Utvid spillet ditt slik at resultatet til slutt skrives ut i et grafisk brukergrensesnitt, et GUI-vindu.

a) Legg en knapp merket «Avslutt» som kan lukke vinduet, nedenfor resultatet (5 poeng).
b) Lag en tråd som starter opp rett før det vinduet du laget i oppgave 3a) vises fram. Denne tråden skal sove i 5 sekunder før den utføre «Platform.exit()» (og dermed lukker vinduet hvis det ikke allerede var lukket med Avslutt-knappen) (5 poeng).

Kjør programmene og ta bilder av skjermen. Legg bildene og løsningen på oppgave 3 sammen med de klassene du trenger fra oppgavene 1 og 2 i en mappe slik at alt som trengs for å kjøre programmet ligger i denne mappen. Komprimer denne mappen til en zip-fil. Last opp denne zip-filen som svar på oppgave 3 i Inspera.

Oppgave 4. Spill med flere utganger fra hvert sted (25 poeng)

Du skal nå videreutvikle spillet fra oppgave 2 slik at en spiller får oppgitt alternative utganger fra stedet den er, og bestemmer via brukergrensesnittet hvilken utvei som skal tas videre.

a) Etter at du har lest resten av oppgave 4, tegn en ny datastruktur som ligner på den du laget i oppgave 2a. Forskjellen er at nå skal hvert sted ha en liste eller array av utganger som refererer til steder der spilleren kan gå videre. Ellers er datastrukturen den samme. Ta et bilde eller lag en pdf-versjon av tegningen din og lever det som en del av besvarelsen av oppgave 4.

I det følgende skal du skrive subklasser av tre klasser du laget i oppgave 2. Du må gjerne gå tilbake til oppgave 2 og forandre klassene der slik at det blir lettere å lage subklasser av dem. Klassene du leverte i oppgave 2 skal leveres på nytt i oppgave 4, men de må gjerne være implementert forskjellig i svaret på oppgave 4.

b) Skriv klassen VeivalgSted som en subklasse av Sted. I tillegg til listen eller arrayen av utganger som refererer til steder der spilleren kan gå videre, skal klassen også ha en beskrivelse av disse utgangene (f. eks. «hoeyre», «venstre» og «rett frem»).
c) Skriv klassen VeivalgSpiller som en subklasse av Spiller. Når en VeivalgSpiller gjør et nytt trekk skal den først gjøre det samme som Spiller, nemlig selge og ta med seg gjenstander. Men istedenfor å bare gå videre til eneste mulige neste sted, må en VeivalgSpiller kommunisere med brukergrensesnittet for å velge veien videre.
d) Skriv klassen VeivalgTerreng som en subklasse av klassen Terreng som kan bygge opp et terreng med alternative veivalg. I VeivalgTerreng kan du plassere skattkister og etablere en vei gjennom alle stedene som i oppgave 2 – men i tillegg skal resten av utgangene fra hvert sted deretter lenkes mot tilfeldige andre steder.
e) Modifiser resten av klassene fra oppgave 2 slik at når spillet startes opp kan det velges mellom enkelt terreng og et terreng med veivalg, og du får et kjørbart program som virker etter hensikten.
f) Kompiler og kjør programmet på de samme filene som i oppgave 2. Ta et eller flere skjermbilder som viser en kjøring med veivalg og brukergrensesnittet Terminal. Kjør og ta skjermbilder av kjøringer med robot-grensesnittet også hvis du har det.

Legg alle filene fra oppgave 4 (husk tegningen i 4a og bildene av kjøringen(e)) sammen med de klassene du trenger fra oppgavene 1, 2 og 3 i en mappe slik at alt som trengs for å kjøre programmet ligger i denne mappen. Komprimer denne mappen til en zip-fil. Last opp denne zip-filen som svar på oppgave 4 i Inspera.

Oppgave 5. Flere spillere (5 poeng)

Hvis du ikke har løst oppgave 4 kan du basere løsningen av oppgave 5 på oppgave 2.

Du skal nå utvide spillet ditt til flere spillere. Main-metoden spør om antall spillere som skal være med, og programmet oppretter deretter spillere med navn og enten terminal- eller robotgrensesnitt. Tips: Det blir enklere å følge med hvis det som skrives ut innledes med spillernavn når et nytt trekk starter. Lag gjerne en skillelinje før eller etter at hver spiller har gjort sitt trekk.

Kompiler og kjør programmet med de samme filene som før.
Ta og lever et eller flere skjermbilder som viser en kjøring, gjerne med både robot-brukere og terminal-brukere. Legg alle filene fra oppgave 5 sammen med de filene du trenger fra oppgave 1, 2, 3 og 4 i en mappe slik at alt som trengs for å kjøre programmet ligger i denne mappen. Komprimer denne mappen til en zip-fil. Last opp denne zip-filen som svar på oppgave 5 i Inspera.

Oppgave 6. Spillere som egne tråder (10 poeng)

Også i denne oppgaven skal du utvide spillet til flere spillere, men på en annen måte enn i oppgave 5, slik at oppgave 6 kan løses uavhengig av oppgave 5. På samme måte som for oppgave 5 kan du løse oppgave 6 basert på oppgave 2 hvis du ikke har løst oppgave 4.

Du skal nå utvide spillet ditt til flere spillere slik at hver spiller utføres av en tråd. Main-metoden spør om antall spillere, og programmet oppretter spillere med terminal- eller robotgrensesnitt.

Du har to utfordringer:

1. Du må sørge for at bare en spiller/tråd har adgang til skattkisten på et sted om gangen
2. Du må la én og én tråd/spiller kommuniserer med terminalen om gangen. Det er derfor i orden at parallelliteten mellom terminal-trådene ikke er særlig høy.

Vi krever ikke noen perfekt løsning på dette. Skriv en kommentar i Spiller-klassen eller i en «lesmeg.txt»-fil om hva du burde ha gjort og hva du gjorde for å løse oppgaven. Kompiler og kjør programmet med 3 spillere og 3 tråder.

Legg alle java-filene som trengs for å kjøre programmet (dvs. ta også med klasser du laget i oppgavene 1, 2, 3 og 4) og bilder du tar av kjøringen av oppgave 6, i en mappe som du lager en zip-fil av. Last opp zip-filen i Inspera som svar på oppgavene 6.

Lykke til!

Siri Moe Jensen og Stein Gjessing