Sensorveiledning IN1010 våren 2020 Spill

Versjon 22. juni 2020

En ukes hjemmeeksamen med varighet inntil 2 dager.

Besvarelsene skal vurderes til bestått / ikke bestått.

IN1010-eksamenssettet er vektet og for å bestå må kandidaten ha 40 poeng pluss at oppgave 1a må besvares. Det står i oppgavesettet: «Programmer som leveres men som ikke kompilerer og kjører, vil ikke gi særlig uttelling».

Hver oppgave sensureres og antall poeng på hver oppgave bestemmes. Her er forslag til trekk i tilfellet oppgaven ikke er helt perfekt besvart:

Trekk		
0 - 10%	Oppgaven er nesten perfekt.	Gir vanligvis en A
10 - 20%	Oppgaven er svært bra løst.	Vanligvis en B
20 - 40%	Oppgaven er middels bra løst	Vanligvis en C
40 - 50%	Det er noen alvorlige mangler i løsningen	Vanligvis en D
50 – 60%	Løsningen har store mangler	Vanligvis en E
60 - 100%	Løsningen har så store mangler at den kan ikke ansees som løst - stryk	

Generelt om kode som ikke kompilerer og kjører: Er koden fornuftig og ser ut til å være skrevet for å løse oppgaven slik studentene har lært i IN1010, kan det gis noen poeng. Det gis ikke poeng for kode som ikke løser problemet og heller ikke for svært dårlig (inkludert ikke-objektorientert) kode.

Om sensurering av de enkelte oppgavene:

Oppgave 1. Brukergrensesnitt 10 poeng

Her skal studentene vise at de forstår interface, at de skjønner at flere klasser kan implementere et interface på forskjellige måter og at en klasse som implementerer et interface må definere instansvariabler og innmaten i metodene.

Her er det viktig at interfacet Brukergrensesnitt er riktig programmert og at minst én klasse, Terminal, implementerer dette. Om klassen Terminal ikke implementerer dette interfacet (men allikevel gjør som det skal) vil oppgaven 1a være godkjent, men det må gis svært lite poeng. Om klassen Terminal ikke er skrevet eller åpenbart ikke virker får kandidaten karakteren ikke bestått på hele oppgavesettet (det står i oppgaveteksten at oppgave 1a må være bestått for å få karakteren bestått). Dette er rimelig fordi temaet er en overordnet og sentral del av pensum som studentene har trent på og har vist at de kan, for eksempel i obligatoriske oppgaver. Hvis dette punktet ikke er løst vil de resterende oppgavene ikke kunne kommunisere med en bruker. **Om resten av oppgavene kjører og allikevel klarer å kommuniserer med bruker og er besvart til en klar bestått-karakter i seg selv, skal kandidaten allikevel få bestått.**

Oppgave 1a. 6 poeng
Det gis poeng for a) riktig konstruktør b) riktig utskrift c) riktig innlesing av svar fra terminal, d) riktig retur av svaret.

Oppgave 1b. 4 poeng
Her KAN det gjøres utskrift, men det viktige her er at beOmKommando trekker et tilfeldig tall i lengden av arrayen «alternativer» og returnerer denne verdien.

Oppgave 2 Spill med én vei videre 40 poeng

Dette er oppgavesettets klart viktigste oppgave og her skal kandidatene vise at de kan programmere et litt større program strukturert og objektorientert. Eventuelle trekk i poengsummen må gis som en helhetlig vurdering av alle aspekter ved tegningen og programmet.

Det er viktig at spillet kompilerer og kjører med kommunikasjon med bruker. Om spillet ikke kompilerer og kjører må det gis mindre enn 16 poeng på denne oppgaven (under grensen for stryk på denne oppgaven isolert sett). Om spillet kompilerer og kjører kan det allikevel gis svært få poeng (under 16) hvis koden er svært uryddig, ikke er objektorientert eller på andre måter ikke er skrevet slik studentene lærer i IN1010.

Oppgave 2a.
Her må det tegnes datastrukturer med objekter og instansvariabler som inneholder referanser til andre objekter på tegningen. Se løsningsforslaget.

Oppgave 2b – 2g
Her må de forskjellige klassene skrives rimelig riktig for at programmet i det hele tatt skal virke. Hver av klassene må inneholde det som står beskrevet i oppgaveteksten for å få full uttelling. Se også løsningsforslaget. For hver del som mangler trekkes poeng.

Oppgave 2h: “The proof of the pudding . . . “
Hvis programmer som leveres ikke kompilerer og kjører, må sensor se på koden. Hvis alt virker vel strukturert og koden er veldig bra kan det gis opp til 16 poeng.

Virkende program med alle klasser representert og OK kode bør minst gi 32 poeng (Karakteren B eller bedre). Avvik fra krav i oppgaveteksten vil normalt gi trekk, men om det resulterende programmet allikevel er godt strukturert og objektorientert og virker gis det svært lite (om noe) trekk for avvik.

Hvordan skattkisten virker er ikke av særlig betydning bare studenten viser hvordan gjenstander tas ut av (også tilfeldig) og legges ned i skattkisten (og returnerer en pris som varierer litt tilfeldig fra gjenstandens egentlige pris). Kreative alternativer som gir et godt spill men som avviker fra oppgaveteksten (uten å forenkle programmeringen) vil gi full uttelling.

Oppgave 3 GUI 10 poeng

Her skal studentene vise at de klarer å opprette et vindu og at de skjønner hendelseshåndtering og klarer å lage en metode som behandler en hendelse.

Oppgave 3a.
Her gis det poeng for: Vinduet kommer opp, resultatet synes i vinduet, det er en knapp i vinduet og denne knappen lukker vinduet når den trykkes på. Det må finnes en hendelseshåndteringsmetode som er knyttet opp mot knappen «Avslutt».

Oppgave 3b.
Her gis det poeng for å lage og starte en tråd (før vinduet åpnes), det gis poeng for at tråden venter (sleep) og det gis poeng for at tråden lukker vinduet når den er ferdig med å vente.

Oppgave 4. Spill med flere utganger 25 poeng

Her skal studenten vise at de kan strukturere et program med subklasser og polymorfi og at de skjønner og kan programmere med referanser til flere andre objekter og til objekter i et klassehierarki.

Her (som i oppgave 2) må det gjøres en helhetsvurdering av hele programmet. Trekk i poengsummen må eventuelt gis som en helhetlig vurdering av alle aspekter ved tegningen og programmet.

Det er viktig at spillet kompilerer og kjører med kommunikasjon med bruker og at det er mulig å velge veien videre. Om spillet ikke kompilerer og kjører må det gis mindre enn 10 poeng på denne oppgaven (under grensen for stryk på denne oppgaven isolert sett). Det spiller ingen rolle om programmet bruker GUI eller ikke til resultatene.

Oppgave 4a.
Her må det tegnes datastrukturer med objekter og instansvariabler som inneholder referanser til andre objekter på tegningen. I forhold til oppgave 2a er det viktig i oppgave 4a at hvert sted har en liste eller array av referanser til neste sted å gå til. Se løsningsforslaget.

Oppgave 4b – 4d.
Om ingen av klassene er subklasser gis det et trekk på minst 20% (maksimum 20 poeng på hele oppgave 4. Hvis klassene er subklasser er det også viktig at dette er gjort på en fornuftig måte. Dårlige designede subklasser og dårlig bruk av subklasser gir også lett et trekk på 5 poeng.

Oppgave 4e – 4f.
Hvis programmet som leveres kompilerer og kjører, og følger noen av de prinsippene for programmering som foreleses i IN1010, er det vanskelig å gi mindre enn 10 poeng på oppgave 4.

Hvis programmer som leveres ikke kompilerer og kjører, må sensor se på koden. Hvis alt virker vel strukturert og objektorientert med god kode kan det gis opp til 10 poeng

Oppgave 5 Flere spillere 5 poeng

Her skal studenten vise at de har oversikt over et større program og kan modifisere dette.

Alt som trengs her er en for-løkke som genererer spillere og en for-løkke som ber alle spillerne trekke. Se løsningsforslaget. Grunnen til at det bare gis opp til 5 poeng her er jo at løsningen er så enkel og krever så lite kode.

Det gis trekk for lang og komplisert kode.

Oppgave 6. Flere spillere med tråder 10 poeng

Her skal studentene vise at de kan programmere med tråder og felles ressurser.

Her er det viktigste at det starter en tråd for hver spiller og at disse trådene går i løkke i sine run()-metoder og kaller spillerne sine nyttTrekk()-metode. Se løsningsforslaget. Ellers må sensor se på helheten i programmet.

Spillerne må passe på å ikke aksessere de samme skattkistene samtidig (slik det står i oppgaveteksten). Når programmet kjører bør utskrift fra forskjellige tråder gå minst mulig i bena på hverandre, men her gis det lite trekk selv om dette skjer.

Sensor må se i koden for å se hvordan disse to utfordringene løses. Sensor må også lese lesmeg.txt-filen eller kommentarer i koden for å se om studenten ønsker og klarer å lage kritiske regioner for å beskytte hver skattkiste og utskriften. Det trekkes ikke selv om trådhåndteringen gir mye overhead ved synkronisering i forhold til uthyttelsen av parallellitet.

I tillegg bør besvarelsen inneholde en beskrivelse av hvordan dette gjøres som finnes enten i Spiller-klassen eller i en egen lesmeg.txt-fil.

Slutt sensorveiledning IN1010 V2020