Sensorveiledning IN1010 våren 2021

En skog med stier og turgåere

I dette oppgavesettet skal studentene vise at de behersker å sette seg inn i et nytt domene og programmere forskjellige oppgaver i dette domenet på en objektorientert måte.

Oppgave 1. Skogen, stiene og stikryssene 30 poeng

I denne oppgaven skal studentene vis at de forstår og kan implementere klasser i et klassehierarki med arv som også inneholder et interface. De skal også vise at de forstår beskrivelsen av og kan programmere en kompleks og ny datastruktur.

Oppgave 1a 3 poeng

Tegn klassehierarkiet for stiene i skogsområdet

I denne deloppgaven skal studentene vise at de ut fra en tekstlig beskrivelse av et domene kan lage og tegne et klassehierarki. Det viktigste som bør være med er:

- Superklassen Sti er abstrakt.
- Det er to grener av hierarkiet med de to klassene som implementerer interfacet nederst.
- Interfacet er tydelig angitt som et interface (tegnet eller tekstlig).

Oppgave 1b. 3 poeng

Tegn en datastruktur med tre objekter av klassen Kryss og to objekter av en subklasse av klassen Sti. Du trenger ikke ta med metoder.

I denne oppgaven skal studentene vise at de forstår og kan illustrere i en tegning hvordan objekter i en kompleks datastruktur refererer til hverandre. Det legges vekt på:

- Tegningen er grei å forstå. Alle referansevariabler er tegnet som variabler og referansene (pekerne, pilene) ut fra disse starter inne i disse variablene og ender i kanten av et objekt
- Tegningen viser hvordan hvert Kryss-objekt har en datastruktur som refererer 0-n stier ut fra dette krysset
- Hvert Sti-objekt refererer de to kryssene i hver ende

Oppgave 1c. 23 poeng

Skriv klassene Skog, Kryss og alle klassene og interfacet du beskrev i oppgave 1a.

I denne oppgaven skal studentene vise at de kan implementere et interface, bruke exceptions og klasser med arv. Videre at de kan programmere på en god måte ulike typer relasjoner med datastrukturer som arrayer og referanser mellom objekter. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Instansvariabler og –metoder/ konstruktører for Sti-hierarki med interface Utsikt er plassert i rett klasse med angitt funksjonalitet, konstruktørene kaller super() riktig.
- En subklasse av RunTimeException er implementert og instansieres riktig.
- Klassen Kryss med datastruktur som refererer Stiene (f.eks. en ArrayList) og metodene hentTilfeldigSti og isolert.
- Klassen Skog oppretter og refererer til kryssene.
- Klassen Skog oppretter riktig antallet stier, trekker med ~lik sannsynlighet type sti, oppretter Sti-objektet, setter endepunktene til to tilfeldige kryss og setter stien inn i de to kryssene.

Oppgave 1d. 1 poeng

Skriv klassen Trekk

I denne oppgaven skal studenten vise at han/ hun kan deklarere en statisk metode som tilpasser funksjonaliteten i en annen metode til sitt formål. Det legges vekt på:

- trekkInt er statisk. Returverdi er av riktig type, i riktig intervall med ca korrekt fordeling.

Oppgave 2. Simulering av turgåere i skogen 45 poeng

I denne oppgaven skal studentene vise at de kan lage og bruke et ganske komplekst rammeverk med en beholder som implementerer en spesiell kø, en abstrakt superklasse som definerer en byggekloss (aktivitet) og en generell klasse for simulering. I tillegg skal dette rammeverket brukes i en spesiell kontekst (turgåere i skogen).

Oppgave 2a 4 poeng

Skriv klassen Aktivitet

Her skal studentene vise at de kan lage en abstrakt superklasse og implementere en lenket liste med nytt design. Kontroll av verdier som oppdaterer tid kreves ikke her (se 2f). Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Klassen Aktivitet er abstrakt og implementerer Comparable.
- compareTo er programmert riktig.
- Inneholder en instansvariabel "tid", samt forrige- og neste-pekere.

Oppgave 2b 10 poeng

Skriv klassen PrioKo

Her skal studentene vise at de kan manipulere en lenket liste som de selv implementerer. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Inneholder første- og siste-pekere.
- Klarer å sette inn helt bakerst, helt først og i mellom.
- Tar hensyn til tom liste og virker også når det er bare ett element i listen.
- Tar ut første element i normaltilfellet.
- Setter både første og sistepeker til null når lista blir tom.
- Setter inn ved å lete bakfra (i elementer med størst tid).

Oppgave 2c 10 poeng

Skriv den fullstendige klassen Simulator

I denne oppgaven skal studentene vise at de kan implementere en spesifisert kompleks algoritme. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Det opprettes en PrioKo i klassen.
- Konstruktør som tar en array av Aktiviteter som parameter og går gjennom hele arrayen og setter alle aktivitetene der inn i PrioKo-objektet.
- Klassen inneholder en metode simuler(int t) som går i løkke så lenge globaltiden er mindre enn parameteren t.
- Løkken i simuler henter en aktivitet ut, oppdaterer globaltid, utfører handlingene i aktiviteten og setter aktiviteten tilbake i PrioKo-objektet.

Oppgave 2d 10 poeng

Skriv klassen Turgaaer.

I denne oppgaven skal studentene vise at de kan bruke et rammeverk de har laget selv og implementere en spesifisert algoritme (handlingen som gir oppførsel i et kryss i skogen). Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Klassen er en subklasse av Aktivitet.
- Inneholder instansvariablene hastighet og krysset den er i / kommer til.
- Har en konstruktør som setter startkryss og hastighet.
- Implementerer metoden handling().
    - Finner et tilfeldig kryss å gå videre til
    - Finner tiden det tar å gå stien til det nye krysset
    - Setter krysset den er i /kommer til som det nye krysset
    - Oppdaterer egen tid med ankomsttid til dette nye krysset

Oppgave 2e 9 poeng

Skriv klassen TestSimulator med en main()-metode.

Her skal studentene vise at de kan instansiere et rammeverk og bruke det. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Klassen inneholder konstanter for antall kryss, stier og turgåere.
- Et objekt av klassen Skog opprettes med riktige parametere til konstruktøren.
- Det opprettes en array av typen Aktivitet eller Turgaaer og tilsvarende mange Turgaaer-objekter.
- Simulatoren opprettes med arrayen som parameter til konstruktøren og metoden simuler() kalles med simuleringstiden som parameter.

Oppgave 2f 2 poeng

Her skal studentene vise at de kan lage en abstrakt superklasse mer robust ved å sjekke lovlighet av nye verdier i en set-metode, og bruke modifikatorene i Java for å kontrollere tilgang til instansvariabelen. Det legges vekt på:

- Aksessmodifikator for instansvariabelen tid i Aktivitet2 er «private».
- En metode for å sette ny tid i Aktivitet2 sjekker at tiden øker.
- Det finnes en metode i Aktivitet2 for å lese av verdien i tid.
- Metoden for å endre tid i Aktivitet2 er «protected».

Oppgave 3: Turgåere som tråder 15 poeng

Her skal studentene vise at de behersker å lage tråder og programmere riktig når trådene modifiserer en felles datastruktur. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- main()-metoden oppretter skogen som før oppretter og starter deretter en tråd for hver turgåer.
- Klassen Turgaaer implementerer Runnable.
- Aktiviteten i handling() metoden er nå implementert i run()-metoden.
  Forskjellen fra handling()-metoden er at
    - turgåeren kaller turgaaerKommer (som ber om sete) i krysset
    - Deretter sover den en stund (kaller Thread.sleep)
    - Deretter kaller den en metode i krysset som frigir setet

    - Til slutt kaller den Thread.sleep() for å sove så lang tid det tar å komme frem til neste kryss.
- Data som endres av tråder (ledige plasser i klasen Kryss) beskyttes med lås og Condition-variabel. Metoder som modifiserer ledige plasser beskyttes av låsen.
- Venting på ledig plass gjøres ved hjelp av await() ved ankomst og signal() ved avgang, await() beskyttes av try-catch (InterruptedException e).

Oppgave 4: Unngå at skogen inneholder rundturer 10 poeng

I denne oppgaven skal studentene vise at de kan skrive og bruke en rekursiv metode. Det legges vekt på:

- Helhet, inkludert velstrukturert og oversiktlig kode.
- Det er deklarert en rekursiv metode finnesVeiTil i klassen Kryss.
- Metoden har en rekursjonsbunn (returnerer true om den kalles på et kryss som også er argument).
- Metoden kaller seg selv korrekt på nabokryssene - unntatt det nabokrysset den ble kalt fra (sjekkes typisk mot en parameter som angir forrige kryss).
- Konstruktøren i Skog trekker som før to kryss, og kaller finnesVeiTil i det ene med det andre som parameter. finnesVeiTil kalles på nytt inntil den returnerer false, men maksimalt 100 forsøk.