OBJ2000 H24, Arbeidskrav

Innleveringsfrist: Mandag 4.11. kl 23:59 Versjon: 2024.1.0

Obligatorisk oppgave 1: Diktgenerator

Du skal lage et program som skriver dikt (grafisk eller ikke-grafisk). Du skal lage en enkel versjon uten strenge krav til hvordan diktene skal se ut. Legg vekt på å beskrive programmet med pseudokode og klassediagram før du begynner å programmere. Trinnvis forfining av programmet vil være en god idé. Legg vekt på først å få hovedkravene til programmet oppfylt, deretter kan du utvide med mer funksjonalitet. Legg også vekt på kommentarer i programkoden, samt gode ledetekster i menyen.

Meny grafisk kan f.eks. lages ved å bruke klassen JOptionPane. Hovedmenyen kan ha to valg: Enkelt dikt og avslutt. De to første valgene skal åpne nye menyer med tre valg: registrer ord, skriv dikt og avslutt.

Programmet skal bruke kontrollobjekt til å kontrollere programutførelsen og et grensesnittobjekt for kommunikasjon med brukeren. Kontrollobjektet skal bruke en array til å håndtere ordene. Innlesing av ord gjennom grensesnittobjektet kan gjøres ved å bruke klassen JOptionPane.

Arbeidet gjøres individuelt. Innlevering som zip-filer av prosjektet i Canvas.

Bruk etternavnet ditt som en del av både prosjektnavnet og filnavnet. Det gjør det lettere for meg å teste programmene.

Enkel diktgenerator

Diktgeneratoren skal lage meningsløse dikt. Det trenger følgende funksjoner, som kan være utgangspunkt for menyvalg:

- Registrere ord
- Skrive dikt
- Avslutt

Avhengig av hvilke ord som er lest inn, skal nå programmet kunne lage et ”tulledikt” på ett vers, der verset har 4 linjer og hver linje 4 ord. Ordene skal velges tilfeldig fra de ordene du har tastet inn:

håper gulrot virrer grønn
vinter snerrer borte lenge
snakker borte grønn inne
derfra sover gulrot borte

Vær oppmerksom på at dere trenger en del ord i ordlisten for at det skal bli noe særlig ut av diktene.

Obligatorisk oppgave 2: Dataklynge

Innledning

I denne oppgaven skal du lage et program for å holde oversikt over alle komponentene i en dataklynge ("computer cluster" på engelsk). En slik dataklynge kan brukes til å fordele tunge beregninger på mange maskiner slik at de kan jobbe i parallell. På den måten kan en simulering som ville tatt en måned å kjøre på en vanlig maskin, kjøres på dataklyngen på noen timer i stedet.

Dataklyngens bestanddeler

En dataklynge består av ett eller flere rack (et kabinett med skinner) hvor flere noder kan monteres over hverandre. En node er en selvstendig maskin med et hovedkort med minne og én eller flere prosessorer, i tillegg til en del andre ting. I denne oppgaven skal vi bare se på antall prosessorer (maks 2) og størrelsen på minnet. Du kan derfor anta at en node har enten en eller to prosessorer og et heltallig antall GB med minne.

Programdesign

Programmet skal designes med tre klasser som representerer henholdsvis node, rack og dataklynge. Et objekt av klassen dataklynge skal kunne referere til ett eller flere rack-objekter, der hvert rack-objekt igjen refererer til ett eller flere node-objekter. Noen flere krav og tips til design av den enkelte klassen finner du videre i oppgaven.

Under overskriften **Oppgaver og levering** nedenfor finner du beskrivelsen av hva programmet ditt skal kunne utføre. Det kan i tillegg være nyttig å implementere andre hjelpemetoder - gjør egne vurderinger og gi en begrunnelse for disse gjennom kommentarer i koden.

Klassen Node

Når et Node-objekt blir opprettet, opprettes dette med ønsket minnestørrelse og prosessorantall. Disse er klassens instansvariabler og settes i klassens konstruktør.
For øvrig skal klassen tilby tjenester (metoder) som trengs i andre deler av programmet. (Hvilke metoder som trengs vil bli klarere i del C og D.)

Klassen Rack

Klassen Rack skal lagre Node-objektene som hører til et rack i en liste. Vi skal kunne legge til noder i racket hvis det er færre enn maks antall noder der fra før. For enkelhets skyld skal vi anta at hvert rack i dataklyngen har plass til like mange noder. Opprett andre instansvariable og metoder etter behov. (Del C og D)

Klassen Dataklynge

Klassen Dataklynge skal holde rede på en liste med racks, og må tilby en metode som tar imot et nodeobjekt og plasserer det i et rack med ledig plass. Hvis alle rackene er fulle, skal det lages et nytt Rack-objekt som legges inn i listen, og noden plasseres i det nye racket.

Tips: Det kan være lurt å ta inn antall noder per rack i konstruktøren til Dataklynge.

Oppgaver og levering

Oppgavene du skal løse er beskrevet under. Lever den ferdige løsningen i Canvas med .java filen til hver klasse:

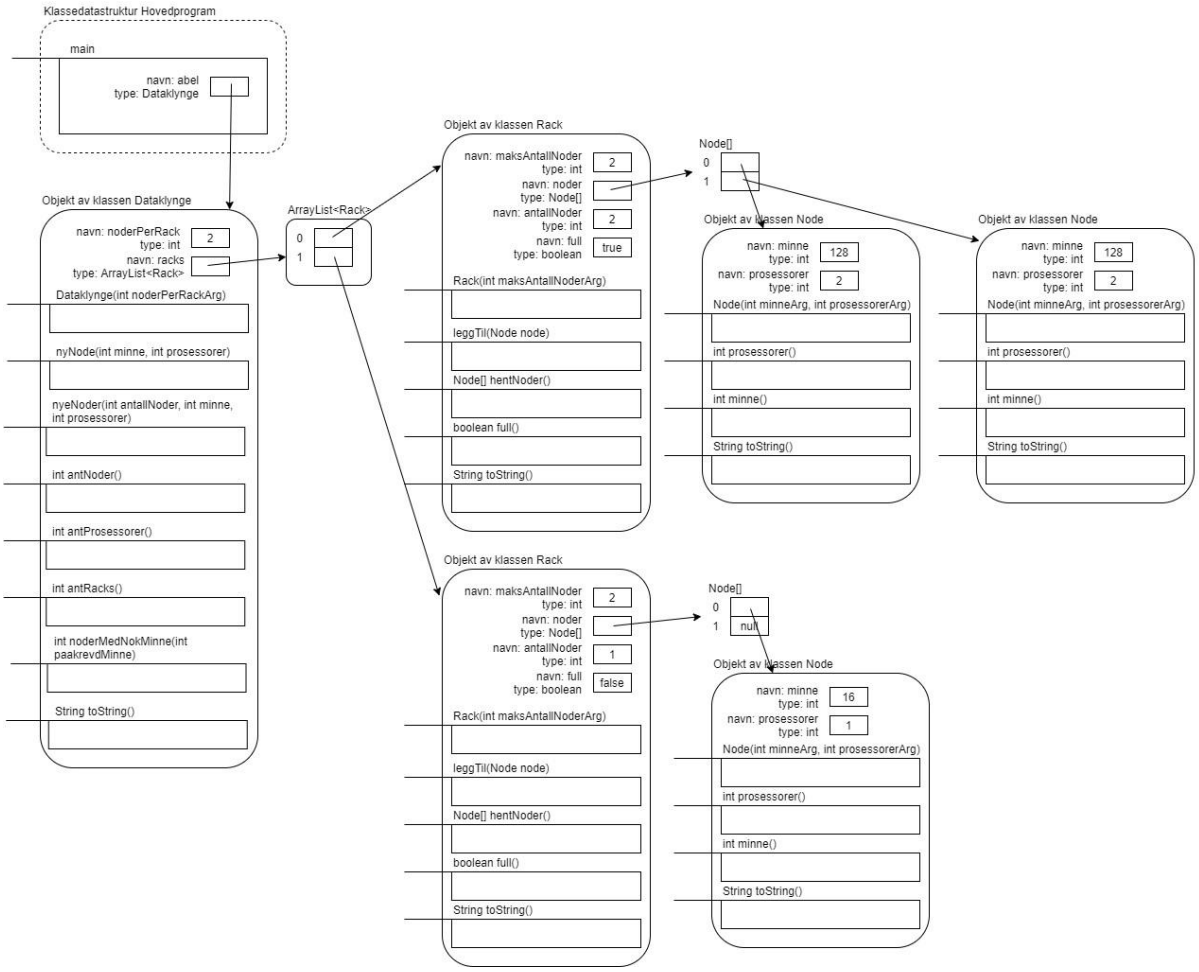
- Dataklynge.java
- Node.java
- Rack.java
- Hovedprogram.java

Legg også ved eventuelle tekstfiler som er nødvendige for at hovedprogrammet skal fungere.

Besvarelsen skal kunne kjøres. Det vil si at de skal kunne kompileres og kjøres med Java, og IDE'er skal ikke være nødvendig for at programmene skal fungere.

Del A: Datastrukturtegning

Studer tegningen under for hvordan løsningen kan eventuelt kan se ut. Hvis du gjør noe annerledes, så skal du tegne og forklare ved å legge ved en ny tegning (bruk f.eks. draw.io).



Del B: Klasser

Skriv klassene beskrevet under “Programdesign”. Der det ikke er oppgitt hva slags datatype som skal brukes (for eksempel ved valg av listetype) forventes det at du selv gjør fornuftige valg og begrunner disse gjennom kommentarer i koden.

Del C: Antall prosessorer og minnekrav

Lag en metode antProsessorer i Dataklynge som returnerer det totale antall prosessorer i dataklyngen.

Noen programmer trenger mye minne, typisk et gitt antall GB med minne på hver node vi bruker. Vi er derfor interessert i å vite hvor mange noder som har nok minne til at vi kan bruke dem. Lag en metode noderMedNokMinne(int paakrevdMinne) i Dataklynge som returnerer antall noder med minst paakrevdMinne GB minne.

Utvid klassene Node og Rack slik at de støtter implementeringen av disse metodene.

Del D: Hovedprogram

Skriv en klasse Hovedprogram med en main-metode for å teste at klassene virker som de skal. Lag en dataklynge, med navn abel (https://no.wikipedia.org/wiki/Niels_Henrik_Abel), og la det være plass til 12 noder i hvert rack. Legg inn 650 noder med 64 GB minne og en prosessor hver. Legg også inn 16 noder med 1024 GB minne og to prosessorer.

Sjekk hvor mange noder som har minst 32 GB, 64 GB og 128 GB minne. Finn totalt antall prosessorer, og sjekk hvor mange rack som brukes. Skriv ut svarene i terminalen. Utskriften kan f.eks. se slik ut:

```
Noder med minst 32 GB: 666
Noder med minst 64 GB: 666
Noder med minst 128 GB: 16
Antall prosessorer: 682
Antall rack: 56
```

Del E: Lese fra fil

Skriv en ny konstruktør til klassen Dataklynge. Denne skal ha et filnavn som parameter (i stedet for max antall noder per rack), og lese alle data om dataklyngen fra fil. Filen er bygget opp slik at første linje beskriver antall noder per rack, mens de neste linjene beskriver hvor mange noder, med hvor mye minne og antall prosessorer som skal settes inn, slik:

```
Max antall noder per rack
AntallNoder MinnePerNode AntallProsessorerPerNode
AntallNoder MinnePerNode AntallProsessorerPerNode
...
```

For dataklyngen med bestanddeler som i deloppgave D blir innholdet i filen slik (dataklynge.txt):

```
12
650 64 1
16 1024 2
```

Endre hovedprogrammet fra deloppgave D slik at du oppretter en dataklynge med data fra filen "dataklynge.txt" (som vist over) før det gjør beregningene beskrevet i deloppgave D). Test gjerne med flere variasjoner av data i filen og sjekk at du får riktig resultat.

Merk: Du trenger bare å levere den siste versjonen av klassene for Dataklynge og Hovedprogrammet, det vil si klassene slik de ser ut etter at du har løst denne deloppgaven.

Oppgave slutt.