UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	INF1010 — Objektorientert programmering
Dato:	18. august 2016
Tid for eksamen:	09.00 – 15.00 (6 timer)

Oppgavesettet er på 3 sider.

Vedlegg:	4
Tillatte hjelpemidler:	Læreboka (med notater i)

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

Les gjennom hele oppgaveteksten før du begynner å svare. Noter ned uklarheter/spørsmål under gjennomlesningen slik at du har spørsmålene klare når faglærer kommer. Husk å skrive i besvarelsen der du gjør egne forutsetninger. Forklar med ord hva en kode det ikke eksplisitt er spurt etter, gjør.

Denne oppgaven består av ett program som er delvis ferdigskrevet. Programmet leser inn ord fra en fil og legger dem inn ett og ett i en beholder som bruker ei dobbel lenkeliste for å ta vare på ordene. Du skal i oppgavene fullføre klassen DLListe ved å skrive kode som er fjerna. Klassen har flere indre klasser, bl.a. en klasse som implementerer en iterator, og en annen som definerer en sorteringstråd som skal brukes av metoden quicksort.

Metoden quicksort ordner (sorterer) objektene i stigende orden ved hjelp av tråder.

Det er veldig viktig å forstå strukturen i programmet før du går igang med programmeringen.

Du bestemmer selv i hvilken rekkefølge du vil gjøre oppgavene, men nummerorden er anbefalt. I programmet er oppgavene skrevet i denne rekkefølgen:

```
oppgave 6
oppgave 4
oppgave 3
oppgave 5
oppgave 7
oppgave 8
oppgave 11
oppgave 10
oppgave 9
oppgave 2
oppgave 1
```

Deloppgavene er skrevet som kommentarer inn i programmet der de hører hjemme, men her følger en litt mer utfyllende beskrivelse av hver oppgave:

(Fortsettes på side 2.)

Oppgave 1 — 6%

Tegn datastrukturen i objektet som variabelen ordListe peker på. Du trenger ikke tegne klassedatastrukturen til objektet, heller ikke indre klasser. Du trenger ikke skrive kode i konstruktører og metoder.

Oppgave 2 — 4%

Skriv metoden fjernForan i DLListe. Metoden fjerner noden som ligger foran i lenkelista, ikke listehodet. Hvis lista er tom returneres null. (Det kan hende du vil løse oppgave 5 før denne).

Oppgave 3 — 6%

Programmer metoden settInnOrdnet i klassen Node. Metoden bruker sammenlig-metoden til å avgjøre om den nye noden er større enn, lik eller mindre enn denne noden. Ordningen skal være slik at den minste noden ligger foran, og så etter stigende orden mot slutten av lista. Hvis alle nodene blir satt inn med denne metoden vil hele lista være ordnet/sortert (innstikksortering). Metoden *skal* være rekursiv.

Oppgave 4 — 2%

Skriv implementasjonen av den samme metoden i Listehode. Den *skal* skrives kortere/enklere enn settInnOrdnet i oppgave 3.

Oppgave 5 — 6%

Fullfør metoden fjern i klassen Node. Når den kalles i en node, fjernes noden fra lenkelista. (Denne metoden kan brukes av iteratorens remove-metode (oppg 7) og av metoden fjernForan (oppg 2))

Oppgave 6 — 8%

Vi fjerner klassen AbstrNode og erstatter det med grensesnittet ANode. Skriv hele klassen Node på nytt med de endringer som er nødvendige for at programmet skal virke som før byttet. Forklar også hvilke endringer du må gjøre i Listehode og Listehale når AbstrNode erstattes med det nye grensesnittet.

Oppgave 7 — 16%

Skriv klassen ListeIterator som implementerer en iterator i lenkelista. Metoden remove skal kaste unntak av typen IllegalStateException hvis next ennå ikke er kalt, eller remove allerede er kalt etter det forrige kallet på next, jf. beskrivelsen av metoden i Javas API (vedlagt).

Oppgave 8 — 6%

Skriv metoden som skjøter (limer) sammen denne lista med lista som er gitt som parameter. Denne metoden brukes av quicksort for å skjøte sammen to sorterte lister.

Merknad: Metoden limSammen *(oppg 8) ligger i* DLListe*. De to neste metodene (oppg 9 og 10) ligger i en indre klasse,* QSortTr*, i beholderen. Dette*

(Fortsettes på side 3.)

er viktig å merke seg for at metodene skal bli korrekte.

Oppgave 9 — 4%

Hvis vi kun bruker settInnOrdnet når vi legger til noder, blir nodene ordnet i stigende rekkefølge, dvs. sortert. Bruk dette til å lage metoden sorterInnstikk i klassen QSortTr.

Oppgave 10 — 6%

Denne metoden skal dele en beholder i to. Legg merke til at den ligger i klassen QSortTr. Hvis vi har en beholder beh, og en T-verdi p skal kallet

```
DLListe<T> små = mindreEnnPivot( beh, p ) ;
```

gjøre følgende:

beh.antall() + små.antall() er lik beh.antall() før kallet
Alle noder i små «er mindre enn eller lik» p
Alle noder i beh «er større enn» p

Oppgave 11 — 36%

Siste oppgave er å skrive en trådklasse som parallelliserer quicksort. I programmet er klassen laget som en subklasse til Thread. Du kan gjerne lage den med grensesnittet Runnable hvis du foretrekker det. I denne oppgaven står du fritt til å lage andre klasser også. Det eneste som er gitt, er en beholder som skal inneholde det ferdigsorterte resultatet, samt signaturen til en metode som skal vente på at den første tråden som startes skal avslutte. Bortsett fra dette er det opp til deg å løse denne oppgaven slik den første tråden som startes til slutt inneholder en beholder som er sortert.

Hvis vi skulle bruke quicksort uten tråder, ville følgende metode sortert beholderen vår. Det anbefales å bruke samme algoritme, men parallellisert.

```
public void quickSort() { // definert inne i klassen DLListe
    if ( antall() > 5000 ) {
        T pivot = this.fjernForan();
        Lenkeliste<T> mep = mindreEnnPivot(pivot);

        mep.quickSort();

        this.quickSort();

        mep.settInnBak(pivot);
        mep.limSammen(this);

    } else {
      // sorter med innstikksortering
    }
}
```

Oppgavesett slutt.

Lykke til!

Stein Michael Storleer