**Datastrukturer & Algoritmer, Datalogi C**
Henning Christiansen 19/10-2004

## Opgaver til forelæsningen 19/10-2004

**Afleveringsopgave**: Opgave 3 nedenfor; afleveringsfristen er **torsdag 4 november**; opgaven kan afleveres elektronisk som pdf-fil sendt til henning@ruc.dk [husk at indkludere dine programfiler i pdf'en], eller i papirform (Hennings postbakke i hus 42.1 eller personligt ved forelæsningen).

**Opgave 1.** Handler om håndsimulering af quicksort; opdel jer i grupper på 3–4 personer, gerne omkring en tavle eller lignende. Tegn et array på 25 elementer og fyldt en passende sekvens af usorterede heltal ind i den. Udfør i fællesskab algoritmen for quicksort side 310 i bogen med værdien af CUTOFF sat til 4. Hvis der går skudder-mudder i processen må I starte forfra indtil I har fundet en systematisk måde at holde styr på de mange rekursive kald.

**Opgave 2.** Bogens opgaver side 316, 8.4 + 8.5. Tag kun underspørgsmål a, c, d.

**Opgave 3, AFLEVERINGSOPGAVE**
Opgaven handler om anvendelse af sortering på samlinger af objekter, hvor det er relevant at kunne sortere efter forskellige kriterier. De objekter, vi vil fokusere på, er e-mails, som man kan forestille sig, de er er repræsenteret inde i en mail-klient. Vi vil i denne opgave forstå en e-mail som et objekt med følgende felter:

> *Løbenr*. som svarer til den rækkefølge, som e-mailene er modtaget. Konkret i opgaven, den rækkefølge objekterne er oprettet i (dvs. første gang der siges »new« skabes objekt med løbenr 0, næste gang med løbenr. 1, osv.).
>
> *Dato,* som man kan vælge at repræsentere enten ved et tal (f.eks. 20031022, så er de nemme at sortere) eller en prædefineret dato-klasse hvis man kan finde en i Java API.
>
> *Afsender,* som er en string
>
> *Modtager,* som er en string
>
> *Indhold,* som er en string

Det skal være muligt at sammenligne — og dermed sortere — e-mails efter Løbenr., Dato, Afsender eller Modtager.

**Spørgsmål 3.1**
Implementér en sådan klasse af e-mails. Konstruktoren skal have 4 argumenter svarende til de nævnte felter minus løbenummeret, som tildeles automatisk. Udstyr klassen med

fire forskellige Comparator-objekter, som svarer til hver af de 4 sorteringskriterier (dvs. alle felter pånær Indhold). E-mails skal også udstyres med en toString-metode og evt. andre ting, der måtte blive brug for.

Skriv et lille testprogram, hvor du en opretter et antal e-mails og udskriver dem.

**Spørgsmål 3.2**
Nu skal du tilpasse den version af quicksort, som blev præsenteret ved forelæsningen, så den kan sortere objekter efter forskellige kriterier, dvs. den skal nu tage en Comparator som et ekstra argument (i stedet for at virke på Comparable-objekter ved at benytte deres standard-compareTo-metode).

Du kan finde den oprindelige kildetekst på filen Sort.java, som er tilgængelig på kursets hjemmeside. Skriv et lille testprogram, som viser, at sorteringen virker. NB: det accepteres ikke at benytte en sorteringsmetode fra Javas medfølgende pakker!!! Du skal selv skrive koden, dvs. modificere filen Sort,java på passende vis.

**Spørgsmål 3.3**
Nu skal der laves en klasse som svarer til en postkasse med et antal breve i. Klassen skal fungere således at man ved at sige »new« flere gange kan få oprette forskellige postkasser.

En postkasse indeholder et array som rummer et antal e-mails, samt en variabel som angiver det aktuelle sorteringskriterium; det kan synes fornuftigt at denne variabel indeholder et Comparator-objekt.

Du bruger klassen ved først at sige »new mailbox«, så får du en tom postkasse, og så skal du efterfølgende kunne indsætte nye mails i stil med:

```
email e = new email(20031022, “blabla”, ... );
box.indsæt_brev(e);
```

Som udgangspunkt skal mailboxens sorteringskriterium være efter løbenummer, men der skal være metoder, så man kan skifte kriteriet løbende. Der skal også være en metode som udskriver indholdet af en mailbox i listeform (evt. med indhold undertrykt), naturligvis i forhold til det aktuelle sorteringskriterium.

Klassen skal opføre sig således:

- e-mailene skal til enhver tid ligge i sorteret rækkefølge i forhold til det aktuelle kriterium

- Dvs., når du indsætter en email, skal du placere den i dens korrekte position (og sandsynligvis flytte ca. halvdelen af de e-mails som er der i forvejen).

- Når du skifter sorteringskriterium, skal du benytte den sorteringsmetode, du konstruerede i spørgsmål 3.2.

NB: Du må ikke bruge faciliteter i Javas Collection eller Collections i denne opgave, du skal selv sætte det hele op (og du kan sikkerte genbruge en del inspiration fra den første afleveringsopgave).

**ET GODT RÅD:** Opgaven kan synes overvældende, men hvis du sætter dig grundigt ind i, hvordan Comparator-klassen fungerer inden du går løs på kodningen, er den forholdsvis overkommelig. Hvis du begynder at miste kontrollen over dit program undervejs, er det sandsynligvis fordi du har grebet opgaven forkert an.

**KRAV TIL BESVARELSEN AF AFLEVERINGSOPGAVEN:**
I skal aflevere individuelt, og jeres fulde navn skal fremgå af forsiden. (Det er selvfølgelig tilladt at arbejde sammen 2–3 stykker om at forstå opgaven, men det forventes at I hver især har skrevet ned med jeres egne ord, hvordan I har løst opgaven). Besvarelsen skal være formuleret som en lille rapport, der forklarer din løsning, evt. med fragmenter af koden som illustration; programkoden gengives i fuld længde som appendiks. For alle spørgsmål skal der vedlægges testudskrifter, som anskueliggør at programmerne har været kørt. (Hvis dit program kun virker næsten, så vedlæg testudskrifter alligevel og argumenter for, hvordan og hvorfor det burde have virket.) Besvarelse, eksklusiv program- og testudskrifter, bør normalt ikke fylde over 4-5 sider.