Datalogi C
Henning Christiansen 5/11-2003

Opgave til stoffet gennemgået ved forelæsningen 4/11-2002

Der arbejdes med med opgave 1 og 2 til øvelserne 7/11; *spørgsmål 2.2 efterfølgende afleveringsopgave med frist 18. november* (med samme krav som til de forrige afleveringsopgaver).

Opgave 1
Betragt de hægtede lister, som er beskrevet i lærebogens afsnit 17.1. Beskriv tilstanden af L og de iterators, som indføres, efter hver sætning:

```
LinkedList L = new LinkedList();
LinkedListIterator iter = L.zeroth();
L.insert(a,iter);
L.insert(b,iter);
iter.advance();
L.insert(c,iter);
LinkedListIterator anotherIter = L.first();
anotherIter.advance();  anotherIter.advance();
L.insert(d,anotherIter);
iter.advance();
L.insert(e,iter);
```

Opgave 2
A. Jensen har et lille firma som påtager sig kopieringsopgaver. Man afleverer et antal originale sider plus bestilling af hvor mange kopier man vil have. A. Jensen kender sin kopimaskine godt, så han hurtigt kan regne ud, hvor mange minutter, hver opgave vil tage. Og det er en god kopimaskine der aldrig går istykker, og den har en elektronisk dims, som adviserer papirleverandøren som altid er på pletten og fylder papir i, så kopimaskinen aldrig løber tør. A. Jensen organiserer sit arbejde på den måde at store opgaver (defineret ved at tage længere tid end 10 minutter) sættes i en kø, hvorimod små opgaver (10 minutter eller derunder) lægges på en stak. Opgaverne ekspederes i følgende rækkefølge: Næste opgave tages altid fra stakken, hvis der er nogen, ellers tages den fra køen.

Spørgsmål 2.1
Konstruér en historie med et antal indkommende opgaver (hvor store de er og på hvilke tidspunkter), og lav en håndsimulering af A. Jensens arbejdsgang ved at tegne køen og stakken på en tavle eller et stykke papir. Prøv at variere historien så I får et indblik i, i hvilke situationer nogle opgaver kommer til at vente urimeligt længe.

Spørgsmål 2.2
Vi skal nu skrive et program som simulerer en typisk arbejdsdag for A. Jensen; han arbejder ca. 8 timer, og da han er på slangekur springer han frokosten (og andre pauser) over. Vi antager et ur, som tæller hele minutter og som starter ved kl. 0. Simuleringen foretages efter følgende abstrakte algoritme (som ikke behøver en yderligere kø af »begivenheder«). Bemærk at de »ca. 8 timer« skal forstås sådan, at A.Jensen ikke modtager ny opgaver efter 8 timer, men han gør alle opgaver færdige inden han går hjem.

klokken = 0;
opgaveFærdig=0;
næsteOpgave=0;

Gentag følgende:
    Hvis næsteOpgave ≤ opgaveFærdig og næsteOpgave < 480
        Generér en ny opgave med tilfældig varighed mellem 1 og 30 minutter;
        Sæt opgave i kø eller stak som beskrevet;
        Generér et tidsinterval med tilfældig varighed D mellem 2 og 35 minutter
            for hvor længe der er til næste opgave ankommer;
        Sæt næsteOpgave = klokken + D;
    Ellers, hvis der er opgave at vælge:
        Tag en opgave ud af kø eller stak som beskrevet, og kald dens varighed V;
        Sæt opgaveFærdig = klokken + V;
    Ellers stop;
    klokken = den mindste af NæsteOpgave og OpgaveFærdig;

Simuleringsprogrammet skal genererere følgende udskrifter, som skal kunne slås til og fra (f.eks. ved nogle boolske variable i starten af programmet, som kan sættes til true eller false):

- en rapport om samtlige begivenheder i historien, ny opgave ankommer..., opgave påbegyndes ..., opgave afsluttes,
- gennemsnitlig og maksimal kølængde og stakhøjde, maksimal ventetid for opgaver.
- hvor lang A. Jensens arbejdsdag ender med at blive.

Udvid eventuelt derefter programmet, så det simulerer 100 arbejdsdage, hvor man så undlader at udskrive de enkelte begivenheder, men beregner

- gennemsnitlig og maksimal kølængde og stakhøjde, maksimal ventetid for opgaver i løbet af samtlige 100 dage.
- hvor lang A. Jensens arbejdsdage er i gennemsnit.

Det anbefales at man benytter Javas standardbiblioteker til køer, stakke og tilfældige tal. Hvis der er fejl eller uhensigtsmæssigheder i den abstrakte algoritme ovenfor, forventes I selv at korrigere dem.