

Opgave til forelæsningen 29/10-2002

Følgende opgave, som også er afleveringsopgave for dat C'erne; afleveringsfrist fredag d. 15. november. Opgaven kan måske tage lidt længere tid end de øvrige, da der skal eksperimenteres lidt for at få den grafiske side til at makke ret, men burde ikke være voldsomt svær at løse på det principielle plan. Iøvrigt er der ingen forelæsning 5/11, så der skulle være tid nok.

Opgaven handler om at tegne binære søgetræer, og man skal benytte de grafiske primitiver, som også er blevet brugt i programmet "drawRuler" side 247 i bogen (plus et par stykker mere, som forklares nedenfor); programtekst med hovedprogram kan findes via link på kursushjemmesiden.

Om binære søgetræer:

Der er lavet en tilpasning af bogens binære søgetræer (afsnit 19.1), så de også har de metoder, som blev beskrevet i kapitel 18 for "almindelige" binære træer. Du skal have følgende filer i dit arbejdskatalog, og du skal compilere dem i den rækkefølge de fremgår her:

```
DuplicateItemException.java  
ItemNotFoundException.java  
UnderflowException.java  
Stack.java  
Queue.java  
ArrayStack.java  
ArrayQueue.java  
BinaryNode.java  
BinarySearchTree.java  
TestTreeIterators.java
```

Det er modificerede udgaver af filer ("fra bogen"), og du kan finde dem i kataloget

```
draw_tree_files
```

som du kan finde link til på kursets hjemmeside. Hvis problemer, så spørg Lars Mogenssen.

Programstruktur som kan bruges til opgaven:

Her følger et skelet du evt. kan bruge til at strukturere dit program efter.

```
import java.awt.Frame;
```

```

import java.awt.Graphics;
import java.awt.Color;

public class TreeDraw extends Frame
{ private static final int theSize = 500;// konstant som benyttes nedenfor;

private static BinarySearchTree theTree = new BinarySearchTree();

// Man SKAL definere en 'paint' metode; som så på magisk vis
// aktiveres af 'show' nedenfor

public void paint( Graphics g )
{ setBackground( Color.white );
  g.setColor( Color.red ); // den farve der tegnes og skrives med
  // Her kan du kalde din egen tegnemetode med theTree som argument
}

// private void ... dine tegnemetoder.

// For simplicity, must terminate from console with a control-c

public static void main( String [ ] args )
{ Frame f = new TreeDraw( ); // det her skal der altså bare stå
  f.setSize( theSize, theSize); // det her skal der altså bare stå
  //Initialiser et træm f.eks.:
  theTree.insert(new Integer(55));
  theTree.insert(new Integer(35));
  theTree.insert(new Integer(88));
  theTree.insert(new Integer(40));
  theTree.insert(new Integer(20));
  theTree.insert(new Integer(45));
  theTree.insert(new Integer(117));

  // denne her skal bare kaldes, og så kommer din tegning ud på skærmen :)
  f.show( );
}
}

```

Om de grafiske primitiver:

Vi kan beskrive den tegneflade, vi arbejder indenfor, ved et koordinatsystem med en X-akse, som peger den vej fra venstre mod højre og en Y-akse som peger nedefter og med punktet (0,0) i øverste venstre hjørne. I Mac-versionen (måske også på PC) er der iøvrigt den underlige egenskab at Y-aksen starter med at tælle i den bjælke, som er på vinduet, så hvis man sætter en prik i (117, 0) kan man ikke se den, mens en prik i (117,50) kan ses i den øverste del af vinduet.

Til at løse opgaven er der brug for følgende metoder:

```
g.drawString(String str, int x, int y)
g.drawLine(int x1, int y1, int x2, int y2)
```

hvor "g" er et Graphics-objekt (se hvordan der er brugt i bogens programmer).

Den første udskriver en streng (men kan også godt finde ud af tal), så det nederste venstre hjørne af "billedet" svarende til strengen ligger i punkt (x,y). F.eks. svarende i foden af P'et i "Pilsner". Metoden drawLine tegner en streg mellem (x1,y1) og (x2,y2).

Endelig har vi brug for at skifte farve, og skal man f.eks. skrive noget i en anden farve end resten af tegningen, skal man 1) skifte farve, 2) skrive, og 3.) skifte farven tilbage. F.eks. sådan her,

```
g.setColor( Color.green );
g.drawString("Pilsner"), 100, 100);
g.setColor( Color.red );
```

hvis vi aktuelt tegner med rødt og vil skrive en enkelt grøn streng.

Nu kommer opgaven — Spørgsmål 1

Skriv en første version af et program, som tegner et binært søgetræ på skærmen, så det så nogenlunde ligger midt i vinduer, og vi "drejer" blot lige meget til højre og venstre, når vi går ned i et deltræ. Det kan illustreres således for vort eksempeltræ:



Spørgsmål 2

Men hvis træet er større opstår der et problem, da knuder fra forskellige grene vil blive tegnet over i hinanden. Vi bliver nødt til at udstyre vores tegnemetode med et breddeargument. Det optimale ville nok være at skrive en metode, som beregner bredden af et træ, men vi kan nøjes med at benytte den eksisterende højdemetode. Vi bruger så højden som udgangspunkt for et startværdi for hvor meget grenene skræver og halverer (eller noget deromkring) i hvert skridt.

Træet skulle nu gerne komme ud noget i denne her retning:

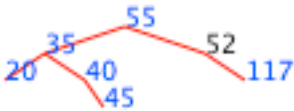


Spørgsmål 3

I vores hakede udgave af binære søgetræer kan vi risikere at et fejagtigt program kan modificere træet, således at den definerende egenskab for binære søgetræer (hvilken?) ødelægges. F.eks. hvis det træ vi har brugt hidtil udsættes for følgende:

```
theTree.getRoot().right.element=new Integer (52);
```

Udvid dit program, så det fremhæves på tegningen, hvor der er problemer i træet, ved i stedet for at bruge farven blå til tallet at benytte farven sort. For det modificerede træ får vi følgende billede:



(Hvis du ikke kan se farverne, så er 52 sort og de øvrige tal blå).

Spørgsmål 4 (kun for specielt interesserede perfektionister)

Sørg for, at knuden fremstår som en kasse med afrundede hjørner med tallet inden i, og så grenede ned til undertræerne rammer disse kasser perfekt.