Uitwerking tentamen Imperatief Programmeren 20 december 2001
Cursieve opmerkingen behoren niet tot het antwoord, maar vormen een toelichting daarop. Die hoefden bij beantwoording van de vragen er niet bij geschreven te worden.

Opgave 1a
Als je de declaratie weglaat is het symbool 't' niet geïntroduceerd, en dus wordt dat symbool niet herkend als het in de andere twee opdrachten wordt gebruikt. Er onstaat een compile-time foutmelding 'cannot resolve symbol'.
Als je de initialisatie weglaat is t een verwijzing naar een object die niet daadwerkelijk naar een object wijst. Roep je er dan toch een methode van aan, dan ontstaat de run-time foutmelding 'null pointer exception'. Bij een goede compiler komt het trouwens niet zo ver, omdat de compiler weigert een programma te accepteren waarin 't might not have been initialized'. *(Beide antwoorden zijn goedgerekend).*

Opgave 1b
Een Thread-object kun je gebruiken om meer dan één ding tegelijk te laten gebeuren bij het uitvoeren van het programma. Een voorbeeld daarvan is animatie.
De methode start roept de methode run aan en keert daar direct van terug, zelfs als de methode run nog niet is afgelopen. Daarbij wordt het object onder handen genomen dat als parameter aan de constructor was meegegeven.

Opgave 1c
Een Component zit hoger in de klasse-hierarchie dan een Button. Je kunt een 'laag' object alleen naar een 'hoog' object laten wijzen als je met een cast garandeert dat het 'hoge' object inderdaad van het juiste type is (en niet bijvoorbeeld een TextField). Andersom kan het zonder cast, maar dat werd niet gevraagd. Dus:
```
b = (Button) c;
```

Voor het bepalen van de getalswaarde van een string is een standaardmethode beschikbaar:
```
n = Integer.parseInt(s);
```

Als character h een hoofdletter bevat, dan kun je daar een kleine letter van maken door het verschil in waarde van de unicode-coderingen daarbij op te tellen, dat is h+32. Je telt daarmee een char en een int op. Bij optelling van ongelijksoortige typen heeft het resultaat het 'grootste' type, in dit geval dus int. Wil je dat in een char-variabele opslaan, dan moet je het eerst terug-casten, dus (char)(h+32). Als je die waarde 32 uit je hooft kent, dan is dat mooi meegenomen, maar dat hoeft eigenlijk niet, want je kunt het verschil tussen de codering van hoofdletters en de overeenkomstige kleine letters berekenen als 'a'-'A'. Het makkelijkste antwoord is dus:
```
k= (char)(h+'a'-'A');
```

De + operator tussen en string en iets anders, converteert dat andere naar string, en voegt dan de strings samen. Je kunt dat gebruiken om alleen de conversie uit te voeren, door samen te voegen met een lege string:
```
s= "" + d;
```

Opgave 1d
Bij een BorderLaout worden de toegevoegde componenten langs de randen of in het centrum gerangschikt, afhankelijk van een aan add meegegeven windrichting.
Bij een FlowLayout komen de componenten naast elkaar in de volgorde waarop ze ge-add zijn. Als de 'regel' vol is gaat het verder op de volgende regel. Elke regel wordt gecentreerd.
Bij een GridLayout worden de componenten gerangschikt in een gelijkmatig raster, waarvan het aantal rijen en kolommen bij constructie is meegegeven.
Je kunt een GridLayout verkrijgen met de opdracht
this.setLayout(new GridLayout(3,4,5,5));

Opgave 1e
Er zijn in totaal 40 lijnen. In dit programma worden ze getekend als 10x4 lijnen. Vergeet ook het kader niet!

```
public void paint(Graphics g)
{    int d = 200;
     g.drawRect(0,0,d,d);
     for (int n=0; n<=d; n+=(d/10))
     {
          g.drawLine(n, 0, d, d-n );
          g.drawLine(0, n, d-n, d );
          g.drawLine(0, n, n, 0 );
          g.drawLine(d-n, d, d, d-n );
     }
}
```

Opgave 2a
Het programma is geen applet, maar een stand-alone applicatie. Daarom is de klasse geen subklasse van Applet.
Het programma maakt geen window, maar doet al zijn uitvoer op de Java-console (*in de practicumomgeving:* het zwarte DOS-venster). Daarom is extends Frame niet nodig.

Opgave 2b
[Wat is out in System.out.println(…) ?]
Voor de punt van out staat System. System is een klasse, dat wordt ook door de hoofdletter gesuggereerd. Als er voor de punt een klasse staat, dan moet het ding achter de punt wel
static
zijn. Omdat er achter 'out' geen haakjes staan, is het geen methode maar een
object
die dus in de klasse
System
is gedefinieerd. In de appendix kun je dan opzoeken wat het type ervan is:
PrintStream
Dat klopt ook wel, want een PrintStream-object kent een methode println.

Opdracht 2c

In een try-catch constructie worden eerst de opdrachten achter try uitgevoerd. Als daarbij een exception optreedt worden de rest van de opdrachten overgeslagen, en gaat het verder bij catch *(waarachter een parameter van het type overeenkomend met de opgeworpen exception staat gedeclareerd).* Zijn er geen exceptions, dan wordt het catch-gedeelte overgeslagen.

Opdracht 2d

```
BufferedReader br;
String regel;
br = new BufferedReader(new FileReader(naam));
while ((regel = br.readLine())!=null)
     this.verwerk(regel);
```

Opdracht 2e

```
StringTokenizer st;
st = new StringTokenizer(regel, " ");
while (st.hasMoreTokens())
     woorden.add(st.nextToken());
```

Opdracht 2f

```
for (i=0; i<woorden.size(); i++)
{
     for (j=0; j<ander.woorden.size(); j++)
     {
          String s, t;
          s = (String)woorden.elementAt(i);
          t = (String)ander.woorden.elementAt(j);
          if (s.equals(t))
               n++;
     }
}
```

Opdracht 2g

```
if (j!=i)
{
     m = info[i].common(info[j]);
     if (m>max)
     {
          max = m;
          paar = info[i].getNaam() + " en " +
                 info[j].getNaam();
     }
}
```

Opdracht 3

```
public class Mode extends Applet
implements ActionListener, MouseListener
{
```

/ De hele geschiedenis van alle aangeklikte punten moet worden opgeslagen, zodat paint het hele plaatje kan tekenen. Dat is altijd zo in dit soort programma's; probeer dus niet een extra lijntje "erbij" te tekenen in mouseclicked, maar sla het aangeklikte punt op, en laat paint het werk doen.*

In een eenvoudig programma zou je arrays voor de x- en y-coordinaten declareren:

```
   int [] xs; int [] ys;
```

Maar hier is het ingewikkelder, want er is niet een stel lijnen maar drie. We maken daarom een twee-dimensionale array:

**/*

```
  int [][]xs;
  int [][]ys;
```

/ En er is niet één aantal, maar drie, die we handig in een array zetten: */*

```
  int aantal[];
  boolean met;
  Button clear, undo, lijnen;
  TextField mode;

  public void init()
  {
```

/ De arrays xs en ys bevatten de aangeklikte punten voor de drie typen punten. De array aantal bevat de drie totale aantallen. De boolean met geeft aan of er met lijnen getekend moet worden. Aan het begin is dat nog niet zo, dus deze boolean wordt initieel false. */*

```
    xs = new int[3][100];
    ys = new int[3][100];
    aantal = new int[3];
    met = false;
```

/ De opbouw van de userinterface is standaard. */*

```
    clear  = new Button("clear");
    undo   = new Button("undo");
    lijnen = new Button("lijnen");
    mode   = new TextField("0");
    add(mode);
    add(clear);
    add(undo);
    add(lijnen);
    clear.addActionListener(this);
    undo.addActionListener(this);
    lijnen.addActionListener(this);
    this.addMouseListener(this);
  }

  public void paint(Graphics gr)
  {
```

```
    int i, m;
/*Deze methode moet alle punten en lijnen tekenen. */

/* Teken alle punten van type 0 (rondjes) */

    for (i=0; i<aantal[0]; i++)
        gr.drawOval(xs[0][i]-5, ys[0][i]-5, 10, 10);
/* Teken alle punten van type 1 (vierkantjes) */
    for (i=0; i<aantal[1]; i++)
        gr.fillRect(xs[1][i]-5, ys[1][i]-5, 10, 10);
/* Teken alle punten van type 2 (twee gekruiste lijntjes)*/
    for (i=0; i<aantal[2]; i++)
    {   gr.drawLine(xs[2][i]-5, ys[2][i]-5, xs[2][i]+5,
ys[2][i]+5 );
        gr.drawLine(xs[2][i]+5, ys[2][i]-5, xs[2][i]-5,
ys[2][i]+5 );
    }

    if (met)
    for (m=0; m<3; m++) // voor alle drie types…
        for (i=0; i<aantal[m]; i++) // teken alle verbindingen
            gr.drawLine(xs[m][i], ys[m][i],
xs[m][(i+1)%aantal[m]], ys[m][(i+1)%aantal[m]]);
        /* De % maakt dat het laatste punt weer met het eerste wordt verbonden */

  }

  public void actionPerformed(ActionEvent e)
  {  int m;
    if (e.getSource()==clear)
      aantal[0] = aantal[1] = aantal[2] = 0;
    else if (e.getSource()==lijnen)
      met = !met;
    else if (e.getSource()==undo)
    {   /* welke mode is er momenteel gekozen? */
        m = Integer.parseInt(mode.getText());
        /*daarvan vermindert het aantal! */
        if (aantal[m]>0) aantal[m]--;
    }
    repaint();

  }

  public void mousePressed(MouseEvent e)
  {
    int m;
    m = Integer.parseInt(mode.getText());
```

```
      if (aantal[m]<100)
      {
      /* sla het aangeklikte punt op in de arrays van de juiste mode */

        xs[m][aantal[m]] = e.getX();
        ys[m][aantal[m]] = e.getY();
        aantal[m]++;
        repaint();
      }
  }
}
```