Tentamen Imperatief Programmeren

20 december 2001, 14-17 uur

- Het tentamen bestaat uit 3 opgaven, die alle drie even zwaar tellen.
- Schrijf op elk ingeleverd blad je naam, en op het eerste blad ook je collegekaartnummer en het aantal ingeleverde bladen (indien >1).
- Toon bij het inleveren je collegekaart.
- Als je een deel van de opgave niet kunt maken (bijvoorbeeld een van de methoden) probeer dan toch de rest te doen! Daarbij mag je niet-geschreven methoden gewoon aanroepen.

Opgave 1 *(de vijf onderdelen wegen even zwaar in de beoordeling)*

a. Beschouw het volgende programma-fragment:

```
TextField t;            // declaratie
t = new TextField(5);   // initialisatie
t.setText("hallo");     // gebruik
```

Beschrijf de foutmelding die je krijgt als je declaratie weglaat, en de foutmelding die je krijgt als je de initialisatie weglaat. Op welk moment krijg je deze foutmeldingen?

b. Voor welk doel kun je een `Thread`-object gebruiken?
Beschrijf kort wat de methode `start` uit de klasse `Thread` doet, en waarvoor de parameter van de constructor-methode dient.

c. Gegeven zijn de volgende declaraties:

```
Component c; Button b; char h, k;
int n; double d; String s, t;
```

Neem aan dat:
- `c` naar een `Button` verwijst
- `s` uitsluitend cijfers bevat
- `h` een hoofdletter tussen A en Z bevat

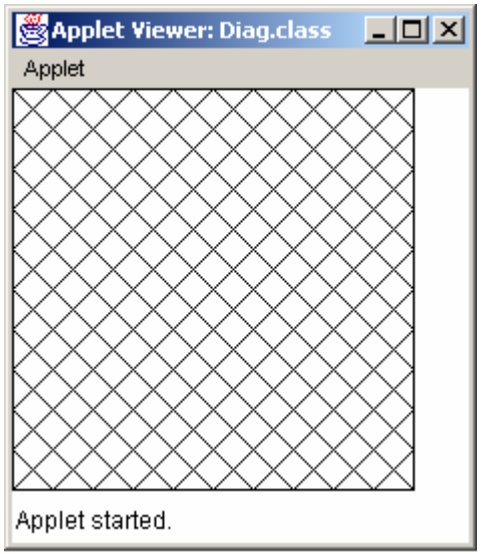
Schrijf nu toekenningsopdrachten en de benodigde conversies zodat:
- `b` naar hetzelfde object als `c` gaat verwijzen
- `n` de getalswaarde van `s` krijgt
- `k` de overeenkomstige kleine letter van `h` wordt
- `t` een String wordt met de weergave van `d`

d. In de package java.awt worden een aantal `LayoutManager` klassen gedefinieerd, waaronder:

```
BorderLayout
FlowLayout
GridLayout
```

Beschrijf in het kort hoe de opbouw van het scherm er uit komt te zien bij gebruik van deze layout-managers.
Hoe kun je in een applet zorgen dat een GridLayout wordt gebruikt?

e. Schrijf de methode `paint` van een applet met output zoals in de afbeelding hieronder. De breedte van het vierkant is 200 beeldpunten.



Zie ommezijde

Opgave 2

Bekijk het programma in de listing op de bijlage.
Het bestaat uit de klassen `Match` en `Data`.

Dit programma leest een aantal files, en slaat van elke file alle woorden op in een Data-object. Vervolgens vergelijkt het programma alle tweetallen files met elkaar, en rapporteert welke twee files het meest op elkaar lijken. Het "op elkaar lijken" van files is gemodelleerd als "veel woorden gemeenschappelijk hebben". Dus hoe meer woorden gemeenschappelijk, hoe beter ze op elkaar lijken.

Het programma rapporteert uiteindelijk een regel zoals:

```
aap.txt en noot.txt hebben 37 dezelfde woorden
```

waarbij er dus 37 gemeenschappelijk woorden in de files aap.txt en noot.txt staan, wat meer is dan het aantal gemeenschappelijke woorden in bijvoorbeeld aap.txt en mies.txt, en alle andere combinaties van files.

De deelopgaven a t/m c kun je met tekst beantwoorden

a. Waarom is het in deze klassen niet nodig om `extends Applet` te schrijven, en ook niet `extends Frame` ?

b. In de methode rapportage komt, vlakbij het eind, de naam `out` voor.
- wat is dat voor iets (methode, object, of klasse? static of niet?)
- in welke klasse is dit gedefinieerd?
- wat is het type ervan?
- wat stelt het voor?

c. In de methode `lees` staat een `try-catch` opdracht.
Wanneer/in welke volgorde worden de bodies van try en catch uitgevoerd?

De deelopgaven d t/m g vragen een stukje programma

d. Vul het ontbrekende stuk van de methode `lees` aan.

e. Vul het ontbrekende stuk van de methode `verwerk` aan.

f. Vul het ontbrekende stuk van de methode `common` aan. Deze methode moet opleveren hoeveel woorden er gelijk zijn in het object dat de methode onderhanden heeft, en in het object dat als parameter wordt meegegeven.
Hint: vergelijk *alle* paren van woorden.

g. Vul het ontbrekende stuk van de methode `rapportage` aan.
Hint: let op dat je een file niet met zichzelf vergelijkt!

(Verdeling van de 10 punten die met deze vraag te verdienen zijn:
a, e: elk 1 punt; b, c, d, f: elk 1.5 punt; g: 2 punten)

Opgave 3

Schrijf een applet met de volgende specificaties.
Op de bijlage staan enkele illustraties van dit programma in werking.

- Boven in beeld is een tekstveld zichtbaar en een drietal knoppen, met als opschrift respectievelijk "clear", "undo" en "lijnen". De hele rest van het window kan door de gebruiker overal worden aangeklikt.
- Het tekstveld is de *mode* waarin het programma zich bevindt. Initieel staat er 0 in het tekstveld, maar de gebruiker kan daar ook 1 of 2 van maken. Je mag er zonder controle van uitgaan dat de gebruiker geen andere *mode* dan 0, 1 of 2 intikt.
- Iedere keer als de gebruiker een punt van het window aanklikt, verschijnt gecentreerd op die plaats een symbool met een doorsnede van 10 beeldpunten. Welk symbool dat is hangt af van de mode op dat moment: mode 0 geeft een open rondje, mode 1 een gevuld vierkant, mode 2 een kruis.
- Van elk symbool kunnen maximaal 100 exemplaren worden getekend. Als de gebruiker toch meer punten probeert aan te klikken, gebeurt er niets.
- Als de gebruiker op de knop "clear" drukt, verdwijnen alle symbolen, en kan de gebruiker aan een nieuwe tekening beginnen.
- Als de gebruiker op de knop "lijnen" klikt, worden alle gelijke symbolen onderling verbonden. De verbindingen vormen een ring: de laatste is weer verbonden met de eerste.
- Drukt de gebruiker nogmaals op "lijnen", dan verdwijnen de lijnen weer, bij een derde keer verschijnen ze weer, enz.
- Als de gebruiker op "undo" klikt, verdwijnt het laatste symbool van de op dit moment gekozen *mode* (als dat symbool er tenminste is). Eventuele lijnen worden ook aangepast.

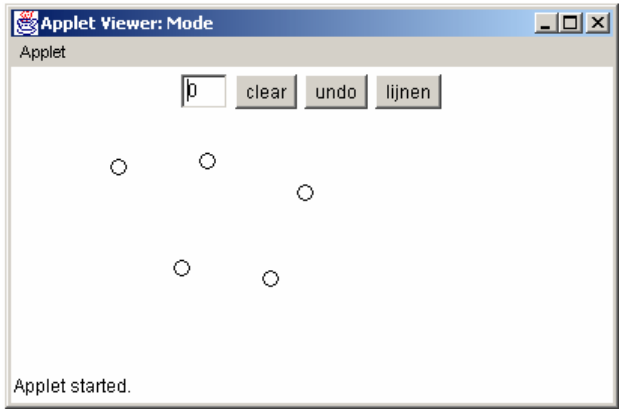
Hint: gebruik twee-dimensionale array(s) van 3 bij 100.

De import-aanwijzingen, en de muis-methoden met een lege body mag je weglaten. De HTML-file hoef je ook niet te schrijven.

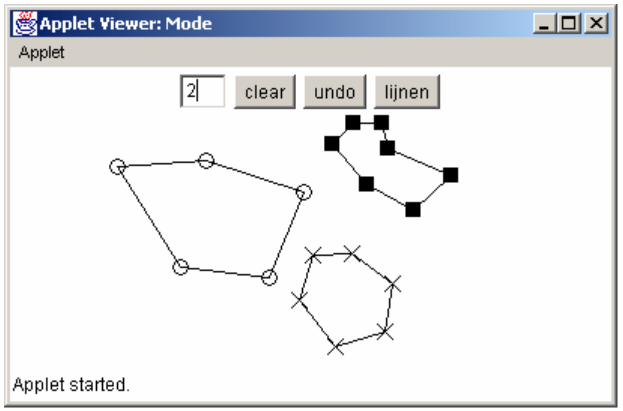
Einde

bijlage bij opgave 3

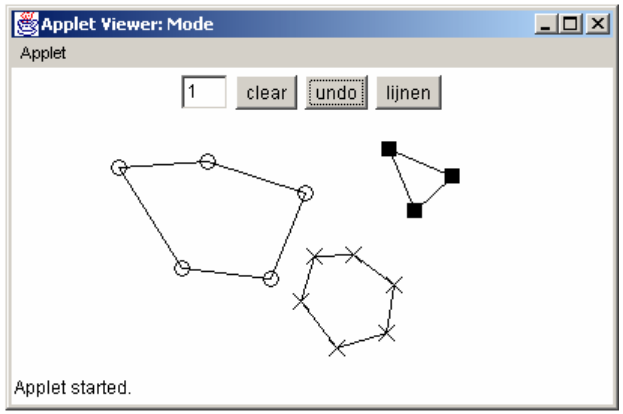
1. De gebruiker heeft vijf punten aangeklikt in mode 0



2. De gebruiker heeft de mode op 1 gezet, zeven punten aangeklikt, de mode op 2 gezet, en nog zes punten aangeklikt. Bovendien is er op de knop “lijnen” gedrukt.



3. De gebruiker heeft de mode weer op 1 gezet, en vier keer op de knop “undo” gedrukt.



bijlage bij opgave 2

```
import java.io.*;
import java.util.*;

public class Match
{
  public static void main(String [] ps)
  {
    Data [] informatie;
    informatie = new Data[ps.length];

    for (int i=0; i<ps.length; i++)
    {
      informatie[i] = new Data();
      informatie[i].lees(ps[i]);
    }

    if (ps.length>2)
      rapportage(informatie);
  }

  private static void rapportage(Data [] info)
  {
    int i, j, max;
    String paar;
    paar = "";
    max = 0;
    for (i=0; i<info.length; i++)
    {
      for (j=0; j<info.length; j++)
      {
```

g

```
      }
    }
    System.out.println(paar + " hebben "
      + max + " dezelfde woorden");
  }
}
```

```
class Data
{  private String naam;
   public Vector woorden;

   public Data()
   {  woorden = new Vector();
   }

   public void lees(String s)
   {  naam = s;
      try
      {  // lees de file met naam s, en roep
         // voor elke regel de methode  verwerk  aan
```

d

```
      }
      catch (Exception e)
      { System.out.println(naam + " is niet leesbaar");
      }
   }

   private void verwerk(String regel)
   {  // voeg alle woorden van de String regel
      // toe aan de Vector woorden
```

e

```
   }

   public int common(Data ander)
   {  // hoeveel woorden zijn er hetzelfde?
      int n = 0;
```

f

```
      return n;
   }

   public String getNaam()
   { return naam;
   }
}
```