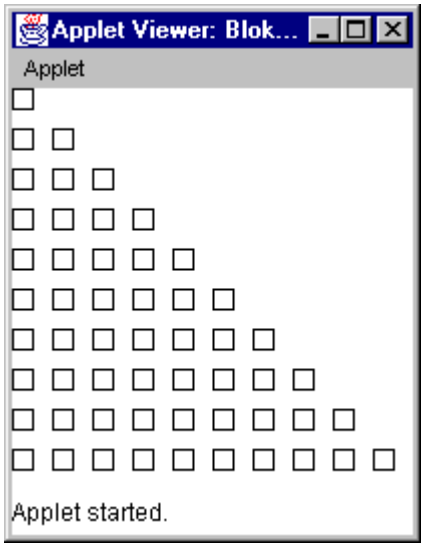
Tentamen Imperatief Programmeren

8 maart 2001, 9-12 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.
- In alle programma's mag je de *import*-regels weglaten.

Opgave 1

a. Schrijf de methode `paint` van een applet met de output zoals in de figuur. Gebruik daarbij geen while-, maar for-opdrachten.



b. Wat is het verschil tussen static methoden en niet-static methoden?
Wat zijn de consequenties voor de aanroep ervan?
Niet-static methoden uit dezelfde klasse kunnen elkaar aanroepen zonder iets voor de punt te vermelden. Kan op deze manier een static methode een niet-static methode aanroepen? En andersom?

c. Twee strings s en t kunnen vergeleken worden met `s==t` en met `s.equals(t)`. Wat is het verschil? Vul bovendien het juiste woord in in de volgende uitspraken, en licht het antwoord toe:

- als `s==t` dan geldt [altijd | soms | nooit] `s.equals(t)`
- als `s.equals(t)` dan geldt [altijd | soms | nooit] `s==t`

d. Wat zijn de overeenkomsten tussen een array en een `Vector`-object?
Geef van elk van deze twee een voordeel boven de andere.

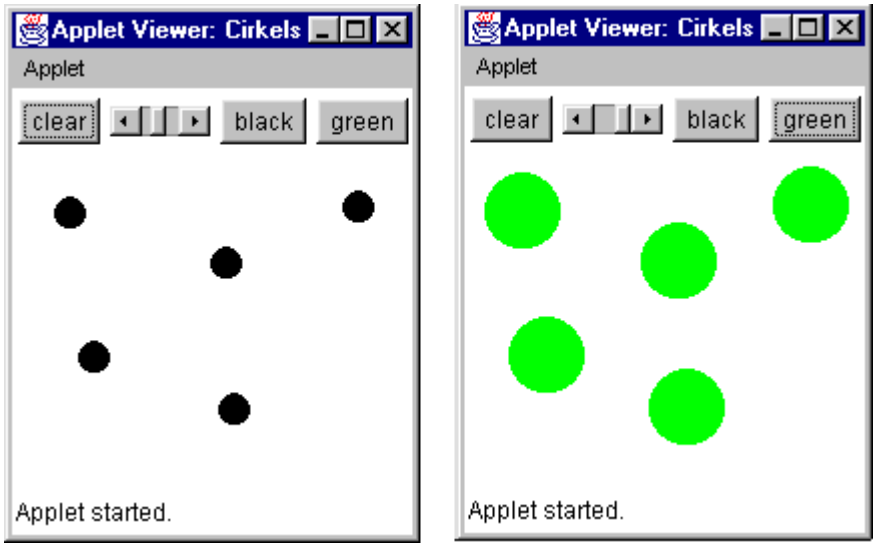
e. Om is de kleur geel te gaan tekenen, kun je de volgende opdracht gebruiken:

```
g.setColor(Color.yellow);
```

Hoe kun je hetzelfde effect bereiken, als de constante `Color.yellow` niet beschikbaar zou zijn geweest?

Opgave 2

Schrijf een applet met de volgende specificaties.
Boven in beeld is een knop met opschrift “clear” aanwezig, daarnaast staat een schuifregelaar, en daarnaast nog twee knoppen met opschrift “black” en “green”. De hele rest van het window kan door de gebruiker overal worden aangeklikt.
Iedere keer als de gebruiker een punt van het window aanklikt, verschijnt gecentreerd op die plaats een cirkel. Maximaal mogen 100 punten worden aangeklikt. Als de gebruiker toch meer punten probeert aan te klikken, gebeurt er niets.
De gebruiker kan de straal van de cirkels bepalen met de schuifregelaar: als die helemaal naar links wordt geschoven wordt de straal van alle cirkels 2, helemaal naar rechts 20. Aan het begin is de straal (dus niet de diameter!) van de cirkels gelijk aan 8.
Als de gebruiker op de knop “clear” drukt, verdwijnen alle cirkels, en kan de gebruiker weer beginnen met het aanklikken van maximaal 100 nieuwe punten.
Als de gebruiker op de knop “green” drukt, worden alle cirkels groen; met een druk op de knop “black” worden ze weer zwart, zoals in het begin.
Om het schrijfwerk te verminderen, mag je de import-regels bovenaan het programma weglaten, en ook de muismethoden die een lege body zouden hebben.



De afbeelding toont links het programma na het aanklikken van een vijftal punten. In de afbeelding rechts is de schuifregelaar naar rechts geschoven, en de knop “green” ingedrukt: de cirkels zijn daardoor groter en groen geworden.

Opgave 3

Bekijk het programma in de listing op de bijlage. Het bestaat uit de klassen `Summary` en `Data`.
Dit programma leest een aantal files, en slaat van elke file enkele gegevens op in een Data-object. Vervolgens rapporteert het programma van alle letters A tot en met Z in welk van de gelezen files deze letter het vaakste voorkomt.

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

a. Hoe kan de gebruiker van dit programma bepalen welke files er gelezen worden?

b. In de methode `main` staat de expressie `new Data()`. Waarom is dat nodig, terwijl enkele regels eerder ook al `new Data` staat? Licht het antwoord toe met een schets van de situatie in het geheugen.

c. Verderop in `main` staat de test

```
if (ps.length>1)
```

Als deze test wordt weggelaten kan er een foutmelding volgen. Welke fout is dat? Treedt die fout op bij het compileren of het runnen van het programma?

De deelopgaven d t/m g vragen om een stukje programma.

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

e. Schrijf de methode `turf`. Zorg ervoor dat zowel hoofdletters als kleine letters worden meegeteld; de aantallen voor overeenkomstige hoofd- en kleineletter mogen samen worden genomen. Andere symbolen (zoals cijfers, leestekens, en letters met accenten) moeten worden genegeerd.

f. Vul het ontbrekende stuk van de methode `rapportage` aan.

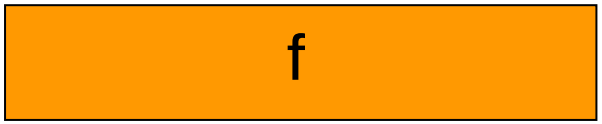
g. We gaan het programma nu veranderen, zodat het van elke file ook de gemiddelde regellengte kan bepalen. De opdracht die in de methode main in commentaar staat, wordt daarom nu toegevoegd, en de opdracht waarmee het Data-object wordt gemaakt, wordt vervangen door

```
informatie[i] = new ExtraData();
```

Schrijf de klasse `ExtraData`.
Hints: het is natuurlijk niet de bedoeling om de hele klasse `Data` over te gaan schrijven. Het kan en moet veel handiger. Let verder op dat je vanuit de klasse `ExtraData` de bestaande methode `turf` niet kunt aanroepen, omdat die private is.

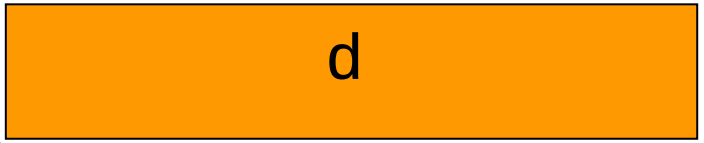
(Verdeling van de 10 punten die met deze vraag te verdienen zijn:
a, c, e: elk 1 punt; d, f: elk 1.5 punt; b, g: elk 2 punten)

```
import java.io.*;

public class Summary
{
  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]);
      // System.out.println( ps[i]
      // + " heeft regellengte: "
      // + informatie[i].gemRegLen());
    }

    if (ps.length>1)
      rapportage(informatie);
  }

  private static void rapportage(Data [] info)
  {
    int m;
    for (char c='A'; c<='Z'; c++)
    { // bepaal in welke file letter c
      // het vaakste voorkomt
```

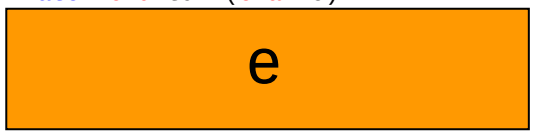



```
      System.out.println( "letter "
        + c + " komt het vaakst voor in "
        + info[m].getNaam());
    }
  }
}
```

```
class Data
{
  private String naam;
  private int [] aantal;

  public Data()
  {
    naam = "";
    aantal = new int[26];
  }

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




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

  public void verwerk(String regel)
  {
    for (int i=0; i<regel.length(); i++)
      turf(regel.charAt(i));
  }

  private void turf(char c)
  {
```




```
  }
  public int freq(char c)
  {
    if (c>='A' && c<='Z')
      return aantal[c-'A'];
    return 0;
  }

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