Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A−Eskwadraat.
Het college INFOIMP werd in 2005/2006 gegeven door Jeroen Fokker.
De uitwerking is samengesteld/gemaakt door Roland Vaandrager.

Uitwerking[1] Imperatief Programmeren (INFOIMP) 28 september 2005

N.B. De uitwerking van dit tentamen is samengesteld aan de hand van PowerPoint slides, die door de docent gemaakt zijn.

Opgave 1

Opgave 1 *(20 punten)*

a) Wat is het verschil tussen een *compiler* en een *interpreter*?

b) Wat is het verschil tussen een *expressie* en een *opdracht*?

c) De aanroep van een methode vormt soms een expressie, en soms een opdracht. Hoe kunt je dat zien aan de *header* van de methode?

d) Hoe kun je dat zien aan de *body* van de methode?

e) In veel klassen zit een methode met dezelfde naam als de klasse: de zogeheten *constructormethode*. Zo'n constructormethode wordt op een bijzondere manier aangeroepen. Hoe? En wat gebeurt er dan?

Antwoorden:

a) Een *compiler* vertaalt broncode naar machine-uitvoerbare code; een *interpreter* voert broncode direct uit.

b) Een *expressie* kun je uitrekenen, het heeft een waarde. Een *opdracht* kun je uitvoeren.

c) Als de methode een header met het woord "void" heeft, is de methode een *opdracht*.

d) Als de body van de methode het woord "return" bevat, is de methode een expressie.

e) Een constructormethode roep je aan met:

```
new Klassenaam(...)
```

Dan gebeurt er het volgende:

- er wordt een nieuw object van die klasse aangemaakt
- het wordt alvast onderhanden genomen door de constructor-methode
- het levert een verwijzing naar het nieuwe object als resultaat op

Opgave 2 *(20 punten)*

a) Waarvoor dient een declaratie?

b) Een declaratie kan op drie wezenlijk verschillende plekken in een programma staan.

Geef van elk van deze drie mogelijkheden aan:

- Op welke plek staat deze soort declaratie?
- Waar wordt een declaratie op deze plek voor gebruikt?

[1] Deze uitwerkingen zijn met de grootste zorg gemaakt. In geval van fouten kan de TBC niet verantwoordelijk worden gesteld, maar wordt zij wel graag op de hoogte gesteld: `tbc@A-Eskwadraat.nl`

- Waar en hoe krijgt de variabele die zo wordt gedeclareerd zijn waarde?
  (in totaal dus drie keer drie (korte) antwoorden geven)

Antwoorden:
De drie mogelijkheden voor declaraties zijn: *objectvariabelen*, *parameters* en *lokale variabelen*.

a) De drie soorten staan op de volgende plaatsen in dit voorbeeldprogramma:

```
public class Mixer ...
{
      Scrollbar rood;                                   object-variabele

      public void paint ( Graphics gr )                 Graphics gr is een parameter
      {
            int rw;                                     lokale variabele
            rw = rood . getValue( );
            gr.setColor(new Color(rw,...));
      }
}
```

b) Een object-variabele wordt gebruikt als permanente variabele; een parameter is bedoeld voor communicatie tussen methoden en een lokale variabele is bedoeld voor tijdelijke opslag van gegevens.

c) Een object-variabele krijgt zijn waarde in de methode `init`; een parameter krijgt zijn waarde bij de aanroep van de methode en de lokale variabele krijgt zijn waarde d.m.v. waardetoekenning.

Opgave 3 *(30 punten)*

Een benadering van de logaritme van een (reëel) getal a kun je als volgt uitrekenen:
Bereken eerst $x = a - 1$. Bereken vervolgens

$$\frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \frac{x^6}{6} + \ldots$$

Let op: de termen worden dus afwisselend bij het resultaat opgeteld en afgetrokken. Het resultaat is ook weer een reëel getal.
(Deze formule werkt alleen als $0 < a < 2$, maar daarop hoef je niet te controleren).
Schrijf een methode `logaritme` die deze benadering door 20 van deze termen te sommeren, en dat als resultaat oplevert.
Het is hierbij niet toegestaan om de bestaande methodes, zoals `log`, uit de klasse `Math` te gebruiken.
Ook is het niet toegestaan om alle 20 termen helemaal uit te schrijven.
Je mag wel (maar hoeft niet) extra hulp-methoden definiëren.

Een mogelijk programma:

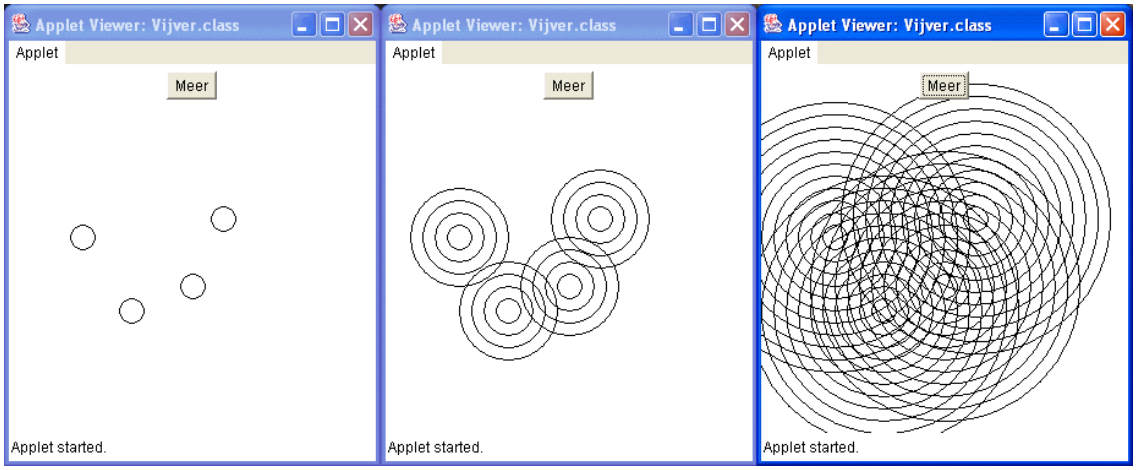
```
double logaritme (double a)
{
    int t;
    double x;
    double m;
    double res;
    int s;
    x = a-1;
    m = x;
    res = 0;
    s = 1;

    for (t=1; t<=20; t++)
    {
        m = this.macht(x,t);
        res = res + s * m / t;
```

```
        m = m * x;
        s = -s;
    }
    return res;
}
```

Opgave 4 *(30 punten)*

Schrijf een complete applet met de volgende werking. Je hoeft alleen de Java-file te schrijven, en je mag daarin de import-regels weglaten. De HTML-file is hieronder gegeven.
Op het scherm is een button zichtbaar, en vier cirkels. De cirkels stellen de kringen voor die vier in een vijver gegooide steentjes verspreiden. Het centrum van de vier cirkels bevindt zich respectievelijk op positie $(60, 140)$, $(100, 200)$, $(150, 180)$ en $(175, 125)$. Het eerste plaatje hieronder laat de beginsituatie zien.
Elke keer als de gebruiker op de knop drukt, breiden de kringen zich uit. In het tweede plaatje heeft de gebruiker driemaal op de knop gedrukt, en zijn er rond elk steentje dus vier kringen zichtbaar. In het derde plaatje heeft de gebruiker tienmaal op de knop gedrukt.



De HTML-file waarmee deze applet wordt gestart is als volgt:

```
<HTML>
  <APPLET code=Vijver.class width=300 height=300>
    <PARAM name=afstand value=10>
  </APPLET>
</HTML>
```

De waarde die in de PARAM genaamd afstand wordt gespecificeerd (in dit voorbeeld is dat 10) moet in de applet gebruikt worden als afstand tussen de kringen.
Het is in dit programma niet toegestaan om de code voor zo'n groep cirkels viermaal helemaal uit te schrijven. (Bespaar je de moeite om het toch te proberen: dit levert geen extra punten op). Gebruik in plaats daarvan een mechanisme dat in Java beschikbaar is om dit soort situaties compact te formuleren.

Een mogelijk programma:

```
public class Vijver extends Applet implements ActionListener
{
    int n;
    int a = Integer.parseInt(getParameter("afstand"));
```

```
    public void init()
    {
        Button b;
        b = new Button("meer");
        this.add(b);
        b.addActionListener(this);
        n = 1;
    }

    public void actionPerformed(AE e )
    {
        n++;
        this.repaint();
    }

    public void paint(Graphics g)
    {
        this.steen(g, 60,  140);
        this.steen(g, 100, 200);
        this.steen(g, 150, 180);
        this.steen(g, 175, 125);
    }

    public void steen(Graphics g, int x, int y )
    {
        int t;
        for (t=1; t<=n; t++)
        {
            g.drawOval(x-t*a, y-t*a, 2*t*a, 2*t*a);
        }
    }
}
```