Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TEX van A−Eskwadraat.
Het college INFOIMP werd in 2004/2005 gegeven door Jeroen Fokker.

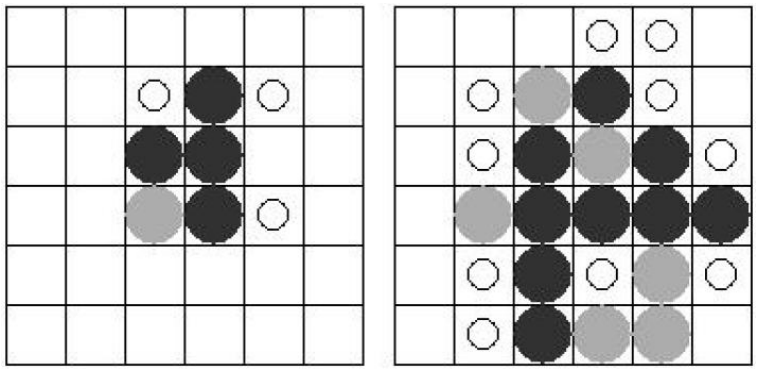
Imperatief Programmeren, toets 3 (INFOIMP)
5 november 2004

Opgave 1 *(20 punten)*

Bij het spel "reversi" leggen twee spelers om de beurt een gekleurde steen op een veld van een rechthoekig speelbord. Een steen mag alleen maar worden neergelegd op een veld als

- het veld nog leeg is, en
- met deze zet een rij van een of meer stenen van de andere kleur wordt ingesloten tussen de nieuwe steen en een al op het bord liggende steen van de eigen kleur.

De ingesloten stenen kunnen in acht mogelijke richtingen naast de nieuwe steen liggen: horizontaal, verticaal of diagonaal. Stenen insluiten in meerdere richtingen mag ook. In de figuur is voor twee voorbeelden met open cirkels aangegeven op welke velden de speler met de lichte stenen mag zetten.



Als gevolg van een zet veranderen alle ingesloten stenen van kleur. In een programma wordt de situatie opgeslagen in een twee-dimensionale array:

```
int bord[][] = new int[6][6];
```

Lege velden zijn gecodeerd met 0, gevulde velden met 1 of -1 voor de twee kleuren. De opgave: Schrijf een methode

```
boolean mag (int kleur, int x, int y)
```

die controleert of speler `kleur` een steen op veld `(x,y)` mag zetten. (De zet wordt dus nog niet uitgevoerd!). Hint: het is toegestaan (en handig) om een extra methode te schrijven die het insluiten in één van de 8 richtingen controleert.

Opgave 2 *(30 punten)*

In opgave 2 en 3 gaat het over het programma dat in de bijlage is gegeven. Opgave 2 bestaat uit theorie-vragen die met tekst beantwoord kunnen worden, bij opgave 3 moet je zelf stukjes programma schrijven.
We beschrijven nu eerst het programma (zie ook de snapshots op de achterkant). Het programma bestaat uit drie klassen: Test, Stad en Kaart.

Als dit programma gerund wordt, maakt het een window met een tekstveld (bovenin), en canvas (links) een meer-regelige textarea (rechts) en een button (onderin). De gebruiker moet de naam van een stad in het tekstveld bovenaan invullen en op Enter drukken, en daarna met de muis op het canvas klikken. Op die plaats verschijnt dan een vierkantje met de naam van de stad. Door dit te herhalen kan de gebruiker een hele kaart opbouwen. Als de gebruiker op de eerste regel van de textarea de naam van een stad intikt, en daarna op de 'Zoek'-knop drukt, wordt het vierkantje van de gekozen stad blauw (tenzij die stad niet bestaat). (Als de gebruiker meerdere steden met dezelfde naam heeft gemaakt, hoeft er maar ééntje blauw te worden).

a) Boven in de klasse Test worden vier variabelen gedeclareerd: k, t, a en b. Hadden (sommige van) deze variabelen in plaats daarvan in de constructor-methode gedeclareerd kunnen worden? Zo ja: welke, en waarom? Zo nee, waarom niet?

b) In de constructor van Test is een regel weg-gecommentarieerd (bij het pijltje). Maak een schets hoe het window er uit had gezien als deze regel echt in het programma had gestaan.

c) In de klasse Test worden zeven methoden gedefinieerd waarvan de naam met window... begint. Dat is natuurlijk veel gedoe. Hoe had dit opgelost kunnen worden zonder deze methoden allemaal te schrijven?

d) In de methode actionPerformed van de klasse Kaart wordt de cast-notatie `(TextField)` gebruikt. Waarom is dat nodig? Is dit veilig, of kan er nu een runtime fout (class cast exception) optreden? Licht het antwoord kort toe.

e) In de methoden schrijf en lees van de klasse Kaart wordt een try-catch constructie gebruikt. Leg uit hoe de opdrachten in zo'n constructie worden uitgevoerd. En wat is het alternatief voor het schrijven van een try-catch-opdracht in situaties waarin een Exception kan worden verwacht?

Opgave 3 *(50 punten)*

a) Schrijf het ontbrekende gedeelte van de constructor van de klasse `Kaart`.

b) Schrijf de methode `mousePressed` van de klasse `Kaart`.

c) Schrijf de methode `waar` van de klasse `Kaart`, die oplevert op welke punt een stad met de opgegeven naam ligt. Als er geen stad is met die naam, dan moet de methode `null` opleveren.

d) Schrijf de ontbrekende gedeelten van de methoden `lees` en `schrijf` van de klasse `Kaart`. De methode `schrijf` moet de naam en plaats van alle steden naar een file met de gegeven naam schrijven, op zo'n manier dat de methode lees de situatie weer kan reconstrueren. (Het blauw-zijn van de stad 'hier' hoeft niet bewaard te worden).

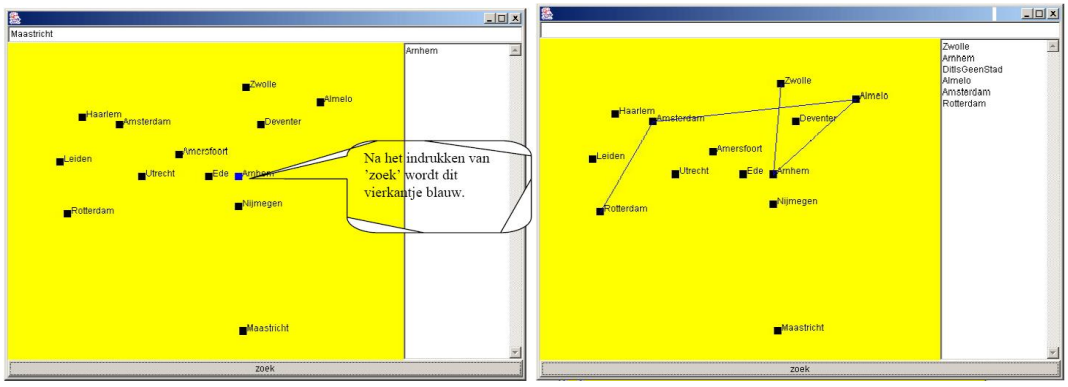
e) We gaan het programma nu aanpassen: in de constructor van de klasse Test wordt de toekenning

```
k = new Kaart();
```

vervangen door

```
k = new RouteKaart();
```

Schrijf de klasse RouteKaart zodat het programma zich als volgt gaat gedragen: In de textarea kan de gebruiker nu meerdere stad-namen tikken, elk op een aparte regel. Als hij daarna op de 'Zoek'-knop drukt, tekent het programma lijnen tussen de gekozen steden. (Eventuele niet-bestaande steden worden daarbij overgeslagen).



f) Wat moet er waar in het programma worden toegevoegd zodat de gebruiker de steden die hij heeft neergezet de volgende keer dat hij het programma gebruikt meteen weer ziet staan?

(Verdeling van de 10 punten die met deze vraag te verdienen zijn: a, b, f: elk 1 punt; c, d: elk 2 punten; e: 3 punten)

Bijlage bij opgaven 2 en 3

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
class \textbf{Test} extends Frame implements ActionListener, WindowListener
{   Kaart k;    TextField t;
    TextArea a; Button b;

    public Test()
    {  this.setSize(700,500);
       // this.setLayout(new FlowLayout());
       k = new Kaart(); k.addMouseListener(k);
       t = new TextField(20); t.addActionListener(k);
       b = new Button("zoek"); b.addActionListener(this);
       a = new TextArea(10,20);

       this.add(k, BorderLayout.CENTER);
       this.add(t, BorderLayout.NORTH);
       this.add(b, BorderLayout.SOUTH);
```

```
        this.add(a, BorderLayout.EAST);
        this.addWindowListener(this);
    }

    public static void main(String [] args)
    {  new Test2004().setVisible(true);
    }

    public void actionPerformed(ActionEvent e)
    { k.zoek(a.getText());
    }

    public void windowClosing(WindowEvent e)
    { System.exit(0);
    }
    public void windowOpened(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
}

import java.awt.Point;
class \textbf{Stad}
{
    String naam;
    Point locatie;

    public Stad(String s, Point p )
    {  naam = s;
       locatie = p;
    }
}
```

Einde tentamen