Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat.
Het college INFOIMP werd in 2006/2007 gegeven door Jeroen Fokker.

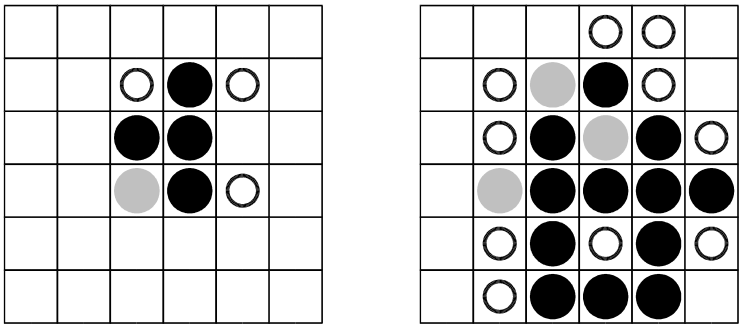
Imperatief Programmeren, derde deeltentamen (INFOIMP)
3 november 2006

Opgave 1 *(25 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 *(35 punten)*

a) Sommige programma's maken geen gebruik van een window, maar communiceren met de gebruiker via een commandoregel.

- Hoe kun je in het programma de woorden (bijvoorbeeld filenamen) te pakken krijgen die de gebruiker bij het opstarten van het programma heeft meegegeven?

- Hoe kun je een string tonen aan de gebruiker van zo'n programma?

b) Bekijk de volgende implementatie van de methode `actionPerformed`:

```
public void actionPerformed(ActionEvent e)
{
  TextField tf;
  tf = (TextField) e.getSource();
  tf.setText("hallo");
}
```

Waarom staat er `(TextField)` in de toekenningsopdracht?

Wat gaat er fout als je dit weglaat?

Wat kan er toch nog fout gaan nu het er wel staat?

c) Omschrijf in woorden voor welk doel je een DataInputStream zinvol kunt gebruiken. Geef met regel Java aan hoe zo'n object kan worden gemaakt (declaratie plus initialisatie).

d) Omschrijf in woorden voor welk doel je een Iterator zinvol kunt gebruiken.

Geef met een programma-fragment aan hoe dat in zijn werk gaat.

e) Een object is een groepje variabelen dat door methoden onder handen genomen wordt. Als het type van een object een klasse uit een standaard-package is, is het voor de programmeur niet altijd bekend uit welke variabelen zo'n object bestaat, maar uit het gedrag dat het object vertoont kun je het soms toch afleiden. Geef twee voorbeelden van een variabelen die blijkbaar onderdeel uitmaken van een `Graphics`-object.

Geef ook twee voorbeelden van eigenschappen die in een `Graphics`-object *niet*, maar in een `Graphics2D`-object *wel* worden bijgehouden.

Opgave 3

(45 punten)

Het is in deze opgave toegestaan (maar niet per se noodzakelijk) om naast de gevraagde methodes ook nog extra hulp-methodes te schrijven.

Het programma in deze opgave leest een bestand in waarin de ligging van een aantal steden wordt vastgelegd. Het programma toont deze steden op het scherm als zwarte vierkantjes met de naam ernaast.

De gebruiker kan deze vierkantjes aanklikken. De namen van de aangeklikte steden verschijnen in een lijst rechts van het plaatje. Bovendien zie je rond de steden een cirkel, die bij elke klik groter wordt.

De gebruiker kan ook de lijst rechts op het scherm editten. Als hij/zij daarna op de knop onderaan het scherm drukt, worden de cirkeltjes aangepast op grond van het aantal keer dat de naam van de stad in het lijstje voorkomt.

In onderdeel g) passen we het programma aan, zo dat de grootste cirkel(s) in een andere kleur worden getoond.

In de bijlage staat een deel van het programma gegeven. Het bestaat uit vier klassen: `Test`, `Kaart`, `Counter`, en `Stad`. Een aantal onderdelen moet nog worden ingevuld. Let op dat sommige variabelen private zijn gedeclareerd: die mogen van buiten de klasse niet worden gebruikt.

a) Schrijf de methode `raak` in de klasse `Stad`, die oplevert of de parameters een punt binnen het zwarte vierkantje van de betreffende stad aanduiden.

b) Schrijf de declaraties en initialisaties van de variabelen die nodig zijn bovenin de klasse `Kaart`.

c) Schrijf het ontbrekende deel van de methode `lees` van de klasse `Kaart`, die de steden inleest vanuit een bestand. De regels van dat bestand bevatten steeds 2 getallen (de x- en y-coordinaat) en precies één woord, dus regels zoals

```
250 110 Amsterdam
280 180 Utrecht
```

d) Als met de muis een vierkantje wordt aangeklikt, moeten er twee dingen gebeuren:

- de naam van de stad wordt toegevoegd aan het lijstje rechts op het scherm
- de cirkel die om de stad wordt getekend wordt groter

Schrijf de methodes `welkeStad` en `stadPlus` die daarvoor nodig zijn.

e) Schrijf de methode `paint` die alle steden in beeld brengt. De kleur van de cirkel rond elke stad moet zijn zoals de gegeven methode `kleurVan` dat voorschrijft (momenteel groen voor alle steden).

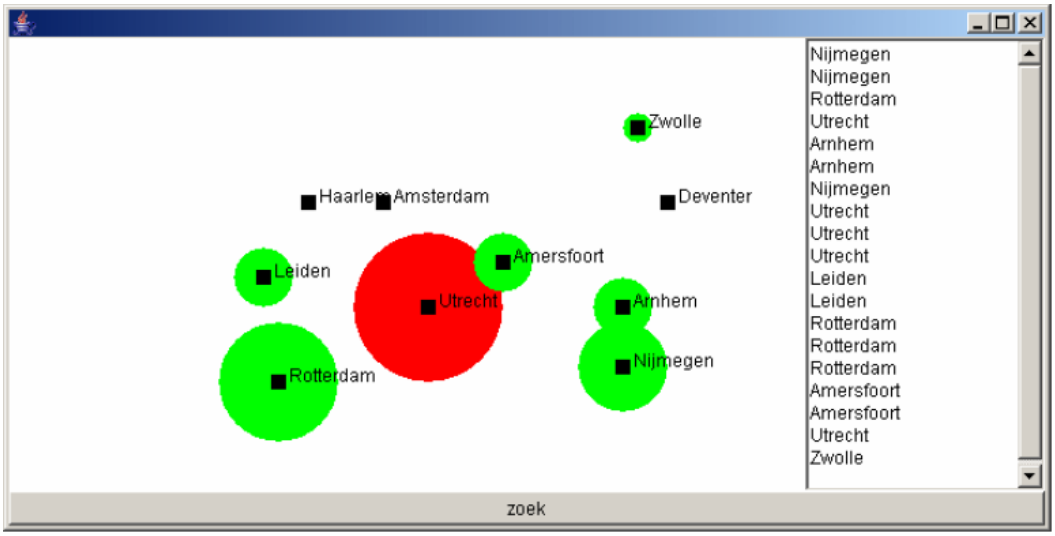
f) De gebruiker kan ook met het toetsenbord de lijst van steden aanpassen. Als hij/zij daarna op de knop "zoek" onderin beeld klikt, dan worden de cirkels rond de steden aangepast op grond van het aantal keer dat de stad dan in de lijst voorkomt. Namen met een spelfout erin worden genegeerd.

Schrijf de methode verwerk die daarvoor nodig is.

g) We gaan het programma nu aanpassen. In de klasse `Test` wordt het type `Kaart` in de declaratie en initialisatie van de variabele `k` vervangen door het type `Kaart2`. Schrijf de klasse `Kaart2` zodat het programma zich als volgt gaat gedragen:

bij de stad met de grootste cirkel eromheen (of steden, als er meerdere steden de grootste zijn) wordt die cirkel niet groen maar rood.

In de klasse `Kaart2` moet zo veel mogelijk van het werk dat al in de klasse `Kaart` is gedaan worden hergebruikt; het mag dus niet opnieuw worden opgeschreven.



Bijlage bij opgave 3

```
class Test extends Frame implements WindowListener
                                  , ActionListener
{   Kaart k;
    TextArea a;
    Button b;
    public Test()
    {   this.setSize(700,500);
        k = new Kaart2();       k.addMouseListener(k);
        b = new Button("doen"); b.addActionListener(this);
        a = new TextArea(10,20);
        k.setParent(this);
        k.lees("steden.txt");
        this.add(k, BorderLayout.CENTER);
        this.add(b, BorderLayout.SOUTH);
        this.add(a, BorderLayout.EAST);
        this.addWindowListener(this);
    }
    public static void main(String [] args)
    {   new Test().setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {   k.verwerk(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) {}
}



class Counter
{
    private int x;
    void reset()
    {   x = 0;
    }
    void increment()
    {   x++;
    }
    int value()
    {   return x;
    }
}
```

```
class Stad extends Counter
{   private String naam;
    private Point loc;
    public Stad(String s, Point p )
    {   naam = s;
        loc = p;
    }
    public String getNaam()
    {   return naam;
    }
    public boolean raak(int x, int y)
    {
        /*
         * Opgave 3 a
         */
    }
    public void teken(Graphics g, Color c)
    {
        int d = this.value();
        g.setColor(c);
        g.fillOval(loc.x-10*d, loc.y-10*d, 20*d, 20*d );
        g.setColor(Color.BLACK);
        g.fillRect(loc.x-5, loc.y-5, 10, 10);
        g.drawString(naam, loc.x+7, loc.y);
    }
}


class Kaart extends Canvas implements MouseListener
{
    /*
     * Opgave 3 b
     */
    public void setParent(Test p)
    {   parent = p;
    }
    public void lees(String naam)
    {   try
        {
            /*
             * Opgave 3 c
             */
        }
        catch(Exception e)
        {   System.out.println(""+e);
        }
    }
    public void mousePressed(MouseEvent e)
    {   Stad s = this.welkeStad(e.getX(), e.getY());
        if (s!=null)
        this.stadPlus(s);
        this.repaint();
    }
    public Stad welkeStad(int x, int y)
    {
        /*
```

```
         * Opgave 3 d
         */
    }
    public void stadPlus(Stad s)
    {
        /*
         * Opgave 3 d
         */
    }
    public Color kleurVan(Stad s)
    {   return Color.GREEN;
    }
    public void paint(Graphics g)
    {
        /*
         * Opgave 3 e
         */
    }
    public void verwerk(String str)
    {
        /*
         * Opgave 3 f
         */
        this.repaint();
    }
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    }
}
```