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.

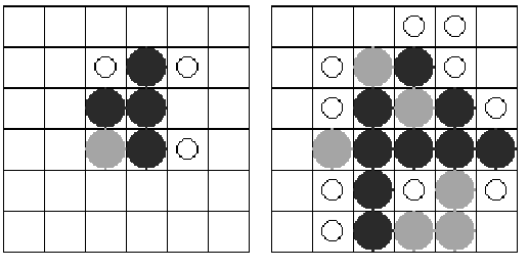
Imperatief Programmeren, derde deeltentamen (INFOIMP)
4 november 2005

Opgave 1 *(2 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.
Gegeven zijn volgende methoden (die hoef je dus *niet* te schrijven)

```
boolean mag (int kleur, int x, int y)
boolean sluit(int kleur, int x, int y, int dx, iny dy)
```

De methode `sluit` test of speler `kleur`, door te zetten op veld (x, y), één of meer stenen van de tegenstander insluit in de richting $(\mathrm{d}x, \mathrm{d}y)$, waarbij $\mathrm{d}x$ en $\mathrm{d}y$ $-1, 0$ of 1 zijn. De methode `mag` bepaalt of speler `kleur` mag zetten op veld (x, y).

Opdracht: schrijf nu een methode
`public boolean doeZet (int kleur, int x, int y)`
die, als dat is toegestaan, de zet voor speler kleur op veld (x, y) uitvoert.
Hint: het is toegestaan (en handig) om een extra methode te schrijven die het veranderen van kleur in één van de 8 richtingen uitvoert. En natuurlijk kun je de methoden `mag` en `sluit` ook gebruiken.

Opgave 2 *(4 punten)*

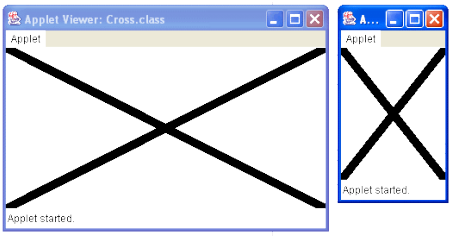
a) Wat is het verschil tussen een `InputStream` en een `Reader`?

b) In welke situatie is het zinvol om een methode aan te roepen door middel van `super.naam()` in plaats van `this.naam()`?

c) De Java GUI-library "Swing" is een lichtgewicht GUI-toolkit.

1. Wat is het verschil met een zwaargewicht toolkit, zoals AWT?
2. Noem een voordeel en een nadeel van zo?n lichtgewicht toolkit.

d) Schrijf de methode `paint` van een applet dat een diagonaal kruis tekent, net zo groot als het window is, met lijndikte 10. De methode moet kunnen werken in applets van allerlei afmetingen; zie de twee voorbeelden in het plaatje.



e) Bij sommige klassen staat er abstract in de header van de klasse.

1. Waarin onderscheidt zo'n abstracte klasse zich van een gewone klasse?
2. Hoe kun je zo'n abstracte klasse elders in het programma gebruiken?

f) In Java 5.0 is het mogelijk om *generic* Collections te maken.

1. Wat houdt dat in?
2. Wat is hiervan het voordeel boven 'gewone' Collections zoals die in eerdere versies van Java al mogelijk waren?

Opgave 3

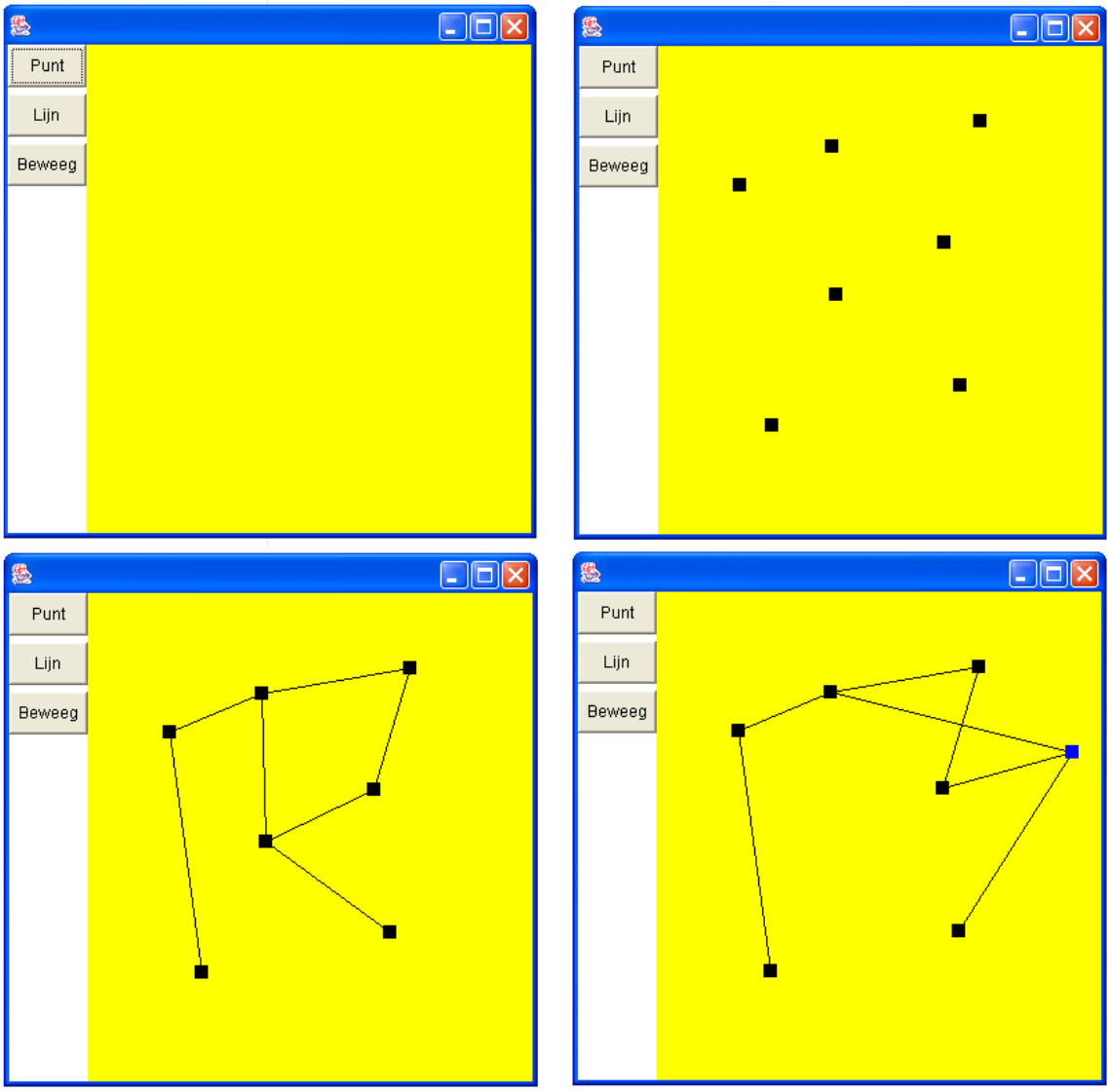
(4 punten)

Bekijk de listing van de klasse `Drie` op de bijlage.
Als dit programma gerund wordt, maakt het een window met drie buttons en een canvas.
Met de buttons kan de gebruiker een van de drie modes uitkiezen: Punt, Lijn, of Beweeg. Afhankelijk van de huidige mode kan hij in het canvas rechts van de knoppen iets doen:

- In de Punt-mode: punten aanklikken (een onbeperkt aantal), die als zwart blokje getekend worden.
- In de Lijn-mode: eerst een van de al bestaande zwarte blokjes aanklikken; die wordt dan blauw. Daarna nog een zwart blokje aanklikken. Er onstaat dan een lijn tussen die twee punten.
- In de Beweeg-mode: een zwart blokje ?vastpakken?, en verschuiven naar een andere plek. De lijnen die aan dat blokje vastzitten verschuiven mee!

In de klasse `Tekening` staat het gedrag van het canvas. In de klasse is een variabele mode gedeclareerd, die wordt veranderd in `'P'`, `'L'` of `'B'` als reactie op het indrukken van een van de knoppen.
Verder zijn er nog twee extra klassen `Punt` en `Lijn`, zodat in het programma gemakkelijk punten en lijnen in objecten kunnen worden opgeslagen.

1. In de klasse Drie wordt een variabele filenaam gedeclareerd, en op twee plaatsen gebruikt (zie de pijlen). Wat merkt de gebruiker daarvan bij het runnen van het programma?
2. In de klasse `Tekening` wordt een `try-catch` opdracht gebruikt. Wat is de betekenis daarvan, en waarom wordt die hier gebruikt?


Punt
Lijn
Beweeg
Punt
Lijn
Beweeg
Punt
Lijn
Beweeg
Punt
Lijn
Beweeg

3. Schrijf de body van de methode `raak` in klasse Punt. Deze methode moet bepalen of de parameters een positie aangeven die binnen het blokje liggen zoals dat getekend wordt.

4. Declareer en initialiseer de benodigde objectvariabelen in de klasse `Tekening`, en schrijf de methode `paint`.

5. Schrijf de methode `zoek` in de klasse `Tekening`, die het Punt zoekt dat overlapt met de meegegeven coordinaten. Als er geen enkel punt voldoet, geeft de methode `null` terug. (Hint: gebruik `raak`!)

6. Schrijf het ontbrekende deel van de methode `mousePressed`, en eventuele andere muis-mehoden die er nog nodig zijn (maar de methoden met een lege body mag je weglaten).

7. Schrijf de ontbrekende delen van de methoden `lees` en `schrijf` van de klasse `Tekening`. De methode `schrijf` moet het plaatje in een zodanig vorm in de tekstfile waarvan de naam gegeven is opslaan, dat het later met behulp van `lees` weer gereconstrueerd kan worden.

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

Bijlagen

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
class Tekening extends Canvas
implements MouseListener, MouseMotionListener
{
    char mode;
    .......... opgave d ..........
    Tekening()
    { mode = 'P';
      .......... opgave d ..........
    }
    public void paint(Graphics g)
    {
      .......... opgave d ..........
    }
    Punt zoek(int ax, int ay)
    {
      .......... opgave e ..........
    }
    public void mousePressed(MouseEvent e)
    {
        switch(mode)
        {
          .......... opgave f ..........
        }
        this.repaint();
    }
    void schrijf(String naam)
    { try
    {
        .......... opgave g ..........
    }
    catch(Exception e)
    { System.out.println("schrijven?");
```

```
    }
    }
    void lees(String naam)
    { try
    {
        .......... opgave g ..........
    }
    catch(Exception e)
    { System.out.println("lezen?");
    }
}

import java.awt.*;
import java.awt.event.*;
class Drie extends Frame
implements ActionListener, WindowListener
{
    Tekening t;
    String filenaam = "tekening.txt";
    public Drie()
    {
        this.setSize(400,400);
        Panel p; Button b;
        t = new Tekening(); t.lees(filenaam);
        p = new Panel();
        p.setLayout( new GridLayout(10,1,5,5));
        b = new Button("Punt");
        p.add(b); b.addActionListener(this);
        b = new Button("Lijn");
        p.add(b); b.addActionListener(this);
        b = new Button("Beweeg");
        p.add(b); b.addActionListener(this);
        this.add(p, BorderLayout.WEST);
        this.add(t, BorderLayout.CENTER);
        t.addMouseListener(t);
        t.addMouseMotionListener(t);
        this.addWindowListener(this);
    }
    public static void main(String [] ps)
    { new Drie().setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    { t.mode = (Button)e.getSource()).
    getLabel().charAt(0);
    }
    public void windowClosing(WindowEvent e)
    { t.schrijf(filenaam);
    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.Graphics;
class Punt
{
    int x, y, d;
    Punt(int ax, int ay)
    { d = 10;
    this.set(ax, ay);
    }
    void verplaats(int ax, int ay)
    { x = ax;
    y = ay;
    }
    void teken(Graphics g)
    { g.fillRect(x-d/2, y-d/2, d, d);
    }
    boolean raak(int ax, int ay)
    {
        .......... opgave c ..........
    }
}

import java.awt.Graphics;
class Lijn
{
    Punt begin, eind;
    Lijn(Punt p1, Punt p2)
    { begin=p1; eind=p2;
    }
    void teken(Graphics g)
    { g.drawLine( begin.x, begin.y,
    eind.x , eind.y );
    }
}
```