Departement Informatica en Informatiekunde, Faculteit Bètawetenschappen, UU.
In elektronische vorm beschikbaar gemaakt door de TBC van A–Eskwadraat.
Het college INFOIMP werd in 2008-2009 gegeven door drs. J. D. Fokker.

Derde deeltentamen Imperatief Programmeren (INFOIMP)
7 november 2008

Opgave 1

(**20 %**)
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.

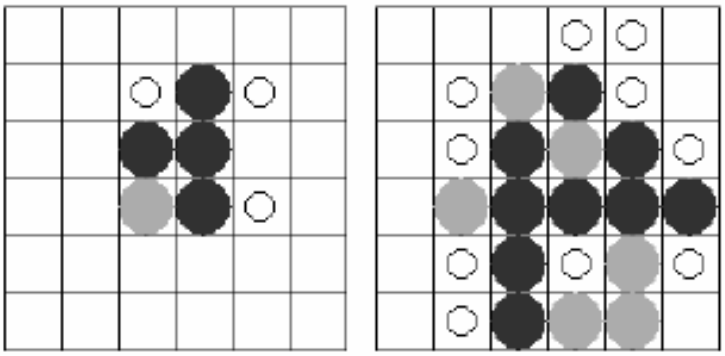
De ingesloten stenen kunnen in acht mogelijke richtingen naast de nieuwe steen liggen: horizontaal, verticaal of diagonaal. Stenen insluiten in meerdere richtingen mag ook. In de figuur is voor twee voorbeelden met open cirkels aangegeven op welke velden de speler met de lichte stenen mag zetten. Als 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 [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, int 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 `(dx,dy)`, waarbij `dx` en `dy` -1, 0 of 1 zijn. De methode `mag` bepaalt of speler `kleur` mag zetten op veld `(x,y)`.

Opdracht: schrijf nu een methode
`public void 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

(**40 %**)
Van onderstaande zes onderdelen mag je er één overslaan.
De andere vijf tellen elk voor 2 punten.
Maak je ze toch alle zes, dan tellen ze elk voor 1.67 punten (als je er eentje fout beantwoordt, had je hem dus beter over kunnen overslaan!)

a) In de body van methode `paint` met parameter `Graphics g` kun je gebruik willen maken van anti-aliasing. Welke opdracht is er nodig om dit te doen?
En wat is anti-aliasing?

b) Eén van de volgende drie declaraties-met-toekenningen is correct. Welke is dat, en waarom zijn de andere twee niet correct?
```
ArrayList a = new Collection(); // versie 1
Collection c = new ArrayList(); // versie 2
Collection c = new Collection(); // versie 3
```
Beschrijf een situatie waarin de correcte versie van bovenstaande regels een voordeel heeft, vergeleken met het in ieder geval ook correcte:
```
ArrayList a = new ArrayList(); // versie 4
```

c) De syntax van de while-opdracht is: `while` *(voorwaarde) opdracht*
De semantiek ervan is dat de opdracht steeds opnieuw wordt uitgevoerd zolang de voorwaarde `true` is.
Opgave: beschrijf nu zelf de syntax en semantiek van de `try`-opdracht. Leg daarbij ook uit hoe er gekozen wordt uit meerdere `catch`-gedeeltes.

d) Wat is een *abstracte methode*?
Abstracte methodes kunnen gedefinieerd worden in een abstracte klasse. In wat voor soort situaties kan dat nuttig zijn?

e) In de userinterface van een programma kun je *menu's* gebruiken. Vergelijk aan de hand van de klasse-hiërarchie de manier waarop dat in AWT en Swing gebeurt. Waarom heeft men in AWT niet voor de Swing-aanpak kunnen kiezen?

f) Bespreek het verschil tussen een variabele met type `double` en een variabele met type `Double`
In welke situatie is het beter om het type `Double` te gebruiken dan het type `double`?

Opgave 3

(**40%**)
Bekijk de application `Test` (met vier andere hulpklassen) op de bijlage. Als dit programma gerund wordt, maakt het een window met drie grote buttons (langs de rechterrand), een scrollbar (onderaan) en een canvas (de rest van het window). Met de buttons kan de gebruiker een van de drie modes uitkiezen: Draw, Top, of Move. De huidige mode wordt als tekst linksboven op het scherm getoond. Aan het begin is dat 'Draw?, maar dat verandert dus als de gebruiker op een van de buttons drukt.

Verder kan de gebruiker met de muis in het canvas werken. De gebruiker drukt de muisknop in, en beweegt de muis dan met ingedrukte muisknop. Er wordt dan een zwarte lijn zichtbaar van het startpunt tot het punt waar de muis zich bevindt. Als de gebruiker de muisknop weer loslaat, verdwijnt de lijn weer. Bovendien gebeurt er dan nog iets, afhankelijk van de gekozen mode:

1. In de Draw-mode: er onstaat een rechthoek met de zojuist verdwenen lijn als diagonaal. De kleur van de rechthoek is een grijstoon die afhangt van de stand van de scrollbar op het moment van loslaten: van zwart (helemaal links) tot wit (rechts).

2. In de Move-mode: één van de rechthoeken verplaatst zich: het punt van de rechthoek dat werd aangeklikt verplaatst van het begin naar het eind van de lijn.

3. In de Top-mode: de aangeklikte rechthoek wordt in zijn geheel zichtbaar (komt 'bovenop? de stapel overlappende rechthoeken te liggen).

a) Schrijf het ontbrekende deel van de methode `beginnen` in klasse `Programma`, waarin de userinterface wordt opgebouwd.

b) Geef de declaraties van de benodigde objectvariabelen en de constructormethode van de klasse `Ruimte`. (Hint: misschien wil je eerst opgave *c, d* en *e* maken).

c) Schrijf de body van methode `actionPerformed` in klasse `Ruimte`.

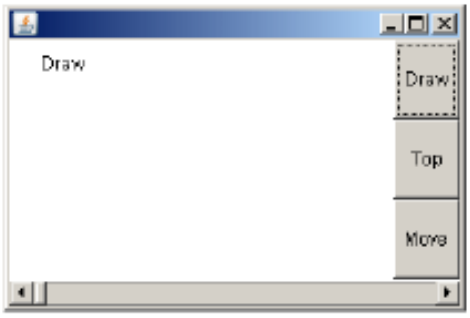
d) Schrijf de body van methode `paint` in klasse `Ruimte`.

e) Schrijf de body van methode `mouseReleased` in klasse `Ruimte`. Hint: houd rekening met de door de gebruiker gekozen mode!

f) Schrijf de ontbrekende delen van de methoden `lees` en `schrijf` van de klasse `Ruimte`. De methode `schrijf` moet het plaatje in een zodanige vorm in een tekstfile (waarvan de naam meegegeven is) opslaan, dat het later met behulp van `lees` weer gereconstrueerd kan worden.

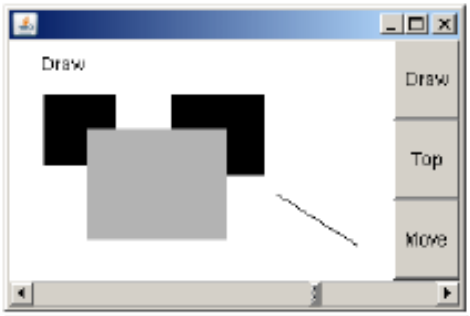
g) We veranderen nu de body van methode main in:
`new ProgrammaPlus().beginnen();`
Schrijf de klasse `ProgrammaPlus`, zodat het programma als extra functionaliteit krijgt dat het plaatje bij afsluiten van het programma automatisch wordt opgeslagen, en bij opnieuw opstarten van het programma weer zichtbaar wordt.

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

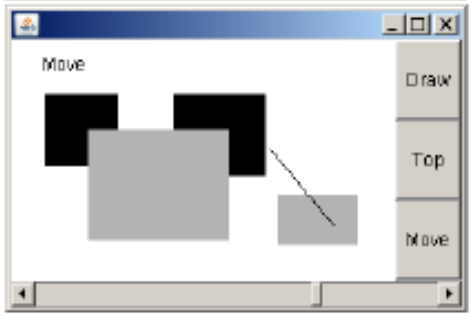


Een paar screenshots als voorbeeld:

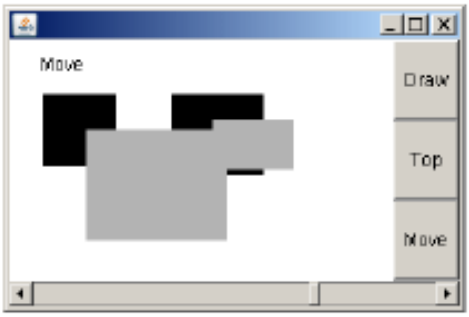
De beginsituatie.



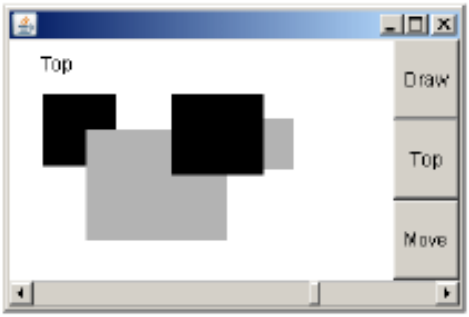
De gebruiker heeft in Draw-mode drie rechthoeken getekend. Bij de derde rechthoek stond de scrollbar een stuk naar rechts.
De gebruiker is nu net bezig om een vierde rechthoek te tekenen (maar heeft de muisknop nog niet losgelaten).



De vierde rechthoek staat er nu.
De gebruiker heeft de Move-mode gekozen, en is nu bezig om de vierde rechthoek een stuk naar linksboven te verplaatsen.



Na het loslaten van de muisknop is de vierde rechthoek verplaatst.



De gebruiker heeft in Top-mode de rechter van de twee zwarte rechthoeken aangeklikt. Die ligt daarom nu weer ‘bovenop’.

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;

public class Ruimte extends Canvas
  implements  MouseListener, MouseMotionListener,
              AdjustmentListener, ActionListener
{
```

Opgave b

```
    public void mousePressed(MouseEvent e)
    {
        start = new Punt(e.getX(), e.getY());
        this.repaint();
    }
    public void mouseDragged(MouseEvent e)
    {
        eind = new Punt(e.getX(), e.getY());
        this.repaint();
    }
    public void adjustmentValueChanged(AdjustmentEvent e)
    {
        k = e.getValue();
    }
    public void actionPerformed(ActionEvent e)
    {
```

Opgave c

```
    }
    public void mouseReleased(MouseEvent e)
    {
```

Opgave e

```
        eind = null;
        this.repaint();
    }

// zie vervolg hiernaast
```

```
// Vervolg van de klasse Ruimte
   public void paint(Graphics g)
    {
```

Opgave d

```
    }

    public Ding welke(int x, int y)
    { // geeft het Ding terug dat op punt (x,y)
      // bovenop ligt, of null als hier niets staat
```

details zijn niet van belang

```
    }

    public void schrijf(String naam)
    {
        try
        {
```

Opgave f

```
        }
        catch(Exception e)
        {   System.out.println(""+e);
        }
    }

    public void lees(String naam)
    {   try
        {
```

Opgave f

```
        }
        catch(Exception e)
        {   System.out.println(""+e);
        }
    }

    public void mouseMoved(MouseEvent e)  {}
    public void mouseClicked(MouseEvent e)  {}
    public void mouseEntered(MouseEvent e)  {}
    public void mouseExited(MouseEvent e)   {}
}
```

N.B.: de afmetingen van de grijze hokken zijn *geen* indicatie voor de hoeveelheid afgedekte tekst.

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class Programma extends Frame
{
    Ruimte r;
    Scrollbar s;

    public void beginnen()
    {
        this.setSize(700,500);
        r = new Ruimte();
        s = new Scrollbar(Scrollbar.HORIZONTAL,
                          0, 1, 0, 255);
        Button b1, b2, b3;
        b1 = new Button("Draw");
        b2 = new Button("Top");
        b3 = new Button("Move");

        b1.addActionListener(r);
        b2.addActionListener(r);
        b3.addActionListener(r);
        r.addMouseListener(r);
        r.addMouseMotionListener(r);
        s.addAdjustmentListener(r);
```


```
        this.setVisible(true);
    }
}
```

```
public class Test
{
    public static void main(String [] args)
    {
        new Programma().beginnen();
    }
}
```

```
import java.awt.Graphics;
import java.awt.Color;

public class Ding
{
    int x, y, w, h;
    Color c;

    public Ding(Punt p1, Punt p2, int k )
    {
        x = Math.min(p1.x, p2.x);
        y = Math.min(p1.y, p2.y);
        w = Math.abs(p2.x - p1.x);
        h = Math.abs(p2.y - p1.y);
        c = new Color(k,k,k);
    }
    public boolean raak(int xa, int ya)
    {
        return xa>=x && xa<=x+w
            && ya>=y && ya<=y+h;
    }
    public void teken(Graphics g)
    {
        g.setColor(c);
        g.fillRect(x, y, w, h);
    }

    public String tekst()
    {
        return x + " " + y + " " + (x+w) + " "
                 +   (y+h) + " " + c.getRed();
    }
}
```

```
public class Punt
{
    public int x, y;

    public Punt(int x0, int y0)
    {   x = x0;
        y = y0;
    }
}
```