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, tweede deeltentamen (INFOIMP) 14 oktober 2005

Opgave 1

(25 punten)

In de klasse String zit onder andere een methode `compareTo`, met de volgende header:

```
int compareTo(String s)
```

Deze methode levert 0 op als de string die door de methode onder handen wordt genomen, en de string die als parameter wordt meegegeven, precies gelijk zijn.
De methode levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de string onder handen kleiner is dan de parameter, en een positief getal (bijvoorbeeld 1) als de string onder handen groter is dan de parameter. Met 'kleiner' en 'groter' wordt hier de woordenboek-ordening bedoeld: de eerste letter waar de strings verschillen bepaalt de ordening (volgens de Unicodes van die letters).
Is de ene string een beginstuk van de andere, dan is de kortste de kleinste. Spaties en leestekens tellen gewoon mee, die hoeven dus niet speciaal behandeld te worden.

Voorbeelden:

`‘‘aap’’ .compareTo(‘‘noot’’)`	geeft negatief getal, want 'a'<'n'
`‘‘noot’’.compareTo(‘‘nieten’’)`	geeft positief getal, want 'o'>'i'
`‘‘niet’’.compareTo(‘‘nietmachine’’)`	geeft negatief getal vanwege lengte
`‘‘noot’’.compareTo(‘‘noot’’)`	geeft nul, want precies gelijk
`‘‘noot’’.compareTo(‘‘NOOT’’)`	geeft positief getal, want 'n'>'N'

Stel dat je de auteur van de klasse `String` bent. Je hebt alle methoden van die klasse al geschreven, behalve de methoden `compareTo` en `equals`.

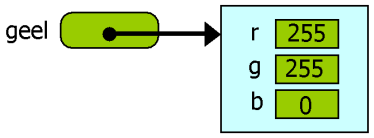
De opgave: Schrijf nu de methode `compareTo`.

Opgave 2

(20 punten)

Gegeven zijn de volgende klasse-definities:

```
class Een
{   int x;
    public Een()
    {   x = 5;
    }
}
class Twee
{   int n;
    Een e;
    public Twee(int x)
    {   e = new Een();
        n = x+1;
    }
}
class Drie extends Een
{   int y;
    Twee p, q;
    public Drie()
    {   y = 3;
        p = new Twee(y);
        q = p;
    }
}
```



Het plaatje rechtsboven toont de situatie in het geheugen die onstaat na uitvoering van

```
Color geel = new Color(255,255,0);
```

Teken, in dezelfde stijl als het voorbeeld-plaatje, de situatie die in het geheugen ontstaat na uitvoering van

```
Drie d = new Drie();
```

Maak, net als in het voorbeeld, duidelijk onderscheid tussen de naam en de waarde van de variabelen: de naam staat *naast* de hokjes, de waarde er *in*.
Object-verwijzingen moeten, net als in het voorbeeld, met een duidelijke stip beginnen in het hokje van de verwijzings-variabele, en wijzen naar de *rand* van het object.

Opgave 3

(25 punten)

Bekijk de listing hieronder, waarin de klassen Test, Gebied, en Ruimte worden gedefinieerd. Dit programma wordt door de compiler zonder foutmeldingen geaccepteerd.

a) Welke methode(n) moet(en) er nog worden gedefinieerd in de klasse Test, welke in Gebied, en welke in Ruimte?

b) Maak een schets van hoe het scherm van de applet in werking er uit komt te zien, en geef daarin aan wat er groen, blauw, geel en rood is.

c) Bekijk de aanroep

```
g1 = new Gebied(Color.GREEN, this, this);
```

Wat is hier het type van `this`, en waarom is dat voor de tweede èn de derde parameter acceptabel?

d) Met de aanroep `t.start()` wordt een animatie gestart. Hoe kan deze animatie weer worden gestopt?

Listing bij opgave 3

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

public class Test extends Applet
                      implements ......
{
    public void init()
    {
       Gebied g1, g2;

        g1 = new Gebied(Color.GREEN, this, this);
        g2 = new Gebied(Color.BLUE , g1,   this);
        this.add(g1);
        g1  .add(g2);

        Thread t;
        t = new Thread(g1);
        t.start();
    }

    ......
}

class Gebied extends Panel
                implements ......
{
    Ruimte r1, r2;

    Gebied(Color k, Container c, ActionListener a)
    {
        Button b;
        b = new Button("hoi");
        this.setBackground(k);
        r1 = new Ruimte(100);
        r2 = new Ruimte(50);
        this.add(r1);
        c.add(b);
        b.addActionListener(a);
        this.add(r2);
    }

    ......
}

class Ruimte extends Canvas
                implements ......
{
    Ruimte(int n)
    {
        this.setBackground(Color.YELLOW);
        this.setSize(n,n);
        this.addMouseMotionListener(this);
    }

    public void paint(Graphics g)
```

```
    {
        g.setColor(Color.RED);
        g.fillRect(25,25,50,50);
    }

......
}
```

Opgave 4 *(35 punten)*

Om het schrijfwerk te beperken mag je in deze opgave:

- de HTML-file weglaten: je hoeft alleen de Java-file te schrijven;
- de import-regels bovenaan het programma weglaten;
- de muis-methoden die een lege body hebben weglaten;
- door afrondfouten onstane kleine afwijkingen laten maken door je programma.

Schrijf een applet met de volgende eigenschappen:

1. De gebruiker ziet bovenaan het scherm:
   - een button met het opschrift "Leeg"
   - een schuifregelaar
   - een button met het opschrift "Lijn"
2. Aan het begin staat het schuivertje van de schuifregelaar ongeveer in het midden.
3. Iedere keer als de gebruiker ergens klikt ontstaat er, gecentreerd op dat punt, een vierkant met lengte en breedte 20.
4. De kleur van het vierkant kan allerlei grijstinten aannemen, met wit en zwart als uitersten. De grijswaarde wordt bepaald door de stand van de schuifregelaar op het moment dat het punt wordt aangeklikt: hoe verder naar links, hoe lichter. De schuifregelaar helemaal naar rechts is zwart.
5. Rondom de vierkantjes zit altijd een zwarte rand.
6. Er zijn maximaal 100 vierkanten zichtbaar. Als de gebruiker daarna toch meer punten aanklikt, gebeurt er niets (maar er mag ook geen run-time fout optreden!).
7. Na het indrukken van de "Lijn" knop verschijnt er een lijn tussen het lichtste en het donkerste vierkant. Als er twee vierkanten even licht of donker zijn, kies je daarvan het vierkant dat het eerst was aangeklikt.
8. Na het nogmaals indrukken van "Lijn" verdwijnt de lijn weer, bij de derde klik verschijnt hij weer, enz.
9. Na het indrukken van "Leeg" verdwijnen alle vierkanten. De gebruiker kan dan weer met 100 nieuwe kliks beginnen.

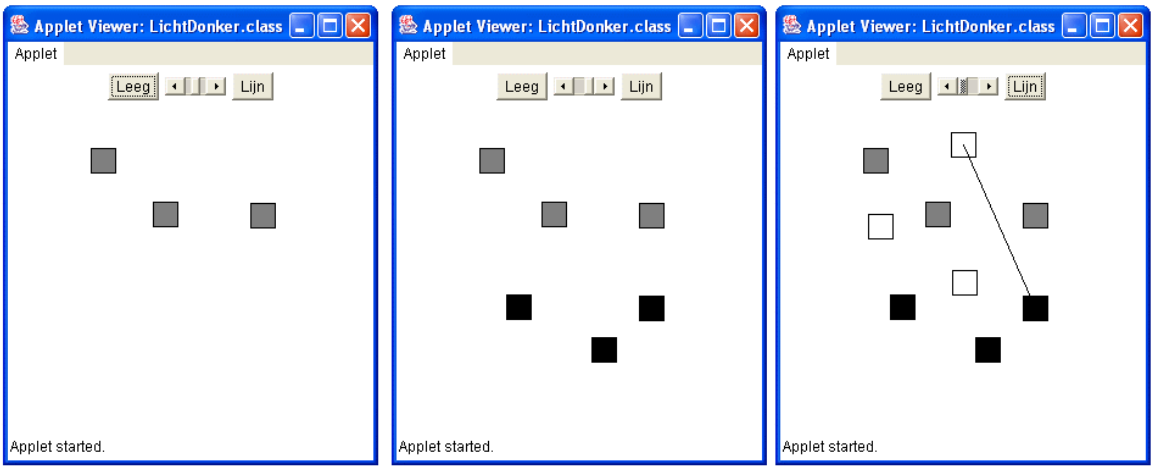
Op de afbeelding zie je achtereenvolgens de situatie:

a) Na het aanklikken van 3 punten: er verschijnt een grijs vierkant (want de schuifregelaar staat nog in het midden) met een zwarte rand op de aangeklikte plaatsen.

b) Na het verschuiven van het schuivertje helemaal naar rechts, en het aanklikken van nog drie punten: hier verschijnen zwarte vierkanten.

c) Na het verschuiven van het schuivertje helemaal naar links, het aanklikken van nog drie punten, en het indrukken van "Lijn":

   er verschijnen witte vierkanten, en er komt een lijn van het eerst aangeklikte witste vierkant naar het eerst aangeklikte zwartste vierkant.


Applet Viewer: LichtDonker.class
Applet
Leeg
Lijn
Applet started.
Applet Viewer: LichtDonker.class
Applet
Leeg
Lijn
Applet started.
Applet Viewer: LichtDonker.class
Applet
Leeg
Lijn
Applet started.