Tentamen Objectgeorienteerd Programmeren IN1205
15 januari 2007 9.00-12.00
Afdeling ST Faculteit EWI TU Delft

Bij dit tentamen mag u gebruik maken van: Barnes, Object-Oriented Programming with Java en de Inleiding Algoritmiek.

puntenverdeling:

1	2	3	4	5	6
6 * 2	3 * 6	6 + 9	6 + 6	6 * 3	15

Opgave 1.
Bijlage A bevat de code van de klasse Vat. Deze klasse beschrijft vaten die kunnen worden gevuld met olie. Klasse Vat heeft 3 attributen: code: String, volume: double en inhoud: double. Het **volume** geeft aan hoeveel olie een vat kan bevatten, de **inhoud** hoeveel olie er werkelijk aanwezig is.

Gegeven de volgende declaraties en opdrachten:

```
Vat vat1 = new Vat("a12", 30.8);
Vat vat2 = new Vat("a12", 30.8);
Vat vat3 = new Vat("b34", 23.7);

vat1.vulBij(21.7);
vat2.vulBij(12.6);
vat3.vulbij(12.6);
vat1 = vat3;
```

Wat is nadien de waarde van elk van de volgende expressies? Licht je antwoord toe! Mocht een van de expressies aanleiding geven tot een compileer of run-time fout, dan dien je dat aan te geven samen met de reden van de fout.

```
a.   vat1 == vat2
b.   vat1 == vat3
c.   vat1.equals(vat2)
d.   vat1.getInhoud() > vat2.getInhoud()
e.   vat1.getCode().equals(vat2.getCode())
f.   vat1.getCode() == vat3.getCode()
```

Opgave 2
Met behulp van klasse Vat construeren we een klasse Vaten die een lijst bevat van een aantal vaten. Een deel van de implementatie van Vaten is hieronder gegeven:

```
public class Vaten{

     public Vaten(int n){
          vaten = new Vat[n];
          aantal = 0;
     }
```

```
    public void voegToe(Vat v){
    //pre:  er is nog ruimte
    //post: v is toegevoegd
         vaten[aantal] = v;
         aantal = aantal + 1;
    }

    private Vat[] vaten;
    private int aantal;//het aantal vaten in het array
}
```

Let erop dat het volume van de vaten onderling kan verschillen.

Implementeer in de klasse Vaten de volgende methoden:

```
a.  public double totaleInhoud()
    //post: retourneert de som van de inhoud van de
    //      vaten.
```

De relatieve vulling van een Vat geeft de verhouding aan tussen de inhoud en het volume. Is de inhoud 0, dan is de relatieve vulling 0%. Is de inhoud == volume, dan is de relatieve vulling 100%.

```
b.  public int indexGrootsteRelVulling()
    //pre: aantal > 0
    //post: retourneert de index van het vat met de
    //      grootste relatieve vulling (GRV).
    //     Indien de GRV meer dan eenmaal voorkomt
    //      wordt de grootste index teruggegeven
    //      van een vat met de GRV.
```

Stel dat de inhoud van de vaten zo wordt herverdeeld, dat de vaten met index 0 t/m i, volledig zijn gevuld, vat met index i+1 leeg is of gedeeltelijk gevuld en de overige vaten leeg. Wat is dan de index van het eerste lege vat?

```
c.  public int indexEersteLegeVat()
    //post: retourneert de index van het eerste lege
    //      vat, indien de inhoud is herverdeeld op de
    //      manier die hierboven is toegelicht.
```

Opgave 3
In een tekstfile staan een aantal vaten gerepresenteerd als tekst. De eerste regel van de file geeft het aantal vaten weer, elke volgende regel bevat de data van een vat.
De data van een vat zijn: code, volume en inhoud, gescheiden door spaties.
Voorbeeld:

```
3
a12  30.8 21.7
a12 30.8  12.6
b343 23.71 12.63
```

Voeg aan de klasse Vaten de volgende twee methoden toe:

a.
```
public void schrijfNaar(String bestand)
//post: de inhoud van dit vat is naar het
//      tekstbestand genaamd 'bestand'
//      geschreven, in bovenstaand formaat
```

b.
```
public static Vaten leesVan(String bestand)
//pre : het tekstbestand genaamd 'bestand' bevat
//      nul of meer Vaten, in bovenstaand formaat
//post: retourneert een object van type Vaten
//      met de vaten uit dit bestand.
```

NB U mag hier **niet** de klasse SimpleInput van Barnes gebruiken, wel (zonodig) de klasse StringTokenizer.
NB Deze methoden werpen geen Exceptions.

Opgave 4

a. Geef een volledig klassendiagram van de klassen uit bijlage B
b. Geef een sequencediagram van de volgende methode:

```
public double test(){
    Functie f1 = new Constant(3);
    Functie f2 = new Multiplier(4);
    Functie f3 = new Compose(f1, f2);
    return f3.apply(5);
}
```

Opgave 5

Onderstaande zes programmafragmenten zijn gebaseerd op de definities uit bijlage B. Geef van elk van de volgende programmafragmenten aan:

- of het tijdens het vertalen fouten oplevert (zo ja, welke),
- zo neen, of het tijdens het verwerken fouten oplevert (zo ja, welke),
- wat (in geval van correcte verwerking) de waarde van x wordt.

a.
```
Id multiplier  = new Multiplier(7);
Functie f = multiplier.afgeleide();
double x = f.apply(2);
```

b.
```
Functie functie  = new Multiplier(6);
Constant constant = (Constant)functie;
double x = constant.apply(2);
```

c.
```
Id id  = new Multiplier(5);
Constant constant = (Constant)id;
double x = constant.apply(2);
```

d.
```
Id id  = new Multiplier(4);
Multiplier multiplier = (Multiplier)id;
double x = multiplier.apply(2);
```

e.
```
Id id  = new Multiplier(3);
double x = id.apply(2);
```

f.
```
Functie functie  = new Id();
double x = functie.apply(2);
```

Opgave 6
Deze opgave speelt zich af in de klasse Getalrij:

```
public class Getalrij{
    int [] rij;
    int aantal;
    ...
}
```

U mag aannemen dat het array rij gecreëerd is, en nul of meer getallen bevat (namelijk het aantal gegeven door het attribuut aantal). De rij in deze Getalrij is niet-dalend.

Implementeer met behulp van het zeven-stappen-plan de volgende methode in de klasse Getalrij:
```
public int aantalVerschillende()
// post: retourneert het aantal verschillende elementen
//         in de rij.
```

NB Punten voor deze opgave worden verdiend door juiste toepassing van het zeven-stappen-plan. Alleen een implementatie levert weinig op - zelfs als hij correct is.

Bijlage A

```
public class Vat{

   public Vat(String c, double v){
      volume = v;
      code = c;
   }

   public double getVolume(){
      return volume;
   }

   public double getInhoud(){
       return inhoud;
   }

   public String getCode(){
       return code;
   }

   public void vulBij(double hoeveel){
      inhoud += hoeveel;
   }

   public boolean equals(Object o){
      if (o instanceof Vat){
         Vat other = (Vat) o;
         return this.code.equals(other.code) &&
                this.volume == other.volume;
      }
      else
         return false;
   }

   private double inhoud;
   private double volume;
   private String code;

}
```

Bijlage B

```
public abstract class Functie{
   public double apply(double x){
      return x;
   }
}

public class Id extends Functie{
}

public class Constant extends Functie{

   public Constant (double c){
      this.c = c;
   }

   public double apply(double x){
      return c;
   }

   private double c;
}

public class Multiplier extends Id{
   public Multiplier(double f){
      factor = f;
   }

   public double apply(double k){
      return k * factor;
   }

   public Functie afgeleide(){
      return new Constant(factor);
   }

   private double factor;
}

class Compose extends Functie{
    public Compose (Functie first, Functie second){
       this.first = first;
       this.second = second;
    }

    public double apply(double x){
       double y = first.apply(x);
       return second.apply(y);
    }

    private Functie first, second;
}
```