Tentamen Objectgeorienteerd Programmeren IN1205
15 juni 2009 9.00-12.00
Afdeling ST Faculteit EWI TU Delft

Bij dit tentamen mag u gebruik maken van: Java in Two Semesters (Charatan & Kans) en de Inleiding Algoritmiek.
Dit tentamen bestaat uit 6 opgaven, en heeft drie bijlagen.

Opgave 1 (5 * 3)
Voor de ontwikkeling van een programma dat betalingen beheert, zijn de klassen Overschrijving en Betalingen geschreven. Klasse Overschrijving bevat de gegevens (bedrag en rekening(nummer)) van 1 overschrijving. Klasse Betalingen bevat 0 of meer overschrijvingen. In Bijlage A is een deel van de implementatie van deze klassen gegeven.

In klasse Betalingen bevat een methode `test()`. Wat is na de uitvoering van deze methode de waarde van onderstaande expressies? Licht uw antwoord toe.

Mocht een van de expressies aanleiding geven tot een compileer- of runtime-fout, dan dient u dit aan te geven samen met de reden van de fout.

```
a) betalingen[0] == betalingen[2];
b) betalingen[1] == betalingen[2];
c) betalingen[0].getRekening();
d) betalingen[1].getRekening();
e) betalingen[2].getRekening() == betalingen[1].getRekening();
```

Opgave 2 (3 + 5 + 7)
Aan klasse Betalingen moeten 3 methoden worden toegevoegd. De specificatie van de methoden is hieronder gegeven.

a) Geef een implementatie van methode somBedragen() die de som van de bedragen berekent van de overschrijvingen die in betalingen zijn opgeslagen.

```
somBedragen() : double
post: retourneert de som van de bedragen van de overschrijvingen die
zijn opgeslagen in betalingen.
```

b) Geef een implementatie van methode betalingenAan die alle overschrijvingen naar een opgegeven rekening geeft.

```
betalingenAan(rek: Rekening) : Betalingen
post: retourneert een Betalingen-object dat alle overschrijvingen
naar rekening rek bevat.
```

c) Geef een implementatie van methode bevatRekeningMetNOverschrijvingen die vaststelt of betalingen een rekening bevat waarnaar *n* overschrijvingen worden gedaan.

```
bevatRekeningMetNOverschrijvingen(n : int) : boolean
post: retourneert true dsd er in betalingen een rekening voorkomt met
precies n overschrijvingen.
```

Opgave 3 (8 + 7)

a) Geef een *volledig* klassediagram van de definities in bijlage B.

b) Geef een sequencediagram van een aanroep van de methode test:

```
public static boolean test(int k){
   Divides divides100 = new Divides(100);
   Between between7and12 = new Between(7,12);
   And both = new And(divides100, between7and12);
   return both.isSatisfiedBy(k);
}
```

Opgave 4 (5 * 3 punten)

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

- Of het tijdens het compileren fouten oplevert (en zo ja, welke)
- Zo neen, of er tijdens het verwerken fouten oplevert (zo ja, welke)
- Wat (in geval van correcte werking) na afloop de waarde van de variabele `bool` is.

```
a.  Predikaat p = new AtLeast(5);
    AtLeast al = p;
    boolean bool = al.isSatisfiedBy(7);

b.  Predikaat p = new AtLeast(5);
    Between b  = (Between) p;
    boolean bool = b.isSatisfiedBy(7);

c.  AtLeast al = new Between(5, 10);
    boolean bool = al.isSatisfiedBy(30);

d.  Predikaat p = new AtLeast(5);
    AtLeast al = (AtLeast) p;
    boolean bool = al.isSatisfiedBy(7);

e.  AtLeast be = new Between(5, 10);
    boolean bool = be.isFalse();
```

Opgave 5 (5 + 4 + 6 punten)

De klassen uit bijlage C vormen een (incomplete) implementatie van een simulatie, en een Gui, waarop de simulatie zichtbaar wordt. In de methode stap van de Simulatie wordt een simulatiestap uitgevoerd. Het resultaat daarvan wordt zichtbaar gemaakt op het tekenPanel.

a) De simulatie moet starten (methode gaMaar in de SimulatieGui) als er op de startknop wordt geklikt. Maak een inner class in de klasse SimulatieGui die events van de startknop op gepaste wijze afhandelt.

b) De simulatie loopt zolang de variabele `ready` in Simulatie de waarde **`false`** heeft. Omdat deze waarde nooit verandert stopt de simulatie nooit. De bedoeling van de stopknop is, dat de variabele ready de waarde **`true`** krijgt als er op de stopknop wordt geklikt. Maak van de klasse Simulatie een ActionListener, die events van de stopknop op gepaste wijze afhandelt. Gebruik **geen** inner class. Geef nauwkeurig aan welke wijzigingen u in welke klasse(n) aanbrengt.

c) De oplossing onder b) werkt niet goed, omdat de thread die de simulatie laat werken dezelfde is als de thread die het event van de stopknop afhandelt. Als er op de stopknop gedrukt wordt, blijft dat event dus liggen tot de simulatie klaar is. Zorg dat de simulatie in een aparte thread wordt afgehandeld, zodat er een thread beschikbaar blijft voor het afhandelen van events. Geef nauwkeurig aan wat u waar wijzigt.

Opgave 6 (15 punten)

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 één of meer getallen bevat (namelijk het aantal gegeven door het attribuut aantal). De rij in deze Getalrij is niet-dalend, dat betekent (onder meer) dat een getal meer dan één keer kan voorkomen. Onder de frequentie van een getal verstaan we het aantal malen dat dat getal voorkomt. We willen weten wat de grootst voorkomende frequentie is, dat wil zeggen, hoevaak het getal voorkomt dat het vaakst voorkomt.
Voorbeeld: in de rij 0, 1, 2, 2, 2, 3, 4, 4, 4, 5 is de grootst voorkomende frequentie 3.
Implementeer met behulp van het zeven-stappen-plan in deze klasse de methode :
`public int hoogsteFrequentie()`
geeft het aantal malen dat het meest voorkomende getal voorkomt.

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 Overschrijving{

    private double bedrag;
    private String rekening;

    public Overschrijving(double b, String r){
        setBedrag(b);
       setRekening(r);
    }

    public void setBedrag(double b){
        bedrag = b;
    }

    public double getBedrag(){
        return bedrag;
    }

    public void setRekening(String r){
        rekening = r;
    }

    public String getRekening(){
        return rekening;
    }
}


public class Betalingen{

    private Overschrijving[] betalingen;
    private int aantal;

    public Betalingen(){
        betalingen = new Overschrijving[200];
        aantal = 0;
    }

    public void voegToe(Overschrijving ov){
        betalingen[aantal] = ov;
        aantal = aantal + 1;
    }

    public void test(){
        Overschrijving ov = new Overschrijving(100.00,"A15");
        voegToe(ov);

        ov = new Overschrijving(200.00,"B33");
        voegToe(ov);

        ov.setBedrag(betalingen[0].getBedrag());
        ov.setRekening(betalingen[0].getRekening());
        voegToe(ov);
    }
}
```

Bijlage B

```
class Predikaat{
   boolean isSatisfiedBy(int k){
      return true;
   }
}

class Divides extends Predikaat{
   int multiple;
   public Divides(int m){
       multiple = m;
   }

   public boolean isSatisfiedBy(int n){
       return multiple % n  == 0;
   }
}

class AtLeast extends Predikaat{
   int minimum;
   public AtLeast(int m){
      minimum = m;
   }

   public boolean isSatisfiedBy(int n){
      return n >= minimum;
   }
}

class Between extends AtLeast{
   private int maximum;

   public Between(int min, int max){
      super(min);
      maximum = max;
   }

   public boolean isSatisfiedBy(int n){
      return super.isSatisfiedBy(n) && n <= maximum ;
   }

   public boolean isFalse(){
       return maximum < minimum;
   }
}

class And extends Predikaat{
   Predikaat p1, p2;
   public And(Predikaat p1, Predikaat p2){
      this.p1 = p1;
      this.p2 = p2;
   }

   public boolean isSatisfiedBy(int n){
      return p1.isSatisfiedBy(n) && p2.isSatisfiedBy(n);
   }
}
```

Bijlage C

```
import java.awt.*;
public class SimulatieGui extends Frame{

  private Button stopknop       = new Button();
  private Button startknop      = new Button();
  private Panel simulatiePanel  = new Panel();

  public SimulatieGui(){
    ...
    add(simulatiePanel);
    ...
    add(startknop);
    ...
    add(stopknop);
    ...
  }

  public void gaMaar(){
     Simulatie simulatie = new Simulatie(simulatiePanel);
     ...
     simulatie.go();
  }

  public static void main (String [] args){
     SimulatieGui gui = new SimulatieGui();
  }

}


public class Simulatie{

  private Panel tekenPanel;
  private boolean ready = false;

  public Simulatie(Panel panel){
    tekenPanel = panel;
      ...
  }

  public void go(){
    while (!ready)
      stap();
  }

  private void stap(){
     // doe een simulatiestap
     // en laat het resultaat zien op het tekenPanel.
  }
```