Tentamen Inleiding Programmeren (IN1608WI), 1 november 2011, 14.00-16.00
Technische Universiteit Delft, Faculteit EWI, Afdeling 2
Gesloten boek tentamen. Je mag geen gebruik maken van een tekstboek of slides.

Opgave 1 (15 punten)

Voor een financieel programma zijn de klassen Overschrijving en Betalingen geschreven. Klasse Overschrijving bevat het bedrag en rekening(nummer) van 1 overschrijving. Klasse Betaling bevat een lijst met 0 of meer overschrijvingen. In Bijlage A is een deel van de implementatie van deze klassen gegeven.

Klasse Betalingen bevat een methode **`test().`** Wat is na de uitvoering van deze methode de waarde van onderstaande expressies? Licht je antwoord toe.

Mocht een van de expressies aanleiding geven tot een compileer- of runtime-fout, dan dient je 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(30) punten)

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 n
overschrijvingen.
```

Opgave 3 (35 punten)

Gegeven de specificatie van een klasse Verzameling die getallen kan bevatten uit het interval [og,bg], met 0 <= og <= bg, Geef een implementatie van deze klasse.

Verzameling
-elementen: boolean[] **-onderGrens : int** **-bovenGrens : int**
+Verzameling(og : int, bg : int) **pre: 0 <= og <= bg** **post: indien pre geldt is er een lege verzameling gecreëerd van getallen uit het interval [og,bg], anders is het interval leeg, dwz [].** **+bevat(e : int): boolean** **pre: onderGrens <= e <= bovenGrens** **post: retourneert true indien pre geldt en e in de verzameling voorkomt, retourneert anders false** **+voegToe(e : int)** **pre: onderGrens <= e <= bovenGrens** **post: indien pre geldt en !bevat(e), is e aan de verzameling toegevoegd** **+doorsnijding(that: Verzameling) : Verzameling** **post: retourneert de verzameling van alle getallen die zowel in this als in that voorkomen**

Toelichting van het attribuut elementen: de indices van elementen corresponderen met de getallen uit het interval [og,bg]. Als elementen[i] true is, betekent dat i in de verzameling voorkomt, is elementen[i] false, dan komt i niet in de verzameling voor.

Opgave 4 (10 punten)

In de schakeltechniek wordt de werking van netwerken van schakelaars beschreven mbv logische operatoren. Naast de bekende logische operatoren AND en OR wordt daarbij de operator NOR gebruikt.

De waarheidstabel van NOR ziet er als volgt uit:

a	**b**	**a NOR b**
true	**true**	**false**
true	**false**	**false**
false	**true**	**false**
false	**false**	**true**

Geef een implementatie van methode NOR die aan de waarheidstabel voldoet.

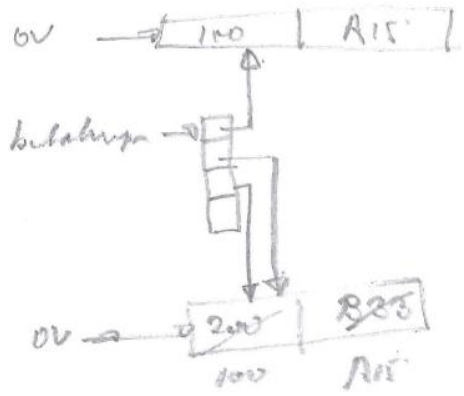
```
public static boolean NOR(boolean a, boolean b)
post: retourneert de waarde die voldoet aan de waarheidstabel van NOR
```

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);
	}
}
```

Uitwerking:
1 (5 x 3 punten), 1,5 punt voor antwoord, 1,5 voor toelichting

Tekening situatie:



a. false, niet dezelfde objecten
b. true, dezelfde objecten
c. "A15", oorspronkelijke waarde van rekeningnummer
d. "A15", waarde van aangepast object ov
e. True, zelfde objecten

```
2a.
public double somBetalingen(){
      double res = 0;
      for(int i = 0;i < aantal; i++)
            res = res + betalingen[i].getBedrag();
      return res;
}

2b.
public Betalingen betalingenAan(String rekening){
      Betalingen res = new Betalingen();
      for(int i = 0; i < aantal; i++)
            if (betalingen[i].getRekening().equals(rekening))
                  res.voegToe(betalingen[i]);
      return res;
}
2c.
public boolean rekeningMetNBetalingen(int n){
      for(int i = 0; i < aantal; i++){
            int count = 0;
            String rekening = betalingen[i].getRekening();
            for(int j = 0; j < aantal; j++)
                  if (betalingen[j].getRekening().equals(rekening))
                        count = count + 1;
            if (count == n)
                  return true;
      }
      return false;
   }
}
```

3.

```
public class Verzameling{

        private int onderGrens;
        private int bovenGrens;
        private boolean[] elementen;

        public Verzameling(int og, int bg){
               if (0 <= og && og <= bg){
                  elementen = new boolean[bg+1];
                  onderGrens = og;
                  bovenGrens = bg;
               }
               else
                   elementen = new boolean[0];
        }

        public boolean bevat(int e){
               if (onderGrens <= e && e <= bovenGrens)
                  return elementen[e];
               else
                   return false;
        }

        public void voegToe(int e){
               if (onderGrens <= e && e <= bovenGrens)
                  elementen[e] = true;
        }

        public Verzameling doorsnijding(Verzameling that){
               int og1 = this.onderGrens;
               int og2 = that.onderGrens;
               int bg1 = this.bovenGrens;
               int bg2 = that.bovenGrens;
               int og = og1;
               if (og2 > og)
                  og = og2;
               int bg = bg1;
               if (bg2 < bg)
                  bg = bg2;
               Verzameling res = new Verzameling(og,bg);
               for(int i = og;i <= bg;i++)
                       if (this.bevat(i) && that.bevat(i))
                          res.voegToe(i);
               return res;
        }
}
```

4.

```
public static boolean NOR(boolean a, boolean b){
      return !(a || b);
}
```