Tentamen Objectgeorienteerd Programmeren TI1200
26 januari 2011 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 slides over het stuk Algoritmen.
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 vier programmafragmenten zijn gebaseerd op de definities uit bijlage B.

a. De code in bijlage bevat 1 fout, geef aan waar de fout zit en waarom de Java compiler het er niet mee eens is. Let wel, het gaat om compileerfouten, niet semantische fouten!

Eenmaal de code in bijlage B verbeterd is, kijk dan naar de 4 programmafragmenten en bepaal:

- Of het tijdens het compileren fouten oplevert (en zo ja, welke). Hoe kan je dit oplossen (schrijf het volledig verbeterde statement op!)
- Zo neen, of er tijdens het verwerken fouten oplevert (zo ja, welke). Hoe kan je dit oplossen? (schrijf het volledig verbeterde statement op!)
- Wat is na afloop de waarde dit wordt afgedrukt op het scherm? (indien het programma goed werkt, maar ook na het oplossen van het probleem!)

b.
```
Voertuig vervoermiddel1 = new HybrideAuto("Honda Insight");
System.out.println(vervoermiddel1.getVerbruik());
```

c.
```
Voertuig vervoermiddel2 = new ElektrischeFiets("Gazelle");
Fiets fiets = vervoermiddel2;
System.out.println(fiets.CO2_Uitstoot());
System.out.println(fiets.getMerk());
```

d.
```
Voertuig vervoermiddel3 = new HybrideAuto("Honda CRZ");
System.out.println(vervoermiddel3.getMerk());
```

e.
```
Voertuig vervoermiddel4 = new Auto("Volkswagen Golf");
Auto auto = (HybrideAuto)vervoermiddel4;
System.out.println(auto.CO2_Uitstoot());
```

Opgave 5 (7.5 + 7.5 punten)

De klassen uit bijlage C vormen een (incomplete) implementatie van een PING applicatie. Je ziet hier de server-kant van de applicatie die binnenkomende oproepen ontvangt en dan “PING” terugstuurt. Er zijn echter wat problemen met deze applicatie, voer de volgende 2 stappen uit om de applicatie te verbeteren:

a) Zorg ervoor dat de “PING” wordt teruggestuurd als er op de knop “Send PING over network” wordt gedrukt. Maak hiervoor gebruik van een ActionListener. Wijzig enkel de klasse **ClientWorker** en geef nauwkeurig aan welke wijzigingen je doorvoert.

b) Zorg er ook voor dat er meerdere verbinden tegelijk kunnen behandeld worden. M.a.w. zorg dat elke binnenkomende oproep wordt afgehandeld door een eigen thread. Wijzig opnieuw enkel de klasse **ClientWorker** en geef nauwkeurig aan welke wijzigingen je doorvoert.

Opgave 6 (15 punten)

Deze opgave speelt zich af in de klasse Getalrij:

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

De rij in een Getalrij is niet-dalend. Gegeven is een getal q. Ga na of er twee getallen in de rij staan, waarvan de som gelijk is aan q.

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. Ga na of er twee getallen in de rij staan, waarvan de som gelijk is aan q. (1 instantie van deze situatie vinden is genoeg)

Voorbeeld: stel dat q 7 is en de array volgende elementen bevat 0, 1, 1, 2, 2, 3, 4, 5, 6, 7, 8 dan moet de methode “true” teruggeven omdat 3 en 4 naast elkaar staan en hun som 7 is.

Implementeer met behulp van het zeven-stappen-plan in deze klasse de methode : **`public boolean`** `bevatSom(int q),` die true teruggeeft als er 2 elementen in de rij naast elkaar staan waarvan de som gelijk is aan de parameter q en false in alle andere gevallen.

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

```
public abstract class Voertuig{
	protected static final int fUitstoot = 95;

	private String fMerk;

	public Voertuig(String merk)
	{
		fMerk = merk;
	}

	public String getMerk()
	{
		return fMerk;
	}

	public abstract int CO2_Uitstoot();
}

public class Auto extends Voertuig{
	protected static final int fUitstoot = 130;

	public Auto(String merk)
	{
		super(merk);
	}
}

public class HybrideAuto extends Auto{
	private static final double fVerbruik = 3.0;

	public HybrideAuto(String merk)
	{
		super(merk);
	}

	public String getMerk()
	{
		return "Toyota Prius";
	}

	public int CO2_Uitstoot()
	{
		return fUitstoot;
	}

	public double getVerbruik()
	{
		return fVerbruik;
	}
}



public class Fiets extends Voertuig{
	private String fMerk;
	private static final int fUitstoot = 0;
```

```
	public Fiets(String merk)
	{
		super(merk);
	}

	public int CO2_Uitstoot()
	{
		return fUitstoot;
	}

	public String getMerk()
	{
		return fMerk;
	}
}

public class ElektrischeFiets extends Fiets{
	private static final int fUitstoot = 10;

	public ElektrischeFiets(String merk)
	{
		super(merk);
	}

	public int CO2_Uitstoot() {
		return fUitstoot;
	}
}
```

Bijlage C

```
public class MessengerServer
{
	private int fPort;

	public static void main(String args[])
	{
		MessengerServer server = new MessengerServer(827);
		server.listen();
	}

	public MessengerServer(int port)
	{
		fPort = port;
	}

	public void listen()
	{
		try {
			ServerSocket listenTo = new ServerSocket(fPort);
			while(true)
			{
				Socket clientsocket = listenTo.accept();
				ClientWorker worker = new
				ClientWorker(clientsocket);
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

```
public class ClientWorker extends JFrame {

	private Socket fSocket;
	private JButton button = new JButton("Send PING over network");
	private PrintWriter out;

	public ClientWorker(Socket socket) {
		fSocket = socket;
	}

	public void run() {
		try {
			out = new PrintWriter(fSocket.getOutputStream());
		} catch (IOException e) {
			e.printStackTrace();
		}

		this.setSize(250, 250);
		this.setLocation(300,200);
		this.getContentPane().add(BorderLayout.CENTER, button);
		this.setVisible(true);
		out.write("PING");
		out.flush();

	}


}
```