Tentamen Objectgeorienteerd Programmeren IN1205
Voorbeeld

Afdeling ST Faculteit EWI TU Delft

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

puntenverdeling

1	2	3	4	5	6	7	8
15	30	10 + 10	5 * 3	15	12	12	16

Opgave 1.

Gegeven is de klasse Document uit bijlage A.
De volgende declaraties en opdrachten worden uitgevoerd:

```
Document d1 = new Document("inhoud1","titel1","versie1");
Document d2 = new Document("inhoud2","titel1","versie2");
Document d3 = new Document("inhoud3","titel2","versie2");
Document d4 = d2;
d4.setInhoud("inhoud3");
d2 = null;
```

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.    d1.getInhoud().length == 4
b.    d2.equals(d4)
c.    d4.equals(d3)
d.    d4.getTitel() == d1.getTitel()
e.    d1.getVersie() <= d3.getVersie()
```

Opgave 2

Gegeven is de klasse Ordner, waarin een aantal documenten kan worden opgeborgen.

```
public class Ordner{

   private Document[] document;
   private int aantal;

   public Ordner(int n){
      document = new Document[n];
      aantal = 0;
   }
```

```
   public void voegToe(Document d){
      if (aantal < document.length){
         document[aantal] = d;
         aantal = aantal + 1;
      }
   }
}
```

Aan deze klasse moet een tweetal methoden worden toegevoegd.

a. Geef de implementatie van methode langsteDocument() die het document met de grootste inhoud geeft. Als er meer dan een document met de grootste inhoud is, dan wordt van deze documenten het laatste document in de ordner gegeven.

```
langsteDocument(): Document
//pre:  ordner is niet leeg
//post: retourneert document met de grootste inhoud, indien er
//      meerdere documenten zijn die hieraan voldoen, wordt het
//      laatst gevonden document dat voldoet geretourneerd
```

b. Geef een implementatie van methode overzicht() die een String retourneert, bestaande uit een aantal regels. Elke regel bevat de titel en versie van een document.

```
overzicht(): String
//post: retourneert een String bestaande uit een aantal regels, elke
//      regel bevat de titel en versie van een document
```

Voorbeeld: als de ordner 2 documenten bevat met de titels: `titel1` en `titel2` en versies `versie1` en `versie2` is de waarde van overzicht:

```
"titel1,versie1,\n,titel2,versie2"
```

Aanwijzing: om ervoor te zorgen dat de implementatie efficiënt met het geheugen omgaat, **moet** bij de implementatie van deze methode gebruik te worden gemaakt van klasse StringBuffer.

Opgave 3

a. Geef een *volledig* klassendiagram 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

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();
```

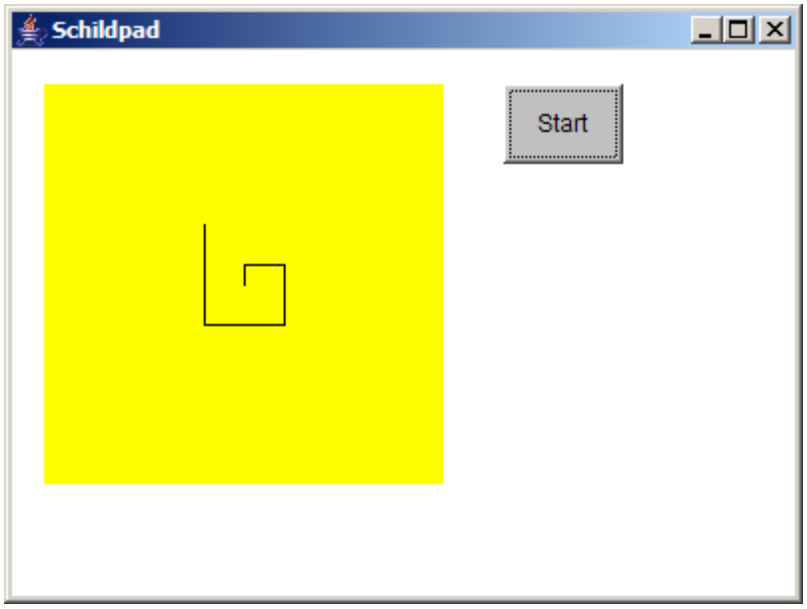
Het programma Schildpad is een interpretator van een zeer eenvoudige taal. Hieronder ziet u een stukje programma:

```
N 10
O 20
Z 30
W 40
N 50
```

Elke regel bestaat uit een van de hoofdletters 'N', 'O', 'Z', 'W' gevolgd een of meer spaties, gevolgd door een getal. Een programma in deze taal bestaat uit een aantal van deze regels. Interpretatie van een programma betekent: onder invloed van de instructies een punt in een plat vlak laten bewegen. De betekenis van elk van de instructies is als volgt:

`N` *num*: doe een stap ter grootte *num* naar het noorden,
`O` *num*: doe een stap ter grootte *num* naar het oosten,
`Z` *num*: doe een stap ter grootte *num* naar het zuiden,
`W` *num*: doe een stap ter grootte *num* naar het westen,

Hieronder ziet u het resultaat van uitvoering van bovenstaand programma.



In bijlage C vindt u een drietal klassen die deze interpretatie realiseren.

- De klasse **Schildpad** bevat een attribuut 'hier' van type Point, dit punt beweegt onder invloed van het programma.
- De gebruikersinterface is gerealiseerd de klasse **SchildpadVenster**.
- In de klasse **TekenCanvas** worden stappen van de schildpad zichtbaar gemaakt. Daartoe onthoudt hetTekenCanvas de laatste positie van de schildpad. Om een stap zichtbaar te maken wordt in het TekenCanvas een lijn van de laatste positie naar de nieuwe positie getrokken, en wordt de laatste positie aangepast (methode gaNaar) .

In opgave 5, 6 en 7 moet u het gegeven programma uitbreiden.

1. Vang waar nodig Exceptions af
2. Importeer waar nodig andere packages
3. Zorg dat steeds duidelijk is **wat** u **waar** wijzigt.

Opgave 5

Implementeer in de klasse Schildpad de methode execute:

```
public void execute()
```

Deze methode opent de tekstfile waarvan de naam in INFILE staat, leest daaruit stuk voor stuk de regels, en laat die uitvoeren (methode stap). U mag hier **niet** de klasse SimpleInput van Barnes gebruiken, wel (zonodig) de klasse StringTokenizer.

NB Als u de methode readLine van de klasse BufferedReader wilt gebruiken, is het handig te weten dat die methode **`null`** teruggeeft als het einde van de stream bereikt is.

Opgave 6

Als op de startknop van het SchildpadFrame wordt gedrukt moet de methode execute van Schildpad (die u zo-even heeft geschreven) worden uitgevoerd. Zorg dat dit gerealiseerd wordt, door van de Schildpad een ActionListener te maken. Geef nauwkeurig aan waar u welke aanpassingen maakt.

Opgave 7

Gebruik Observer en Observable om de veranderingen in de schildpad op het tekenCanvas zichtbaar te maken. Geef nauwkeurig aan waar u welke aanpassingen maakt.

Opgave 8

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 Document{
   private String inhoud;
   private String titel;
   private String versie;

   public Document(String in, String ti, String ver){
      inhoud = in;
      titel = ti;
      versie = ver;
   }

   public String getInhoud(){
      return inhoud;
   }

   public String getTitel(){
      return titel;
   }

   public String getVersie(){
      return versie;
   }

   public void setInhoud(String newInhoud){
      inhoud = newInhoud;
   }

   public boolean equals(Object other){
      if (other instanceof Document){
          Document otherDoc = (Document) other;
          return this.inhoud.equals(otherDoc.inhoud) &&
                 this.versie.equals(otherDoc.versie);
      }
      return false;
   }
}
```

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 SchildpadVenster extends Frame{

    private Button startknop;
    private TekenCanvas tekenCanvas;
    private Schildpad schildpad;

    public SchildpadVenster(){
       ... // create graphics
      startknop = new Button();
      add(startknop);

      tekenCanvas = new TekenCanvas(new Point(100,100));
      add(tekenCanvas) ;

      schildpad = new Schildpad();

    }

    public static void main(String [] args) {
       SchildpadVenster sv = new SchildpadVenster();
       sv.show();
    }

}
```

```
import java.awt.*;
public class Schildpad {
    private final static String INFILE = "program.txt" ;
    private Point hier = new Point(100,100);

    public void execute(){
    }

    private void stap(char c, int step){
        switch (c){
            case 'N': hier.y = hier.y - step; break;
            case 'O': hier.x = hier.x + step; break;
            case 'Z': hier.y = hier.y + step; break;
            case 'W': hier.x = hier.x - step; break;
        }
    }

}
```

```
import java.awt.*;
public class TekenCanvas extends Canvas{
    private Point hier;

    public TekenCanvas(Point startpunt){
        hier = startpunt;
        setBackground(Color.yellow);
        setSize(200, 200);
    }

    private void gaNaar(Point daar){
        Graphics g = getGraphics();
        g.drawLine(hier.x, hier.y, daar.x, daar.y);
        hier.x  = daar.x;
        hier.y  = daar.y;
    }
}
```