**TECHNISCHE UNIVERSITEIT DELFT**
**Faculteit Elektrotechniek, Wiskunde en Informatica**

TU Delft

# IN1205 Objectgeorienteerd Programmeren

28 januari 2009 9.00-12.00

Bij dit tentamen mag u gebruik maken van: Java in Two Semesters (Charatan & Kans)
en de Inleiding Algoritmiek.

## Opgave 1 (5 * 3 punten)

Bijlage 1 bevat de implementatie van klasse Document. Gebruik deze bijlage bij het beantwoorden van de volgende vragen:

Gegeven zijn de volgende declaraties en opdrachten:

```
String s1 = "titel";
String s2 = "1";

Document d1 = new Document("inhoud1", s1 + s2, 1);
Document d2 = new Document("inhoud2", "titel1", 2);
Document d3 = new Document("inhoud3", "titel2", 1);
Document d4 = d2;
d4.setInhoud("inhoud3");
d2 = null;
```

Hieronder vind je een aantal expressies: sommige expressies kunnen tot een compileer- of runtime fout leiden. Als dat zo is, geef dan de reden van de fout.
Als dat niet zo is, geef dan de waarde van de expressie. Licht je antwoord toe.

```
a.   d1.getInhoud().length == 7
b.   d2.equals(d4)
c.   d4.equals(d3)
d.   d4.getTitel() == d1.getTitel()
e.   d1.getVersie() == d3.getVersie()
```

**Opgave 2 (5+5+5 punten)**

Gegeven klasse Documenten die een lijst van documenten (zie bijlage 1) bevat. Een deel van de implementatie van de klasse is hieronder gegeven:

```
public class Documenten{

   private Document[] documenten;
   private int aantal;

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

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

Aan deze klasse moeten 3 methoden worden toegevoegd.

Methode geefTitels maakt een ArrayList van de titels van de documenten die zijn opgeslagen. Alle titels in deze ArrayList zijn uniek, een titel mag maximaal een keer voorkomen.

Methode voegSamen voegt de inhoud van twee documenten samen tot een nieuw document.

Methode aantalVersies bepaalt hoeveel verschillende versies er in de lijst voorkomen van een document met een opgegeven titel.

Geef een implementatie van de 3 methoden. Indien nodig mag je gebruik maken van een of meer handige hulpmethoden, die je zelf moet implementeren.

a.

```
ArrayList<String> geefTitels()
post: retourneert een ArrayList met de titels van de
documenten in de lijst. De titels zijn uniek, d.w.z. dat ze
maar 1 keer in de ArrayList voorkomen
```

b.

```
Document voegSamen(String nieuweTitel, int pos1, int pos2)
pre: 0 <= pos1, pos2 < aantal
post: retourneert (indien pre) een nieuw document met titel =
nieuweTitel, inhoud = de samenvoeging van de inhoud van de
documenten op pos1 en pos2 en versie = 1,
retourneert anders null
```

c.

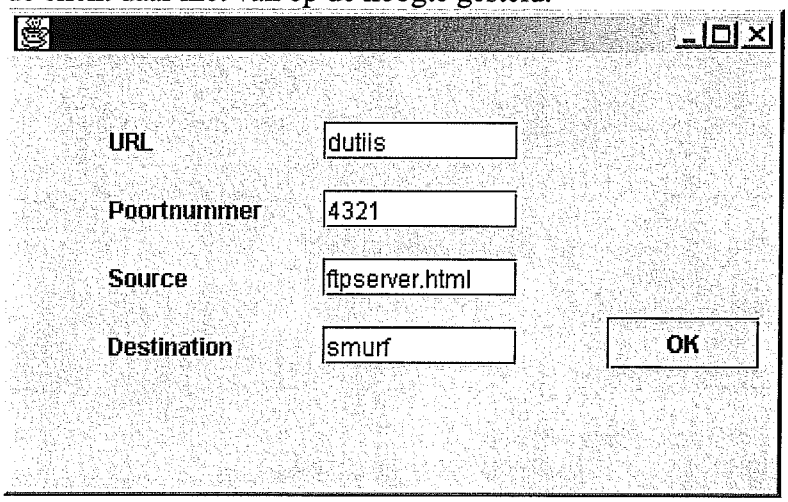
```
int aantalVersies(String titl)
post: retourneert het aantal versies in de lijst van het
document met titel = titl
```

**Opgave 3 (8+7 punten)**

a. Geef een sequencediagram van de methode voegSamen, die u in opgave 2b heeft geschreven

b. Geef een volledig klassediagram van de klassen uit bijlage 2

**Opgave 4 (15 punten)**
Hieronder is de interface gegeven van een programma om files van een andere computer op te halen. De naam van de computer en het poortnummer van de server moeten worden opgegeven, alsmede de naam van de file die moet worden opgehaald, en de naam waaronder die file moet worden opgeslagen. Wordt in onderstaand plaatje op de OK-knop gedrukt, dan wordt het bestand "ftpserver.html" op de machine "dutiis" opgehaald (server heeft poortadres 4321) en lokaal weggeschreven onder de naam "smurf".
Het programma is weinig gebruikersvriendelijk: als de gevraagde file niet bestaat wordt de client daar niet van op de hoogte gesteld.




In bijlage 3 vindt u de implementatie van de serverklasse. In de methode listen van de server wordt de methode read uit de klasse InputStream gebruikt. Deze is als volgt gespecificeerd:

`public int read() throws IOException`

> Reads the next byte of data from the input stream. The value byte is returned as an `int` in the range `0` to `255`. If no byte is available because the end of the stream has been reached, the value `-1` is returned.
>
> **Returns:**
> the next byte of data, or `-1` if the end of the stream is reached.
> **Throws:**
> `IOException` - if an I/O error occurs

Bijlage 3 bevat ook een begin van de implementatie van de client (FtpFrame). U mag aannemen dat alle grafische componenten gedeclareerd zijn, en in de constructor zijn geïnitialiseerd. In deze implementatie heeft het drukken op de okKnop geen enkel effect. Pas de klasse FtpFrame zodanig aan dat het drukken op de okKnop het bovenbeschreven effect heeft:

- maak van de klasse FtpFrame een ActionListener,
- zorg dat deze luistert naar het klikken op de okKnop,
- geef aan wat u waar wijzigt,
- geef zonodig aan wat u importeert.

**Opgave 5 (5* 3 punten)**
Onderstaande vijf programmafragmenten zijn gebaseerd op de definities uit bijlage 2.
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.
```
Numbers numbers = new OddNumbers();
int x = numbers.nextInt();
x = numbers.nextInt();
```

b.
```
Numbers numbers = new OddNumbers();
OddNumbers oddNumbers = numbers;
int x = numbers.nextInt();
x = numbers.nextInt();
```

c.
```
Numbers numbers = new BoundedNumbers(1);
int x = numbers.nextInt();
if (!numbers.hasMoreInts())
   numbers.reset();
x = numbers.nextInt();
```

d.
```
Numbers numbers = new OddNumbers();
OddNumbers oddNumbers = (OddNumbers) numbers;
int x = numbers.nextInt();
x = numbers.nextInt();
```

e.
```
Numbers numbers = new IntEnum();
int x = numbers.nextInt();
```

**Opgave 6 (15 punten)**
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.
Zorg dat uw algoritme niet onnodig inefficient is.
Gegeven is 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).
Onder een *niet-dalende* rij verstaan we een rij waarvoor geldt: $x(i) \leq x(i+1)$ voor alle relevante i.
Implementeer in Getalrij de methode
`public int lengteLangsteNDD()`
post: retourneert de lengte van de langste niet-dalende deelrij in rij.

## Bijlage 1

```
public class Document{

   private String inhoud;
   private String titel;
   private int versie;

   public Document(String inh, String tit, int ver){
      inhoud = inh;
      titel  = tit;
      versie = ver;
   }

   public String getInhoud(){
      return inhoud;
   }

   public String getTitel(){
      return titel;
   }

   public int getVersie(){
      return versie;
   }

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

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

## Bijlage 2

```
abstract class IntEnum{
   int next;
   boolean hasMoreInts(){
      return true;
   }
   abstract int nextInt();
}

class Numbers extends IntEnum{
   public Numbers(){
      next = 1;
   }

   public int nextInt(){
      return next++;
   }
}

class OddNumbers extends Numbers{
   public int nextInt(){
      int answer = next;
      next += 2;
      return answer;
   }
}

class BoundedNumbers extends Numbers{
   private int last;

   public BoundedNumbers(int last){
      this.last = last;
   }

   public boolean hasMoreInts(){
      return next <= last;
   }
   public void reset(){
      next = 1;
   }
}
```

## Bijlage 3

```
import java.io.*;
import java.net.*;
/**
 * Title:        FTP-project
 * Description:  Een simpel FTP programmaatje voor een toets.
 */

public class Server {

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

  public void listen(){
    ServerSocket serversocket = null;
    try{
       serversocket = new ServerSocket(4321);
       while(true){
          Socket client = serversocket.accept();
          BufferedReader in = new BufferedReader(
                                  new InputStreamReader(
                                    client.getInputStream()));
          OutputStream out  = client.getOutputStream();
          String sourceName  = in.readLine();
          InputStream source = new FileInputStream(sourceName);
          int b = source.read();
          while(b > 0) {
             out.write(b);
             int b = source.read();
          }
          source.close();
          client.close();
       }
    }
    catch (IOException e) {
        System.out.println(e.toString());
    }
}
```

```
import  java.awt.*;
import  java.awt.event.*;

/**
 * Title:        FTP-project
 * Description:  Een simpel FTP programmaatje voor een toets.
 */

public class FtpFrame extends Frame{ implements ActionListener

   TextField urlTextField;
   TextField poortTextField;
   TextField sourceTextField;
   TextField destinationTextField;
   Button okKnop;
   Label uRLLabel;
   ... // overige labels

   public FtpFrame() {
      setSize(400,250);
      setLayout(null);
      okKnop.addActionListener(this)
      ... // initialisatie van grafische componenten
   }

   public static void main(String[] args) {
      FtpFrame ftpFrame = new FtpFrame();
      ftpFrame.setVisible(true);
   }

}
```