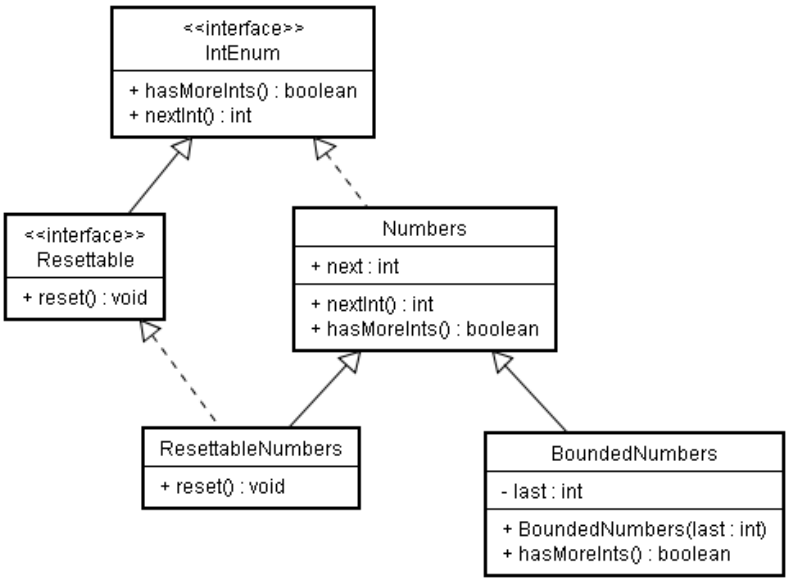
Voorbeeldtentamen Inleiding programmeren 2 (IN1608WI),
Technische Universiteit Delft, Faculteit EWI, Afdeling 2.

Gesloten boek tentamen: bij het tentamen mag **geen** gebruik worden gemaakt van het studieboek, college slides, practicumopdrachten of ander materiaal.

Opgave 1

In bijlage A is een aantal interfaces en klassen gedefinieerd. Bestudeer de definities goed voordat je begint aan het beantwoorden van de vraag.

Geef een volledig klassendiagram van de klassen en interfaces.



Opgave 2

De volgende code fragmenten verwijzen naar de definities van Bijlage A.

Geef voor elk codefragment aan:
- of er tijdens het compileren een compileerfout zal optreden, en zo ja waarom,
- zo niet, of er tijdens de uitvoering een runtime error zal optreden, en zo ja waarom,
- zo niet, wat de waarde van x is na de uitvoering.

a.
```
Numbers numbers = new BoundedNumbers(1);
//BoundedNumbers: last = 1
int x = numbers.nextInt();
//Numbers: retourneert next (= 1), next wordt 2
if(numbers.hasMoreInts())
    //let op de hasMoreInts() van BoundedNumbers wordt
    //gebruikt: is next <= last => false
    x = numbers.nextInt();
else
    //deze tak wordt gekozen
    x = 0;
System.out.println(x);
// er wordt een 0 afgedrukt
```

```
b.//Compileerfout: numbers is van klasse Numbers, deze
   //klasse heeft geen reset methode
  Numbers numbers = new ResettableNumbers();
  int x = numbers.nextInt();
  numbers.reset();
  x = numbers.nextInt();
  System.out.println(x);

c.Numbers numbers = new ResettableNumbers();
  //next = 1
  int x = numbers.nextInt();
  //retourneert next (= 1), next wordt 2
 ((ResettableNumbers)numbers).reset();
  //(ResettableNumbers) numbers heeft type
 //ResettableNumbers, deze heeft wel een reset methode
  //next = 1
  x = numbers.nextInt();
  //retourneert next (= 1), next wordt 2
  System.out.println(x);
   //er wordt een 1 afgedrukt

d.Numbers numbers = new BoundedNumbers(1);
  //BoundedNumbers: last  = 1
  int x = numbers.nextInt();
  //retourneert next (= 1), next wordt 2
  x = numbers.nextInt();
  //retourneert next (= 2), next wordt 3
  System.out.println(x);
  //er wordt een 2 afgedrukt

e.IntEnum numbers = new Numbers();
  //dit kan, je kunt een object binden aan een variable van
 //een Interfacetype,
  //Numbers: next = 1
  int x = numbers.nextInt();
  //retourneert next (= 1), next wordt 2
  ResettableNumbers rn = (ResettableNumbers)numbers;
  //Runtime fout: ClassCastException: tijdens de uitvoering
 //ontdekt het runtime systeem dat het object van het type
 //Numbers is en niet van het type Resettable numbers
  x = rn.nextInt();
```

Opgave 3

In bijlage B zijn de klassen Boodschap en Postbus gegeven. Deze klassen zijn onderdeel van een eenvoudig e-mail systeem.

Elke gebruiker van het e-mail systeem heeft een Postbus die een lijst (lijstIn) bevat met ingaande boodschappen en een lijst (lijstUit) met uitgaande boodschappen. De lijsten kunnen een groot aantal boodschappen bevatten.

De Postbus worden regelmatig aangepast door een mailserver, die is verbonden met internet. De server ontvangt nieuwe boodschappen en voegt deze toe aan lijstIn. Ook verzendt de server de boodschappen uit lijstUit en verwijdert ze daarna uit de lijst.

Aan klasse Postbus moeten enkele methoden worden toegevoegd voor het opslaan van gegevens in een text-bestand en opvragen van gegevens uit een text-bestand. De data worden als volgt opgeslagen in het bestand. Het commentaar aan de rechterkant hoeft niet in het bestand te worden opgeslagen!

```
1                           //aantal boodschappen in Lijst In
Jan                         //afzender 1e boodschap
Els                         //ontvanger 1e boodschap
Vanavond naar de film?      //inhoud 1e boodschap
                            //lege regel
1                           //aantal boodschappen in Lijst Uit
Els                         //afzender 1e boodschap
Jan                         //ontvanger 1e boodschap
Afgesproken, tot straks.    //inhoud 1e boodschap
```

+schrijf(bNaam : String)
post: heeft de inhoud van het Postbus-object naar bestand met de naam bNaam geschreven in het formaat zoals hierboven is aangegeven

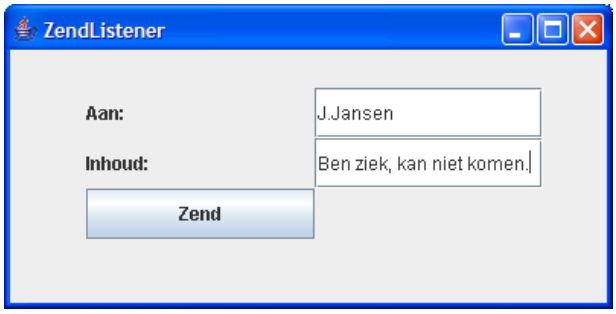
Gebruik een FileWriter voor de verbinding met het bestand, en een PrintWriter om de attributen van een Boodschap-object weg te schrijven. Het geheel moet binnen een try-catch blok. Schrijf dan eerst het aantal Boodschappen, en voor elke Boodschap de attributen weg. Doe dit voor beide lijsten.

+lees(bNaam : String) : Postbus
pre: het bestand met de naam bNaam bevat de data van een Postbus-object in het formaat dat hierboven is aangegeven
post: retourneert een Postbus-object met de gegevens uit het bestand

Gebruik een FileReader voor de verbinding met het bestand, en een Scanners om de elementen van de regels te lezen. Lees eerst het aantal regels, gebruik dan de scanner om de afzonderlijke Strings te lezen (next(), next() en nextLine()). Maak een nieuw Boodschap object en voeg dat aan de lijst toe. Doe dit voor beide lijsten.

Opgave 4

Voor gebruikers van een Postbus is een eenvoudige Grafische User Interface ontwikkeld. De code van de interface kun je vinden in bijlage C. Een afbeelding van de GUI is hiernaast te zien.



Met de GUI kan de ontvanger en de inhoud van een boodschap worden ingevoerd. Na het aanklikken van de Zend knop wordt er een nieuw Boodschap object gemaakt, en wordt dit Boodschap-object aan lijstUit toegevoegd.

a) Als de methode voegToeUit in Postbus wordt aangeroepen terwijl lijstUit vol is, gebeurt er helemaal niets. Pas deze methode aan, zodat hij in dat geval een Exception werpt.

Voeg aan de methode voegToeUit() een else tak toe waarin een RuntimeException wordt "geworpen".

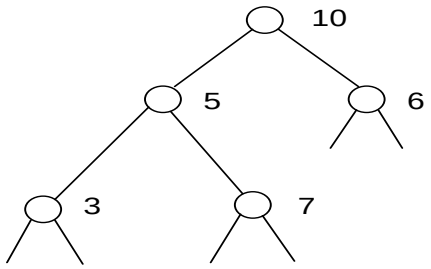
b) Pas de methode actionPerformed uit bijlage C aan, zodat hij passend reageert als de methode voegToeUit een Exception werpt. In dat geval moeten de tekstvelden **niet** worden leeg-gemaakt, en moet er in het messageLabel een foutmelding worden getoond!

Lees eerst de velden en maakt een nieuw Boodschap object aan. Voeg het object toe aan LijstUit. Wordt er geen exception geworpen dan kun je de tekstvelden leegmaken. Wordt er wel een exception geworpen dan moet je in het boodschapLabel een tekst zetten.

Opgave 5

Bijlage D bevat de implementatie van klasse BinNode. Met de knopen van deze klasse kan een binaire boom worden gevormd.

Hieronder is zo'n binaire boom getekend, met 5 knopen die gehele getallen bevatten. De boom telt 3 niveau's. De wortel van de boom bevindt zich op niveau 0. Het volgende niveau bevat twee knopen, de rechterknoop is een blad, dit is een knoop met twee lege subbomen. Op niveau 2 zijn er twee knopen, beide bladeren.



Gevraagd wordt om de implementatie van een aantal statische methoden:

a. **`public static boolean isBlad(BinNode b)`**
`post: retourneert true dsd b een blad is`

b. **`public static int bladerenOpNiveau(BinNode b, int niv)`**
`post: retourneert het aantal bladeren op niveau niv`

```
public static boolean isBlad(BinNode b){
      if (b == null)
            return false;
      else
            return (b.getLinks()==null) && (b.getRechts()==null);
}


public static int bladerenPerNiveau(BinNode b, int niveau){

      if (b == null)
      //lege boom
            return 0;
      else if (niveau == 0)
      //je bent op het gewenste niveau aangekomen
            if (isBlad(b)
                  return 1;
            else
                  return 0;
      //je moet nog dieper in de boom
      else return bladerenPerNiveau(b.getLinks(),niveau-1) +
                  bladerenPerNiveau(b.getRechts(),niveau-1);

}
```

Bijlage A

```
interface IntEnum{
   boolean hasMoreInts();
   int nextInt();
}

interface Resettable extends IntEnum{
   public void reset();
}

public class Numbers implements IntEnum{
   int next = 1;

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

   public boolean hasMoreInts(){
      return true;
   }
}

public class ResettableNumbers extends Numbers
       implements Resettable{
  public void reset(){
     next = 1;
  }
}
```

```
public class BoundedNumbers extends Numbers{
   private int last;

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

   public boolean hasMoreInts(){
      return next <= last;
   }

}
```

Bijlage B

```
public class Boodschap{
      private String afzender;
      private String ontvanger;
      private String inhoud;

      public Boodschap(String afz, String ont, String inh){
            afzender   = afz;
            ontvanger  = ont;
            inhoud     = inh;
      }
     public String getAfzender(){ return afzender;}
     public String getOntvanger(){     return ontvanger;     }
     public String getInhoud(){        return inhoud;        }
}

public class Postbus{
      public final static int capaciteit = 10;
      private Boodschap[] lijstIn  = new Boodschap[capaciteit];
      private Boodschap[] lijstUit = new Boodschap[capaciteit];

      private int aantalIn  = 0;
      private int aantalUit = 0;

      public void voegToeIn(Boodschap b){
            if (aantalIn < capaciteit){
                  lijstIn[aantalIn] = b;
                  aantalIn = aantalIn + 1;
            }
      }

      public void voegToeUit(Boodschap b){
            if (aantalUit < capaciteit){
                  lijstUit[aantalUit] = b;
                  aantalUit = aantalUit + 1;
            }
      }
}
```

Bijlage C

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

public class ZendListener extends JFrame implements ActionListener{
      private JLabel ontvangerLabel = new JLabel("Aan:");
      private JLabel inhoudLabel    = new JLabel("Inhoud:");
      private JLabel boodschapLabel = new JLabel("");

      private JTextField ontvangerVeld    = new JTextField();
      private JTextField inhoudVeld       = new JTextField();
      private JButton verzendKnop         = new JButton("Zend");
      private JPanel panel          = new JPanel(new GridLayout(3,2));

      private final static String afzender = "student";
      private Postbus postbus              = new Postbus();

      public ZendListener(){

            setTitle("ZendListener");
            setBounds(100,100,400,200);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            Container c = getContentPane();
            setLayout(null);

            panel.add(ontvangerLabel);
            panel.add(ontvangerVeld);
            panel.add(inhoudLabel);
            panel.add(inhoudVeld);
            panel.add(verzendKnop);
            panel.add(boodschapLabel);
            panel.setBounds(50,25,300,100);
            c.add(panel);
            verzendKnop.addActionListener(this);
            setVisible(true);
      }

      public void actionPerformed(ActionEvent e){

            String ont = ontvangerVeld.getText();
            ontvangerVeld.setText("");
            String inh = inhoudVeld.getText();
            inhoudVeld.setText("");

            Boodschap b = new Boodschap (afzender,ont,inh);
            postbus.voegToeUit(b);
            boodschapLabel.setText("boodschap verzonden");
      }
}
```

Bijlage D

```
public class BinNode{

      private int waarde;
      private BinNode links, rechts;

      public BinNode(int w, BinNode l, BinNode r){
            setWaarde(w);
            setLinks(l);
            setRechts(r);
      }

      public void setWaarde(int w)     {     waarde =w; }
      public void setLinks(BinNode l)  {     links = l; }
      public void setRechts(BinNode r){      rechts = r;      }

      public int getWaarde()           {     return waarde;   }
      public BinNode getLinks()        {     return links;    }
      public BinNode getRechts()       {     return rechts;   }
}
```