Institutionen för
Datavetenskap
CTH, GU

VT07
TDA550, DIT720
07-12-19

Lösningsförslag till tentamen i Objektorienterad programvarutveckling IT, fk.

DAG : 19 december 2007

Uppg 1:

a) Nej, gränssnit implementerar ingenting !

b) Nej, generiska typer är bara subklasser till '?', dvs `ArrayList<A>` och `ArrayList<B>` är båda subklasser till `ArrayList<?>`

c) Ja, med samma parametertyp gäller de vanliga reglerna.

d) Nej, `private` betyder enbart åtkomliga i klassen

e) Nej, `new Integer(5) == 5` blir `true`,
eftersom automatisk 'unboxing' sker,
medan `new Integer(5) == new Integer(5)` ger `false`,
eftersom det är två olika instanser.

Uppg 2:

```
import java.io.*;
import java.util.*;

public class FindLines {

    private static void errExit(String errMassage) {
        System.err.println( errMassage );
        System.err.println(
          "Usage: java FindLines <digit> <textfile>");
        System.exit(0);
    } // printError

    public static void main( String[] args ) {
        try {
            char n = '0';
            if ( args[0].length() == 1 &&
                 Character.isDigit(args[0].charAt(0)) )
               n =  args[0].charAt(0);
            else
               errExit( " First argument not a digit." );
    Scanner inFile = new Scanner(new File(args[1]));
            int noRows = 0;
            while( inFile.hasNextLine() )
               if ( inFile.nextLine().indexOf(n) > -1 )
                  noRows++;
            System.out.println( "Siffran " + n + " förekom på "
                                + noRows + " rader." );
            inFile.close();
        }
        catch (ArrayIndexOutOfBoundsException aioob) {
            errExit( "Missing agument(s) " );
        }
        catch (FileNotFoundException fnfe) {
            errExit( "File " +  args[1] + " not found " );
        }
    } // main
} // class FindLines
```

Uppg 3:

```
import java.util.*;

public class MyQueue<E> implements Queue<E>{

   private LinkedList<E> que = new LinkedList<E>();

    public void enqueue( E elem ) {
        que.addLast(elem);
    } // enqueue

    public E dequeue()
             throws NoSuchElementException {
        if ( que.size() > 0 )
           return que.removeFirst();
        else
           throw new NoSuchElementException(
                        "Empty queue in dequeue");
    } // dequeue

    public E front()
             throws NoSuchElementException {
        if ( que.size() > 0 )
           return que.getFirst();
        else
           throw new NoSuchElementException(
                          "Empty queue in front");
    } // front

    public int size() {
       return que.size();
    }

} // class MyQueue
```

Uppg 4:

```
import java.util.*;

public class PersonByAge
       extends Person
       implements Comparable<PersonByAge> {

    public PersonByAge( String idNumber,
                        String name       ) {
        super( idNumber, name );
    } // constructor PersonByAge

    public int compareTo( PersonByAge pba ) {
        String today     = "071219",
               thisDate  = idNumber.substring(0,6),
               pbaDate   = pba.idNumber.substring(0,6);
        boolean thisLastCentury = thisDate.compareTo(today) > 0,
                pbaLastCentury  = pbaDate.compareTo(today)  > 0;
        if ( thisLastCentury == pbaLastCentury )
            return pbaDate.compareTo( thisDate );
        else if (thisLastCentury)
            return 1;
        else
            return -1;
    } // compareTo

    public static String toString( Set<Person> sp ) {
        SortedSet<PersonByAge> ssp = new TreeSet<PersonByAge>();
        String res = "";
        for ( Person p : sp )
            ssp.add( new PersonByAge( p.getId(), p.getName() ));
        for (PersonByAge pba : ssp )
            res = res + pba.getId() + " " + pba.getName() + "\n";
        return res;
    } // toString
```

Och ett litet testprogram som inte ingick i uppgiften:

```
    public static void main(String[] args) {
        HashSet<Person> hsp = new HashSet<Person>();
        hsp.add(new Person( "460702-xxxx","Kalle" ));
        hsp.add(new Person( "560702-xxxx","Kajsa" ));
        hsp.add(new Person( "040702-xxxx","Knatte" ));
        hsp.add(new Person( "050702-xxxx","Fnatte" ));
        hsp.add(new Person( "050703-xxxx","Tjatte" ));

        System.out.println( toString(hsp));

    } // main

} // class PersonByAge
```

Vilket ger vid exekvering:

```
> javac PersonByAge.java
> java PersonByAge
050703-xxxx Tjatte
050702-xxxx Fnatte
040702-xxxx Knatte
560702-xxxx Kajsa
460702-xxxx Kalle
```

Uppg 5: **a)**

```
public class IncAndDecrOneMillion {

    public static void main( String[] args ) {

        MyThread mt1 = new MyThread(  1, 1000000 );
        MyThread mt2 = new MyThread( -1, 1000000 );
        mt1.start();
        mt2.start();
        try {
            mt1.join();
            mt2.join();
        }
        catch( InterruptedException ie) { }

        System.out.println( MyThread.shared );

    } // main
} // class IncAndDecrOneMillion
```

b) Det beror på att koden kan bli 'saxad' (interleaved)

c) De enda ändringarna som behövs är merkerade med `/**/`

```
public class  MyThread extends Thread {

    private int whatToAdd;
    private int howMany;

    public static int shared = 0;
    public static Object lockObj
                        = new Object();      /**/

    public MyThread( int toAdd, int times ) {
       whatToAdd = toAdd;
       howMany   = times;
    }  // constructor MyThread

    public void run() {
        int temp = 0;
        for ( int i = 0; i < howMany; i++ ) {
           synchronized(lockObj) {           /**/
               temp   = shared + whatToAdd;
               shared = temp;
           }                                 /**/
        }
    } // run

} // class MyThread
```

d)

```
    public static void main( String[] args ) {

        int iterMt1 = Integer.parseInt( args[0] );
        int iterMt2 = Integer.parseInt( args[1] );

        MyThread mt1 = new MyThread(  1, iterMt1 );
        MyThread mt2 = new MyThread( -1, iterMt2 );
```

Resten som tidigare.