Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 10 dec 2010

Examination in

PROGRAMMERINGSTEKNIK F1 TIN211

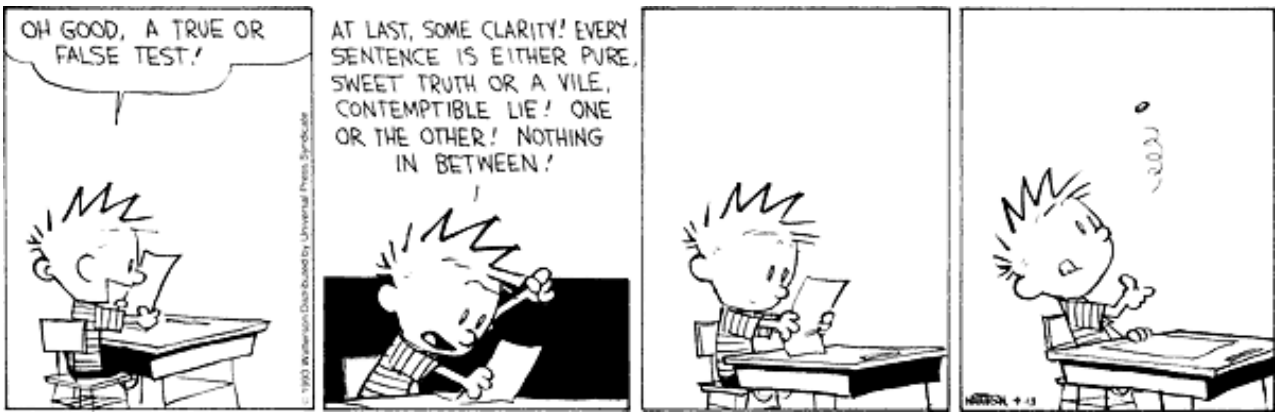
DAY: SATURDAY DATE: 2010-12-11 TIME: 14.00-19.00 ROOM: V

Responsible teacher:	Erland Holmström tel. 1007, home 0708-710600
Results:	Are sent by mail from Ladok.
Solutions:	Are eventually posted on homepage.
Inspection of grading:	The exam can be found in our study expedition after posting of results.
	Time for complaints about grading are announced on homepage after the result are published or mail me and we find a time.
Grade limits:	CTH: 3=26p, 4=36p, 5= 46p, max 60p
Aids on the exam:	**Bravaco, Simonson Java Programming From the Grounf Up**
	or Horstmann Java Concepts and A print of Chapter 18.

Observe:

- Start by reading through all questions so you can ask questions when I come. I usually will come after appr. 2 hours.
- All answers must be motivated when applicable.
- Write legible! Draw figures. Solutions that are difficult to read are not evaluated!
- Answer concisely and to the point.
- The advice and directions given during course must be followed.
- Programs should be written in Java, indent properly, use comments and so on.
- Start every new problem on a new sheet of paper.

Good Luck!



Problem 1. Vilka av följande påståenden är korrekta? Korta motiveringar.

a) Uttrycket p1 == p2 har värdet true om objekten som referensvariablerna p1 och p2 refererar till har samma värden på alla sina attribut.

b) När specifikationen av en klass är given kan man implementera klassen på flera olika sätt.

c) En klass som beskriver en kvadrat måste alltid ha attributen x och y (koordinaterna för kvadratens läge) och side (kvadratens sidlängd).

d) Lokala variabler i olika metoder i samma klass kan ha samma namn.

e) Om man i en metod deklarerar en lokal int-variabel utan att ge den ett startvärde får variabeln automatiskt värdet 0.

och slutligen

f) Beskriv kort idén med en hashtabell. En figur behövs.

(7p)

Problem 2. Övar enkel rekursion. I klassen Integer finns metoden

```
static String toString(int i, int radix)
   Returns a string representation of the first argument in
the radix specified by the second argument.
```

dvs givet ett heltal `"i"` så returnerar metoden detta konverterat till ett tal i talbasen "`radix`". Om radix är 2 så får vi alltså ett binärt tal.

Ex:

toString(23, 2) ger "10111", toString(23, 10) ger "23", toString(-23, 2) ger "-10111"

Du skall nu skriva en egen variant av Integer.toString som vi kallar int2base. Skriv alltså en *rekursiv* funktion som omvandlar ett heltal till ett binärt tal i form av en sträng. Metodspecifikationen skall se ut så här:

```
static String int2base(int i, int base) {
```

"base" skall alltid vara 2 här och anropar man med något annat så skall du kasta en `NumberFormatException` exception med texten

`"nat2Base: Otillåten bas, måste vara 2 "`

För att göra detaljerna lite enklare så kan ni använda `Integer.toString` för omvandlingen av små tal 0..9 tex för `Integer.toString(1)` som då är "1".

Skriv också ett huvudprogram som tabellerar talen 1 till 20 och deras motsvarande binära tal. Glöm inte fånga en exception och använd felobjektet för att göra en utskrift.

Tips: Man omvandlar ett heltal, tex 23, till ett binärt tal genom att heltals-dividera med talbasen som för binära tal är 2. Siffrorna längst till höger nedan är resten vid divisionen (alltid 0 eller 1 vid bas 2) och det är dessa som sedan bildar det binära talet. Man slutar när resultatet är noll (sista raden nedan). Resultatet kan sedan läsas nerifrån och upp dvs 10111 så int2bin(23, 2) skall ge "10111".

$23/2 = 11 + 1$

$11/2 = 5 + 1$

$5/2 = 2 + 1$

$2/2 = 1 + 0$

$1/2 = 0 + 1$ (12p)

Problem 3. interface, metoder, loopar,

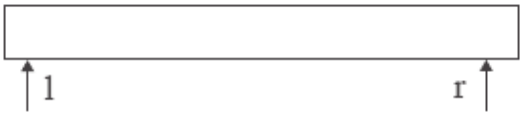
En av operationerna i sorteringsmetoden quicksort är att "partitionera" (Sv. avdela, fördela) men det är en ganska generell operation som även används på andra ställen. Den generella operationen "att partitionera" innebär att man delar upp element i två grupper: de element som uppfyller ett villkor och de element som inte uppfyller villkoret.

Partitioneringen innebär vanligen att elementen flyttas till början eller slutet av vektorn (man kan också tänka sig att returnera de element som uppfyller ett visst kriterium men så gör vi inte här).

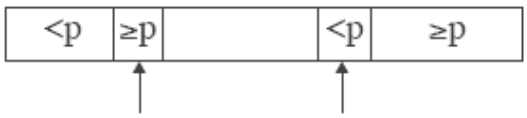
Exempel: partitionering av vektorn {1, 2, 3, 4, 5, 6} med villkoret "jämnt tal" ska medföra att vektorn blir {2, 4, 6, 1, 5, 3} (ordningen mellan de tre första talen är godtycklig, liksom mellan de tre sista).

Den algoritm för att partitionera (dvs för att dela upp fältet) som används i quicksort är denna:

1) använd två pekare low (l) och high (r) som från början pekar på första resp sista elementet i vektorn



2) scan: flytta low och high tills dess att du har situationen till höger.



Där står <p för att villkoret är uppfyllt (dvs talen är jämna i exemplet ovan) och ≥p står för att villkoret inte är uppfyllt.

3) switch: byt f(low) och f(high) och du har situationen till höger

4) test: fortsätt tills low>high

5) returnera low som anger gränsen mellan de objekt som uppfyller/inte uppfyller villkoret.

Nu vill vi implementera en generell algoritm i Java för partition dvs vi vill kunna ha villkoret som indata till algoritmen så vi kan använda olika partitioneringsvillkor enligt

```
/** Flyttar om talen i vektorn f så att de tal som
*  uppfyller villkoret filter hamnar före de tal som
*  inte uppfyller villkoret */
public static int partition(Object[] f, Filter filter){
```

För att göras detta använder vi ett interface

```
/** @return true om obj uppfyller villkoret,
    false annars */
public interface Filter {
  public boolean accept(Object obj);
}
```

För varje specifikt villkor skriver man sedan en subklass till Filter. Till exempel ser ett skal för en klass för villkoret "jämnt tal" ut så här:

```
/** @return true om obj är ett jämnt tal,
*   false annars */
public final class Even implements Filter {
  public boolean accept(Object obj) {
    ...
  }
} ... forts...
```

Nu kan man skicka med ett Even-objekt till algoritmen för att partitionera enligt villkoret jämnt tal och objekt av andra subklasser för att partitionera enligt andra villkor tex alla geometriska objekt med fler än 3 hörn eller alla svarta bilar osv.

a) Skriv metoden partition.

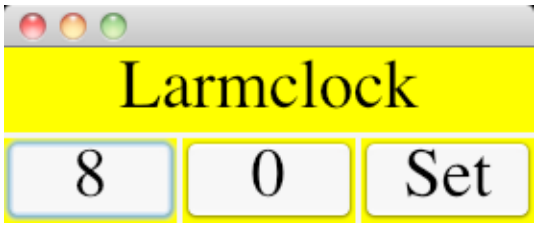
b) Skriv färdigt klassen Even. Tänk på att du får ett fält med Objekt som parameter, inte int eller Integer.

c) Skriv ett huvudprogram som skapar en vektor med talen 0..8 och sedan partitionerar med Even.

(17p)

Problem 4. Grafiskt gränssnitt, händelsehantering

I den här uppgiften skall du konstruera en enkel och ofärdig larmklocka typ äggklocka. Gränssnittet skall se ut som figuren nedan.



Den vänstra knappen (8) representerar timmar och den mittersta minuter. När man klickar på timknappen så ökas den med ett tills den når 24 då den blir 0. Minutrarna ökar i steg om 10 varje gång man klickar och när man kommer till 60 så blir dom 0 och timmarna ökas med ett. Klickar man på Set knappen så kommer programmet ihåg tiden och texten ”Larmet satt på 9 50” skrivs ut på standard output (om tiden på klockan var 9:50).

Några krav och friheter:

Textfältet (Larmclock) skall vara gult.

Du skall använda minst en gridlayout.

Fonten skall vara plain 36p.

Du behöver inte ”larma” på annat sätt än med utskriften.

Du får ha knapparna som instansvariabler om du vill.

(20p)

Problem 5. Även inom programmering måste man kunna ta fram sina mjuka sidor:

Skriv en dikt om kursen på minst 4 rader med rim och med minst 2 ord som har med programmering/pt/java/kursen att göra.

(4p)

Uppgift 1

1a falskt. pekarna skall vara lika
1b sant. hur man implementerar beror inte på specen
1c falskt. man kan representera en kvadrat på många sätt
1d sant. scopen är olika
1e falskt. endast instansvariabler initieras
1f stort adressintervall och liten tabell. figur krävs.

```java
// Provkörd
public class Int2Base {
/** ********** ********** ********** ********** **********
* Omvandlar ett naturligt tal 'nat' till ett tal i basen base
* Det nya talet representeras i form av en sträng
* endast om 2 <= base <= 10 och nat >= 0
* Detta är en enkel variant av Integer.toString(nat, base);
********** ********** ********** ********** ********** */
    // Omvandlar nat till ett tal i basen 'base'
    // 2<= base <= 10 och nbr >= 0, kollas ej
    static String int2base(int nat, int base) {
        if ( base<2 || base>10) {
            throw new NumberFormatException( "nat2Base: "
                                + "Otillåten bas, måste vara 2..10 " );
        } else if ( nat < 0 ) {
            return "-" + int2base(-nat, base);
        } else {
            if (nat < base) {
                return Integer.toString(nat);
                //return "" + nat;
            } else {
                return int2base(nat/base, base) + (nat%base);
                //return int2base(nat/base, base) + Integer.toString(nat%base);
            }
        }
    } // nat2base


    // enklare men sämre slutvillkor, detta räckte på tentan
    static String int2base2(int nat, int base) { //8
        if ( base != 2 ) {
            throw new NumberFormatException( "nat2Base: "
                                + "Otillåten bas, måste vara 2" );
        } else if ( nat < 0 ) {
            return "-" + int2base2(-nat, base);
        } else if ( nat == 0 ) {
                return "0";
        } else if ( nat == 1 ) { // onödig men avsaknad ger inledande nolla
        //      return "1";
        } else {
                return int2base2(nat/base, base) + Integer.toString(nat%base);
        }
    } // nat2base


    // ---------- ---------- ---------- ----------
// krävs bara 2 kolumner på tentan och bara 1..20
    public static void main(String[] args) { //4
        try {
            System.out.println("Bas 10\tBas 2\tBas 3\tBas 4\tBas 5\tBas 6");
            for ( int i = -2; i<=16; i++ ) {
                System.out.print(i + " \t");
                for ( int j=2; j<=6; j++ ) {
                    System.out.print(int2base(i,j) + " \t" );
                }
                System.out.println();
            }
            System.out.println();
            System.out.println(int2base(23, 2) );
            System.out.println(int2base(-23, 2) );
```

```
        } catch (NumberFormatException ex) {
            System.out.println();
            System.out.println("## ex.getMessage() ger utskriften: \n" + ex.getMessage());
            System.out.println("## ex.printStackTrace() ger utskriften: " );
            ex.printStackTrace();
        }
    } // end main
} // end Int2Base
```

```
public final class Partition {

    public static int partition (Object[] f, Filter filter){
        if ( f == null || f.length == 0) {
            return -1;
        }
        int low = 0;
        int high = f.length-1;
        while( low < high ) {
            while ( low <= high && filter.accept(f[low]) ){
                low = low + 1;
            }
            while (low < high && !filter.accept(f[high]) ){
                high = high - 1;
            }
            if ( low < high ) {
                //swap
                Object temp = f[low];
                f[low] = f[high];
                f[high] = temp;
            }
        }
        // we return the first position of the part that don't
        // fulfil the accept criteria. Sometimes this is outside the array.
        return low;
    } // end partition


    public static void main(String[] args) {

        Object[][] arr = new Object[9][]; // Integrer också ok men inte int
        String[] correctAns = new String[9]; // bara för utskrifterna

        // null och icke integers
        arr[0] = new Object[5];
        correctAns[0] = " should be 1";
        // allt som inte är en Integer returnerar false
        arr[0][0] = ("kalle");
        arr[0][1] = null;       // speciellt här returnerar accept false
        arr[0][2] = ("sdfg");
        arr[0][3] = ("23");
        arr[0][4] = ((Integer)24);

        // blandat
        arr[1] = new Object[8];
        correctAns[1] = " should be 4";
        arr[1][0] = 0;
        arr[1][1] = 1;
        arr[1][2] = 2;
        arr[1][3] = 3;
        arr[1][4] = 4;
        arr[1][5] = -3;
        arr[1][6] = -6;
        arr[1][7] = 7;

        // alla jämna
        arr[2] = new Object[4];
        correctAns[2] = " should be 4";
        arr[2][0] = 2;
        arr[2][1] = 2;
        arr[2][2] = 2;
```

```
        arr[2][3] = 2;

        // alla udda
        arr[3] = new Object[4];
        correctAns[3] = " should be 0";
        arr[3][0] = 1;
        arr[3][1] = 1;
        arr[3][2] = 1;
        arr[3][3] = 1;

        // alla udda utom sista
        arr[4] = new Object[4];
        correctAns[4] = " should be 1";
        arr[4][0] = 1;
        arr[4][1] = 1;
        arr[4][2] = 1;
        arr[4][3] = 2;

        //alla jämna utom sista
        arr[5] = new Object[4];
        correctAns[5] = " should be 3";
        arr[5][0] = 2;
        arr[5][1] = 2;
        arr[5][2] = 2;
        arr[5][3] = 1;

        // alla udda utom första
        arr[6] = new Object[4];
        correctAns[6] = " should be 1";
        arr[6][0] = 2;
        arr[6][1] = 1;
        arr[6][2] = 1;
        arr[6][3] = 1;

        //alla jämna utom första
        arr[7] = new Object[4];
        correctAns[7] = " should be 3";
        arr[7][0] = 1;
        arr[7][1] = 2;
        arr[7][2] = 2;
        arr[7][3] = 2;

        // null fält
        correctAns[8] = " should be -1";

        //StringsMoreThan3 smt3 = new StringsMoreThan3();
        Even even = new Even();
        //LargerThanZero ltz = new LargerThanZero();
        //FilterOutInteger integ = new FilterOutInteger();

        for (int i = 0; i<arr.length; i++) {
            int middle = partition(arr[i], even);
            System.out.print("arr " + i + ": middle = " + middle + "   ");
            System.out.print(correctAns[i] + " ** ");
            if (middle == -1) {
                System.out.println("null array ");
                break;
            } else {
                for (int j = 0; j<arr[i].length; j++) {
                    System.out.print(arr[i][j] + " ");
                }
```

```
            }
            System.out.println();
        }
    }
}
```

```java
public final class Even implements Filter {

    public boolean accept(Object obj) {
        if ( obj instanceof Integer ) {
            return ((Integer)obj)%2==0;  // funkar utan intValue
            //return ((Integer)obj).intValue()%2==0;
            //return obj%2==0;  // funkar inte
        } else {
            return false;
        }
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Larmklocka extends JFrame implements ActionListener {

    private JButton tim;
    private JButton min;
    private JButton setAlarm;
    private int timmar = 8;
    private int minuter = 0;
    private int larmTimmar = 0;
    private int larmMinuter = 0;

    public Larmklocka() {
        setTitle("");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //setLocation(130,50);
        // The horizontal and vertical gaps are set to the specified values (3,3)
        // not needed on exam
        setLayout(new GridLayout(2, 1, 3, 3));
        // Build GUI
        JLabel rubrik = new JLabel("Larmclock",  JLabel.CENTER);
        add(rubrik);

        JPanel tidsrad = new JPanel();
        tidsrad.setLayout(new GridLayout(1,3,3,3));
        tim = new JButton("8");
        min = new JButton("0");
        setAlarm = new JButton("Set");
        tidsrad.add(tim);
        tidsrad.add(min);
        tidsrad.add(setAlarm);
        add(tidsrad);

        tim.addActionListener(this);
        min.addActionListener(this);
        setAlarm.addActionListener(this);

        rubrik.setBackground(Color.yellow);
        rubrik.setOpaque(true);

        Font font = new Font("Times", Font.PLAIN, 36);
        rubrik.setFont(font);
        tim.setFont(font);
        min.setFont(font);
        setAlarm.setFont(font);

        pack();
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == tim) {
            timmar = timmar + 1;
            if (timmar==24) {timmar = 0;}
            tim.setText("" + timmar);
        }
        if (e.getSource() == min) {
            minuter = minuter + 10;
            if (minuter >= 60) {
                minuter = 0;
```

```
                timmar = timmar + 1;
                if (timmar == 24) {timmar = 0;}
                tim.setText("" + timmar);
            }
            min.setText("" + minuter);
        }
        if (e.getSource() == setAlarm) {
            larmTimmar = timmar;
            larmMinuter = minuter;
            System.out.println("Larmet satt på " + timmar + " " + minuter);
        }
    }
    public static void main(String[] s) {
        Larmklocka enLarmklocka = new Larmklocka();
    }
}
```

Några dikter
(Stavfel och felaktiga avskrivningra är helt mina fel)

Att jämföra rader i en matris,
jag trodde jag blev galen, nu var det kris.
Men efter att ha kollat på min metod,
jag löste det och se, det ökade på mitt mod.
Så nu tog jag mig an en liten sekant,
och likt Kalle, när han ser tillbaks på backen,
ja, nu känner jag likadant.

Vi kämpat, svettats, stitigt vårt hår
Försökt att förstå semantiken
Om till slut vi tyckte att "Ja, jag förstår!"
var det istället problem med tekniken.
Klasser, metoder, intar och strängar
Vad har vi fått för detta som tack?
Godis, presenter, bullar och pengar?
Nej, Kalle, Hobbe och propaganda om Mac.
Så om i slutet precis under gränsen vi ligger
Vi springer till Erland fortare än kvickt
Visar 25:an, bönar och tigger:
"Det var ju en jättefin dikt!"

Java i mitt hjärta, låt mig besjunga dig nu,
bringar en massa smärta men bättre än C++ är du.
Av kurser jag känner i världen,
är det den här kursen som fått allt,
genom framens kärlek till panelen,
får vi nåt riktigt ballt!

I DD springer Karl och hjälper till,
interface här, JButtons där,
aldrig får han sitta still!
Handledarna har för mycket att göra,
och bara klagomål och uppgivenhet får dom höra.
När programmen inte kompileras ok,
skriker labgruperna: Nej, nej, nej!
Men Bartolomeus är alltid lik apositiv och glad,
kanske borde vi ge honom choklad?

Det är synd att det snart är jul
programmering är ju så kul.
Det är mycket man kommer att sakna
men inte att till Erlands tuta vakna
Det är bra att kunna Java
det är något man nu med sig i livet kommer att hava
och kanske kommer jag på lovet att under granen sitta
och på min fina CrystalModel titta

Nu när jag får använda lite poesi,
får jag nästan en känsla av eufori.
Programmering och dikt kombinerat till ett,
av dessa två blir man väl aldrig mätt.
Även om jag inte fattar uppgift 1b
hoppas jag dikten får rättaren att le.
Kvalitetsmässigt blir den nog inte så vass,
nu måste jag nog nödrimma på metod eller klass,
kasta ett exception och fånga det igen,
kompilatorn är numera min käratse vän.
Förlåt snälle Erland för att jag inte pluggade igår,
förreste, hålls det någom omtenta i vår?
Osammanhängande blev dikten nog nu,
men om den blev bra bestämmer ju du.

Nu börjar jag längta hem till min säng,
så jag avslutar dikten och hoppas på poäng.

I Javas "bibliotek" finns paketet javax.swing
som gör det lättare att skapa grafiskt bling
Dock är inte detta min största passion,
jag önskar hela tentan hade handlat om rekursion.

Programmeringsteknik i all ära,
det är roligt, men mycket att lära.
Eclips blev snabbt en nära vän,
men att programmera på papper är jag inte lika bra på, än.

Förrut visste jag inte vad programmering var
men nu kan jag kasta exceptions som en hel karl!
Och Erlands skämt för att ambitiöst få klassen att höra på
gjorde att jag till slut fick mina while-loopar att funka som få
Så tack för min första programmeringskurs i Java
den fick verligen mig och min labbkompis att slav!

Ah, då var tentan äntligen över
nu jag något stark i mig behöver
programmeringskursen var ju jättesvår!
det blir nog en omtenta i vår.
Men Erland du skall ändå ha ett tack
för jag har faktiskt bytt min PC mot en Mac!

F1pt 20101211

Några vanligt förekommande fel utan inbördes ordning

U2 int2base
- glömmer hantera negativa tal
- gör try-catch i int2base istället för i huvudproghrammet
- ordningen felaktig dvs det binära talet byggs upp i omvänd ordning
- rekursionen fungerar inte
  tex använder globala variabler
- ingenting funkar

U3 partition
- glömmer allt om tomma fält, tex kan man göra
   if (f==null || f.length==0) return false
- flyttar index utan att kolla bounds och får inexOutOfBounds
  eller har dom i fel ordning
  if (low<high && filter.accept(...)) måste det vara

U3 Even
- typfel av olika slag
  - testar tex obj instanceOf int (istället för Integer)
  - typcastar till ett int fält (istället för Integer)
  - gör obj%2 utan typecast
  - deklarerar fältet som int[] trots att partition vill ha ett fält med objekt

U4 larmclock
- ökar inte timmarna när mnutrarna passerar 60
- sparar inte larmtiden trots att det explicit står i tesen
- olika problem med fonter
-

U5 dikter
(detta är väl inte fel i vanlig mening men inte heller poänggivande)
- flera 2-radare helt utan inbördes sammanhang
- rim på java - lava: Det är svårt att hitta på nåt meningsfullt med de orden
  Det är lättare med jul - kul men inte heller det blir nån höjdare för det mesta.
- allmänt dåligt flyt