Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 27 okt 2013

Examination in

Objektorienterad programvaruutveckling IT1 TDA545

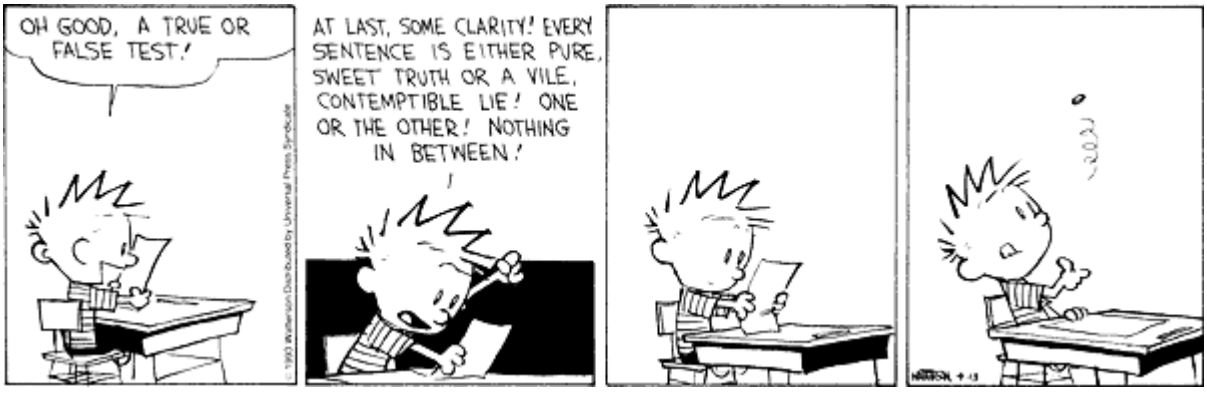
DAY: THURSDAY DATE: 2013-10-24 TIME: 14.00-18.00 ROOM: M

Responsible teacher:	Erland Holmström phone 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=28p, 4=38p, 5= 48p, max 60p
Aids on the exam:	En sida **"Tentamenshjälpmedel – språkkonstruktioner i Java"**
utanför	på denna får man också skriva vad man vill **på de vita ytorna** textrutorna. **Lappen lämnas in med tentan.**

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. Uppvärmning

a) Förklara de bägge begreppen polymorfism och överlagring. Ge exempel.

b) Skriv om följande if sats (skriven i pseudokod) så enkelt som möjligt utan att använda någon if sats.

```
if prime(a) and a>1 then
  if prime(b) then
    return true;
  else
    return false;
else
  return false;
end if
```

c) Vad skrivs ut av följande program inspirerat av George Orwells roman Animal farm?

```
public class AnimalFarm extends Object {
  public static void main(String[] args) {
    final String pig = "Snowball";
    ...println("All Animals are equal: " + pig==pig);
  }
}
```

d) Antag att vi har en klass `Circle` (med radie) och en klass `Cylinder` (med radie och längd) som ärver `Circle`. Många böcker skriver equals metoderna såhär

I Circle:

```
public boolean equals (Object rhs){
    if ( rhs instanceof Circle ) {
      return this.getRadius() == ((Circle)rhs).getRadius();
    } else {
      return false;
    }
  }
```

I Cylinder:

```
  public boolean equals (Object rhs){
    if ( rhs instanceof Cylinder )
      return super.equals(rhs) &&
        this.getLength() == ((Cylinder)rhs).getLength();
    else {
      return false;
    }
  }
```

Ge ett exempel på när dessa definitioner inte fungerar som tänkt och förklara varför.

(2+1+3+3p = 9)

Glöm inte svara på kursenkäten.

(fel är rättade)

Exam Help lappen skall lämnas in med tentan,
ni får tillbaka den med tentan igen.

Problem 2. Testar: rekursion

En (tal)palindrom är samma tal oavsett om man läser det fram eller baklänges. Talet 12321 är tex en palindrom men 55558 är det inte. Talet 21 i basen 10 (vi skriver 21_{10}) är ingen palindrom men 21_{10} i basen 2 är det (10101_2).
Talpalindromer har inte inledande (eller avslutande) nollor, 0220 är alltså ingen palindrom.

a) Skriv först en rekursiv funktion

```
static String nat2base(int nat, int base)
```

som omvandlar ett naturligt tal, `nat`, (i basen 10) till ett tal i basen `base` ($2 \leq base \leq 10$) i form av en sträng (indata skall kontrolleras, använd en wrapperfunktion). `nat2base(21,2)` skall alltså returnera strängen "10101". Inga inledande nollor. (`nat2base` är just vår egen version av `Integer.toString` så den får du naturligtvis *inte* använda.

b) Skriv också en rekursiv funktion som vänder på en sträng

```
static String reverse(String s)
```

dvs reverse("Kalle") ≡ "ellaK"

c) Skriv sedan ett program som läser in två heltal `nbr` ($1 \leq nbr \leq 25$) och `from` ($0 \leq from \leq 100000$) och som skriver ut de första `nbr` talen större än `from` och mindre än 100000 som är palindromer i någon bas ($2 \leq basen \leq 10$). Utdata skall vara `nbr` stycken rader. På varje rad skall det stå ett heltal som är palindrom i någon av baserna 2-10. Ex: Om indata är 8 25 så skall utdata vara

```
tal= 26 bas= 3 222
tal= 26 bas= 5 101
tal= 27 bas= 2 11011
tal= 27 bas= 8 33
tal= 28 bas= 3 1001
tal= 28 bas= 6 44
tal= 29 bas= 4 131
tal= 30 bas= 9 33
```

Du kan anta att man anger två tal vid inmatningen (och att dom är tal) men måste kolla att dom uppfyller kraven ovan.

(13p)

Problem 3. Testar: val, metoder, logiska uttryck

Skriv en metod som avgör om en tidpunkt är ok, se Uppg 6. Om den inte är det så skall lämplig åtgärd vidtagas.

```
boolean isDateOk(int y, int m, int d, int h, int min, int s)
```

Du kan anta att det finns en metod `isLeapYear(y)` som gör det uppenbara.

Du skall skriva en hjälpmetod `daysInMonth(int yearNbr, int monthNbr)` som avgör antalet dagar i en månad. Månaderna 4, 6, 9 och 11 har 30 dagar.

Du får inte använda if/case sats i `isDateOk` och skall använda en case i `daysInMonth`.

(4p)

Problem 4. Testar: Enkel klass

Vi behöver en klass som hanterar ett kombinationslås.

Vårt lås har en 3-siffrig kombination. För att öppna matar man in en siffra i taget via metoden `enter`, om man matar in de tre rätta siffrorna i rätt ordning så öppnas låset. Låset måste alltså komma ihåg inmatade siffror. Konstruktorn tar låsets kombination som parameter i form av ett 3-siffrigt heltal dvs 123 skall tolkas som att kombinationen är siffrorna 1, 2 och 3.

Felaktiga indata hanteras på lämpligt sätt.

```
public class CombLock {
  // @check 0<=combination<=999
  public CombLock(int combination) ...
  public void close() ...
  public boolean isOpen()  ...
  // @check 0<=digit<=9
  public void enter(int digit) ...
```

Några fall:

- kommandot `close` nollställer dvs låset blir låst och inmatade siffror nollställs.

- om rätt kombination delvis matats in och man matar in fel siffra så måste man ibland börja om och ibland inte.
Exempel: om kombinationen är 123 och man matar in 122 så måste man börja om från början och mata in 123 igen eftersom man matar in siffrorna i en ordning och den måste vara rätt (dom tre sist inmatade siffrorna skall vara rätt). Men om kombinationen är 223 och man matat in 222 så räcker det att man matar in 3 för att öppna låset eftersom de tre sista siffrorna i 2223 är rätt kombination.

- det spelar ingen roll hur många siffror man matar in och

- när låset väl är öppet så ändras inte det om man matar in fler siffror.

Exempel: Antag att koden är 1,2,3

inmatat siffra	låsets status
4	closed
1	closed
2	closed
5	closed
1	closed
2	closed
3	open
5	open
1	open

Skriv klassen.

(10p)

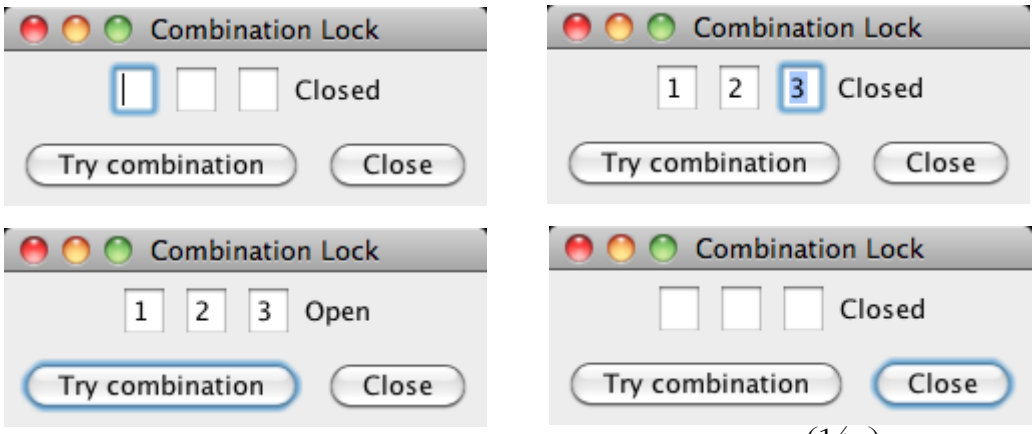
Problem 5. Testar: Swing, händelsehantering

Vi skall också presentera vårt lås grafiskt enligt nedan.

Den första bilden visar hur låset ser ut när vi startar. ”Closed” visar att låset är låst, knappen ”Try combination” används för att mata in siffrorna i låset och knappen ”Close” för att låsa låset.

I den andra bilden har jag matat in en kombination och i den tredje har jag tryckt på ”Try combination” och eftersom kombinationen var rätt så har låset öppnats.

I den fjärde så har jag tryckt på knappen ”Close” så låset är låst och all inmatning glömd.



(14p)

Problem 6. Testar: läsning från fil, exceptions, val, loopar, läsa API, använda objekt

Låt en *tidpunkt* vara given som 6 heltal y, m, d, h, min, s där $1970<y<2030$, $1\leq m\leq 12$, $1\leq d\leq 31^{*}$, $0\leq h\leq 24$, $0\leq min<60$, $0\leq s<60$.
Tex 1997 12 31 23 59 59. (*maxvärde, aktuellt värde beror av värdet på andra data) Tidpunkter startar som vanligt dvs ett år startar med 1 januari, en månad på sin första dag och en dag klockan 00:00:00.

En *period* anges som ett par av ett positivt heltal och ett ord som uttrycker en (tids)enhet. Dom är avskilda med mellanslag. Ordet kan vara year eller month eller day eller hour eller minute eller second. Tex 24 *hour* eller 1 *day*. En period tar slut efter sin sista sekund.

Skriv ett program som beräknar hur många perioder av en given längd som får rum mellan två givna tidpunkter, d_1 och d_2.

Note: För att förenkla den här uppgiften något så kan du anta att alla månader har 30 dagar.

Exempel: Antag att tidpunkterna är 1997 03 15 14 00 00 och 1997 03 15 16 02 00 dvs det är två timmar och 2 minuter mellan dem och att perioden är 1 hour, då skall programmet svara 2 eftersom det får rum två timmar. Samma svar om perioden varit 60 minute. Om perioden varit 1 minute skulle svaret vara 122.

Indata

Indata finns i filen ”indata.txt” (som kanske inte existerar) och består av ”block” med 3 rader. Första och andra raden innehåller tidpunkter (d_1, d_2) med $d_1<d_2$. Alla tal i en tidpunkt avgränsas med minst ett mellanslag.

Den tredje raden innehåller en tidperiod.

Mellan blocken står en tom rad.

Du kan anta att indata är korrekt angivna.

Exempel:

```
1997 12 31 23 59 59
1998 1 1 0 0 0
1 second


2000 2 29 0 0 0
2000 2 29 23 59 59
1 day


2000 2 29 0 0 0
2000 3 1 0 0 0
24 hour


1996 12 31 20 30 0
1997 1 1 7 30 0
60 minute


1996 12 31 20 30 0
1997 1 1 7 30 0
1 hour
```

Utdata

Det skall vara en utdatarad per indatablock med en siffra som anger hur många av de angivna perioderna som får plats mellan de givna tidpunkterna.

Exempel för indata enligt ovan

```
1
0
1
11
11
```

Tips: Du kan ha användning av klassen Calendar, se API i slutet. Tex kan man skapa två datum av indata och sedan omvandla till millisekunder. Funkar enkelt om alla månader har 30 dagar och alla år 365, annars blir det inte lika lätt :-)

(10p)