Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström

Göteborg 1 xxx 20xx

Examination in

PROGRAMMERINGSTEKNIK F1 TIN211

DAY: THURSDAY DATE: 20xx-xx-xx TIME: 8.30-12.30 ROOM: xx

Responsible teacher:	Erland Holmström tel. 1007,
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=26/28p, 4=36/38p, 5= 46/48p, max 60p
Aids on the exam:	...

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,

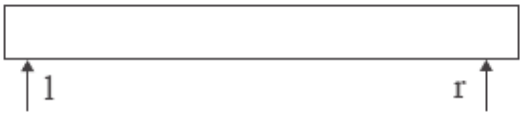
En av operationerna i sorteringsmetoden quicksort är att “partitionera” (Sv. avdela, fördela) men det är en ganska generell operation som även används på andra ställen. Den generella operationen “att partitionera” innebär att man delar upp element i två grupper: de element som uppfyller ett villkor och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

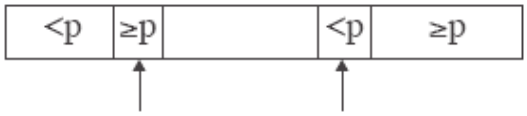
Exempel: partitionering av vektorn {1, 2, 3, 4, 5, 6} med villkoret ”jämnt tal” ska medföra att vektorn blir {2, 4, 6, 1, 5, 3} (ordningen mellan de tre första talen är godtycklig, liksom mellan de tre sista).

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

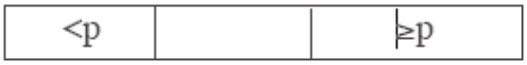


2) scan: flytta low och high tills dess att du har situationen till höger.



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.

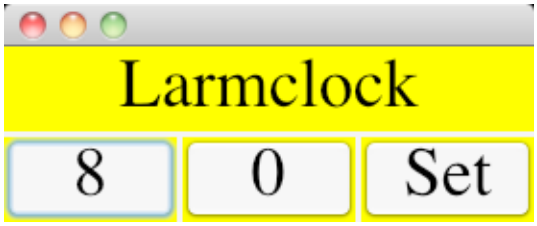
b) Skriv färdigt klassen Even. Tänk på att du får ett fält med Objekt som parameter, inte int eller Integer.

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)