Data och Informationsteknik / Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Erland Holmström Göteborg 6 dec 2016

Dugga i

PROGRAMMERINGSTEKNIK F1/TM1 TIN212

Dag: Måndag Datum: 2016-12-12 Tid: 8.30-11.30 (OBS 3 tim) Rum: M

Ansvarig lärare:	Erland Holmström tel. 1007, mobil 0708-710 600
Resultat:	Skickas med mail från Ladok.
Lösningar:	Läggs eventuellt på hemsidan.
Tentagranskning:	Tentan finns efter att resultatet publiserats på vår expedition. Tid för granskning av rättningen annonseras på hemsidan efter att resultatet är publiserat (eller maila mig så försöker vi hitta en tid).
Betygsgränser:	G=15p, max 30p
Hjälpmedel:	Inga.

Observera:

- Börja med att läsa igenom alla problem så du kan ställa frågor när jag kommer.
  Jag brukar komma efter ca 1 timme (men det är typ 6 salar så det kan bli senare).
- Alla svar måste motiveras där så är möjligt/lämpligt.
- Skriv läsligt! Rita figurer. Lösningar som är svårlästa bedöms inte.
- Skriv kortfattat och precist.
- Råd och anvisningar som getts under kursen skall följas.
- Programs skall skivas i Java, skall vara indenterade och lämpligt kommenterade.
- Börja nya problem på ny sida.
-
- *Lämna inte in tesen med tentan utan behåll den*. Lämna inte heller in kladdpapper/skisser.

Lycka till!

Du behöver inte skriva några import satser i uppgifterna.

Problem 1.

a) Vad skrivs ut av följande programkod om vi antar att klassen Card (se problem 2) finns i samma mapp.
Förklara varför det blir som det blir (svar utan rätt förklaring får inga poäng).

```
public class Test {
  public static void main(String[] args) {
    System.out.println("12" + 5 + 3);
    Card c1 = new Card(100);
    Card c2 = new Card(100);
    System.out.println(c1);
    System.out.println(c2==c1);
    System.out.println(c2=c1);
  }
}
```

(4p)

Problem 2. Testar:klasser, val

Du skall implementera en klass som simulerar ett kort, typ Västtrafiks kontoladdningskort. Den skall innehålla följande:

- två instansvariabler: en som håller reda på saldot (double) och en som håller reda på när nuvarande stämpling på kortet löper ut (long).
- en konstruktor

  `public Card(double saldo)`

som skapar ett nytt kort med saldot som är givet som parameter. Den skall också sätta tiden på lämpligt sätt, värdet skall vara sådant att om man anropar metoden ”stamp” nedan så skall den inte ge svaret ”byte” (sätt den tex till ”nu”-1).

- en metod

  `public void load(double amount)`

som laddar kortet med beloppet som är givet. (dvs adderar till nuvarande värdet)

- två ”getter”-metoder för instansvariablerna.
- och slutligen en metod

  `public String stamp()`

som simulerar att man stämplar sitt kort. Metoden skall returnera en sträng som innehåller någon av ”ok”, ”fail”, ”byte”. Implementeringen skall uppfylla tre regler:

- om den nuvarande tidsstämpeln inte gått ut så returneras bara strängen ”byte”.
- om det nuvarande saldot på kortet är mindre än 10 kronor skall metoden returnera ”fail”.
- annars skall metoden minska saldot med kostnaden för resan dvs 22 kronor och sätta en ny tidsstämpel 90 minuter efter nuvarande tid. Slutligen skall den returnera ”ok”.

Du skall använda konstanter där det är lämpligt, statiska variabler/metoder där det är lämpligt och glöm inte att deklarera synbarheten public/private.

Note: Man kan få reda på tiden genom att anropa den statiska metoden

`long System.currentTimeMillis()`

Den ger nuvarande tid i millisekunder. Varje ny stämpling skall vara giltig i 90 minuter = 5 400 000 millisekunder. Du behöver inte översätta millisekundrarna till en riktig tid.

(9p)

Problem 3. *Testar: iteration, metoder, fält, scanner mm*

a) Implementera metoden

```
public static int localMaximum(int[] a)
```

som får ett fält med osorterade heltal. Metoden skall returnera index för första lokala maximat i fältet

Ett lokalt maximum är ett tal som är större än båda sina grannar. Till exempel om metoden anropas med talen {2, -1, 3, 6, 0, 7, 8} så skall den returnera 3 eftersom på index 3 finns talet 6 och 6 är större än 3 och 0. Första talet har ingen föregångare och sista talet har ingen efterföljare och kan därmed inte vara lokala maxima.
Om det inte finns något lokalt maximum så skall metoden returnera -1.
Lägg vikt vid att få indexen i loopen rätt.

b) Skriv också en klass som testar metoden ovan genom att läsa in tal från en användare. Talen sparas i ett fält och sedan anropas metoden ovan samt slutligen skrivs resultatet ut. Första talet som läses in är antalet tal som följer. Ge lämpliga utskrifter.
Du kan anta att talen är korrekta heltal.

(7p)

Problem 4. *Testar: rekursion, metoder, val*

Ingen kan väl ha undgått att det har varit presidentval i USA nyligen. Valet baseras på att erövra så många elektors röster (ER) som möjligt och varje stat har ER som motsvarer deras storlek i antalet invånare, se en lista nedan. Av tradition så vinner den som får flest röster i en delstat alla elektors röster.

Det finns 50 delstater + district of Columbia och det totala antalet elektorsröster är 538 så det krävs 270 ER för att vinna.

Frågan är om det kan bli oavgjort dvs att varje kandidat får 269 ER?

Vi löser ett lite mer generellt problem:

Skriv en *rekursiv* metod som undersöker om det är möjligt att givet ett fält, t, med tal skapa en delmängd av talen vars summa är lika med ett givet tal, m (i vårt specialfall är $m = 269$ och t=talen nedan men algoritmen skall fungera även för andra tal och fält)

Du skall alltså skriva metoden

```
boolean subsetSum(t, m)
```

där t är ett heltalsfält och m är ett heltal och den skall returnera true om det går att hitta en delmängd av talen vars summa är lika med m.

Du behöver inte rapportera vilka tal som ingår, bara om det finns en summa.

Om jag skrivit av rätt så ser listan med ER ut så här: (men du har ingen användning för den i uppgiften bara om du vill provköra när du kommer hem.)

3,3,3,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5,6,6,6,7,7,7,7,8,8,9,9,9,10,10,10,10,11,11,11,11,12,13,15,15,15,17,20,21,21,27,31,34,55

Tips1: Du behöver en wrapper metod som inför ett index som håller reda på var i fältet vi är och som gör lämpliga felkontroller.

Tips2: ett tal är antingen med i delmängden eller inte.

(10p)

Dugga programmeringsteknik 2016-12-12 Lösningar

Uppgift 1
=========
Utskrift:
1253
Card@677327b6
false
Card@677327b6

System.out.println("12" + 5 + 3);
All beräkning sker från vänster till höger om man inte har parenteser.
strängen "12" läggs ihop med talet 5 efter att 5 automatiskt omvandlats till en sträng.
Sedan läggs på samma sätt talet 3 till strängen.
+ med en sträng till vänster är alltså strängkonkatenering ev. med automatisk typkonvertering.

System.out.println(c1);
Systemet letar efter en toString metod i Card klassen. Eftersom det inte finns någon så letar systemet vidare i Objekt klassen och där finns en som används. Den skriver dock bara ut klassens namn, ett @-tecken och objektets adress i minnet.

System.out.println(c2==c1);
Här jämförs adresserna och dom är olika så "false" skrivs ut.

System.out.println(c2=c1);
Här sker först en tilldelning av adressen i c1 till c2 och sedan skrivs c2 ut på samma sätt som System.out.println(c1); Därför skrivs C1's adress ut

Uppgift 2
=========

```
public class Card {
    private double saldo = 0.0;
    private long validTo = 0;

    private static final long   TICKET_VALIDITY = 5400000;
    private static final double TICKET_PRICE    = 22;
    private static final double SALDO_LIMIT     = 10;

    public Card(double saldo) {
        if (saldo<10) {
            throw new IllegalArgumentException("beloppet måste vara > 10");
            // en exception är rätt här för man kan inget annat göra jmf U3
        }
        this.saldo   = saldo;
        this.validTo = System.currentTimeMillis()-1;
    }

    public void load(double amount) {
        // samma test som i konstuktorn
        saldo = saldo + amount;
    }

    public double getSaldo() {
        return saldo;
    }

    public long validTo() {
        return validTo;
    }

    public String stamp() {
        long time = System.currentTimeMillis();
        if( time < validTo ) {
            return "byte";
        } else if( saldo < SALDO_LIMIT ) {
            return "fail";
        }
        validTo = time + TICKET_VALIDITY;
        saldo = saldo - TICKET_PRICE;
        return "ok";
    }
}
```

Uppgift 3
=========

```
import java.util.Scanner; // behövs ej på duggan
public class Question {
    public static int localMaximum(int[] a) {
        if( t==null || t.length==0 ) {
           return -1;
           // här bör man returnera -1 dvs att det inte finns lokalt maxima
            // en exception är fel här för det finns ett legalt värde jmf U2
        for (int i = 1; i < a.length-1; i++) {
            if (a[i] > a[i+1] && a[i] > a[i-1])
                return i;
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("mata in antalet tal: ");
        int x = sc.nextInt();
        // ganska dålig felhantering, bättre åtgärd är bra
        // men det viktiga är att indikera att här behövs nåt
        if (x<0) {x=0;}
        int[] a = new int[x];
        System.out.println("mata in talen ");
        for (int i = 0; i < a.length; i++) {
            a[i] = sc.nextInt();
        }
        System.out.println("index på " + localMaximum(a));
    }
}
```

Uppgift 4
=========

Detta är övningsuppgift 7 på övning 3
(Och eftersom ni fått lösningen på övning så minskar svårighetsgraden kraftigt. Såväl övningar som labbar ingår ju i kursen så jag förutsätter att ni gjort dom. Annars är detta en rätt svår uppgift. Uppgiften finns även som problem 4 på sid 344 i boken)

```
public static boolean subsetSum(int[] t, int m) {
    if( t==null || t.length==0 ) {
        return false;
        // false är rätt svar här, inte en exception
    } else {
        return subsetSumLocal(t, m, t.length-1);
    }
}

private static boolean subsetSumLocal
                (int [] t,  // weights
                 int m,     // knapsack
                 int k) {   // index into t
    if ( m == 0 ) { // a solution
        return true;
    } else if (m > 0 && k >= 0) {
        return subsetSumLocal(t, m-t[k], k-1)  // with t(k)
            || subsetSumLocal(t, m, k-1);       // without t(k)
    } else {
        return false;  // no more elements or m<0
    }
}
```