Tentamen för TDA143 PROGRAMMERADE SYSTEM

DAG: 17-03-14 TID: 8:30 – 13:30

Ansvarig:		Christer Carlsson, ankn 1038
Förfrågningar:		Christer Carlsson
Resultat:		erhålls via Ladok
Allmän info:		tentamen är uppdelad i två delar: del 1 omfattar översikt av datateknik och del 2 omfattar programmering
Betygsgränser:	3:a	10 poäng på del 1 och 26 poäng på del 2
	4:a	13 poäng på del 1 och 36 poäng på del 2
	5:a	16 poäng på del 1 och 48 poäng på del 2
	maxpoäng	20 poäng på del 1 och 60 poäng på del 2
Siffror inom parentes:		anger maximal poäng på uppgiften.
Granskning:		Fredag 21/4 kl 12-13 och måndag 24/9 kl 12-13, rum 6128 i EDIT-huset.
Hjälpmedel:		En valfri lärobok i Java eller de på kursen utdelade föreläsningsanteckningarna (OH-bilderna) som behandlar **programmeringsdelen** av kursen.
		Förtydligande noteringar får finnas i boken resp. föreläsningsanteckningarna.
Var vänlig och:		Skriv tydligt och disponera papperet på lämpligt sätt.
		Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.
Observera:		Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva.
		Alla program skall vara väl strukturerade, lätta att överskåda samt enkla att förstå.
		Vid rättning av uppgifter där programkod ingår bedöms principiella fel allvarligare än smärre språkfel.
		På de programmeringsuppgifter som ingår i tentamenstesen kan vissa poäng erhållas även om ett fullständigt program inte redovisas, detta kräver dock att en algoritm som löser problemet presenteras.

LYCKA TILL!!!!

Uppgift 1.

Denna uppgift består av 10 påståenden. Frågorna skall besvaras med ”sant” eller ”falskt”.

Ett korrekt svar ger 1 poäng, ett felaktigt svar ger -0.5 poäng och ett utelämnat svar ger 0 poäng. Totalt på uppgiften kan aldrig minuspoäng erhållas.

a) Talområdet för ett 8-bitars heltal på tvåkomplementsform är -127 till 128.

b) Virtuellt minne innebär att ett program får illusionen av att det har allt primärminne för sig självt.

c) Hypertext Transfer Protocol (HTTP) är det vanligaste kommunikationsprotokollet för att leverera elektronisk post.

d) Nedanstående algoritm har tidskomplexiteten $\Theta(n^2)$.

```
for (int i =1; i <= n;  i++)
   for (int j =1; j <= n;  j++)
       for (int k =1; k <= n;  k++)
           if (a[i]+a[j] == a[k])
               return true;
return false;
```

e) Om ett binärt träd har exakt 5 interna noder (dvs noder som inte är löv eller rot), kan trädet ha maximalt 7 löv.

f) Datormaskar (worms) är en typ av datavirus som ofta används av hackers för att överbelasta någon specifik webbplasts och därmed förhindra normal användning av webbplatsen.

g) I en databas eftersträvar man *redundans* för att öka tillgängligheten till den information som finns lagrad i databasen.

h) Begreppet t*exturing*, som används inom datorgrafik, innebär att en bild projiceras på ett geometriskt objekt

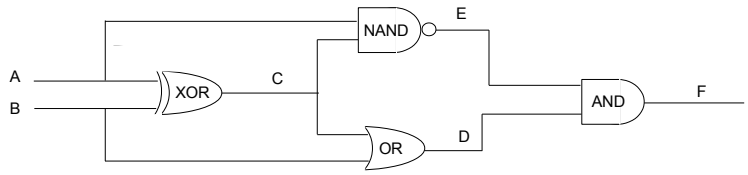
i) En programmerare använder vanligtvis *black box testing* för att testa de programmoduler han/hon skrivit.

j) Inom mjukvaruindustrin är korta utvecklingscykler den viktigaste aspekten för överlevnad och framgång.

(10 poäng)

Uppgift 2.

Ange sanningstabell för den logiska kretsen nedan.



(2 poäng)

Uppgift 3.

Ge en kort beskrivning av primär- respektive sekundärminne och förklara skillnaden. Ofta beskrivs minnen som flyktiga respektive icke-flyktig, vad menas med detta? (2 poäng)

Uppgift 4.

Beskriv datastrukturerna *stack* och *kö.* (2 poäng)

Uppgift 5.

Antag att du har följande databastabell:

Student:

Name	School	Age	Distance
Anna Hansson	Lilleputtskolan	13	7,1
Klas Stålnacke	Hagaskolan	11	6,9
Stefan Lilja	Lilleputtskolan	14	3,5
Jens Andersson	Stjärnaby skola	9	2,1
Clara Skoog	Stjärnaby skola	7	5,4

Hur ser tabellen ut som erhålls när nedanstående SQL-query körs?

```
SELECT Name, School
FROM Student
WHERE Age < 12 AND Distance > 5
```

(2 poäng)

Uppgift 6.

Förklara vad som menas med begreppet *software ecosystem*. (2 poäng)

DEL 2: Programmeringsteknik

Uppgift 7.

a) Betrakta nedanstående program:

```
public class Mystery {
  public static void main(String[] args) {
      System.out.println(mystery(8));
      System.out.println(mystery(15));
  } //main
  public static String mystery(int num) {
      if (num < 0)
          throw new IllegalArgumentException();
      if (num == 0)
          return "0";
      String str = "";
      while (num > 0) {
          str = num%2 + str;
          num = num/2;
      }
      return str;
  } //mystery
}//Mystery
```

i) Vad blir utskriften när programmet exekveras?

ii) Förklara med ord vad metoden mystery gör (= vilket syfte metoden har).

(3 poäng)

b) Betrakta nedanstående program

```
import java.util.Scanner;
public class FirstProgram {
  public static void main(String[] args) {
      if (proceed())
          System.out.println("I'm glad you will continue!");
      else
          System.out.println("Good-bye.");
  }//main

  public static boolean proceed() {
      Scanner scannerObject = new Scanner(System.in);
      while (true) {
          System.out.println("Answer yes to continue, and no to stop.");
          String answer = scannerObject.next();
          if (answer == "yes")
              return true;
          if (answer == "no")
              return false;
          else
              System.out.println("Your answer should be yes or no!");
      }
  }//proceed
}//FirstProgram
```

Avsikten med metoden proceed är att upprepa en fråga tills användaren besvarar frågan med "yes" eller med "no". Besvaras frågan med "yes" returnerar metoden värdet **true**, och om frågan besvaras med "no" returnerar metoden **false**. Metoden fungerar inte, oavsett vad användaren svara upprepas frågan! Förklara vad som är fel och korrigera felet.

(2 poäng)

c) Betrakta nedanstående program, vars syfte är att läsa in ett tal och skriva ut talets kvadratrot:

```
import javax.swing.*;
public class  Math {
   public static void main (String[] arg) {
        String indata = JOptionPane.showInputDialog("Ange ett tal: ");
        double  tal = Double.parseDouble(indata);
        double  root = Math.sqrt(tal);
        JOptionPane.showMessageDialog(null, "Roten ur talet " + tal + " är " + root);
   } //main
} //Math
```

När programmet kompileras fås följande felmeddelande

```
Math.java:6: cannot resolve symbol
symbol  : method sqrt (double)
location: class Math
double root = Math.sqrt(tal);
                        ^
```

Förklara vad som är felaktigt i programmet och åtgärda felet.

(2 poäng)

Uppgift 8.

Spelet V75 går ut på att finna de vinnande hästarna i 7 travlopp. En kupong i V75 har följande utseende:

Lopp 1: 1 3 5 7 8 11
Lopp 2: 5 7
Lopp 3: 2 9
Lopp 4: 3
Lopp 5: 1 2 4 5 6 11
Lopp 6: 2 3 4 6 9
Lopp 7: 2 5 8 10 12

På en kupong kan man tippa flera hästar i samma lopp. På kupongen ovan har spelaren t.ex. tagit ut 6 hästar i lopp 1 (hästarna med nummer 1, 3, 5, 7, 8, 11) och kupongen innehåller totalt 6 * 2 * 2 * 1* 6 * 5 * 5 = 3600 rader (som kostar 50 öre per styck).

V75 har fått sitt namn pga att vinstutdelning sker för 7, 6 och 5 rätt.

Om en kupong innehåller *en rad med 7 rätt* och vi betecknar antalet tippade hästar i lopp *i* som n_i erhålls antalet rader med 6 rätt av formlen $\sum_{i=1}^{7} n_i - 1$ och antalet rader med 5 rätt av formeln $\sum_{i=1}^{6} \sum_{j=i+1}^{7} (n_i - 1) \times (n_j - 1)$.

a) Skriv en klass ATG som innehåller en klassmetod

public static int evaluateNrOf6(**int**[] nrOfHorses)

som tar ett heltalsfält nrOfHorses, vilket innehåller antalet tippade hästar i respektive lopp på en V75-kupong som innehåller en rad med 7 rätt, och returnerar antalet rader med 6 rätt på kupongen (enligt formeln ovan).

b) Utöka klassen ATG med en metod

public static int evaluateNrOf5(**int**[] nrOfHorses)

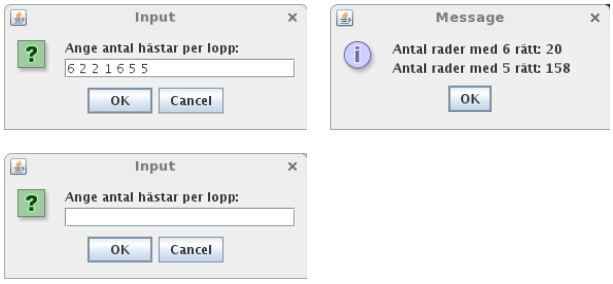
som tar ett heltalsfält nrOfHorses, vilket innehåller antalet tippade hästar i respektive lopp på en V75-kupong som innehåller en rad med 7 rätt, och returnerar antalet rader med 5 rätt på kupongen (enligt formeln ovan)

c) Utöka klassen ATG med en main-metod som upprepade gånger läser in antalet tippade hästar per lopp på en V75-kupong, där en rad finns med 7 rätt, och skriver ut antalet rader med 6 respektive 5 rätt. Du får själv välja om du vill göra in- och utmatning via dialogrutor eller använda System.in respektive System.out (se körningsexempel nedan). Exekveringen av programmet avbryts vid användning av dialogrutor genom att användaren trycker på Cancel-knappen och vid användning av System.in genom att användaren lämpligen ger ctrl z.

För full poäng skall ett Scanner-objekt användas vid inläsningen.

Observera att du kan lösa denna deluppgift utan att du löst föregående deluppgifter genom att anta att dessa metoder finns tillgängliga.

Med användning av dialogrutor



Med användning av System.in resp System.out

```
Antalrader med 6 rätt: 20
Antal rader med 5 rätt: 158
Ange antal hästar per lopp:
```

(15 poäng)

Uppgift 9

I ett program för att hantera godstransporter finns klassen Goods för att avbilda styckegods. Klassen har följande publika instansmetoder:

int getID()	som returnerar godsets unika id-nummer
String getContents()	som returnerar godsets innehåll
int getWeight()	som returnerar godsets vikt (i kg)
String getDestination()	som returnerar godsets destination
String toString()	som ger en strängrepresentation på formen: (239163, Kompressor, 9870, Lysekil)

Din uppgift är att skapa en klass Ship för att avbilda lastfartyg. Klassen Ship skall ha följande privata instansvariabler:

String name	som anger fartygets namn
int capacity	som anger fartygets lastkapacitet i kg
ArrayList<Goods>cargo	som innehåller lasten (godset) som finns ombord på fartyget

Klassen Ship skall innehålla:

- en konstruktorer, som har parametrar för att ange fartygets namn och lastkapacitet. När ett fartyg skapas innehåller fartyget ingen last.
- metoder för att avläsa fartygets namn respektive lastkapacitet.
- en metod **int** getTotalWeight(), som returnerar totala vikten av lasten som finns ombord.
- en metod **boolean** loadGoods(Goods item) som tar ett objekt item av typen Goods och lastar item ombord på fartyget om det är möjligt, dvs om lastningen av item inte innebär att fartygets totala lastkapacitet överskrids. Metoden returnerar **true** om lastningen lyckas, annars returner metoden **false**.
- en metod Goods unloadGoods(**int** id) för att lasta av ett gods från fartyget. Metoden tar id-numret för Goods-objektet som skall lastas av. Om det finns ett objekt med angivet id-nummer ombord lastas detta objekt av och metoden returnerar en referens till objektet, annars returnerar metoden **null**.
- en metod ArrayList<Goods> willBeUnloadAt(String port), som tar en destination, port, och returnerar en ArrayList vilken innehåller det gods som skall lastas av på den angivna destinationen. Finns inget gods till angiven destination returneras en tom lista.

(11 poäng)

Uppgift 10

En digital bild kan representeras som ett tvådimensionellt fält av bildpunkter. I en digital färgbild utgörs varje bildpunkt av *tre* heltalsvärden i intervallet 0-255, där de enskilda värdena representerar intensiteten av färgerna rött, grönt och blått. En färgbild kan avbildas med ett tredimensionellt fält av typen **int**[][][], är den första dimensionen definierar bildens höjd, den andra dimensionen definierar bildens bredd och den tredje dimensionen representerar färgerna rött, grönt och blått.

Din uppgift är att skriva en metod

public static int[][][] scale(**int**[][][] picture, **double** factor)

som tar en bild picture och returnerar en ny bild som är en kopia av picture vars storlek är factor * storleken av picture.

Orginal bild

Skalad med faktor 0.75

Skalad med faktor 1.25

(6 poäng)

Uppgift 11

Ett *anagram* är en fras eller ett ord som bildas genom att kasta om bokstäverna i en annan fras eller i ett annat ord. Nedan ges några exempel:

katt	takt
statsminister	Sitsen smittar
Mona Sahlin	Himla osann
Peter Forsberg	ger sportfeber
Moderaterna	roar dementa
Kristdemokraterna	rikedomen skrattar
Florence Nightingale	Flit on, cheering angel!

Som framgår av exemplen ovan, är det endast bokstäverna som ingår i en fras som avgör om två fraser är anagram eller inte, och ingen skillnad görs mellan versaler och gemener. Således gäller det att en fras är ett anagram till en annan fras om de båda fraserna innehåller samma uppsättning bokstäver (utan hänsyn till om de är versaler eller gemener) och varje bokstav förekommer lika många gånger i de båda fraserna.

a) Skriv en metod

public static int numberOfTimesOccurring(String str, **char** ch)

som tar en sträng str samt ett tecken ch, och returnerar hur många gånger tecknet ch förekommer i strängen str. Ingen skillnad skall görs mellan stora och små bokstäver.

Exempel:

anropet numberOfTimesOccurring("Sitsen smittar", 's',) skall returnera värdet 3
anropet numberOfTimesOccurring("Sitsen smittar", 'p') skall returnera värdet 0

b) Nyttja metoden numberOfTimesOccurring i deluppgift a) för att skriva en metod

public static boolean lettersOccuringSameNumberOfTimes(String str1, String str2)

som tar två strängar str1 samt str2, och avgör *om varje bokstav i strängen str1* förekommer lika många gånger i strängarna str1 och str2.

Exempel:

anropet lettersOccuringSameNumberOfTimes("katt", "skratt") skall returnera värdet **true**
anropet lettersOccuringSameNumberOfTimes("skratt", "katt") skall returnera värdet **false**

c) Nyttja metoden lettersOccuringSameNumberOfTimes i deluppgift b) för att skriva en metod

public static boolean isAnagram(String str1, String str2)

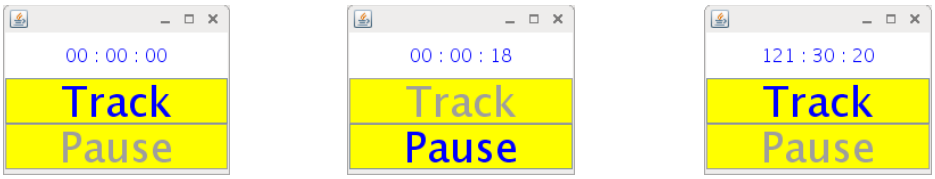
som tar två strängar str1 samt str2, och avgör om strängarna är anagram eller inte.

Obs: Du kan lösa deluppgift b) utan att du löst deluppgift a) och deluppgift c) utan att du löst deluppgift b).

(10 poäng)

Uppgift 12

Du skall skriva ett enkel grafiskt program för att hålla reda på tiden som läggs på ett viss projekt. Användargränssnittet skall vara enligt figurerna nedan:



När programmet startas (bilden till vänster ovan) erhålls ett fönster som innehåller två knappar samt en etikett. När man trycker på knappen Track börjar den nedlagda tiden på projektet att räknas upp, och tiden skrivs ut varje sekund i etiketten (bilden i mitten ovan). Detta kommer att pågå till man trycker på knappen Pause, då tiduppräkningen stoppas (bilden till höger ovan). Tiduppräkningen startas igen är man trycker på Track.

När tidsuppräkningen är aktiv skall Track-knappen vara otryckbar och när tidsuppräkningen är stoppad skall Pause-knappen vara otryckbar.

Gränssnittet skall ha lämpliga färger, teckenstorlekar och fonter (se figurerna ovan), och programmet skall kunna avslutas genom att trycka på stängningsrutan i fönsters ram.

Till din hjälp finns klassen Time nedan att tillgå (och som du bör använda):

```
public class Time {
  private long seconds;

  public Time() {
      seconds = 0;
  }//constructor

  public void increase() {
      seconds++;
  }//increase
  public int getHours() {
      return (int) seconds / 3600;
  }//getHours

  public int getMinutes() {
      return (int) (seconds % 3600) / 60;
  }//getMinutes

  public int getSeconds() {
      return (int) seconds % 60;
  }//getSeconds

  public String toString() {
      return String.format("%02 : %02d : %02d", getHours(), getMinutes(), getSeconds());
  }//toString
}//Time
```

(11 poäng)