Tentamen för TDA143 PROGRAMMERADE SYSTEM

DAG: 17-08-18 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:		Onsdag 13/9 kl 12-13 och tisdag 19/9 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) Om vi lagra heltal i 8 bitar på tvåkomplementsform kommer additionen $65_{10} + 63_{10}$ att resultera i *overflow* och det erhållna resultatet blir -128_{10}.

b) *Paging* är en teknik som används för att implementera virtuellt minne.

c) Ett RAM-minne är i allmänhet snabbare än ett ROM-minne.

d) IPv4 (Internet Protocol version 4) anger en IP-adress med 32 bitar och IPv6 (Internet Protocol version 6) anger en IP-adress med ett 64 bitar.

e) Inom datasäkerhet används begreppet *spoofing* för att beteckna skadlig programvara vars syfte är att stjäla information.

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

```
for (int i =1; i <= n;  i++) {
   int k = i;
   while (k > 1) {
        System.out.println(k);
        k = k / 2;
   }
}
```

f) Handelsresande problemet (the traveling salesman problem) är ett s.k. NP-komplett problem, vilket betyder att det inte finns några effektiva algoritmer för att hitta en exakt lösning till problemet.

g) *Data mining* är en teknik för att upptäcka mönster i stora datamängder.

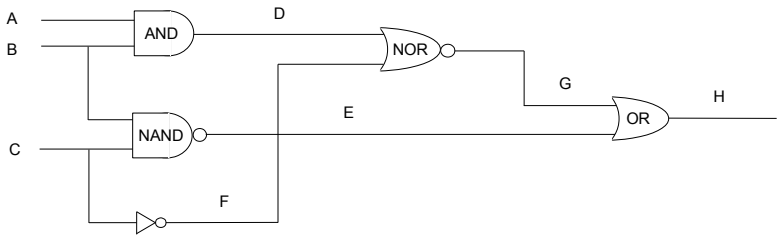
h) När man inom software engineering använder uttrycket *”Need for speed”* syftar man på att dagens allt mer avancerade programsystem kräver allt snabbare datorer.

i) Ett programsystem i vilket modulerna har hög *coupling* och låg *cohesion* är lättare att bygga ut och modifiera än ett programsystem där modulerna har låg *coupling* och hög *cohesion.*

(10 poäng)

Uppgift 2.

Ange sanningstabellen för den logiska kretsen nedan:



(2 poäng)

Uppgift 3.

Målet med datorgenererade bilder är ofta att bilderna skall vara så verklighetstrogna att de inte med blotta ögat skall kunna skiljas från verkliga fotografier. Nämn några aspekter som är viktiga att beakta för att uppnå detta mål.

(2 poäng)

Uppgift 4.

Θ-notationen används för att klassificera algoritmer med avseende på tidskomplexitet. Förklara vad det innebär att en algoritm tillhör en viss Θ-klass, dvs redogör för vilka kriterier hos algoritmen som används för att fastställa dess tidskomplexitet.

(2 poäng)

Uppgift 5.

Förklara vad ett binärt sökträd är, hur det är uppbyggt och dess egenskaper. Ge exempel på hur det används för sökning och insättning.

(2 poäng)

Uppgift 6.

Antag att du har följande databastabeller:

Kunder:

Namn	PersonNr	KontoNr
Jens Andersson	8509093476	345621
Anna Hansson	7904195801	493028
Stefan Lilja	6412120552	475629
Per Persson	9412120412	553322
Clara Skoog	8903041243	292347

Konto:

KontoNr	Saldo
292347	89035
345621	25090
475629	65435
493028	15210
553322	34550

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

```
SELECT Namn, Saldo
FROM Konto, Kunder
WHERE Saldo > 25000 AND Saldo  < 75000
```

(2 poäng)

DEL 2: Programmeringsteknik

Uppgift 7.

a) Vad skrivs ut när main-metoden i klassen Uppgift7a exekveras?

```
public class Uppgift7a {
    public static void main(String[] args) {
        int[] a1 = {1, 2,3,4, 5};
        mystery(a1);
        System.out.println(java.util.Arrays.toString(a1));
    }//main

    public static void mystery(int[] a) {
        for (int i = 0; i < a.length; i = i + 1) {
                a[i] = a[i] + a[a.length - 1 - i];
        }
    }//mystery
}//Uppgift7a
```

(2 poäng)

b) Vad blir utskriften när main-metoden i klassen TestCube nedan exekveras?

```
public class Cube {
  private int side;
  public Cube(int side) {
      this.side = side;
  }
  public void setSide(int side) {
      this.side = side;
  }
  public int getVolume() {
      return side * side * side;
  }
}//Cube

public class TestCube {
  public static void main(String[] args) {
      int n = 1;
      Cube[] cubes = new Cube[4];
      Cube cube = new Cube(n);
      for (int i = 0; i < cubes.length; i++) {
          cubes[i] = cube;
          n++;
          cubes[i].setSide (n);
          System.out.println(cubes[i].getVolume());
      }
      System.out.println(cube.getVolume());
      for (Cube c : cubes) {
          System.out.println(c.getVolume());
      }
  }//TestCube
```

(2 poäng)

Uppgift 8.

På sjön har man sin egen terminologi för att beteckna vindstyrkan. Det gäller att

Vindstyrka m/sek	Benämning
0 - 0.2	stiltje
0.3 - 13.8	bris
13.9 - 24.4	kuling
24.5 - 32.6	storm
32.7 -	orkan

Skriv ett program som läsa in en vindstyrka angiven i m/s och som skriver ut vilken benämning denna vindstyrka anges med på sjön.

Du får själv välja om du vill göra in- och utmatning via dialogrutor eller använda System.in respektive System.out (se exemplen nedan).

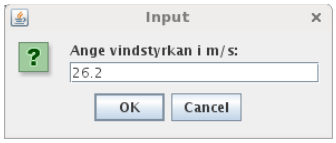
För att erhålla full poäng på uppgiften

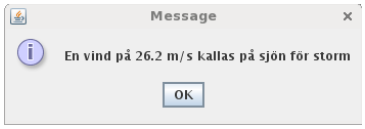
- skall programmet utformas på så sätt att inläsningen upprepas tills användaren avbryter exekveringen (vid användning av dialogrutor genom att användaren trycker på Cancel-knappen och vid användning av System.in genom att användaren lämpligen ger ctrl z)
- skall programmet innehålla en metod

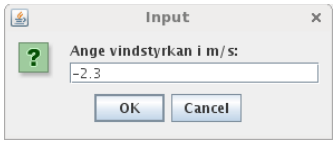
  public static String term(**double** wind)

  som tar en vindstyrka angiven i m/s och returnerar den term som används på sjön för att ange denna vindstyrka.
- skall programmet kontrollera att en korrekt vindstyrka läses in och ge en felutskrift om användaren anger felaktig indata. En korrekt vindstyrka skall ligga i intervaller [0.0, 115.0] (den högsta vindstyrkan som uppmäts är 113.2 m/s och gjordes i cyklonen Olivia 1996).

Med användning av dialogrutor







Med användning av System.in resp System.out

```
Ange vindstyrkan i m/s:  26.2
En vind på 26.2 m/s kallas på sjön för storm

Ange vindstyrkan i m/s:  -2.3
Felaktig indata! Försök igen!

Ange vindstyrkan i m/s:
```

(10 poäng)

Uppgift 9

I tärningsspelet Mexico deltar minst två spelare och spelet är ett elimineringsspel. När spelet börjar har alla spelare lika mycket pengar att spela med (= startavgift, dvs de pengar som spelaren riskerar att förlora).

En omgång går till på så sätt att varje spelare kastar två 6-sidiga tärningar. Spelaren som får lägst poängvärde förlorar omgången och måste lägga en i förväg fastställd insatsen i potten. Detta upprepas tills endast en av spelarna har pengar kvar. Denne är vinnaren och får hela potten.

För att beräkna poängvärdet, beräknas först den sammanlagda *tärningssumman*. Detta görs genom att först multiplicera värdet på tärningen som har högst poäng med 10 och sedan addera värdet på tärningen som har lägst poäng. Har tärningarna poängen 3 och 4 erhålls tärningssumman 43, har tärningarna poängen 5 och 5 erhålls tärningssumman 55, osv.

Tärningssumman ligger till grund för att beräkna *poängvärdet*. Tärningssummornas poängvärde från det högsta till det lägsta är enligt:

21, 66, 55, 44, 33, 22, 11, 65, 64, 63, 62, 61, 54, 53, 52, 51, 43, 42, 41, 32, 31

Högsta poängvärdet är tärningsvärdet 21 (som kallas ”Mexico”), följt av ”paren” (i numerisk ordning), följt av övriga tärningsvärden (i numerisk ordning).

a) Skriv en klass MexicoDice för att avbilda tärningskast i spelet Mexico. Klassen skall innehålla en konstruktor samt metoderna

public void roll()	som kastar tärningarna
public int getValue()	som returnerar tärningsumman av tärningarna enligt ovanstående regler

I implementationen skall du nyttja den färdiga klassen Die, som du tidigare använt i ett par laborationsuppgifter. Klassen Die har följande gränssnitt:

pubic Die()	konstruktor som skapar en sexsidig tärning
public void roll()	kastar tärningen
public int getValue()	returnerar tärningens värde

b) Skriv en klass Util som innehåller en metod

public static int lowestScoreValue(**int** score1, **int** score2)

som tar två tärningssummor, score1 och score2, och returnerar den tärningssumma som har lägsta poängvärdet.

TIPS: Uppgiften innebär att från de möjliga tärningssummorna skapa en ordningsrelation som överensstämmer med tärningssummornas poängvärde. Ett sätt att skapa värdena i denna ordningsrelation är enligt följande:

- tärningssumman 21 får ett maxvärde (t.ex 1000)
- tärningssummor som är ett ”par” får ett värde som är tärningssumman multiplicerad med 10
- övriga tärningssummor får ett värde som motsvarar tärningssumman.

Ordningsrelationen som då erhålls blir

1000, 660, 550, 440, 330, 220, 110, 65, 64, 63, 62, 61, 54, 53, 52, 51, 43, 42, 41, 32, 31

Lämpligen inför du en hjälpmetod

private static int toSerialNumber(**int** score)

som tar en tärningssumma score och returnerar ordningsvärdet enligt ordningsrelationen ovan.

(7 + 7 poäng)

Uppgift 10.

a) Skriv en metod

public static int secondLargest(**int**[] arr)

som givet en heltalsfält arr returnerar det näst största talet i arr. Om inparametern arr är **null** eller har en längd som är mindre än 2 skall metoden kasta en IllegalArgumentException.

Exempel:

Antag att följande deklaration har gjorts

```
int[] f1 = {1, 5, 3, 7, 2};
int[] f2 = {9, 2};
int[] f3 = {7, 7, 7};
int[] f4 = {1};
```

anropet secondLargest(f1) returnerar då värdet 5

anropet secondLargest(f2) returnerar då värdet 2

anropet secondLargest(f3) returnerar då värdet 7

anropet secondLargest(f4) kastar ett IllegalArgumentException

Metoden får inte förändra ordningen av elementen i indatafälten arr eller använda några metoder från klassen java.util.Arrays.

(8 poäng)

b) 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[][][] frosty(**int**[][][] samples)

som tar en bild samples och returnerar en ny bild som ger effekten att se bilden samples genom en frostad glasskiva. Detta görs genom att låta färgerna i punkten (x, y) i den nya bilden tas från en annan, närliggande punkt i samples; denna punkt skall väljs slumpmässigt så att dess x- och y-koordinater ligger högst 5 bildpunkter från punkten (x, y).

Original bild

Filtrerad bild

TIPS: Skriv en hjälpmetod

public static int distube(**int** v, **int** max)

som tar två heltalsvärden v och max. Metoden returnerar ett slumpmässigt heltal ≥ 0 och ≤ max som samt ligger i intervallet [v-5, v+5].

(8 poäng)

Uppgift 11.

Du skall skriva ett program som visar upp ett fönster, enligt vidstående bild. Fönstret innehåller ett grafiskt objekt av klassen SimpleEggTimer som simulerar en enkel äggklocka.

Klassen SimpleEggTimer utökar JPanel och är konstruerad av tre komponenter: en JButton för att ställa in tiden på äggklockan, en JButton för att starta äggklocka samt en JLabel som visar tiden i sekunder.

När programmet startar skall användaren börja med att ställa in den önskade tiden som äggklockan skall räkna ner. Start-knappen är därför inte tryckbar, utan det enda användaren kan göra är att öka upp tiden med hjälp av '+'-knappen.

För varje tryck som användaren gör på '+'-knappen ökas den inställda tiden med 1 sekund och den nya tiden visa på JLabel-objektet.

För att göra det möjligt att starta klockan, görs Start-knappen tillgänglig första gången '+'-knappen trycks.

Antag att användaren ställer in den önskade tiden 30 sekunder. För att starta äggklockan, dvs. påbörja nedräkningen, trycker användaren på Start-knappen, varvid både Start-knappen och '+'-knappen görs otryckbara (användaren kan alltså inte ändra tiden när väl klockan har startats).

För varje sekund som äggklockan går räknas värdet i JLabel-objektet ned med 1. När nedräkningen är klar genereras en ljudsignal (genom att anropa jawa.awt.Toolkit.getDefaultToolkit().beep()) och '+'-knappen görs återigen tryckbar för att användaren skall kunna ställa in en ny tid och använda klockan på nytt.

. . .

a) Skriv klassen SimpleEggTimer.

Tips: För att varje sekund ändra värdet som skrivs ut i JLabel-objektet, måste du använda ett Timer-objekt som genererar en händelse en gång per sekund.

b) Skriv en klass ShowSimpleEggTimer, som är själva fönstret som innehåller ett objekt av klassen SimpleEggTimer. Klassen ShowSimpleEggTimer skall också innehålla en main-metod som skapar ett ShowSimpleEggTimer-objekt. Programmet skall kunna avslutas genom att trycka på stängningsrutan i fönstrets ram.

Obs: Du kan lösa denna deluppgift utan att gjort deluppgift a).

(16 poäng)