Hemtentamen i Objektorienterad Programmering, DAT043

Joachim von Hacht

Datum: 2020-03-14

Tid: 08.30-12.30

Hjälpmedel: Allt

Betygsgränser:

U: -23, 3:24-37, 4:38-47, 5:48-60 (max 60)

Lärare: Joachim von Hacht, tel. 031/772 10 03.

Granskning: Meddelas via Canvas. Alternativt: Kopiera tentan på studieexpeditionen, ta hem och granska, kontakta mig vid tveksamheter. E-posta och ange noggrant vad du anser är fel så återkommer jag (ev. ta en bild och skicka med).

Instruktioner:

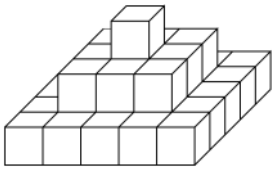
- SKRIV ALLA SVAR I KODEN I REPEKTIVE FIL: Q1andQ2, Q3, Q4, ... direkt i IntelliJ-projektet. Om du inte har tillgång till IntelliJ skriv lösningar i vilken typ av program som helst men lägg in lösningarna i IntelliJ-projektet. Filerna skall heta som ovan (Q1and2, Q3, ..)
- Lösningarna måste klara de fall som anges *samt fall som är principiellt lika*. Lösningar som bara klarar exemplen räcker *inte*. Överkomplicerade lösningar kan ge poängavdrag.
- Färdiga klasser m.m. som får användas anges för varje uppgift. Anges inget får man alltid använda de grundläggande språkliga konstruktionerna, arrayer, egna metoder och egna klasser (men inte t.ex. List/ArrayList/String/m.fl.).

LYCKA TILL...

1. Uppgift 1 och 2 är sammanslagna till en enda uppgift (denna). 6p

   Skriv ett program som beräknar hur hög pyramid man kan bygga om man har tillgång till ett visst antal stenblock. Vi antar att pyramiden är kompakt, d.v.s. det finns inga hålrum inuti. Vidare byggs den enligt principen i figuren nedan. Varje lager är alltså kvadratiskt med en sidlängd som är två block mindre än det underliggande lagrets. Det översta lagret består alltid av ett ensamt block. Programmet ska fråga efter antalet tillgängliga block (högst hundra miljoner) och skriva ut höjden (i block räknat) för den största pyramid som kan byggas. Det gör ingenting om det blir block över, men det får inte saknas ett enda block. Exempel:

   ```
   Number of blocks? 83
   Max height: 3
   ```

2. —

3. Skriv en metod toArray som omvandlar ett positivt heltal (int) till en array av siffrorna i talet (en int-array). Exempel: 6p

   ```
   toArray(12345) ger arrayen [1, 2, 3, 4, 5]
   toArray(1000)  ger arrayen [1, 0, 0, 0]
   toArray(1)     ger arrayen [1]
   ```

4. Givet en kvadratisk matris med heltal och med antal element ≥ 9. Skriv en metod som avgör om en given array finns i någon rad eller kolumn i matrisen. Elementen i arrayen skall finnas i samma ordning i matrisen (vänster/höger eller upp/ner). Vi antar att användaren anger en rimlig array (storlek). För full poäng krävs en lämplig funktionell nedbrytning. Exempel: 12p

```
m1 = {1, 2, 4},       m2 = {1, 3, 2, 1},
     {6, 7, 9},            {4, 5, 1, 7},
     {10, 12, 13}          {1, 2, 6, 0},
                           {1, 7, 9, 2}
```

Matris	Array	Resultat (index börjar på 0)
m1	{9}	true (rad 1 alt. kolumn 2)
m1	{2, 7}	true (kolumn 1)
m1	{7, 6}	false
m2	{2, 6, 0}	true (rad 2)
m2	{5, 2, 6}	false

5. Skriv en metod som "expanderar" en given sträng . Alla metoder i appendix är tillåtna. Expansionen sker enligt följande

```
Given sträng           Expanderad
-----------            ----------
"3(ab)"                "abababab"
"5(x)2(cde)"           "xxxxxcdecde"
"0(x)2(cde)1(xy)"      "cdecdexy"
```

   D.v.s strängen består av en eller flera grupper där varje grupp inleds med en siffra och därefter en parentes med ett antal tecken. Strängen expanderas så att tecknen i parentesen upprepas lika många gånger som siffran anger. Alla parenteser tas bort. 8p

6. Rita en bild som visar variabler, värden, referenser och objekt samt hur dessa förhåller sig till varann före, respektive efter anropet av metoden doIt. Rita som vi ritat under kursen; namn, lådor, pilar o.s.v. Strängobjekt kan ritas förenklat t.ex. "abc". Rita bilden i något ritprogram eller för hand och lägg in den i IntelliJ-projektet med namn: Uppg6.png (eller gif eller jpg) 8p

```
A a = new A(1);
a.a[0] = new A(2);  // Before
a.a[1] = doIt(a);  // Call
                   // After
A doIt(A a) {
   A aa = new A(3);
   a.a[1] = aa;
   aa.a = a.a;
   aa.a[0].i = 9;
   return aa;
}

class A {
   int i;
   A[] a = new A[2];
   public A(int i) { this.i = i;}
}
```

7. Skapa en objektorienterad modell för ett mail-system. Klasserna skall vara så icke-muterbara som möjligt och dölja så mycket som möjligt av sin data (information hiding). Alla klasser/metoder i Appendix är tillåtna. Följande klass skall användas;

```
public class Person {
   private String login;
   private String passwd;
   // Setter/getter, equals, hashCode omitted but present
}
```

   a) Skapa en klass för ett mail. Ett mail skall ha avsändar- och mottagar-adress och en text (innehållet i mailet). All data sätts då objekten skapas. Inga setter/getter, equals eller hashCode behöver anges, vi antar att dessa finns. Gäller även nedan. 2p

   b) Skapa en klass Account, för ett mail-konto. Ett konto har: en innehavare (personen som äger kontot), en adress, en inbox för alla inkomna mail och en outbox för alla skickade mail. Innehavare och adress sätts då kontot skapas. 3p

   c) Skapa en klass för en mail-server. En server har ett antal konton. Lägg till en metod forward som givet ett avsändarkonto, ett mottagarkonto och ett mail vidarebefordrar mailet från sändare till mottagare, om möjligt. Både sändare och mottagare måste ha konton. Det vidarebefordrade brevet måste finnas hos sändaren. När brevet vidarebefordras flyttas det från sändarens inbox till mottagarens inbox och kopieras till sändarens outbox. 5p

8. Termen digital rot (DR) för heltalet N innebär att man upprepade gånger summerar talets ingående siffror tills summan blir < 10. Till exempel för talet 34783 får man 10p

   $3 + 4 + 7 + 8 + 3 = 25$

   $2 + 5 = 7$

   Talet 34783 har DR 7.

   På liknande sätt definieras multiplikativ digital rot (MDR) för heltalet N, där man istället upprepade gånger multiplicerar ingående siffror tills produkten blir < 10. Till exempel för talet 34783 får man

   $3 \cdot 4 \cdot 7 \cdot 8 \cdot 3 = 2016$

   $2 \cdot 0 \cdot 1 \cdot 6 = 0$

   Talet 34783 har MDR 0. Hos talet 34783 är alltså MDR och DR olika.

   Skriv ett program som tar reda på hur många tal i ett givet intervall för vilka det gäller att DR = MDR. Intervallet ryms alltid mellan 1 och en miljon. TIPS: Rekursion.

```
From? 1000
To? 2000
Number of equals: 33
From? 364
To? 371
Number of equals: 2
```

APPENDIX

```
Färdiga Java-klasser/metoder som får användas om så anges vid uppgiften.

Ur klassen String
- equals(s), avgör om en sträng innehåller samma tecken som en annan.
- charAt(int i), ger tecknet vid index i.
- indexOf(char ch), ger index för tecknet ch, -1 om tecknet saknas.
- length() ger längden av strängen.
- subString(int start, int end), ger en delsträng från
  start (inkl.) till end-1.
- subString(int start), ger en delsträng från start (inkl.)
  till strängens slut.
- toCharArray(), gör om strängen till en array med tecken
Ur klassen StringBuilder
- append(String s), lägger till strängen s sist i
  Stringbuilder-objektet.
- append( char ch ), som ovan
- toString(), omvandlar StringBuilder-objektet till en String.
Ur List/ArrayList
- get(i), ger objektet för index i
- add(o), lägger till objektet o sist i listan
- set(i, o), lägger till objektet vid index i (skriver över).
- remove(o), tar bort objektet o ur listan, returnerar
  true om detta lyckades annars false
- remove(i), tar bort och returnerar objektet vid index i ur listan
- removeAll( list ), tar bort alla element i list.
- contains(o), sant om objektet o finns i listan.
- indexOf(o), ger index för objektet
- size(), ger längden på listan
Ur Math
- sqrt(n), roten ur n.
Ur Character
- isDigit(ch), isLetter(ch), isWhiteSpace(ch), getNumericValue(ch),
  toString(ch)

Alla former av omvandlingar mellan String och annan typ
är tillåtna t.ex. Integer.valueOf(), String.valueOf() m.fl.
Klassen Random med metoden nextInt() är alltid tillåten.
```