Tentamen i Introduktion till objektorientering, DAT044

Joachim von Hacht

Datum: 2022-03-12

Tid: 08.30-12.30

Hjälpmedel: Inget förutom Svensk-Engelskt lexikon.

Betygsgränser:

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

Lärare: Joachim von Hacht.

Granskning: Meddelas via Canvas.

Instruktioner:

- För full poäng på essäfrågor krävs ett läsbart, begripligt och heltäckande svar. Generellt 1p för varje relevant aspekt av problemet. Oprecisa eller alltför generella (vaga) svar ger inga poäng. Konkretisera och/eller ge exempel.
- Det räcker med enbart relevanta kodavsnitt, övrig kod ersätts med "..." (aldrig import, main-metod, etc....). Vi utgår från att användaren alltid skriver rätt och/eller gör rätt (d.v.s ingen felhantering behövs). Om felhantering skall ingå anges detta specifikt.
- 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 som får användas anges för varje uppgift. Anges inget får man alltid använda de grundläggande språkliga konstruktionerna, arrayer, egna metoder och egna klasser.

LYCKA TILL...

1. Redogör kortfattat för följande konstruktioner i Java. Du får gärna illustrera med en skiss eller med kod. 4p

   a) Lokal variabel kontra klassvariabel kontra instansvariabel.

   b) Synlighetsområde

2. Vilka av kodraderna nedan kompilerar och vilka kompilerar inte? Du måste ange för varje rad varför det fungerar eller inte. 4p

```
int[] iarr = {1, 2, 3};          // 1
Integer[] iiarr = {1, 2, 3};     // 2
double[] darr = {1, 2, 3};       // 3
Double[] ddarr = {1, 2, 3};      // 4
```

3. Givet två icke-tomma arrayer a och b med positiva heltal. Skriv en metod move(int[] a, int[] b) som avgör om något element, n, i någon av arrayerna kan flyttas från den ena till den andra så att *båda* arrayernas medelvärden *ökar*. Metoden returnerar: 7p

   - n, om elementet skall flyttas från a till b
   - $-n$, om elementet skall flyttas från b till a
   - 0, om element saknas

   Du behöver inte ta hänsyn till eventuella avrundningfel. Exempel:

```
Arrayer                  Medelvärde
-------                  ----------
int[] a1 = {3, 1, 2};      2
int[] a2 = {4, 3, 5, 4};   4
int[] a3 = {7, 5};         6.5
int[] a4 = {4, 6};         5

Anrop       Resultat
-------      -------
move(a1,a2) -3
move(a3,a4)  0  (medelvärde måste öka!)
move(a1,a4) -4
move(a3,a1)  5
```

4. Skriv en metod, removeCols, som givet en matris, $matr$, $m \times n, m > 1, n > 1$ med heltal och en icke-tom array, arr, med heltal, tar bort alla kolumner i $matr$ som innehåller något av elementen i arr. Om alla kolumner tas bort returneras en singulär matris med enda värde Integer.MIN_VALUE. För full poäng krävs funktionell nedbrytning. Du kan anta att du har tillgång till en metod int[] copy(int[] a) som skapar en kopia av arrayen a (ett nytt array-objekt med samma värden). Exempel 12p

```
int[][] m1 = {          int[][] m2 {
   {1, 2},                   {1, 2, 6, -2},
   {3, 4}                    {7, -4, 1, 9},
}                            {9, 1, 3, 4}
                          }
```

```
Anrop                              Resultat
-----                              --------
removeCols(m1, new int[]{1})       {         (kolumn med 1 borta)
                                     {2},
                                     {4}
                                   }
removeCols(m1, new int[]{1, 2}) {          (alla kolumner borta)
                                     {-2147483648}  (=Integer.MIN_VALUE)
                                   }
removeCols(m1, new int[]{5})       {         (ingen kolumn bort)
                                     {1,2},
                                     {3,4}
                                   }
removeCols(m2, new int[]{9})       {         (kolumner med 9 borta)
                                     {2,6},
                                     {-4,1},
                                     {1,3}
                                   }
removeCols(m2, new int[]{6,12,15}) {          (kolumn med 6 borta)
                                        {1,2,-2},
                                        {7,-4,9},
                                        {9,1,4}
                                    }
```

5. Rotation av en sträng innebär att man flyttar tecknen i strängen "åt vänster". Detta gör genom att ta strängens inledande tecken (längst till vänster) och lägga till på slutet av strängen (till höger) ett godtyckligt antal gånger. Exempel: 7p

Sträng 1	Sträng 2	Är Sträng 2 rotation av Sträng 1?
""	""	Ja
"x"	"x"	Ja
"xy"	"xy"	Ja
"xy"	"yx"	Ja
"xyz"	"yxz"	Nej
"xyzv"	"zvxy"	Ja
"xyzv"	"vzxy"	Nej

Skriv en metod som givet två strängar avgör om den ena strängen är en rotation av den andra. Alla klasser/metoder i Appendix får användas.

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: Olika lådor/boxar, variabelnamn, pilar o.s.v. 8p

```
int[] a = new int[]{1, 2, 3};
MyClass m1 = new MyClass("abc", a);
MyClass m2 = new MyClass("xyz", a);  // Before:
doIt(m1, 0, m2.a[2], m2.s);  // The call
                              // After

void doIt(MyClass m, int i, int v, String s) {
   m.a[i] = v;
   m.s = s;
   int[] tmp = {4, 5, 6};
   m.a = tmp;
}

class MyClass {
   int[] a;
   String s;
   MyClass(String s, int[] a) {
      this.a = a;
      this.s = s;
   }
}
```

7. Vi skall skriva ett program för ett bokförlag (Publisher) som alltså bl.a. säljer böcker till bokhandlare. Implementera klasser och metoder enligt a) - c) nedan. Klasserna skall vara så icke-muterbara som möjligt och dölja så mycket som möjligt av sin data (information hiding). Alla klasser/metoder i Appendix får användas. Följande enum och klass skall användas.

```
enum Role {      // Roles for persons employeed at company
   REVIEWER,
   AUTHOR,
   SALES
}

class Book {    // A book (not a perfect implementation)
   String ISBN;
   String author;
   public Book(String ISBN, String author) {
      this.ISBN = ISBN;
      this.author = author;
  }
}
```

a) Skriv en klass, Person, för en anställd på förlaget. En person har ett namn och ett antal roller, en eller flera. Namnet sätts när personen skapas. Lägg till en metod som kan svara på om personen är författare. Inga setter/getter behöver anges, inte heller equals()/hashCode()/toString(). Gäller även nedan. 2p

b) Skriv en klass Publication (publicering). Klassen kopplar ihop en bok med en eller flera författare. Boken och författare anges då publiceringen skapas[1]. 3p

c) Skriv en klass, Publisher, för hela förlaget (företaget). Förlaget har många anställda personer (varav en del författare), många böcker och många publiceringar. Lägg till en metod som givet en bok och en lista med personer, om möjligt, skapar en ny publicering (och spar denna). Om det lyckades returnerar metoden true annars false. För att lyckas måste författarna var anställda och boken måste finnas hos förlaget. 5p

[1] Det är först när en bok är publicerad som den kan börja säljas antar vi.

8. Betrakta koden nedan och ange för varje rad a) - h) en av följande 8p

   - Kompilerar ej därför att ...
   - Körningsfel därför att ...
   - Om inget av ovan, ange vad som skrivs ut. Därför att ...

   (koden är ibland kompakt skriven av utrymmesskäl, läs raden vänster till höger)

```
a) A a = new C(); a.doIt(1.0);
b) C c = new B(); c.doIt(1.0);
c) A a = new B(); a.doIt(1);
d) Object o = new C(); D d = (D) o; d.doIt(1.0);
e) IX i = new C(); A a = (A) i; a.doIt(1.0);
f) Object[] os;
   A[] as = {new A(), new A()};
   os = as;
   ((A) os[0]).doIt(5);
g) List<Object> os = new ArrayList<>();
   List<A> as = new ArrayList<>();
   os = as;
   os.add(new B());
   ((B) os.get(0)).doIt(5);
h) IX ix = new A(); IY iy = new D(); ix = (IX) iy; ix.doIt(1.0);

interface IX { void doIt(double d); }
interface IY { void doOther(double d);}
class A implements IX {
   public void doIt(double d) { out.println("doIt A"); }
}
class B extends A {
    public void doIt(int i) { out.println("doIt B"); }
}
class C extends A {
   public void doIt(double d) { out.println("doIt C"); }
}
class D implements IY {
   public void doOther(double d) { out.println("doOther D");}
}
```

APPENDIX

Följande klasser/metoder 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
- s1.compareTo(s2), ger -1, 0 eller 1 om s1 är respektive mindre, lika med eller större än s2 i lexikografisk ordning
- str1.contains(str2), ger sant om strängen s2 finns i s1

Ur klassen StringBuilder
- append(String s), lägger till strängen s sist i Stringbuilder-objektet.
- append(char ch), som ovan
- indexOf(ch), ger index för tecknet ch
- deleteCharAt(i), raderar tecken vid index i.
- toString(), omvandlar StringBuilder-objektet till en String.

Ur klassen Character
- isDigit(ch),isLetter(ch) avgör om tecknet är en siffra repsltive bokstav

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, flyttar övriga till höger.
- 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

Klassen Random med metoden nextInt() är alltid tillåten.