Tentamen i Imperativ Programmering med Grundläggande Objektorientering, DIT012

Joachim von Hacht

Datum: 2018-01-09

Tid: 14.00-18.00

Hjälpmedel: Engelskt-Valfritt språk lexikon

Betygsgränser:

U: -23

G: 24-43

VG: 44-60 (max 60)

Lärare: Joachim von Hacht, tel. 031/772 10 03. Någon besöker ca 15.00 och 16.30

Granskning: Tentamen kan granskas på studieexpeditionen. Vid ev. åsikter om rättningen eposta mig och ange noggrant vad du anser är fel så återkommer jag (ev. ta en bild och skicka med).

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 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.

LYCKA TILL...

1. Förklara med en eller ett par meningar. Du får gärna ge kodexempel eller rita. 4p

   a) Returtyp

   b) Konstruktor

2. Betrakta koden nedan.

   a) Koden innehåller ett kompileringfel. Vilket? Förklara!

   b) Om felet åtgärdas (på lämpligt sätt). Kommer programmet att kunna köras utan problem? Om ej, förklara! 4p

```
int[] arr = new int[]{1,2,3,4,5};
swap(arr);

void swap(double[] arr){
    for(int i = 0 ; i < arr.length ; i++){
        double tmp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = tmp;
    }
}
```

3. Givet ett positivt heltal n. Skriv en method som ger resultatet av $\sum_{i=1}^{i=n} f(i)$ där $f(i)$ är summan av alla delare till i. Exempel då n= 4: 6p

```
i   delare     f(i)     ∑
1   1           1       1    (1)
2   1,2         3       4    (1+3)
3   1,3         4       8    (1+3+4)
4   1,2,4       7      15    (1+3+4+7)

Metoden skall alltså returnera 15 för n = 4  (n = 5 ger 21)
```

4. Givet en kvadratisk matris med $m \geq 4$ element och ett heltal $2 \leq n \leq \sqrt{m}$. Skriv en metod som returnerar sant om det finns någon kvadratisk delmatris med sidan n sådan att alla element är 1. För full poäng krävs: En lämplig funktionell nedbrytning och att du för varje metod anger syftet med metoden (en kort kommentar vad den gör). OBS! Inte tillåtet att använda samlingar (t.ex. List). Exempel: 12p

```
   matris          n      resultat
[0, 1, 0, 1]       2      true
[0, 0, 1, 1]
[1, 0, 1, 1]
[0, 1, 0, 0]

[0, 1, 0, 1]       2      false
[0, 0, 1, 0]
[1, 0, 1, 1]
[0, 1, 0, 0]

[0, 1, 1, 0]       3      true
[1, 1, 1, 1]
[1, 1, 1, 0]
[1, 1, 1, 0]
```

5. Rotation av en sträng innebär att man flyttar tecknen i strängen till vänster genom att flytta inledande tecken, ett eller flera, till slutet. Exempel: 5p

```
"ACDA" är en rotation av "AACD"
"CDAB"  är en rotation av "ABCD"
"ACBD" är inte en rotation av "ABCD"
```

Skriv en metod som givet två strängar avgör om den ena strängan är en rotation av den andra. Färdiga metoder (från olika klasser) som får användas:

```
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
- endsWith(s), sant om strängen avslutas med s.
StringBuilder
- append(String s), lägger till strängen s sist i
  Stringbuilder-objektet.
- append( char ch ), som ovan
- setLength(), sätter aktuell längd, setLength(0)
  raderar alla tecken.
- toString(), omvandlar StringBuilder-objektet till en String.
```

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, lådor, pilar o.s.v. Ni *måste* rita! 6p

```
A[] as = new A[2];
as[0] = new A(1);
as[0].n = new A(3);   // Before
doIt( as[0]);         // Call
                      // After
void doIt(A a) { a.n = new A(2); a.n.p = a;}

class A {
    int i; A p; A n;
    A( int i ){ this.i = i;}
}
```

7. Implementera en OO-modell för en databas med bilar och personer (som äger bilarna). 9p

   a) Skriv en klass Person för ägare. All personer har ett unikt id och ett namn. Id och namn skall kunna sättas då man skapar en person. Du kan anta att en korrekt equals()-metod och nödvändiga set/get metoder finns (du behöver alltså inte skriva dem).

   b) Skriv en klass Car för bilar. Alla bilar har en ägare av typen Person. Alla bilar har ett märke t.ex. Volvo eller Toyota. Ägare och märke skall kunna sättas då man skapar en bil. Du kan anta att nödvändiga set/get metoder finns.

   c) Skriv en klass Database för hela databasen. Databasen har en lista av bilar och en lista av personer (ägare). Klassen skall ha en metod add(car, person) som registrerar (lägger till) en bil med en viss ägare *om* ägaren finns i databasen. Annars händer inget. Klassen skall ha en metod remove(person). Metoden tar bort personen och alla bilar denne äger ur databasen.

   Klasserna skall vara så icke-muterbara som möjligt och dölja så mycket som möjligt av sin data (information hiding). Färdiga metoder som får användas:

```
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
```

8. Skriv en metod, int minSum(int[][] m), som givet en triangulär matris beräknar den minsta summa man kan få genom att vandra från toppvärdet (4:an i exemplet) till något värde på sista raden. Förflyttningar får göras "rakt ned" eller "snett till höger/vänster" en rad i taget. Det är tillåtet att förändra parametern m. Exempel: 8p

```
int[][] m = {
    {4},
    {3, 2},
    {1, 2, 1},
    {2, 4, 1, 3}
};
Minsta värde: 8 ges av vägen 4, 2, 1, 1
```

9. Betrakta koden nedan. 6p

   a) Vilka rader kommer inte att kompilera. Förklara! OBS! Detta är en nollsummefråga, anger du fel får du lika mycket i avdrag som du skulle fått till godo om du svarat rätt (dock blir totalpoängen aldrig negativ).

   b) Om de rader som inte kompilerar kommenteras ut, vad kommer programmet att skriva ut? Förklara!

```
A a1 = new B();
A a2 = new C();
B b1 = new C();
B b2 = new A();
IX ix1 = new A();
IX ix2 = new C();
A a3 = (A) ix2;
B b3 = (B) ix2;
a1.doIt();         // Will print?

// -----------------------------------------
public interface IX { void doIt();}

public class A {
    public void doIt() { out.println("A doIt()");}
    public void doOther() { out.println("A doOther()");}
}

public class B extends A {
    public void doIt() { out.println("B doIt()");}
    public void doYetOther(){ out.println("B doYetOther()"); }
}

public class C implements IX {
    public void doIt() { out.println("C doIt()");}
}
```