Tentamen i Objektorienterad Programmering DAT043

Joachim von Hacht

Datum: 2019-03-16

Tid: 08.30-12.30

Hjälpmedel: Engelskt-Valfritt språk lexikon.

Betygsgränser:

U: -23

3: 24-37

4: 38-47

5: 48-60 (max 60)

Lärare: Robin Adams ank. 63 48. Någon besöker ca 09.00 och 11.00

Granskning: Anslås på kurssida.

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 kortfattat och exakt varför... 4p

   a) ... man använder klasser?

   b) ... referenser?

   Det räcker med en eller ett par meningar, du får gärna förtydliga med en skiss eller med kod.

2. Koden nedan ger problem. Varför? Motivera! 2p

```
int count = 0;
double a = 1.23;
double b = 4.56;
while (true) {
    if (a == b) {
        break;
    }
    a = a + 0.01;
    count++;
}
out.println("Count is " + count);
```

3. Två arrayer med positiva heltal, a1 och a2, innehåller exakt samma element förutom att den andra arrayen innehåller ett extra element. Skriv en metod som returnerar index till det extra elementet i den andra arrayen. För full poäng krävs en lämplig funktionell nedbrytning. Inget ur Java Collections (samlingar) eller String är tillåtet. Exempel: 6p

```
a1: [6, 4, 2, 8, 10, 12];
a2: [2, 4, 6, 9, 10, 12, 8];       ger svaret: 3

a1: [1, 5, 2, 4, 3];
a2: [1, 7, 5  4, 2, 3];            ger svaret: 1
```

4. Givet en matris m, med heltal (d.v.s. typ int[][]), implementera en metod som avgör om alla rader i matrisen är permutationer av varann. Exempel: 10p

```
m: [1, 2, 3, 4]      resultat: false
   [4, 1, 2, 3]
   [3, 5, 1, 2]
   [2, 3, 4, 1]

m: [1, 2, 3, 4, 5]    resultat: true
   [4, 5, 1, 3, 2]
   [3, 5, 1, 2, 4]
```

   För full poäng krävs funktionell nedbrytning. Inget ur Java Collections (samlingar) eller String är tillåtet.

5. Klassen Converter nedan kan användas för att omvandla från heltal till t.ex. romerska numeraler. Klassens konstruktor får en String-array med symboler och värden för dessa enligt: 12p

```
String[] romans = {"I:1", "IV:4", "V:5", "IX:9","X:10", "XL:40", "L:50",
                   "XC:90", "C:100", "CD:400", "D:500", "CM:900", "M:1000"};
Converter c = new Converter(romans);
String roman = c.convert(547);     // DXLVII
```

   Fler exempel på resultat fråm metoden convert:

```
Värde       Numeral
5           V
14          XIV
1234        MCCXXXIV
9876        MMMMMMMMMDCCCLXXVI
```

   Implementera metoden convert. OBS! Att man skall kunna skicka in andra liknande representation t.ex. String[] others = {”?:1”, ”!:5”, ”@:7”, ”#:9” ...}. För full poäng krävs att man använder så mycket som möjligt av färdiga API:er (se Appendix) samt att man använder funktionell nedbrytning.

```
class Converter {
    final String[] numeralValue;
    public Converter(String[] numeralValue) {
        this.numeralValue = numeralValue;
    }
    String convert(int number) {
        // TODO
    }}
```

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. Strängar kan ritas förenklat som t.ex. “abc”. 8p

```
A a1 = new A(1);
A a2 = new A(2);
A a3 = new A(3);
a1.n = a2;
a2.p = a1;
a2.n = a3;
a3.p = a2;  // Before
A a4 = a2.doIt(a1, a3); // Call
                        // After
class A {
    int i;
    A p;
    A n;
    public A(int i) {this.i = i;}
    A doIt(A a1, A a2){
        i = a1.i + a2.i;
        a1.n = a2;
        a2.p = a1;
        p = null;
        n = null;
        return this;
    }
}
```

7. Vi skall skapa en objektorienterad modell av ett schemaläggningsprogram för möten. Följande (skissade) klasser skall ingå i modellen: 10p

```
class Slot {           // A time slot
    boolean overlaps(Slot other) {
       return ... // true if other overlaps in time with this slot
    }
    // Constructor, equals, hashCode, setter/getters are here
}
public class Person {
    private final String id;
    private String name;
   // Constructor, equals, hashCode, setter/getters are here
}
```

   a) Skriv en klass Room för mötesrum. Ett rum har ett visst antal platser och en lista med upptagna tider (slots). Klassen skall ha en metod för att avgöra om ett rum är upptaget givet en viss tid (slot). Antal platser skall anges då man instansierar ett rum. Inga setter/getters behöver skrivas ut, vi antar att dessa finns (gäller även nedan).

   b) Skriv en klass Meeting för möten. Ett möte har ett rum, en tid och ett antal deltagare. Rum skall anges då man skapar ett möte. Klassen skall ha två metoder: en, som givet en person, avgör om person skall delta i mötet (finns i deltagarlistan) och en som lägger till en person till deltagarlistan om personen inte redan är där.

   c) Skriv slutligen en klass Schedule för ett helt schema. Ett schema har ett antal möten och ett antal rum. Klassen skall ha en metod som läger till ett möte *om möjligt*. D.v.s. inga dubblebokningar och rummet måste vara tillräckligt stort.

   Klasserna skall vara så icke-muterbara som möjligt och dölja så mycket som möjligt av sin data (information hiding). För full poäng krävs användning av API:er, se Appendix.

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

   - Kompilerar ej (motivera varför!)
   - Körningsfel (motivera varför!)
   - Om inget av ovan, ange vad som skrivs ut.

```
A a = new C(); a.doIt(1.0);      // a
IA ia = new B(); ia.doIt(5.5);   // b
C c = new D(); c.doOther(1.0);   // c
B b = new B(); b.doIt(6.6);      // d
IA ia = new B(); IX ix = (IX) ia; ix.doOther(3);  // e

B[] bs = {new B(), new B()}; // f
A[] as = bs;
as[0] = new A();
as[0].doIt(3.4);

C c = new D(); IA ia = (IA) c; ia.doIt(4.5); // g

List<B> bs = new ArrayList<>();  // h
List<A> as = (List<A>) bs;
as.get(0).doIt(1.2);

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

APPENDIX

```
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
- endsWith(s), sant om strängen avslutas med s.
Ur klassen 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.

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

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.
```