Tentamen i Grundläggande Programvaruutveckling, TDA548

Joachim von Hacht

Datum: 2019-08-19

Tid: 14.00-18.00

Hjälpmedel: Lexikon Engelskt-Valfritt språk.

Betygsgränser:

U: -23

3: 24-37

4: 38-47

5: 48-60 (max 60)

Lärare: Joachim von Hacht. Någon besöker ca 15.00 och 16.30, tel. 031/7721003

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. Vad avses med: 4p

   a) inkrementering?

   b) call by value (värdeanrop).

   Förklara 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 c = 0;
double fst = 1;
double sec = 2;
while (true) {
    if (fst == sec) {
        break;
    }
    fst = fst + 0.1;
    c++;
}
out.println(c);
```

3. Vi vill att en given array, *arr*, skall permuteras utifrån de index som finns i en annan given array, *order*. D.v.s element 0 i *arr* skall hamna på index *order*[0] o.s.v. Exempel: 6p

```
arr = [1, 2, 3, 4, 5]

order               arr efter permutationen
[0, 4, 1, 3, 2]     [1, 3, 5, 4, 2]
[4, 3, 1, 0, 2]     [4, 3, 5, 2, 1]
[1, 0, 4, 2, 3]     [2, 1, 4, 5, 3]
```

   Implementera en metod void reorder(int[] order, int[] arr) för detta. Ingen felhantering krävs, vi förutsätter att båda arrayer är korrekta.

4. Skriv en metod som avgör om en given lösning till ett sudoku, i form av en matris av heltal, är korrekt. Ett exempel på en korrekt lösning av ett sudoku är : 12p

```
int[][] sudoku = {
    {4, 8, 3, 9, 2, 1, 6, 5, 7},
    {9, 6, 7, 3, 4, 5, 8, 2, 1},
    {2, 5, 1, 8, 7, 6, 4, 9, 3},
    {5, 4, 8, 1, 3, 2, 9, 7, 6},
    {7, 2, 9, 5, 6, 4, 1, 3, 8},
    {1, 3, 6, 7, 9, 8, 2, 4, 5},
    {3, 7, 2, 6, 8, 9, 5, 1, 4},
    {8, 1, 4, 2, 5, 3, 7, 6, 9},
    {6, 9, 5, 4, 1, 7, 3, 8, 2}
};
```

För ett lösningen skall vara korrekt måste varje rad, kolumn och varje icke överlappande delmatriser av storlek 3x3 innehålla samtliga av siffrorna 1-9 utplacerade med en siffra för varje position (delmatrisernas översta vänstra element börjar på multipler av 3) . För att markera tom plats används siffran 0. För full poäng krävs funktionell nedbrytning. Du får använda följande givna kod (d.v.s. anropa metoden i din kod).

```
boolean isValidSudokuCol(int col, int[][] sudoku) {
    int sum = 0;
    for (int r = 0; r < 9; r++) {
        sum += sudoku[r][col];
    }
    return sum == 45;
}
```

5. En rotation av ett ord åstadkoms genom att flytta en bokstav från början av ordet till slutet. En lexikografisk jämförelse av två ord innebär att orden jämförs bokstav för bokstav från vänster till höger utifrån placering i alfabetet t.ex.

   ```
   aa = aa, aa < ab (b kommer efter a), ac > ab (b kommer före c)
   ```

   Ett Lyndon-ord är det minsta ordet i lexikografisk ordning bland alla rotationer av ett ord[1]. Exempelvis: 8p

   ```
   ord    rotationer                          Lyndon-ord
   aba    [baa, aab, aba]                     aab
   bbaba  [babab, ababb, babba, abbab, bbaba] ababb
   ```

   Skriv en metod som givet en sträng returnerar Lyndon-ordet för strängen. För full poäng krävs funktionell nedbrytning. Alla metoder i Appendix är tillåtna.

[1] Något förenklat för att passa på en tenta.

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

```
X x1 = new X(new Y(), 2);
X x2 = new X(new Y(), 3);  // Before
doIt(x1, x2);      // Call
                // After

void doIt(X x1, X x2) {
    x1.get().set(x2);
    x1.get().get().set(9);
}

class X {
    int i;
    Y y;
    public X(Y y, int i) {
        this.i = i;
        this.y = y;
        y.set(this);
    }
    Y get() { return y; }
    void set(int i) { this.i = i;}
}

class Y {
    X x;
    void set(X x) { this.x = x; }
    X get() { return x; }
}
```

7. Vi skall skapa en objektorienterad modell av en restaurang (t.ex. bokning av bord). Klasserna nedan är givna och skall användas: 12p

```
public class Customer {
    private final String name;
    private final String phone;
    // Ssetters/getters, equals/hashCode ommited
}
class TimeSlot {
    boolean overlaps(TimeSlot other) {
        // True if this and other overlaps in time
    }
    // Setters/getters, equals/hashCode ommited
}
```

a) Skriv en klass Table för ett bokningsbart bord. Ett bord har ett id och ett antal platser. All data skall anges då man skapar ett bordobjekt. Vi antar att alla setters/getters/equals/hashCode vi ev. behöver finns (gäller även nedan).

b) Skriv en klass Booking för en bokning. En bokning har en tid (TimeSlot), en kund och ett bord. All data anges då bokningen skapas.

c) Skriv en klass Restaurant för hela restaurangen. En restaurang har ett antal bord och ett antal bokningar. Lägg till en metod bookTable som givet en kund, ett antal personer och en tid, skapar en ny bokning. Om det lyckas returnerar metoden true annars false. En kund får bara göra en bokning och bordet som bokas måste vara ledigt och rymma alla. Alla metoder i appendix är tillåtna.

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

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

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

```
a) IX ix = new C(); IY iy = (IY) ix; iy.doIt(new B());
b) A a = new B(); a.doIt(1);
c) B b = new C(); b.doIt(1);
d) A a = new B(); IX ix = (IX) a; C c = (C) ix; c.doIt(1.0);
e) C c = new C(); c.doIt(new B());
f) IY iy = new C(); iy.doIt(iy);
g) List<A> as = new ArrayList<>(); List<B> bs = as; bs.get(0).doIt(5);
h) B[] bs = new B[2]; bs[0] = (B) new C();

// --- Interfaces and classes
interface IX { void doIt(double d); }
interface IY { void doIt(IX ix); }
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 implements IY{
    public void doIt(IX ix) {
        out.println("doIt C");
    }
}
```

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
- endsWith(s), sant om strängen avslutas med s.
- s1.compareTo(s2), ger -1, 0 eller 1 om s1 är respektive mindre, lika med eller stör lexikografisk ordning

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

Klassen Random med metoden nextInt() är alltid tillåten.