Tentamen i Grundläggande programvaruutveckling, TDA548

Joachim von Hacht & Magnus Myreen

Datum: 2016-10-27

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: Joachim von Hacht/Magnus Myreen. Någon besöker ca 10.00 och 11.30, tel. 031/7721003

Granskning: Anslås på kursida.

Instruktioner:

- För full poäng på essä-frågor 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 är aldrig någon risk att vara övertydlig!
- Det räcker med enbart relevanta kodavsnitt, övrig kod ersätts med "..." (aldrig import, main-metod, etc....)
- Överkomplicerade lösningar kan ge poängavdrag.
- 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.

LYCKA TILL...

1. Vad avses med (förklara med en eller ett par meningar, du får gärna förtydliga med en skiss eller med kod)? 4p

   a) Tilldelning.

   b) Returtyp.

2. Vilka av de markerade raderna (1-10) i koden nedan kompilerar inte? Du måste ange rad och motivering (alltså varför raden inte kompilerar)! 3p

```
{
    int x = 0;            // 1
    if (x == 4) {
        int x = 0;        // 2
        int y = 0;        // 3
        out.println(x);   // 4
        out.println(y);   // 5
    }
    int x = 0;           // 6
    int y = 0;           // 7
    out.println(x);      // 8
    out.println(y);      // 9
}
out.println(x); // 10
```

3. Skriv en metod som givet en array av heltal returnerar en array med bara de jämna talen ur arryen. Orginal-arrayen får inte ändras. För full poäng krävs att du använder funktionell nedbrytning (d.v.s. hjälpmetod(er) används). Exempel: 4p

```
Indata:              Utdata:
[ 1,4,3,6,6,1,9 ]    [ 4,6,6 ]
[ 2,2,2 ]            [ 2,2,2 ]
[ 1,1,1 ]            []
```

4. Vad skrivs ut i koden nedan. Förklara resultatet genom att rita en bild som visar variabler och referenser (lådor och pilar). Rita antingen som en tecknad serie eller numrera så att tidsföljden blir klar. 4p

```
H h1 = new H(1);
H h2 = new H(2);
H h3 = new H(3);
swap(h1, h2);
swap(h2, h3);
out.println(h1.i + " " + h2.i + " " + h3.i);  // Prints??

void swap(H h1, H h2) {
    H tmp = h1;
    h1 = h2;
    h2 = tmp;
}

class H {
    int i;
    H(int i) { this.i = i; }
}
```

5. Skriv en metod som gör om en matris (2D-arrays) rader till kolumner. Metoden skall ändra orginalmatrisen. Exempel: 6p

```
Indata:              Utdata:
[ [ 1,2,3 ],         [ [ 1,4,7 ]
  [ 4,5,6 ],           [ 2,5,8 ]
  [ 7,8,9 ] ]          [ 3,6,9 ] ]
```

6. Skriv en klass för 2D vektorer enlig specifikationen a) och b). 7p

   a) Följande konstruktorer skall finnas: En förvald (parameterlös) konstruktor (initierar vektorn till (0,0)), en konstruktor som initerar en godtycklig vektor, en kopieringskonstruktor.

   b) Följande metoder skall finnas: add (adderar aktuell vektor med annan vektor), length (ger vektorn längd, metoden sqrt ur Math får användas), equals (ger sant om två vektorer är lika annars falskt). Du får själv göra en rimlig definition likhet.

7. Skriv en metod boolean isAnagram(String word, String ag) som avgör om strängen ag är ett anagram av strängen word. Ett anagram är en omkastning av bokstäverna i ett ord. Exempel: 8p

```
out.println( isAnagram("word", "wrdo")); // All should print true
out.println( isAnagram("boat", "btoa"));
out.println( ! isAnagram("pure", "in"));
out.println( ! isAnagram("fill", "fil"));
out.println( ! isAnagram("b", "bbb"));
out.println( isAnagram("a", "a"));
```

Tillåtna metoder från String-klassen

```
- 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
- split(String str), delar upp en sträng i en array av delsträngar
  utifrån ett visst tecken. Returnerar en String-array (String[])
  Exempel "aaa:bb:cccc:dd".split(":") -> [ "aaa", "bb", "cccc", "dd" ]
```

8. Vilka av de markerade raderna (1-12) nedan kompilerer inte. Du måste ange rad och motivering! 4p

```
A.a = A.b;                    // 1
A.a = (new A()).b;            // 2
A.b = A.a;                    // 3
(new A()).b = A.a;            // 4
(new A()).a = (new A()).b;    // 5
(new A()).b = (new A()).a;    // 6
A a = A.doIt();               // 7
out.println(a.a);             // 8

class A {
    static int a;
    int b;
    int doIt() {            // 9
       return a;            // 10
    }
    static int doOther() {  // 11
       return b;            // 12
    }
}
```

9. Gränssnittet (interfacet) `Func` deklarerar en metod som, givet en `int`, utför en beräkning och returnerar en `int`. Din uppgift är att implementera en klass `MemoFunc` som kan användas som en optimering runt ett `Func` objekt. Idén är att ett `MemoFunc`-objekt beter sig precis som `Func`-objektet som det innehåller, men `MemoFunc`-objekt kommer också ihåg de snaste 100 beräkningarna. Detta betyder att, om man anropar ett `MemoFunc`-objekt med ett värde som den kommer ihåg, då returnerar den det tidigare resultatet utan att köra beräkningen igen på det interna `Func`-objektet. 8p

```
public interface Func {
    public int compute(int n);
}

// test kod för MemoFunc
Func f = new Func() {
    public int compute(int n) { return fib(n); }
};
Func m = new MemoFunc(f);
System.out.println(m.compute(40)); // långsam
System.out.println(m.compute(40)); // snabb!
System.out.println(m.compute(40)); // snabb!
```

10. Koden nedan visar hur en stack för integers, `IntStack`, kan användas.

```
IntStack s1,s2,s3;
s1 = new IntStack();  // skapar en tom stack
s1 = s1.push(3);
s1 = s1.push(5);
System.out.println(s1.top());  // skriver ut 5
s2 = s1.pop();
s3 = s1.pop();
System.out.println(s1.top());  // skriver ut 5
System.out.println(s2.top());  // skriver ut 3
System.out.println(s3.top());  // skriver ut 3
System.out.println(s3.pop().top());  // kastar en exception
```

   a) `IntStack`-objekt är immutable. Förklara vad immutable innebär och hur man ser att `IntStack`-objekt är immutable genom koden och kommentarerna ovan. 2p

   b) Implementera `IntStack` klassen, inklusive metoderna `push`, `top` och `pop`. Metoden `top` returnerar värdet som en `pop` skulle ta bort. Ett anrop till `pop` eller `top` på en tom stack ska kasta en exception. Din klass bör vara immutable och koden måste undvika *onödig* kopiering. 10p

   Obs: För >5 poäng måste man undvika all kopiering som går att undvika.

   Tips: Tänk att en stack är antingen tom eller ett element följt av en stack.