Tentamen i Grundläggande Programvaruutveckling, TDA548

Joachim von Hacht

Datum: 2018-10-31

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 17.00, 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) Konstruktor

   b) Instansvariabel respektive lokal variabel.

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

2. Koden nedan har ett problem. Vilket? Motivera! 2p

```
int doIt() {
    int i = 1;
    int j = 5;
    int k = 3;
    while ( true ) {
        if ( i <= j && k <= i && j <= k) {
          break;
        }else {
          k++;
        }
        i++;
        j++;
    }
    return k;
}
```

3. Man kan beräkna värdet av cosinus m.h.a. serien: $cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...$(x i radianer). Skriv en metod som givet en vinkel i grader och ett positivt heltal n, för antal termer som skall tas med, använder serien för att beräkna värdet av cosinus för vinkeln. Omvandling från grader till radianer görs med formeln $rad = deg(\pi/180)$. 6p

4. Givet en rektangulär matris med heltal. Skriv en metod som returnerar en array med de tal som finns på alla rader. Metoden skall använda arrayer (inte List, ArrayList, etc.). För full poäng krävs en lämplig funktionell nedbrytning. Skriv en kort kommentar vid varje metod, vad är syftet med metoden? Indata och utdata? Exempel: 10p

```
Matris              Värden som finns på alla rader
[1, 2, 3]           []
[4, 5, 6]
[7, 8, 9]

[2, 1, 4, 3]        [2, 3]
[1, 2, 3, 2]
[3, 6, 2, 3]
[5, 2, 5, 3]

[12, 1, 14, 3, 16]   [1, 14, 3]
[14, 2, 1, 3, 35 ]
[14, 1, 14, 3, 11]
[14, 25, 3, 2, 1 ]
[1, 18, 3, 21, 14]
```

5. Vi definierar ett "speciellt" ord som ett ord som börjar på en vokal och slutar på en konsonant eller tvärtom. Skriv en metod som givet en sträng returnerar en lista med alla substrängar till strängen som är speciella. Alla metoder i appendix får användas. För full poäng krävs funktionell nedbrytning. Exempel: 10p

```
Sträng  Substrängar                      Specialla Substrängar
me      [m, me, e]                       [me]
salt    [s, sa, sal, salt, a, al,        [sa, al, alt]
                     alt, l, lt, t]
union   [u, un, uni, unio, union,        [un, union, ni,
           n, ni, nio, nion, i,               nio, ion, on]
              io, ion, o, on, n]
```

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! 8p

```
A[] as = {new A("111"), new A("222")};
B b = new B(as, "bbb");     // Before
as = doIt(b);               // Call
                            // After

A[] doIt(B b) {
    A tmp = b.as[0];
    b.as[0] = b.as[1];
    b.as[1] = tmp;
    return b.as;
}
class B {
    A[] as;
    String s;
    public B(A[] as, String s) {
        this.as = as;
        this.s = s;
    }
}
class A {
    String s;
    public A(String s) {
        this.s = s;
    }
}
```

7. Vi skall skriva ett program för bokningar av flygresor. 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). Ni måste använda de givna klasserna nedan. 12p

   a) Skriv en klass, Flight, som representerar en flygning. En flygning har en flygplats (för avgång), en destination, en tid (datum/tid) då planet avgår och en lista med stolar (Seat). Alla värden utom stolar sätts då man skapar en flygning. Inga setters/getters behöver anges, vi antar att dessa finns och går att använda. Lägg dessutom till en metod getFreeSeat som returnerar någon ledig stol om en sådan finns (d.v.s. stolen är FREE). Om ej returneras null.

   b) Skriv en klass Booking som representerar en flygbokning. En bokning består av en flygning, en stol (platsen för passageraren) och en passagerare. Allt sätts då bokningen skapas.

   c) Skriv en klass Broker (mäklare) som förmedlar bokningar av flyg. En mäklare har en lista med flygningar och en lista med bokningar. Lägg till en metod bookFlight som givet en flygplats, en destination, en tid och en passagerare skapar en bokning för detta om det finns ett passande flyg med någon ledig stol. Om så läggs bokningen till i listan med bokningar och metoden returnerar true. Går det inte att boka returnerar metoden false.

```
enum Status { FREE, OCCUPIED, RESERVED }

public class Seat {
    Status status;
    public Status getStatus() { return status; }
}
public class Passenger {
    private final String passport;
    private final String name;
    public Passenger(String passport, String name) {
        this.passport = passport;
        this.name = name;
    }
        // setters/getters omitted
}
```

8. Ange för varje rad a) - g) nedan om raden ger kompileringsfel eller körningsfel eller är korrekt. Du måste motivera eventuella fel du angivit! Om raden är korrekt måste du ange vad som skrivs ut. 8p

```
a) A a1 = new C(); a1.doA();
b) B b1 = new C(); b1.doA();
c) IA ia2 = new C(); IB ib2 = (IB) ia2; ib2.doB();
d) IB ib4 = new C(); C c = (C) ib4; IA ia4 = c; ia4.doA();
e) C c1 = (C) new B(); c1.doC();
f) A a2 = new B(); C c2 = (C) a2; c2.doA();
g) IA ia1 = new C(); IB ib1 = ia1; ia1.doA();

public interface IA { void doA(); }
public interface IB { void doB(); }

public class A implements IA {
    public void doA() { out.println("A doA");}
}
public class B extends A {
    public void doA() { out.println("B doA");}
}
public class C extends A implements IB {
    public void doB() { out.println("C doB");}
    public void doC() { out.println("C doC");}
}
```

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
```