Tentamen i Grundläggande Programvaruutveckling, TDA545/548

Joachim von Hacht

Datum: 2017-12-19

Tid: 08.30-12.30

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 10.00 och 11.30, 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) Tilldelning

   b) Omslagstyp

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

2. Koden nedan kompilerar inte. Motivera varför! 2p

```
int dubbleOdd (int i){
    if(i % 2 != 0){
       return 2 * i;
    }
}
```

3. Vi antar att vi bara kan överföra beloppen 500kr, 200kr eller 100kr per insättning till ett konto. Nedan visas hur många överföringar som minst måste göras för att komma så nära som möjligt, men inte understiga, ett visst positivt belopp: 6p

```
Belopp          Antal överföringar
   1                  1      (100)
 800                  3      (500+200+100)
 900                  3      (500+200+200)
1800                  5      (500+500+500+200+100)
2321                  6      (500+500+500+500+200+200)
```

   Skriv en metod som givet ett belopp returnerar minsta antal överföringar som måste göras för att överföra beloppet till kontot.

4. Givet en kvadratisk matris med m element och ett positivt heltal $n \leq \sqrt{m}$. Skriv en metod som returnerar maximala summan för talen i någon kvadratisk delmatris med sidan n. Metoden skall använda arrayer för matriser. För full poäng krävs en lämplig funktionell nedbrytning. Exempel: 12p

```
          [-1, 2, 1]
matrix:   [0,  4, 1]   (m = 9)
          [1,  5, 3]

n = 1 ger max 5    n = 2 ger max 13     n = 3 ger max 16
(delmatrisen [5])  (delmatrisen [4, 1]  (delmatrisen är hela matrisen)
                                [5, 3])
```

5. Många websidor har en sträng för att underlätta navigering, vanligen högst upp och/eller längst ner på sidan. Strängen visar aktuell sida och totalt antal sidor. Om man kan navigera till föregående eller efterföljande sida visas dessutom strängen "prev" respektive "next". Exempel: 6p

```
Antal sidor   Aktuell sida       Sträng på sida
     1              1           "[1]"
     3              1           "[1] 2 3 next"
     3              2           "prev 1 [2] 3 next"
     3              3           "prev 1 2 [3]"
     7              4           "prev 1 2 3 [4] 5 6 7 next"
```

Skriv en metod som givet antal sidor och aktuell sida returnerar en sträng enlig ovan.

Tillåtna metoder från olika klasser:

```
String
- 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" ]
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.
Character
- isDigit() (klassmetod). Metoden returnerar sant för tecknen 0-9.
Integer
- valueOf(String s)(klassmetod). Metoden omvandlar en sträng till ett heltal.
  OBS! Att metoden kastar ett undantag för tomma strängen
```

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 föreläsningarna, lådor, pilar o.s.v. Ni *måste* rita! 7p

```
A a = new A(new A[]{new A(-1), new A(1)}, 0);  // Before
doIt(a);     // Call
             // After

void doIt(A a) {
    A[] tmp = new A[2];
    a.as[0].as = tmp;
    tmp[1] = a;
}

class A {
    int i;
    A[] as;
    A(A[] as, int i) { this.i = i; this.as = as;}
    A(int i) {this.i = i;}
}
```

7. Vi skall skriva ett epostprogram. 8p

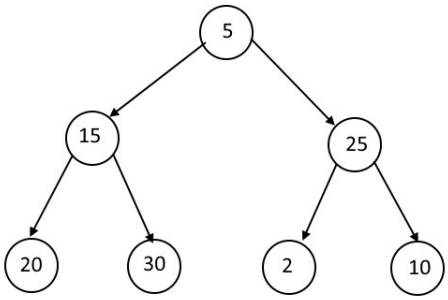
   a) Skriv en klass Mail för ebrev. Ett brev har ett id, en sändare, en mottagare och ett innehåll (en text). Alla värden skall kunna sättas då man skapar ett brevobjekt. Lägg till de set/get metoder som kan behövas i b).

   b) Skriv en klass MailProgram. Klassen skall ha en inbox, för inkomna brev och en outbox för skickade brev, båda i form av listor av Mail. Klassen skall ha en metod forward(id, receiver). Metoden söker efter brevet med givet id i inbox:en. Om detta hittas skapas en kopia av brevet förutom att mottagaren sätts till receiver. Kopian sparas därefter i outbox:en (hur själva sändningen går till bekymrar vi oss inte för).

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

```
Tillåtet att använda List/ArrayList med metoderna
- 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
- contains(o), sant om objektet o finns i listan.
- indexOf(o), ger index för objektet
- size(), ger längden på listan
```

8. Ett binärt träd är en datastruktur enligt bilden (ett upp och nedvänd träd som alltid grenar sig i två). 9p



Trädet i bilden har höjden 2. Antal noder (ringar) i trädet ges av $2^{höjd+1} - 1$. I noderna kan man spara heltal. Din uppgiften är att skapa en klass för binära träd. Klassen skall internt använda en array för att spara heltalsvärdena (representera trädet). Värdena i trädet i bilden skulle i array-format se ut som: [5, 15, 25, 20, 30, 2, 10]. Exempel på metoder som skall finnas i klassen och effekten av dessa (trädet/arrayen visas till höger, alla värden sätts till 0 då trädet skapas):

```
// Create empty tree with height 2
BinTree b = new BinTree(2);          [0, 0, 0, 0, 0, 0, 0]
b.add(1);                            [1, 0, 0, 0, 0, 0, 0]
b.left().add(2);                     [1, 2, 0, 0, 0, 0, 0]
b.up().right().add(3);               [1, 2, 3, 0, 0, 0, 0]
b.up().add(7);                       [7, 2, 3, 0, 0, 0, 0]
b.left().right().add(9);             [7, 2, 3, 0, 9, 0, 0]
```

Metoden pow från Math får användas.

9. Betrakta koden nedan. 6p

   a) Ange för varje rad om raden kompilerar eller inte. Du måste motivera!

   b) Kommer någon rad att ge undantag vid körning? Om så, motivera varför!

   c) Om du åtgärdar felen, så att programmet blir körbart, vad kommer i så fall att skrivas ut? Motivera!

```
Object o = new int[3];               // 1
out.println(o[0]);                   // 2
int[] i = o;                         // 3
int[] j = (int[]) o;                 // 4
out.println(j[0]);                   // 5
out.println(o == new int[3]);        // 6
out.println(o.equals(new int[3]));   // 7
String s1 = o;                       // 8
String s2 = (String) o;              // 9
```