Demo 9 / 16.3

Tämän demon tehtävät 1-4 on mukaeltu aikaisempien vuosien välikokeista. Myös muut tehtävät (erityisesti B1-2) voisivat olla koetehtäviä. Tämän kerran kokeeseen tulee luonnollisesti jotakin testaamiseen liittyvääkin. Taas T:llä merkitty ne joissa ComTest tai JUnit-testi on pakko kirjoittaa. Vähintään kahteen palautettuun tehtävään pitää olla tehtynä testit jotta palautus hyväksytään.

Tietorakenteet voivat olla esim. samanlaisia kuin tehtävän 7 `Taulukko.java` -tehtävässä. Yleensä pyrin laittamaan ainakin yhden "omatoimista soveltamista" mittaavan tehtävän, eli vähän erilaisen kuin näissä harjoituksissa - kuitenkin kurssin sisältöön liittyvän. Muistakaa se yhden A4:n lunttilappu. Esim. tiedostojen avaamisrutiinit kannattaa kirjoittaa siihen.

1. Mitä seuraava ohjelma tulostaa? Tutki PÖYTÄTESTIN avulla! Merkitse pöytätestiin myös milloin mikäkin olio muuttuu roskaksi. Pohjana voi käyttää poyta03tyhja.xls tai poyta03tyhja.html (6p)

```
/**
 * Tutki päytätestin avulla mitä ohjelma tulostaa.
 * Merkitse myös milloin mikäkin olio muuttuu "roskaksi".
 * @author Vesa Lappalainen
 * @version 1.0, 09.03.2003
 */
public class VKseko02 {

  /* 09 */ public static class Sorsa {
  /* 10 */   private int vari;
  /* 11 */   private String nimi;
  /* 12 */
  /* 13 */   public Sorsa() { nimi = "Repe"; vari = 0; }
  /* 14 */   public Sorsa(StringBuffer inimi) {
  /* 15 */     inimi.append("Aku"); vari = 2;
  /* 16 */   }
  /* 17 */
  /* 18 */
  /* 19 */
  /* 20 */   void hyppaa(Sorsa apu) {
  /* 21 */     apu.nimi = "Senkin Sorsa";
  /* 22 */     vari++;
  /* 23 */   }
  /* 24 */   void tulosta() {
  /* 25 */     vari++;
  /* 26 */     System.out.println(nimi + " " + vari);
  /* 27 */   }
  /* 28 */ }
  /* 29 */
  /* 30 */ public static void main(String[] args)
  /* 31 */ {
  /* 32 */   StringBuffer nimi = new StringBuffer("Väiski");
  /* 33 */   Sorsa aku = new Sorsa(nimi);
  /* 34 */   System.out.println(nimi);
  /* 35 */   aku.tulosta();
  /* 36 */   {
  /* 37 */     Sorsa repe = new Sorsa();
```

```
  /* 38 */     repe.tulosta();
  /* 39 */     aku.hyppaa(repe);
  /* 40 */     aku.tulosta();
  /* 41 */     aku = repe;
  /* 42 */   }
  /* 43 */   aku.tulosta();
  /* 44 */ }
}
```

2. Kirjoita funktio `pisinNouseva`, joka palauttaa Javan merkkijonosta pisimmän pelkästään kasvavan (samoja tai aakkosissa aina edeltäjäänsä "suurempia" merkkejä sisältävän) merkkijoukon pituuden

```
Esimerkki:
  "abajiuxac"   => 3  (eli iux)
  "kissa"       => 3  (eli iss)
```

a) Kirjoita ensin algoritmi, saat jakaa ongelman pienempiinkin osiin :-). (2p)

b) Toteuta em. algoritmi Java -kielellä. (T) (4p)

3. Kirjoita ohjelma (kieli vapaa), joka lukee tiedostoa, jossa on välilyönnein toisistaan erotettuja lukuja ja sanoja, ja kopioi toiseen tiedostoon ne rivit, joiden alussa on luku, joka on suurempi kuin 30. (T, tällä kertaa riittää hahmotelma) (6p)

```
Tiedostosta:             tulee tiedosto:
-----------------------------------------------------
33 hiljaa 1 hiipii       33 hiljaa 1 hiipii
hyvä 33 tulee            36 1 3 5 55
36 1 3 5 55
nyt 33 riittää
-----------------------------------------------------
```

4. Kirjoita luokka `Ehdokas`, jota voidaan käyttää kuten seuraavassa pääohjelmassa. Kirjoitettava koko luokka (attribuutit, kaikki metodit yms.) (T) (6p)

```
public static void main(String[] args)  {
  Ehdokas ehd1 = new Ehdokas(100000.0,0),
          ehd2 = new Ehdokas(20000.0,300);
  ehd1.tulosta();  // Tulostaa: Rahaa 100000, ääniä 0
  ehd2.tulosta();  // Tulostaa: Rahaa 20000, ääniä 300
  ehd1.osta(200,100.0);  // Ostaa 200 ääntä, 100 mk/kpl
  ehd1.tulosta();  // Tulostaa: Rahaa 80000, ääniä 200
  boolean onnistui = ehd2.osta(300,100);
  if ( !onnistui  ) System.out.println("Rahat ei riitä :-)");
  ehd2.tulosta();  // Tulostaa: Rahaa 20000, ääniä 300
  if ( ehd1.compareTo(ehd2) > 0 )
    System.out.println("Ehdokas 1 voitti!");
  if ( ehd1.compareTo(ehd2) < 0 )
    System.out.println("Ehdokas 2 voitti!");
  if ( ehd1.compareTo(ehd2) == 0 )
    System.out.println("Taspeli!");
  // Vertailu tehdään äänimäärien perusteella.
  // Esimerkissä tulostuu : Ehdokas 2 voitti!
}
```

5. Javan `Integer` -luokka on muuttumaton (immutable), eli kun kokonaisluku on luotu, sitä ei voi enää muuttaa. Tee oma luokka `Int`, jonka arvo voi muuttaakin (mutable) ja muuta esimerkki dyna/ArrayListMalliGen.java sellaiseksi, että tietorakenteessa oleva luku voidaan muuttaa kutsulla:

```
luvut.get(1).set(4);
```

Luokkaan `Int` pitää tehdä sen verran metodeja, että esimerkki ArrayListMalliGen.java toimii kun kaikki `Integer` sanat muutetaan `Int`. Voit jättää pois tulosta metodin ja sen kutsun. Koska autoboxing ei toimi, täytyy lisäykset tehdä muodossa

```
luvut.add(new Int(0));                    (T)
```

6* Tee aliohjelma

```
int poista(int taulukko[],int lkm,int n)
```

joka "muodollisesti" poistaa taulukosta kaikki luvun `n` esiintymät. Eli poiston jälkeen taulukossa ei ole yhtään lukua `n`. Oikeasti taulukkoa ei voi pienentää, mutta alkioita siirretään alkuun päin ja ilmoitetaan "virallisten" alkioiden `lkm`. Seuraavan esimerkin taulukossa on "poiston" jälkeenkin 6 kokonaislukua, mutta niistä saa käyttää vain 4:ää ensimmäistä. (T)

```
int t[]={4,7,9,3,9,2};
int lkm=6;

lkm = poista(t,lkm,9);  /* => t = {4,7,3,2}, lkm = 4  */
```

7*. Kirjoita dyna/Taulukko.java (ks. moniste 16.3) taulukolle metodi `public Taulukko clone()`, jolla voidaan luoda taulukosta identtisesti käyttäytyvä kopio (T):

```
...
public static void main(String args[]) {
  Taulukko luvut = new Taulukko(7);
  luvut.lisaa(0); luvut.lisaa(2);
  System.out.println(luvut); // Tulostaa " 0 2"
  Taulukko taul = luvut.clone();
  luvut.lisaa(77);
  System.out.println(taul);  // tulostaa saman kuin edellä " 0 2"
}
```

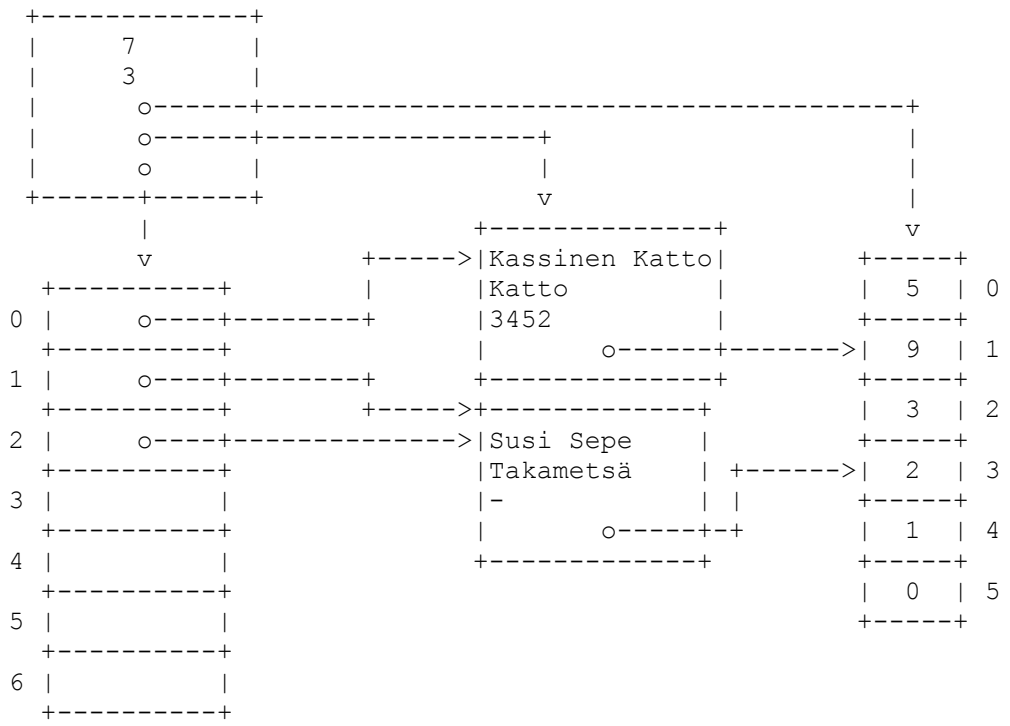
8. Kirjoita `Taulukko` -luokalle 6. tehtävää vastaava metodi `poista`. Mitä parametreja tämä metodi tarvitsee? (T)

9*. Pääteohjauksessa 2 piti kirjoittaa seuraava tiedosto:

```
000 En enää turhaan kirjoita!
001 En enää turhaan kirjoita!
002 En enää turhaan kirjoita!
003 En enää turhaan kirjoita!
...
099 En enää turhaan kirjoita!
```

Tee nyt ohjelma, joka kirjoittaa tiedoston puolestasi. Mitä ohjelma voisi kysellä käyttäjältä? (T, riittää hahmotelma testaamisesta)

B1-2 Kirjoita Java-kieliset luokkamäärittelyt, joilla saat loogisesti seuraavan näköiset rakenteet sekä kirjoita vastaavien muuttujien määrittelyt ja metodikutsut jotta rakenteessa olisi samat tiedot kuin kuvassa:



B3. Ota Pylvas.java ja peri siitä luokka `PylvasPros`, jolla toimii seuraava testiohjelma:

```
public static void main(String[] args) {
    PylvasPros p5 = new PylvasPros(1,5,0);
    PylvasPros p8 = new PylvasPros(1,8,2);

    Window window = new Window();
    window.scale(0,0,5,8);
    p8.move(2,0,0);
    p8.setValiColor(Color.RED);

    window.add(p8);
    window.add(p5);
    window.showWindow();

    Syotto.kysy("Asettaa 5 pylvään välin");      p5.setValiPros(80);
    Syotto.kysy("Pienennä 8 pylvään korkeutta"); p8.setKorkeus(6);
    p8.setLukittu(true); // seuraavassa väli muuttuu 2:sta 1:ksi.
    Syotto.kysy("Pienennä 8 pylvään korkeutta"); p8.setKorkeus(3);
}
```

Myös seuraavan testin tulee toimia:

```
/**
 * Asettaa pylvään korkeuden.  Mikäli lukittu, muuttaa
 * välin tilannettakin.
 * @param korkeus pyvään korkeus
```

```
 *
 * @example
 * <pre name="test">
 *   PylvasPros p = new PylvasPros(1,10,5);
 *   p.getKorkeus()  ~~~ 10.0;
 *   p.getVali()     ~~~  5.0;
 *   p.setKorkeus(20.0);
 *   p.getKorkeus()  ~~~ 20.0;
 *   p.getVali()     ~~~  5.0;
 *   p.getValiPros() ~~~ 25.0;
 *   p.setLukittu(true);
 *   p.setKorkeus(10.0);
 *   p.getVali()     ~~~  2.5;
 * </pre>
 */
@Override
public void setKorkeus(double korkeus) {...}
```

B4-5. Tehtävien 3 ja 9 testit toteutettuna.

Välikokeet 2003 - 2007

Treenausta varten vuosien 2003 - 2007 välikokeet ja niiden vastaukset:

http://www.mit.jyu.fi/vesal/kurssit/ohjelmointi2003/tentti/
http://www.mit.jyu.fi/vesal/kurssit/ohjelmointi2004/tentti/
http://www.mit.jyu.fi/vesal/kurssit/ohjelmointi2005/tentti/
http://www.mit.jyu.fi/vesal/kurssit/ohjelmointi2006/tentti/
http://www.mit.jyu.fi/vesal/kurssit/ohjelmointi2007/tentti/