Demo 5 / 15.2

Tehtävät

1*. Muuta edellisen demon tehtävän 7&8 vastausta (Henkilo.java) seuraavasti:

a) Lisää metodi `toString`, joka palauttaa henkilön tiedot tolppa-erotetussa muodossa.

b) Lisää metodi `parse` joka selvittää henkilön tiedot tolppa-erotetusta muodosta.

Toiminta lisäysten jälkeen:

```
hlo.parse("Sepe|Susi|1948");
hlo.tulosta();
System.out.println(hlo);  // kutsuu hlo.toString();

=> tulostaa
   Sepe Susi 1948
   Sepe|Susi|1948
```

c) Voiko nyt `kysy`-metodin testiä tehdä yhtään ”helpommaksi”?

2*. Toteuta luokka, jolla kuvataan päivämäärä. Kirjoita ainakin sopiva muodostaja ja metodi `toString`, jolla päivämäärä saadaan merkkijonoksi. Luonnollisesti testipääohjelma.

3-4. Toteuta `Vali` -luokka, joka tallettaa suljetun reaalilukuvälin (päätepisteet ei-negatiivisia reaalilukuja). Kirjoita metodit `kysy` ja `comapreTo(vali)`. `kysy` kelpuuttaa seuraavat syötöt (0:n ja 5:n tilalla luonnollisesti voitava olla mitä vaan):

```
Anna väli (0-5) >[ret]       => 0-5
Anna väli (0-5) >3[ret}      => 3-3
Anna väli (0-5) >3-[ret]     => 3-5
Anna väli (0-5) >-3[ret]     => 0-3
Anna väli (0-5) >1-3[ret]    => 1-3
```

Testiohjelma kysyy kaksi väliä ja sitten `compareTo` palauttaa tiedon siitä osuuko toinen väli itse olioon. Testiohjelma voisi olla esimerkiksi:

```
public static void main(String[] args)  {
  Vali v1 = new Vali(1,3), v2 = new Vali(2,4);
  v1.kysy(); v2.kysy();
  System.out.println(v1); System.out.println(v2);
  int osuman_laatu = v1.compareTo(v2);
         // vastaa "vähennyslaskua" ol = v1 - v2;
  if ( osuman_laatu == 0 )
    System.out.println("Välit osuvat toisiinsa");
  else if ( osuman_laatu == 1 )
    System.out.println("Jälkimmäisen välin arvot pienempiä kuin " +
                       "ensimmäisen!");
  else if ( osuman_laatu == -1 )
    System.out.println("Jälkimmäisen välin arvot suurempia kuin " +
```

```
                                "ensimmäisen!");
    }
```

Pohdi onko mielekästä, että `compareTo` palauttaa 0 jos välit osuvat toisiinsa. **Vihje:** Piirrä kuva, miten kaksi väliä käyttäytyy toisiinsa nähden.

5. Toteuta luokka `LinjaAuto`, jossa on paikkojen lukumäärä ja vapaiden paikkojen lukumäärä. Tee metodit `tulosta` sekä `lisaa` ja `vahenna` muuttamaan matkustajien lukumäärää. Kirjoita testipääohjelma.

6.* Modifioi edellistä ratkaisua siten, että luokaa `LinjaAuto` voi käyttää seuraavassa testiohjelmassa:

```
public static void main(String[] args)  {
  LinjaAuto pikkubussi = new LinjaAuto(10);
  LinjaAuto isobussi = new LinjaAuto(45);
  pikkubussi.lisaa(4); pikkubussi.tulosta();
  isobussi.lisaa(30);  isobussi.tulosta();
  int yli = pikkubussi.lisaa(15);
  isobussi.lisaa(yli);
  pikkubussi.tulosta(); isobussi.tulosta();
  if ( pikkubussi.getTilaa() > 0 )
    System.out.println("Pieneen bussiin mahtuu!");
  if ( isobussi.tilaa() )
    System.out.println("Isoon bussiin mahtuu!");
}
```

7. Kirjoita yksinkertainen luokka `Tietokone`, jossa on tietokoneelle tarpeellisia attribuutteja (muistin määrä, kovalevyn koko jne..) sekä tarvittavat muodostajat sekä sopivat metodit. Kirjoita myös testipääohjelma.

8. Etsi sopivista lähteistä (esim. luentomoniste) tietoa Javan bittioperaatioista ja selvitä mitä tapahtuu seuraavassa ohjelmanpätkässä (tutki pöytätestillä):

```
/* 01 */  int a=23,b=13,c=17;
/* 02 */  char m = 'b';
/* 03 */  if ( ( a = b ) != 0 ) c+=0x0f;
/* 04 */  if ( ( a & ~b ) != 0 ) c--;
/* 05 */  m ^= 1 << 5;
/* 06 */  if ( m == 'B' ) b &= c;
/* 07 */  System.out.print(
            "a=" + a + " b=" + b + " c=" + c + " m=" +m );
```

B1-2 Muuta Demo 3:n LueUsers.java guru-tehtävän vastaus sellaiseksi, että siinä on luokka `User` ja attribuutteina luokassa on aliohjelman `kasitteRivi` tarvittavat lokaalit muuttujat. Varsinaisen muunnostyön hoitaa metodi `setAsHTMLString` ja tulos saadaan metodilla `getAsListString`. Muut metodit ja konstruktorit yms. saat määritellä itse.

B3. Täydennä luennolla annettu Astia.java niin että se toimii alkukommenteissa olevien määritysten mukaan. Vastaavan Windows-ohjelman löydät n:\kurssit\winohj\moniste\tentit\v00.

Pääohjelma malliksi

```
public static void main(String[] args) {
  Astia2 astiat[] = { new Astia2("ä",100), new Astia2("5",5),
                      new Astia2("8",8) };
  Astia2 ampari = astiat[0];
  ampari.tayta();

  tulosta_ohje(astiat);

  while ( true ) {
    for ( int i=1; i<astiat.length; i++)
      System.out.println(astiat[i].getTilavuus() +
                         " litran astiassa on " +
                         astiat[i].getMaara() + " litraa nestettä");
    String rivi = Syotto.kysy("Mistä kaadetaan ja mihin");
    if ( rivi.length() == 0 ) break;
    StringBuffer sb = new StringBuffer(rivi);
    String mista = Mjonot.erota(sb);
    String mihin = Mjonot.erota(sb);
    int imista = etsi(astiat,mista);
    int imihin = etsi(astiat,mihin);

    if ( ( imista < 0 ) || ( imihin < 0 ) )
       nimi_ohje(astiat,mista, mihin);
    else
       astiat[imista].kaada(astiat[imihin]);
  }
}
```

K1. Miten seuraava pitäisi kirjoittaa jotta koodirivejä tulisi oleellisesti puolet vähemmän (kun parametrinhakurivejä tulee oikeasti kymmeniä):

```
String beginHour = request.getParameter("beginHour");
String endHour = request.getParameter("endHour");
if (beginHour == null) beginHour="";
if (endHour == null) endHour="";
```

G1. Lisää tehtävään 3-4 vielä tarvittavat metodit ja määrittele tyhjä väli, jotta seuraavat kutsut toimivat:

```
Vali v3 = v1.leikkaus(v2);
System.out.println(v3);
if ( v1.leikkaus(v2) == tyhja )
  System.out.println("Välit eivät osu");
```

Voitaisiinko tehdä välien yhdiste ja mitä ongelmia siitä seuraisi?

K-alkuiset tehtävät ovat Korpista otettuja koodinpätkiä, joissa olisi pitänyt osata alunperinkin tehdä paremmin.