# Demo 5 / 14.2

## Tehtävät

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

a) Lisää luokkaan parametritön muodostaja.

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

c) Kirjoita vielä muodostaja jolla on yksi merkkijonoparametri ja muodostaja kutsuu `parse`-metodia.

d) Lisää luokkaan vielä (valmis) metodi:

```
public void tulosta(OutputStream os) {
    PrintStream out = new PrintStream(os);
    out.println(etunimi + " " + sukunimi + " " + syntymavuosi);
}
```

Lisäysten jälkeen seuraavan pääohjelman:

```
public static void main(String[] args) {
    Henkilo hlo = new Henkilo();
    Henkilo aku = new Henkilo("Aku", "Ankka", 1934);
    System.out.println(hlo);
    hlo.parse("Sepe|Susi|1933");
    hlo.tulosta(System.out);
    aku.tulosta(System.out);
    Henkilo mikki = new Henkilo("Mikki|Hiiri");
    System.out.println(mikki);
}
```

pitäisi tulostaa

```
||0
Sepe Susi 1933
Aku Ankka 1934
Mikki|Hiiri|0
```

2*. Kirjoita luokkaan `Pvm` saantimetodit päivälle, kuukausille ja vuosille. Voisi käyttää esimerkiksi:

```
Pvm pvm = new Pvm();
pvm.parse("3.4.2011");
System.out.println(pvm.getPv()); // tulostaa 3
```

3-4. Tee luennolla tehdyn SwingAanestys.java:n tehtävät. Katso kuva millainen ohjelmasta pitäisi tulla.

5. Toteuta luokka `LinjaAuto`, jossa on paikkojen lukumäärä ja vapaiden paikkojen

lukumäärä. Tee metodit `tulosta` (vrt. Henkilo-tehtävä) sekä `lisaa` ja `vahenna` muuttamaan matkustajien lukumäärää. Kirjoita testipääohjelma.

6.* Modifioi edellistä ratkaisua siten, että luokaa `LinjaAuto` voi käyttää seuraavassa testiohjelmassa (tulostukset voivat olla kauniimpia, esimerkit lyhyyden vuoksi):

```
public static void main(String[] args) {
    LinjaAuto pikkubussi = new LinjaAuto(10);
    LinjaAuto isobussi = new LinjaAuto(45);
    pikkubussi.lisaa(4);    pikkubussi.tulosta(System.out); // 10,4,6
    isobussi.lisaa(30);     isobussi.tulosta(System.out);   // 45,30,15
    int yli = pikkubussi.lisaa(15);
    isobussi.lisaa(yli);
    pikkubussi.tulosta(System.out);                         // 10,10,0
    isobussi.tulosta(System.out);                           // 45,39,6
    if ( pikkubussi.getTilaa() > 0 )
        System.out.println("Pieneen bussiin mahtuu!");   // ei tulosta
    if ( isobussi.tilaa() )
        System.out.println("Isoon bussiin mahtuu!");     // tulostaa
    int vajaa = pikkubussi.vahenna(12);                 // vajaa = -2
    if ( vajaa < 0 )
        System.out.println("Pikkubussissa ei edes ole näin montaa!");
    pikkubussi.tulosta(System.out);                         // 10,0,10
}
```

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) {
    Astia astiat[] = { new Astia("ä", 100),
```

```
                        new Astia("5", 5), new Astia("8", 8) };
        Astia ampari = astiat[0];
        ampari.tayta();

        tulostaOhje(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); // NOPMD
            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) )
                nimiOhje(astiat, mista, mihin);
            else astiat[imista].kaada(astiat[imihin]);
        }
    }
```

B4-5. Toteuta `Vali` -luokka, joka tallettaa suljetun reaalilukuvälin (päätepisteet ei-negatiivisia reaalilukuja). Kirjoita metodit `parse` ja `compareTo(vali)`. `parse` kelpuuttaa seuraavat muodot (oletetaan että väli on alustettu ennen jokaista esimerkkiä väliksi `[0,5]`):

```
vali.parse("")       => 0-5
vali.parse("3")      => 3-3
vali.parse("3-")     => 3-5
vali.parse("-3")     => 0-3
vali.parse("1-3")    => 1-3
```

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

```
public static void main(String[] args) {
    Vali v1 = new Vali(1, 3), v2 = new Vali(2, 4);
    System.out.println(v1); // tulostaa (1.0-3.0)
    System.out.println(v2); // tulostaa (2.0-4.0)
    v1.parse("2-");
    v2.parse("2.5-7");
    System.out.println(v1); // tulostaa (2.0-3.0)
    System.out.println(v2); // tulostaa (2.5-7.0)
    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("v1 kokonaan v2:n oikealla puolella");
    else if ( osuman_laatu == -1 )
        System.out.println("v1 kokonaan v2:n vasemmalla puolella");
}
```

Pohdi onko mielekästä, että `compareTo` palauttaa 0 jos välit osuvat toisiinsa. **Vihje**: Piirrä

kuva, miten kaksi väliä käyttäytyy toisiinsa nähden.

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 B4-5 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.