Demo 6 / 20.10

Voit tehdä nyt tehtäviä samaan luokkaan, jos ne liittyvät samaan aihealueeseen. Jos haluat tehdä tehtäviä eri tiedostoon (sekin järkevää), niin voit kutsua toisessa tiedostossa (vaikka `Laskut.java` olevaa metodia `miidi` toisesta tiedostosta seuraavasti:

```
double m1 = Laskut.miidi(luvut);
```

Toinen tapa olisi laittaa alkuun staattinen `import`:

```
import static demo6.Laskut.miidi;
...
double m1 = miidi(luvut);
```

Tehtävät

1. **Ville:** Tee Villellä (ks. https://trac.cc.jyu.fi/projects/ohj1/wiki/ville) tehtävät: ”Kierros 1: Muuttujat ja ehtolauseet”. Muuta tekemiesi tehtävien määrä suhteeksi välille [0,1] ja laita pistemääräksi 0.2 tarkkuudella pyöristetty luku. Esim. jos teit 120/195 tehtävää = 0.615, niin laita 0.6 pistettä. Jos teit 140/195 = 0.72, laita 0.8 tehtävää). Palauta tiedosto `ville.txt`, johon kirjoitat että auttoiko Ville sinua mitenkään asian oppimisen suhteen.

2. **Reaalilukujen vertailu:** Reaalilukuja ei (erittäin harvoja erikoistilanteita lukuun ottamatta) saa verrata == -operaattorilla. Kirjoita reaalilukujen yhtäsuuruusvertailun avuksi funktioaliohjelmat, joita voisi käyttää esimerkiksi seuraavasti:

```
double a = 7.1001;
double b = 7.1002;
double c = 7.2002;

if ( samat(a,b,0.01) )  System.out.println("Ovat melkein samoja");
if ( !samat(a,b) )      System.out.println("Ovat eri suuria");
if ( !samat(a,c,0.01) ) System.out.println("Ovat eri suuria");
if ( samat(a,c,0.2) )   System.out.println("Ovat sinnepäin");
```

Huomaa että toinen kutsu on vähemmillä parametreillä kuin muut (kuormitettu funktio, function overloading). Kommentoi se ensin pois. Muutenkin tee aina ensin aliohjelma sellaiseksi, että se on syntaktisesti oikein, mutta ei tee mitään järkevää. Esimerkiksi tuo kahden parametrin aliohjelma muotoon:

```
public static boolean samat(double a, double b) {
  return false;
}
```

Sitten aja ohjelma ja totea se syntaktisesti oikeaksi. Ja tämän jälkeen pienillä muutoksilla tee siitä kunnolla toimiva. Kiinnitä myös huomiota aliohjelmien kommentoitiin ja muuttujien nimeämiseen. Eclipsen tekemät nimet eivät aina ole kuvaavia.

3a. **Itseisarvo:** Kirjoita ilman minkään valmiin funktion käyttöä funktioaliohjelma `itseisarvo(double luku)`, joka palauttaa luvun aina positiivisena. Aloita kirjoittamalla sopiva testipääohjelma, jossa kutsut funktiota erilaisilla testattavilla arvoilla.

3b. **Etäisyys:** Kirjoita funktioaliohjelma `etaisyys(a,b)`, joka palauttaa kahden reaaliluvun välisen etäisyyden. Esimerkiksi

```
etaisyys(3.2,8.5) on melkein 5.3, samoin etaisyys(8.5,3.2)
```

4. **Silmukat ja taulukot:** Katso Wikipediasta Keskiluvut -kohdasta erilaisia keskilukuja. Yksi on keskiarvoa lähin joukon alkio (josta tässä käytetään nimitystä `miidi`, ei virallinen nimi). Tee funktioaliohjelma `miidi(luvut)` joka palauttaa reaalilukutaulukon lukujen miidin. Et voi käyttää valitettavasti hyväksesi (eli kutsua) edellisen demon `keskiarvo`-funktiota (koska se oli `int` -taulukolle), vaan joudut kopioimaan sen ja muuttamaan taulukon tyypin. Aloita kuitenkin tekemällä tuo reaalilukutaulukon keskiarvon laskeva `keskiarvo`. Voit testata vaikka aineistolla:

```
      double[] luvut = {1,2,3,2,5}; // keskiarvo == 2.6
      double m1 = miidi(luvut);             // 3
      double m2 = miidi(new double[]{1});   // 1
      double m3 = miidi(new double[]{3,3}); // 3
      double m4 = miidi(new double[]{});    // 0
// tulosta m1-m4
```

Vinkki: Unohda aluksi koko Java ja tee kynällä ja paperilla vastaava tehtävä ja mieti vaiheittain mitä joudut tekemään ja mitä ”apumuuttujia” käyttämään.

5a. **Alueen skaalaaminen**: Tee funktioaliohjelma `skaalaa(luku,min,max)`, joka skaalaa välillä `[0,1]` olevan luvun välille `[min,max]`. Esimerkkejä:

```
skaalaa(0.2,-3,3) => -1.8;
skaalaa(0.2,1,6)  => 2.0;
skaalaa(0.0,1,6)  => 1.0;
skaalaa(1.0,1,6)  => 6.0;
```

Vinkki: jos sinulla on luku väliltä `[0,1[` ja haluat saada siitä luvun välille `[a,b[`, niin mieti mitä pitää tehdä jotta 0:sta tulisi `a` ja 1:stä `b`. (eli `f(x) = a + (b-a)*x`)

5b. **Silmukat ja taulukot:** Tee aliohjelma `arvoLuvut(n,min,max)` joka luo n-paikkaisen reaalilukutaulukon ja arpoo sen täyteen lukuja, joiden arvot ovat välillä `[min,max[` (ks. `Math.random`). Testaa aliohjelma vaikka tulostamalla arvottu taulukko.

6. **Geometriaa:** Muistele koulusta, tai katso Wikipediasta, miten laskettiin suorakulmaisen kolmion hypotenuusa Pythagoraan kaavalla. Päättele miten tämän tiedon avulla voit laskea kahden tason pisteen välisen etäisyyden. Kirjoita sitten funktio

```
etaisyys(x1,y1,x2,y2)
```

joka laskee kahden pisteen välisen euklidisen etäisyyden.

B1. **Jonon paloittelu:** Tee funktioaliohjelma `kysyLuvut(kysymys)`, jonka ansiosta seuraava pääohjelma (vinkki `StringTokenizer`, `Mjonot.erotaDouble`):

```
public static void main(String[] args) {
    double luvut[] = kysyLuvut("Anna luvut");
    tulosta(luvut);
}
```

toimii seuraavasti (erottimina välilyönti ja puolipiste):

```
Anna luvut >2 3 4 5 k      9 ;5
 2,00  3,00  4,00  5,00  0,00  9,00  5,00
```

B2. **Matriisit:** Tee funktioaliohjelma, joka etsii 2-ulotteisen reaalilukutaulukon suurimman alkion. Käyttöesimerkki:

```
public static void main(String[] args) {
    double mat1[][] = {{1,2,3},{2,2,2},{4,2,3}};
    double mat2[][] = {{9,2,8},{1,2,5},{3,19,-3}};

    double suurin1 = suurin(mat1);
    double suurin2 = suurin(mat2);
    System.out.printf("%5.2f %5.2f%n",suurin1,suurin2);
}
```

B3. **Reaalilukujen vertailu:** Tehtävässä 2 vertailit ehkä lukujen absoluuttista suuruutta. Kuitenkin esimerkiksi 1000 ja 1100 ovat samoja 10% tarkkuudella, mutta eivät 0.1:n tarkkuudella. Usein voikin jolla järkevä puhua suhteellisesti yhtäsuuruudesta absoluuttisen yhtäsuuruuden sijaan. Kirjoita vielä yksi funktioaliohjelma, jolle pätee:

```
suhtsamat(0.10,0.12,0.1) == false
suhtsamat(0.10,0.11,0.1) == true
suhtsamat(1.0,1.2,0.1)   == false
suhtsamat(1.0,1.1,0.1)   == true
suhtsamat(10,12,0.1)     == false
suhtsamat(10,11,0.1)     == true
suhtsamat(1000,1200,0.1) == false
suhtsamat(1000,1100,0.1) == true
```

GURU-tehtävät

G1. Suunnittele Hirsipuu-peli. Älä toteuta! Suunnittele konsolipohjainen versio ja mieti mitä näyttöön tulostuisi missäkin vaiheessa. Palauta tekstitiedosto, jossa on kuvattu se, mitä näyttöön tulostuu.