

Demo 7 / 27.10

Tämän kerran aiheena on 2-ulotteiset taulukot ja keskipisteet pistejoukossa. Tähän liittyy kuva [pistejoukonkeskipisteet.png](#), jossa on arvottu tasoon joukko pisteitä ja laskettu pistejoukon painopiste (punainen piste), pistejoukon miidi (keskiarvoa lähin joukon alkio, vihreä ympyrä) sekä koordinaateittain laskettu miidi (sininen ympyrä). Voit tehdä nyt tehtävät 2-5 samaan luokkaan, koska ne liittyvät samaan tehtävään. Jos haluat tehdä tehtäviä eri tiedostoon esimerkiksi sen mukaan, että toiset piirtävät ja toiset vain laskevat (sekin järkevää), niin voit kutsua toisessa tiedostossa (vaikka `Laskut.java` olevaa metodia `miidi` toisesta tiedostosta seuraavasti:

```
double ym = Laskut.miidi(pisteet[1]);
```

Toinen tapa olisi laittaa alkuun staattinen `import`:

```
import static demo6.Lskut.miidi;
...
double ym = miidi(pisteet[1]);
```

Tehtävät

- Ville:** Tee Villellä (ks. <https://trac.cc.jyu.fi/projects/ohj1/wiki/ville>) tehtävät: Merkkijonot, Toistorakenteet ja Metodit. 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).
- 2-ulotteinen taulukko:** Esitetään pisteet (eli lukuparit) 2-ulotteisessa taulukossa niin, että 1. rivissä (indeksi 0) on kaikki x-arvot ja 2. rivissä (indeksi 1) kaikki y-arvot. Tällöin $i:n$ pisteen koordinaatit olisivat:

```
x = pisteet[0][i];
y = pisteet[1][i];
```

Tämä ei ole paras mahdollinen tapa esittää lukupareja, mutta tässä vaiheessa elämää kuitenkin käyttökelpoinen. Ongelmana on se, että tuollaisen taulukon jokaisen rivin ei tarvitse olla välttämättä yhtä pitkiä. Ota kuitenkin pohjaksi [LuvutGraafisesti.java](#) ja tee tarvittavat aliohjelmat, jotta seuraava koodi piirtää aiemmin mainitun kuvan mustat ympyrät:

```
public static void main(String[] args) {
    int pisteita = 100;
    double min = 0;
    double max = 10;
    double r = (max-min)/100;

    double pisteet[][] = new double[2][];
    pisteet[0] = arvoLuvut(pisteita, min, max);
    pisteet[1] = arvoLuvut(pisteita, min, max);
    Window ikkuna = new Window(600,400);
    ikkuna.scale(min-1,min-1,max+1,max+1);
    ikkuna.add(new Axis(100,100,0));
    piirraKuva(ikkuna,pisteet,r);
    ikkuna.showWindow();
}
```

```
}

```

Vinkki: Demo6:ssa tehty suuri osa tarvittavista aliohjelmista: [Keskiluku.java](#).

3. **Etsi lähin:** Kirjoita funktioaliohjelma `lahimmanPaikka(pisteet, x, y)`, joka etsii missä kohti `pisteet` -taulukkoa on pistettä (x, y) lähin piste ja palauttaa tämän indeksin. Voit testata vaikka aineistolla:

```
double[][] luvut = {
    {1,2,1,4 },
    {1,1,2,3 }
};
int i1 = lahimmanPaikka(luvut,0,0); // 0;
int i2 = lahimmanPaikka(luvut,10,10); // 3;
int i3 = lahimmanPaikka(luvut,2.1,2.1); // 1;
```

4. Nyt sitten yhdistä kaikki edellä oleva tietämys niin, että lisäät tehtävän 2 pääohjelmaan lauseet:

```
double xk = keskiarvo(pisteet[0]); // Painopiste
double yk = keskiarvo(pisteet[1]);

int i = lahimmanPaikka(pisteet,xk,yk); // 2D-miidin paikka

ikkuna.add(new Circle(xk,yk,r*1.4)).setColor(Color.RED);
ikkuna.add(new Circle(pisteet[0][i],pisteet[1][i],r*1.2))
    .setColor(Color.GREEN);

double xm = miidi(pisteet[0]); // Miidi komponenteittain
double ym = miidi(pisteet[1]);
ikkuna.add(new Circle(xm,ym,r2)).setColor(Color.CYAN);
```

5. Piirrä tekstitiedostoon (vaikka `ConText`-editorilla) kuva 3-ulotteisesta taulukosta mukaellen Ohj2:n luentomonisteen [kuvaa](#). Piirrä samaan tiedostoon myös kuva 2-ulotteisesta merkkijonotaulukosta mukaellen monisteen 2-ulotteisen taulukon ja [args-taulukon kuvaa](#). Vinkki: [14. luentovideon](#) lopussa on näytetty miten `ConTextia` voi käyttää.

6. **Matriisit:** Tee funktioaliohjelma, joka summaa yhteen kaksi matriisia vastinalkioittain. Esimerkiksi:

```
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 mat3[][] = summaa(mat1,mat2);
    tulosta(mat3);
}
```

tulostaisi (tulosta on tehty luento-esimerkistä [Matriisit.java](#) `double` -matriisille muokaten):

```
10,00  4,00  11,00
 3,00  4,00   7,00
 7,00 21,00   0,00
```

- B1. Tee Java-ohjelma, jossa esittelet ja alustat 3-ulotteisen reaalilukutaulukon. Esitele ja alusta myös 2-ulotteinen merkkijonotaulukko. Tee kaksi tulosta -aliohjelmaa, joilla voit tulostaa em. taulukot.

GURU-tehtävät

- G1-2. **PNS:** Pienimmän neliösumman sovitus on eräs tapa laskea eräänlainen ”keskiluku”tai trendi aineistolle. Esimerkiksi meillä on havaintopisteitä, joiden periaatteessa pitäisi muodostaa ”suora”. Laskemalla PNS-suoran ($y = a + bx$) kertoimet a ja b voimme piirtää aineistoa parhaiten kuvaavan suoran. Katso <http://mathworld.wolfram.com/LeastSquaresFitting.html> kertoimien laskukaavat (12) ja (14) ja tee ohjelma, joka piirtää aineiston ja sitä kuvaavan PNS-suoran. Aineisto piirretään kuten tehtävässä 4, eli voit käyttää sen piirtoaliohjelmaa itse aineiston piirtämiseen (tai edellisen kerran Guru-tehtävän vastausta). Esimerkki: [pns.png](#).