

Demo 11 / 31.3

Taas vähintään kahteen vastaukseen ComTest tai JUnit-testit.

1*2. Kirjoita rahanvaihto-ohjelma, jossa on vakiotaulukko

```

    val.   kur.
+-----+-----+
|mk   |  1.0   |
| $   |  5.7   |
|EUROA| 5.94573|
|SKr  |  0.6   |
+-----+-----+

```

ja jonka toiminta näyttäisi seuraavalta:

```

Määrä ja valuutta>10 Skr [RET]
10.00 Skr on 6.00 mk.
Määrä ja valuutta>2 EUROA [RET]
2 EUROA on 11.89 mk.
Määrä ja valuutta>loppu [RET]
Kiitos!

```

- Muuta edellinen ohjelma sellaiseksi, että valuuttataulukko luetaan tiedostosta.
- Esimerkissä [ListaaKaikki.java](#) tulostetaan yhden hakemiston kaikkien tiedostojen nimet. Muuta esimerkkiä niin, että tulostetaan myös kaikkien alihakemistojen tiedostojen nimet.
- Esimerkiksi [raken_5/Naytto.java](#) metodista `lisaaUusiJasen` voitaisiin piirtää seuraava kutsuhierarkia kun ei merkitä systeemikirjaston kutsuja (`System.out.println`, `rand`, `Mjonot` yms. kutsut):

```

Naytto.lisaaUusiJasen
  Naytto.otsikko
    Naytto.tyhja
      Naytto.tulosta
        Naytto.tulosta
          Naytto.tulosta
            Naytto.tulosta
              Jasen - parametrin konstruktori
                Jasen.rekisteroi
                  Naytto.kysyTiedot
                    Jasen.vastaaAkuAnkka
...

```

Kirjoita vastaava jäsennys omalle ohjelmallesi (ht vaihe 6), metodia `lisaa_uusi_jasen` vastaavalle metodille. Vapaaehtoinen lisäosa: Piirrä sitten (vaikka ihan käsin paperille) puumainen kuva, jossa kukin aliohjelma on vain kerran ja aina nuoli aliohjelmasta/metodista joka kutsuu, siihen aliohjelmaan/metodiin, jota kutsutaan.

Näin saat kuvan, missä nähdään mikä ohjelman osa kutsuu mitäkin aliohjelmaa/metodia.

Tällaisia ohjelman osien riippuvuuksia voidaan kartoittaa myös mekaanisesti, eli on ohjelmia jotka tekevät valmiiksi näitä riippuvuuksia. Jos saadaan käsiteltäväksi valmis"iso" ohjelma, jota pitää korjata, voi tällaisten riippuvuussuhteiden etsimen olla järkevää ennen muutostöiden aloittamista. Näin päästään perille mikä muutos vaikuttaa mihinkä ohjelman osiin.

6*. Kirjoita funktio `fun_min`, jolle viedään parametrina minimoitava funktio-olio ja `x:n` vaihteluväli. Funktio palauttaa funktion "minimin". Katso mallia [oiktark/Integroii2.java](#).

7&8. Ohjelma joka piirtää $\sin(x)$:n kuvaajan seuraavasti (automaattinen skaalaus y-akselille, funktion vaihto helpoksi, ks. [oiktark/Integroii2.java](#) ja tehtävä 6):

```
Piirrän funktion sin(x)
Anna väli ja tiheys jolla piirretään (-5.00 5.00 0.50) >[ret]
```



```
Pääohjelma malliksi
public static void main(String[] args) {
    double x1=-5,x2=5,dx=0.5,y1,y2; String s;
    FunktioRR f = new SinFun();

    System.out.println("Piirrän funktion sin(x)");
    s = Syotto.kysy("Anna väli ja tiheys jolla piirretään",
        Mjonot.fmt(x1,4,2) + " " + Mjonot.fmt(x2,4,2) + " "
+
        Mjonot.fmt(dx,4,2));
    StringBuffer sb = new StringBuffer(s);
    x1 = Mjonot.erota(sb, ' ',x1);
    x2 = Mjonot.erota(sb, ' ',x2);
    dx = Mjonot.erota(sb, ' ',dx);

    y1 = fun_min(f,x1,x2,dx);
    y2 = fun_max(f,x1,x2,dx);

    piirra(f,x1,x2,y1,y2,dx);
}
```

- G1-2 Tee piirtämisestä kunnan graafinen versio. Pohjaksi käy esim. [PiirtoMalli.java](#) tai [PiirtoMalliSwing.java](#).
- G3-4 Muuta graafinen versio sellaiseksi, että klikkaamalla kuvaa jossakin kohti, kuvaan piirretään tätä x:n arvoa vastaavaan kohtaan tangentti käyrälle.