# Viikko 1 labratehtävät – vinkit tehtäviin

**Yleisohje**

Tämän labran tarkoituksena on harjoitella ohjelmointia rakentamalla erillisiä, helposti uudelleenkäytettäviä koodin osia eli metodeja. Tehtävien ratkaisut tulee kirjoittaa omiksi metodeikseen. Nämä metodit eivät saa lukea käyttäjän syötettä tai tulostaa suoraan konsoliin, vaan niiden tulee

palauttaa arvoja. Näin koodi on helposti testattavissa ja modifioitavissa.

Kirjoita jokainen tehtävä omana metodinaan

main-metodin ulkopuolelle, mutta samaan luokkaan. Käytä

main-metodia ainoastaan näiden tehtävämetodien kutsumiseen ja niiden palauttamien arvojen testaamiseen.

*static*-avainsana ja sen käyttö

Koska et vielä opiskele olio-ohjelmointia ja luo olioita, käytämme

*static*-avainsanaa.

*static*-metodit ovat luokkakohtaisia, eivätkä ne liity mihinkään tiettyyn olioon. Tämän ansiosta voit kutsua niitä suoraan luokan nimellä (esim. LabraV1.isEven(8)) ilman, että sinun tarvitsee luoda luokasta erillistä instanssia. Tässä vaiheessa opintoja se on yksinkertaisin ja suoraviivaisin tapa käyttää metodeja, kun ohjelma ei perustu olioihin.

Koodiesimerkki:

A screen shot of a computer program

AI-generated content may be incorrect.

## Tehtävä 1. Parillisuus

**Idea**: % on jakojäännösoperaattori. Parillisuus = jakojäännös 2:lla on 0.  
**Harjoitellaan**: boolean-ehto, operaattorit, yksinkertainen paluuarvo.  
**Tavoite**: osata testata ehto ja palauttaa true/false.  
**Testit**: isEven(8)->true, isEven(7)->false, isEven(0)->true

## Tehtävä 2. Luvun merkki

**Idea**: if/else-rakenne ja haarautuvat paluut.  
**Harjoitellaan**: if-else, merkkijonon palautus.  
**Tavoite**: nimetä positiivinen/negatiivinen/nolla oikein.  
**Testit**: sign(5)->"positiivinen", sign(0)->"nolla", sign(-2)->"negatiivinen"

## Tehtävä 3. Luvun ominaisuudet

**Idea**: uudelleenkäyttö – kutsu aiempia metodeja (sign + isEven).  
**Harjoitellaan**: metodien yhdistäminen, erityistapaus nolla.  
**Tavoite**: tuottaa "positiivinen/negatiivinen parillinen/pariton" tai "nolla".  
**Testit**: describe(8)->"positiivinen parillinen", describe(-5)->"negatiivinen pariton", describe(0)->"nolla"

## Tehtävä 4. Kahden luvun järjestys

**Idea**: Math.min/Math.max käyttämällä saat lyhyemmän ja selkeämmän koodin.  
**Harjoitellaan**: valmiiden apumetodien käyttö, muotoilu.  
**Tavoite**: "pienempi,suurempi".  
**Testit**: order2(5,2)->"2,5"; order2(3,3)->"3,3"

## Tehtävä 5. Kolmen luvun järjestys

**Idea**: "mid" voidaan johtaa summasta: a+b+c - min - max.  
**Harjoitellaan**: looginen päättely, usean arvon yhdistely.  
**Tavoite**: palauttaa "min,mid,max".  
**Testit**: order3(5,2,9)->"2,5,9"; order3(3,7,3)->"3,3,7"; order3(3,2,1)->"1,2,3"

## Tehtävä 6. Kertotaulun rivi

**Idea**: muotoilu täsmälleen testin mukaan + rivinvaihto, joka tehdään näin: \n.  
**Harjoitellaan**: merkkijonon rakentaminen ja kertolasku.  
**Tavoite**: "n x i = tulos".  
**Testi**: line(5,1)->"5 x 1 = 5\n"

## Tehtävä 7. Kertotaulu 1..10

**Idea**: StringBuilder on tehokas toistuvassa liittämisessä; uudelleenkäytä line().  
**Harjoitellaan**: for-silmukka, metodikutsu silmukassa.  
**Tavoite**: koota 10 riviä täsmällisessä muodossa.  
**Testi**: table10(1) alkaa "1 x 1 = 1\n" ja päättyy "1 x 10 = 10\n"

## Tehtävä 8. Salasanan tarkistus

**Idea**: merkkijonien vertailu equalsilla.  
**Harjoitellaan**: equals, null-turvallisuus.  
**Tavoite**: palauttaa true vain täsmälleen oikeasta salasanasta  
**Testit**: check("java123")->true; check("JAVA123")->false; check(null)->false

## Tehtävä 9. Salasanayritykset

**Idea**: rajataan tarkistus enintään kolmeen yritykseen.  
**Harjoitellaan**: taulukon läpikäynti, varhainen palautus.  
**Tavoite**: palauttaa Tervetuloa!/Liian monta yritystä.  
**Testit**: login({"no","java123","x"}) → "Tervetuloa!"; login({"a","b"}) → "Liian monta..."; login({"no","no", "no", "java123"}) → "Liian monta...".

## Tehtävä 10. Taulukon tilastot

**Idea**: käsitellään reunatapaukset (tyhjä taulukko), **kun taulukko on tyhjä**, meidän pitää päättää, mitä funktio palauttaa. max(int[] a) tapauksessa; Jos taulukossa ei ole yhtään alkiota, ei ole olemassa suurinta lukua. Jotta funktio voi kuitenkin palauttaa jonkin arvon, käytetään Java-standardissa määriteltyä ääripäätä: **Integer.MIN\_VALUE**

**Harjoitellaan**: for-each, kertymä, ääripäiden alustus.  
**Tavoite**: average: tyhjälle 0.0; max: tyhjälle Integer.MIN\_VALUE.  
**Testit**: average({5,10,15})->10.0; max({-1,20,3,7})->20; average({})->0.0; max({})->Integer.MIN\_VALUE