

Demo 7 / 26.2

Muista ettei tehtäviä lasketa kuin max. 10 kerrallaan, joten jos teet bonus/guru-tehtäviä, niin voit säästää aikaasi jättämällä muutaman "tavallisen" tekemättä. Bonus-tehtävät on täysin mahdollista tehdä kurssin tiedoilla. Tämän kerran Guru-tehtäväkin on mahdollista tehdä, se vaatii vain hieman miettimistä. Kaikkiin tehtäviin TDD joihin mahdollista. Ja tästä lähtien JUnitilla tehtynä.

- 1*. Mikä (tai mitkä) Java 1.4.1:n (tai uudemman) `String`-luokan metodi sopisi seuraavan ongelman ratkaisemiseen ja miten (kirjoita malli kutsusta):
 - a) Onko merkkijonossa jono muita kirjaimia kuin joukon `k` kirjaimet


```
jono="kissa" k="aik" -> on, k="aiks" -> ei ole
```
 - b) Missä on jonon viimeinen `\`.


```
"C:\mytemp\ohj2\vesal\Koe.java" ->
indeksi "osoittamaan" viimeiseen '\' -merkkiin,
eli jonon \Koe.java:n alkuun.
```
 - c) Onko jonossa jokin kirjain joukosta `k`

```
jono="kissa" k="ibm" -> on , k="pc" -> ei ole
```
2. Edelleen katso `String`-luokasta kuinka saataisiin vastaus kysymyksiin (kirjoita esimerkkikoodit, nyt voi olla ettei yksi rivi enää riitä):
 - a) onko `"matti"` sama kuin `"Matti Nykänen"`?


```
(vastaus: on. Huomaa jokerimerkki, välilyönnit, sekä
isot ja pienet kirjaimet)
```
 - b) Paljonko jonossa `"Kissa istuu puussa"` on yhteensä


```
merkkejä "a-j" tai "r-w" ("a-j" == "abcdefghij")
(vastaus: 2*a + 2*i + 5*s + 1*t + 4*u = 14)
```
3. Monisteen [luvussa 10.5.2](#) on esimerkki funktiosta `postimaksu`. Tee tätä matkien funktio `suurin_kirjeen_paino`, joka palauttaa suurimman kirjeen painon, jonka voi lähettää tietyllä rahasummalla, `if`-toteutus.
- 4*. `suurin_kirjeen_paino`, taulukkoon pohjautuva toteutus. Koeta saada hintojen muuttaminen mahdollisimman helpoksi.
5. Kirjoita funktio `palindromi`, joka palauttaa tiedon siitä (`true`=kyllä, `false`=ei) onko parametrina välitetty SANA palindromi vai ei (esim `abba` on palindromi, `apua` ei ole, sanassa ei ole välilyöntejä tai muita erikoismerkkejä).
- 6*. Kirjoita kaksi eri aliohjelmia (toinen `String`-jonoille ja toinen `StringBuffer`-jonoille) tuhoa_lopusta, jotka loogisessa mielessä "poistavat" merkkijonon `n` viimeistä merkkiä

(muista virhetilanteet!):

```
String s="Kissa istuu";
s = tuhoa_lopusta(s,3);           // => s = "Kissa is"
StringBuffer sb = new StringBuffer("Kissa istuu");
tuhoa_lopusta(sb,3);             // => sb= "Kissa is"
```

- 7*. Seuraavassa C-ohjelman määrittelyt mäkihyppykilpailua varten. Piirrä aluksi kuva kummas-takin tietueesta (toni ja matti) "sijoituksen" jälkeen. Tämän jälkeen esittele vastaavat luokat Java-kielellä. Eli tällä demokerralla tarvitaan vain luokkien nimet ja attribuuttien esittelyt. Jos esimerkki ei riitä C:n tietueista, niin lisää voit katsoa esim: [Ohjelmointi ++](#), [9.2.3 Uuden tietuetyypin määrittely](#) ja [13.7 Tietueet, union ja enum](#).

```
/* tonitiet.c */
/* Malli tietueesta tietueessa
#include <stdio.h>

typedef struct {
    double pituus;           /* hyppyjen pituudet metreinä */
    double tuomarit[5];     /* tuomaripisteet */
    double pisteet;        /* Yhteistulos */
} Kierros_tyyppi;

typedef struct {
    Kierros_tyyppi kierros[2];
    double lopputulos;
} Tulos_tyyppi;

typedef struct {
    char nimi[8];
    int nro;
    Tulos_tyyppi tulos;
} Kilpailija_tyyppi;

int main(void)
{
    Kilpailija_tyyppi toni,matti;
    /* Halutaan tehdä sijoitukset:

        toni:  nimi <- "Toni N"
                nro <- 3
                1. kierroksen pituus <- 107
                2. kierroksen tuomareiden pisteet <-
                    19,18,19.5,18,20
        matti: nimi <- "Matti H"
                nro <- 7
                2. kierroksen pituus <- 109
                1. kierroksen pisteet <- 125
                Lopputulos <- 251

    Esimerkki:
        toni.tulos.kierros[0].pituus = 107;
    */
    ...
    return 0;
}
```

8*. Kirjoita aliohjelmat `paras` (palauttaa reaalilukutaulukon suurimman luvun), `huonoin` (palauttaa reaalilukutaulukon pienimmän luvun) ja `summa` (palauttaa reaalilukutaulukon summan). Näitä käyttäen kirjoita aliohjelma `summa_huonoin_ja_paras_pois`, joka palauttaa reaalilukutaulukon summan kun siitä otetaan huonoin ja paras tulos pois (sopii esim. mäkipisan arvosteluun).

B1-2 Astiapelissä ([AstiaPeli.java](#)) voitaisiin tehdä lopun automaattista tarkistusta auttamaan luokka, jota voitaisiin käyttää seuraavasti (sovitaan että astioiden tilavuudet voivat olla vain kokonaislukuja):

```
public static void main(String[] args) {
    Esiintymat esiintymat = new Esiintymat(1,13);
                                // laskee lukujen 1-13 esiintymiä
    esiintymat.lisaa(0); // ei vaikuta, koska 0 ei ole välillä [1,13]
    esiintymat.lisaa(1); //
    esiintymat.lisaa(8); // lisää yhden esiintymän luvun 8 kohdalle.
    esiintymat.lisaa(5); // lisää yhden esiintymän luvun 5 kohdalle.
    esiintymat.lisaa(13); //
    System.out.println(esiiintymat.loydetyt()); // 1 5 8 13
    System.out.println(esiiintymat.ei_loydetyt()); // 2 3 4 6 7 9 10 11 12
    int loydettyja = esiintymat.getLoydettyja();
    System.out.println("Loydettyja on " + loydettyja); // Löydettyjä on 4
}
```

Toteuta luokka `Esiintymat` ja testaa sitä em. kutsuilla.

G1-2 Edelleen astiapeliin. Löydetyt esiintymät on "helppo" tarkistaa jos tiedetään että käyttöastioita on 2 kappaletta. Mutta jos astioita on `lkm`-kappaletta, niin testaaminen meneekin vaikeammaksi. Hahmottele apuluokka, jota voitaisiin käyttää edellisen tehtävän `Esiintymat`-luokan kanssa apuna käymään läpi kaikki summakombinaatiot, joita astioista voisi muodostaa. Eli jos meillä olisi vaikkapa astioita 3 kappaletta ja niissä olisi nestettä 3, 5 ja 9 litraa, niin saisimme niillä aikaiseksi summakombinaatiot (järjestys ei ole oleellinen):

3	(3+0+0)
5	(0+5+0)
8	(3+5+0)
9	(0+0+9)
12	(3+0+9)
14	(0+5+9)
17	(3+5+9)

Aloita hahmottelu miettimällä kuinka voisit alustaa luokan, mitä metodeja tarvittaisiin ja miten kutsuisit metodeja (vrt. edellinen tehtävä, olkoon luokan nimi vaikkapa `Kombinaatiot`).

HUOM!

Tämä on laskennallisesti viimeinen demokerta demojaksoon 1. Demojaksoon 2 tulevat demot 8,9,10,11,12. Demojaksossa 2 jokaisen kerran maksimipistemäärä on 8 tehtävää (jaksossa 1 se oli 10). Pistelaskennassa prosentteihin ei pyöristellä, vaan KATKAISTAAN. Eli $104.95\% \Rightarrow 104\% < 105\%$.