Demo 7 / 28.2

Muista ettei tehtäviä lasketa kuin max. 10 kerrallaan, joten jos teet bonus-/gurutehtäviä, niin voit säästää aikaasi jättämällä muutaman "tavallisen" tekemättä. Bonustehtävät 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 ComTestillä tai JUnitilla tehtynä. HUOM! Ilman testejä ei saa pisteitä!

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% => 104% < 105%.

1*. Mikä (tai yhdistelemällä mitkä) Java:n `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 `jono`ssa 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 `suurinKirjeenPaino`, joka palauttaa suuriman kirjeen painon, jonka voi lähettää tietyllä rahasummalla, `if`-toteutus. Sovitaan että esim. monisteen esimerkissä olisi <= jokaisen < -merkin tilalla.

4*. `suurinKirjeenPaino`, 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 `tuhoa_lopusta` aliohjelmaa (toinen `String`-jonoille ja toinen `StringBuilder` (tai `StringBuffer`) -jonoille), jotka loogisessa mielessä "poistavat" merkkijonon n viimeistä merkkiä (muista virhetilanteet!):

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

7*. Seuraavassa C-ohjelman määritykset mäkihyppykilpailua varten. Piirrä aluksi kuva kummastakin tietueesta (`toni` ja `matti`) "sijoituksen" jälkeen. Huomaa että C:ssä asiat ovat sisäkkäisiä. 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 `summaHuonoinJaParasPois`, joka palauttaa reaalilukutaulukon summan kun siitä otetaan huonoin ja paras tulos pois (sopii esim. mäkikisan arvosteluun).

B1-4. Seuraavissa Bonus- ja Guru-tehtävissä on tarkoitus tehdä mallin:GraafinenAstiaPeli.jar mukainen GraafinenAstiaPeli. Ota aluksi pohjaksi uudet: Astia.java ja GraafinenAstia.java sekä demo 6:n mallivastauksen AstiaPeli-luokka.

```
Etenemisjärjestys:
  0. Luokkiin Astia, GraafinenAstia ei tarvitse tehdä muutoksia.
  1. Vaihda AstiaPelin sisäisen astiat[] -taulukon tilalle
     ArrayList<Astia> astiat;
  2. Kokeile ja aja ilman muita muutoksia.
  3. Luo uusi Swing Frame nimelle GraafinenAstiaPeli
  4. Yritä tehdä käyttöliittymästä mallin kaltainen, ilman astioita.
  5. Lisää yksi paneli nimelle panelAstiat ja laita siihen
     vaakasuuntainen BoxLayout
  6. Laita GraafinenAstiaPeli-luokan main-metodiin
        frame.setVisible(true);
     edelle koodi:
        AstiaPeli peli = new AstiaPeli();
        peli.lisaaAstia("8", 8);
        peli.lisaaAstia("5", 5);
        frame.alustaAstiat(peli);
  7. Lisää vielä alla olevat metodit luokkaan.
      private void alustaAstiat(AstiaPeli peli) {
        this.peli = peli;
        maxSize = peli.astioidenSumma();
        for (int i = 0; i < peli.getLkm(); i++) {
            Astia astia = peli.anna(i);
            lisaaAstia(astia);
        }
      }
  // alustavasti:
      private void lisaaAstia(Astia astia) {
        GraafinenAstia gastia = new GraafinenAstia();
        gastia.setMaxSize(maxSize);
        gastia.setAstia(astia);
        panelAstiat.add(gastia);
        gastia.addAstiaClickedListener(new AstiaClickedListener() {
            @Override
            public void clicked(GraafinenAstia gastia) {
                astiaKlikattu(gastia);
            }
        });
      }
```

```
8. Itsellesi tehtäväksi jää siis oikeastaan vain metodi
   astiaKlikattu ja lisätä astioidenSumma() AstiaPeli-luokkaan.
   Tämä laskee kaikkien muidne astioiden yhteiskoon paitsi ämpärin.
   Tämän tiedon perusteella osataan astiat piirtää suhteessa oikean
   kokoisiksi. Malliesimerkissä summa on siis 13.
```

Uuden `Astia`-luokan ero vanhaan `Astia`-luokkaan on siinä, että siihen on lisätty

```
addMaaraMuuttuuListener
```

jonka ansiosta voidaan ”kuunnella” määrän muuttumista ja näin graafisessa versiossa reagoida siihen. `GraafinenAstia` on vain ”näytin”, johon voidaan liittää `Astia` ja kun `Astia` muuttuu, niin `GraafinenAstia` huomaa tämän ja muuttaa siksi ”nesteen pintaa”.

Toteuta mallin mukaisesti tomiva `GraafinenAstiaPeli` niin, että siinä ei vielä tarvitse toimia löydettyjen määrien näyttäminen. Kokeile toimiiko ilman muita muutoksia 3:lla lisätyllä astialla.

B5-6. 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(esiintymat.loydetyt());   // 1 5 8 13
    System.out.println(esiintymat.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 (tietysti myös testit) luokka `Esiintymat` ja testaa sitä mm. 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. Toteuta 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       (0+0+3)
5       (0+5+0)
8       (0+5+3)
9       (9+0+0)
12      (9+0+3)
14      (9+5+0)
17      (9+5+3)
```

Aloita hahmottelu miettimällä kuinka voisit alustaa luokan, mitä metodeja tarvittaisiin ja mi-

ten kutsuisit metodeja (vrt. edellinen tehtävä, olkoon luokan nimi vaikkapa `Kombinaatiot`).

G3-4. Lisää `Esiintymat` ja `Kombinaatiot` luokan käyttö sekä komentorivipohjaiseen AstiaPeliin että Graafiseen astiapeliin niin, että käyttäjä aina tietää mitkä nestemäärät hän on onnistunut ratkaisemaan ja mitkä ovat vielä ratkaisematta. Kun kaikki on läydetty, peli huomauttaa tästä.