Demo 10 / 22.3

Taas T:llä merkitty ne joissa JUnit-testi on pakko kirjoittaa. Vähintään kahteen palautettuun tehtävään pitää olla tehtynä testit jotta palautus hyväksytään. Tiedostojen testaamisesta katso edellisen demon vastauksia ja ota uusin `Ali.jar` avuksi.

1*. Kirjoita Java-ohjelma joka tulostaa tiedoston `koe.txt`:

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
Kissa istuu puussa
ja ihmettelee
mualiman menoa
```

sisällön siten, että kunkin rivin alkuun tulee rivinumero ja kustakin rivistä tulostetaan korkeintaan 40 merkkiä (T):

```
/* 01 */ 1234567890123456789012345678901234567890
/* 02 */ Kissa istuu puussa
/* 03 */ ja ihmettelee
/* 04 */ mualiman menoa
```

2. Kirjoita ohjelma, joka kysyy tiedoston nimen ja tulostaa tiedostosta kaikki ne rivit jotka alkavat `**` (T).

3*. Kirjoita ohjelma, joka lukee tiedoston (tietovirtoja käyttäen) ja tulostaa lopuksi kuinka monta kertaa mikäkin aakkosten kirjain esiintyi tiedostossa ('A'='a'). **Vihje:** Mieti ensin mitä itse tekisit jos joutuisit moisen homman tekemään, eli esimerkiksi kertomaan tästä demopaperista monestiko mikäkin kirjain esiintyi. (T)

4. Lisää esimerkkiin dyna/TaulukkoGen.java automaattinen taulukon koon kasvatus, mikäli alkuperäisen varatun taulukon koko käy liian pieneksi.

5. Lisää luokkaan dyna/TaulukkoGen.java metodi `clone()`, joka luo syväkopioinnilla kloonin oliosta. Syväkopiointi tarkoittaa sitä, että olio on kopio toisesta oliosta, mutta niillä ei ole yhteisiä viitteitä, vaan kaikki viitatutkin oliot on syväkopioitu (paitsi Javan tapauksessa immutable luokkia ei tarvitse kopioida). ComTest-testi `clone` -metodille:

```
* <pre name="test">
* #THROWS CloneNotSupportedException
* #import demo.Int;
*  TaulukkoGen<Int> luvut = new TaulukkoGen<Int>();
*  luvut.lisaa(new Int(0)); luvut.lisaa(new Int(2));
*  luvut.lisaa(new Int(99));
*  ;@SuppressWarnings("unchecked")
*  TaulukkoGen<Int> taul = (TaulukkoGen<Int>)luvut.clone();
*  luvut.toString() === " 0 2 99";
*  taul.toString()  === " 0 2 99";
*  luvut.get(1).set(3);
*  luvut.toString() === " 0 3 99";
*  taul.toString()  === " 0 2 99";
*  luvut.lisaa(new Int(2)); luvut.lisaa(new Int(5));
*  luvut.lisaa(new Int(2)); luvut.lisaa(new Int(6));
*  luvut.toString() === " 0 3 99 2 5 2 6";
```

```
     *  taul.toString()  === " 0 2 99";
     *  taul.get(3).intValue() === 2; #THROWS IndexOutOfBoundsException
     *  luvut.poista(new Int(2));
     *  luvut.toString() === " 0 3 99 5 6";
     *  taul.toString()  === " 0 2 99";
     * </pre>
```

Yksinkertaisimmassa ratkaisussa luokan esittelyn voi muuttaa muotoon:

```
public class TaulukkoGen<TYPE extends Int> implements Cloneable
```

jolloin siihen voi tallentaa vain `Int`-luokan olioita tai sen jälkeläisiä. Pitääkö jotakin vielä muuttaa luokassa `Int`?

6. Seuraavana esimerkki linkitetyn listan käytöstä (vrt. luennolla tehty muutos dyna/Taulukko.java tietorakenteeseen: dyna/LinLista.java. Piirrä kuva tietorakenteesta kun pääohjelmassa ollaan menossa rivillä ”PIIRRÄ KUVA”.

```
package demo;
import fi.jyu.mit.ohj2.*;
/**
 * Esimerkki linkitetystä listasta,
 * @author Vesa Lappalainen
 * @version 1.0, 15.03.2003
 */
public class Koulu {

  public static class Oppilas {
    private String nimi;
    private double keskiarvo;
    private Oppilas seuraava;

    public Oppilas(String nimi, double keskiarvo) {
      this.nimi = nimi; this.keskiarvo = keskiarvo;
    }

    public String toString() {
      return Mjonot.fmt(nimi,-22) + " keskiarvo: " +
             Mjonot.fmt(keskiarvo,5,2);
    }
  }

  private String luokka;
  private int oppilaita;
  private Oppilas ensimmainen;
  private Oppilas viimeinen;

  public Koulu(String luokka) { this.luokka = luokka; }

  public void lisaa(Oppilas oppilas) {
  }

  public void tulosta(OutputStream os)  {
    PrintStream out = new PrintStream(os);
  }

  public void poistaKaikki() {
  }
```

```
    public static void main(String[] args)  {
      Koulu luokka = new Koulu("1b");

      luokka.lisaa(new Oppilas("Ankka Aku",5.0));
      luokka.lisaa(new Oppilas("Ankka Tupu",7.0));
      luokka.lisaa(new Oppilas("Hiiri Mikki",9.0));

      luokka.tulosta(System.out);  // PIIRRÄ KUVA
      luokka.poistaKaikki();
      luokka.tulosta(System.out);
    }
  }
```

Seuraavat tehtävät liittyvät tuohon Koulu.java ohjelmaan. Valmis ComTest testi nettiversiossa tai JUnit-testiohjelma demo/test/KouluTest.java.

7*. Täydennä metodi `lisaa.`

8*. Täydennä metodi `tulosta` (tulostaa luokan omat tiedot ja kaikkien oppilaiden tiedot, käyttää metodia `Oppilas.toString`)

GURU ja Bonus-tehtävät

B1-2 Muuta Koulu.java -esimerkin rakenne sellaiseksi, että luokka `Oppilas` ei sisällä viitettä seuraavaan, vaan on erikseen luokkaa `ListanAlkio,` jossa on `Oppilas` -viite ja viite seuraavaan alkioon. Näin luokasta saadaankin luokka, jolla voi tallentaa mitä tahansa objekteja. Pääohjelma ei saa muuttua.

B3 Lisää Koulu.java-esimerkkiin metodi `iterator`. Katso mallia esim: yhteis_51/Jasenet.java ja yhteis_51/Harrastukset.java.

G1-2 Kirjoita Koulu.java:n luokaan `Koulu` metodi `kaanna`, joka kääntää linkitetyn listan päinvastaiseen järjestykseen.

G3-4 Suunnittele ja toteuta luokka `StopWatch` jolla voidaan helposti ottaa väliaikoja ja loppuaikoja ja jolla voidaan kysyä aika eri yksiköissä (ms, s, /kierros). Katso tarve esim: https://korppi.jyu.fi/list-archive/ohj05k/0050.html