Harjoitus 7 (viikko 10)

Nämä ovat kurssin viimeiset harjoitukset. Muista, että hyväksytyistä ratkaisuista ja läsnäoloista kerättyjen pisteiden summan tulee olla vähintään 40 % (23 pistettä) tehtävien ja läsnäolopisteiden kokonaislukumäärien summasta (50 + 7 = 57). Hyvityspisteiden rajat ovat: 60 % (35 pistettä), 70 % (40 pistettä), 80 % (46 pistettä) ja 85 % (49 pistettä).

Voit tehdä myöhemmin julkaistavia lisätehtäviä (korkeintaan 8 kpl), jos aiot suorittaa kurssin, mutta pisteesi ovat alle 40 % -rajan tämän harjoituksen jälkeen. Kurssin vastuuopettaja ottaa yhteyttä opiskelijoihin, jotka voivat saavuttaa 40 %:n rajan lisätehtäviä tekemällä.

Lopuksi vielä tutut ohjeet ja tiedotteet:

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@tuni.fi).
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 11. luku) ja Lausekielinen ohjelmointi II -kurssilla opetettuja hyviä tapoja. Muista muun muassa kirjoittaa metodin otsikkoon liittyvä yleisluonteinen kommentti metodin tarkoituksesta sekä metodin mahdollisesti saamista ja palauttamista tiedoista ellei kyseessä ole aksessori. Tee jokaisen tiedoston alkuun kommentti, jossa on harjoituksen ja tehtävän numero sekä nimesi ja sähköpostiosoitteesi. Noudata myös tämän kurssin kuluessa opetettavia hyviä tapoja: älä muun muassa nimeä aksessoreita kahta kieltä käyttäen.
- Lue syötteet *Scanner*-luokalla ja käytä syötteen lukemiseen vain yhtä *Scanner*-oliota. Esittele lukija vakiomuotoisena luokka-attribuuttina, jos syötteitä on luettava kahdessa tai useammassa metodissa.
- Mikroharjoituksissa saa apua ongelmakohtiin. Perjantaisin pidettävässä ohjatussa ryhmässä avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauusaikaa on jatkettu erityisesti hiihtoloman vuoksi. Palauta vastauksesi WETO-järjestelmään (https://wetodev.sis.uta.fi/) viimeistään sunnuntaina 8.3. klo 23.55.
- WETO tarkistaa palautuksia automaattisesti sekä tulosteita vertailemalla että tutkimalla luokkien rakennetta ja toiminnallisuutta. Automaattisesta tarkistuksesta ja erityisesti WETOn tekemistä oletuksista attribuuttien ja niiden aksessoreiden suhteen on kerrottu kurssisivujen *Harjoitukset* | *Tarkistus* | *Automaattinen* -kohdassa. Tutustu myös yleisempiä ohjeita sisältävään *Harjoitukset* | *Tarkistus* -kohtaan.
- Testaa luokkaasi omatoimisesti tehtävässä kuvatulla tavalla. Oman testauksen tekemättä jättäminen katsotaan virheeksi. Älä tarkista syötteitä niiden lukemisen yhteydessä ellei tarkistusta erikseen pyydetä, koska metodit tarkistavat parametriarvonsa, mikäli mahdollista.
- Kaikki opettajien tarkistamat tehtävät arvioidaan hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys tai puutteellinen kommentointi voi tuottaa nollan, vaikka ohjelma läpäisee WETOn testit. Muista erityisesti metodin otsikkoon liittyvä yleisluonteinen kommentti.
- Ota yhteyttä harjoitusryhmäsi vetäjään tai kurssin vastuuopettajaan, jos et keksi järjellisessä ajassa miksi WETO hylkää palautuksesi.

1. Periytä Javan *LinkedList<E>*-luokasta *OmaLista<E>*-luokka ja tee sille metodi, jonka otsikko on:

   ```
   public int poistaNullArvot()
   ```

   Metodi poistaa listalta kaikki **null**-arvoiset alkiot. Metodin paluuarvo on poistettujen alkioiden lukumäärä. Paluuarvo on nolla, jos lista on tyhjä.

   Jos listan alkioiden arvot ovat esimerkiksi [**null**, **null**, "abba", 42, **null**], niin metodin kutsumisen jälkeen listan alkioiden arvot ovat ["abba", 42]. Metodin paluuarvo on tässä esimerkissä kolme, koska listan ensimmäinen, toinen ja viimeinen alkio poistettiin.

   Testaa listaasi erillisessä *OmaListaTesti1*-luokassa.

2. Periytä *LinkedList<E>*-luokasta *OmaLista<E>*-luokka ja tee sille metodi, jonka otsikko on:

   ```
   public Object[] käänteisestiTaulukkoon()
   ```

   Tämä metodi luo yksiulotteisten *Object*-tyyppisten viitteiden taulukon, jonka koko on sama kuin listan koko. Metodi asettaa taulukon viitteet siten, että taulukon ensimmäinen alkio viittaa listan viimeiseen tietoalkioon, taulukon toinen alkio viittaa listan toiseksi viimeiseen tietoalkioon ja niin edelleen siten, että taulukon viimeinen alkio viittaa listan ensimmäiseen tietoalkioon. Metodi palauttaa lopuksi viitteen taulukkoon. Metodi ei muuta listan tietoalkioiden järjestystä.

   Jos listalla on esimerkiksi alkiot "one", "two" ja "three", taulukon alkiot ovat "three", "two" ja "one".

   Paluuarvo on nollan alkion mittainen tyhjä taulukko, jos lista on tyhjä.

   Testaa listaasi erillisessä *OmaListaTesti2*-luokassa.

3. Periytä Javan *LinkedList<E>*-luokasta *OmaLista<E>*-luokka ja tee sille metodi, jonka otsikko on:

   ```
   public int korvaa(Object korvattava, E korvaava)
   ```

   Metodi hakee listalta tietoalkiot, jotka ovat samoja kuin ensimmäisen parametrin arvo ja korvaa nämä tietoalkiot toisen parametrin arvolla. Paluuarvo on korvattujen tietoalkioiden lukumäärä. Alkio korvataan ja lasketaan mukaan paluuarvoon myös silloin, kun korvattava ja korvaava arvo ovat samat.

   Parametreilla voi olla **null**-arvoja. Samoin listalla voi olla **null**-arvoisia tietoalkioita. Jos *korvattava*-parametri on **null**, metodi korvaa kaikki listan **null**-arvoiset tietoalkiot *korvaava*-parametrin arvolla. Jos *korvattava*-parametri liittyy olioon, metodi korvaa *korvaava*-parametrin arvolla kaikki listan tietoalkiot, jotka havaitaan samoiksi, kun *korvattava*-parametria ja tietoalkiota verrataan toisiinsa *equals*-metodilla.

   Jos parametrit ovat esimerkiksi **null** ja "C" ja listan alkiot ovat ["A", **null**, "D", **null**], ovat alkiot metodin kutsumisen jälkeen ["A", "C", "D", "C"], koska **null**-arvot korvattiin merkkijonolla "C". Metodin paluuarvo on luku 2.

   Jos parametrit ovat "A" ja "B" ja listan alkiot ovat ["A", **null**, "D", "A"], ovat alkiot metodin kutsumisen jälkeen ["B", **null**, "D", "B"], koska ensimmäisen parametrin havaittiin olevan *equals*-metodilla verraten sama kuin ensimmäinen ja viimeinen tietoalkio. Metodin paluuarvo on tässäkin esimerkissä luku 2.

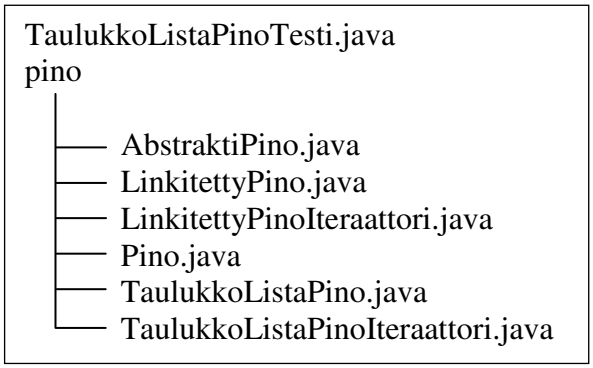
   Testaa listaasi erillisessä *ListaTesti3*-luokassa.

4. Kurssin verkkosivuilla on annettu osoitteessa: https://www.sis.uta.fi/~oope/oope1/kevat-2020/luennot/luento13/esimerkit/linkitetty_pino/ *pino*-pakkaus, jossa on *Pino*-rajapinta, *AbstraktiPino*-luokka, *LinkitettyPino*-luokka ja linkitetyn pinon iteraattori *LinkitettyPinoIteraattori*-luokka. Periytä *AbstraktiPino*-luokasta geneerinen *TaulukkoListaPino*-luokka, joka käyttää tietorakenteenaan Javan *ArrayList*-luokkaa. Tee luokallesi iteraattoriluokka *TaulukkoListaPinoIteraattori*.

   Pinon toteutus on tehokkaampi, kun pinon päällyksenä käytetään viimeistä taulukkolistan tietoalkiota, jolloin *ArrayList*-olion ei tarvitse siirtää tietoalkioita pinoon lisättäessä ja siitä poistettaessa.

   Testaa pinoasi ja sen iteraattoria erillisessä *TaulukkoListaPinoTesti*-luokassa. Huomaa, että yllä annetusta osoitteesta löytyy myös testiluokka, jota voi muokata pienin muutoksin oman testiluokan.

   Lisää uudet pino- ja iteraattoriluokat *pino*-pakkaukseen. Älä lisää testiluokkaa pakkaukseen, vaan sijoita testiluokka pakkaushakemiston välittömään ylihakemistoon. Tiivistä testiluokka ja pakkauksen lähdekoodi zip-muotoiseksi *pino.zip*-paketiksi siten, että tiedostoon tallentuu myös pakkauksen hakemistotieto, jolloin pakkauksen sisältävä hakemisto *pino* muodostuu automaattisesti tiedostoa purettaessa. Oheisessa kuvassa on esitetty pakkauksen ja samalla paketin rakenne. Palauta WETOon vain *pino.zip*-tiedosto.



5. Tee Java-kielellä *Aggregaattori*-luokka, joka lukee käyttäjältä tekstiä, jossa sanat on erotettu toisistaan yhdellä välilyönnillä ja laskee aggregaatio-operaatioita käyttäen sanojen lukumäärän, sanojen pituuksien summan ja sanojen pituuksien keskiarvon.

   Sijoita sanat Javan *LinkedList*-listaan ja laske kukin kolmesta tunnusluvusta soveltamalla aggregaattoreita taulukosta luotuun tietovirtaan. Tarvitset uuden virran kunkin tunnusluvun laskemiseen, koska pääteoperaatio sulkee virran.

   Vinkkejä: Voit pilkkoa tekstin sanoiksi *String*-luokan *split*-metodilla. Eräs tapa täyttää lista sanoilla on muuntaa taulukko listaksi *Arrays*-luokan avulla ja antaa lista parametrina *LinkedList*-listan rakentajalle. Summan ja keskiarvon laskemisessa on tehtävä välioperaatiolla *Stream*-virran muunnos *IntStream*-virraksi. *Average*-operaation palauttaman arvon saa muunnettua **double**-tyypiksi *getAsDouble*-metodilla.

   Voit kirjoittaa kaiken koodisi luokan *main*-metodin sisään.

   Kurssin verkkosivuilla on annettu osoitteessa: https://www.sis.uta.fi/~oope/oope1/kevat-2020/luennot/luento13/esimerkit/aggregaatio/ esimerkkejä aggregaatio-operaatioiden käytöstä.

Esimerkki ohjelman toiminnasta:

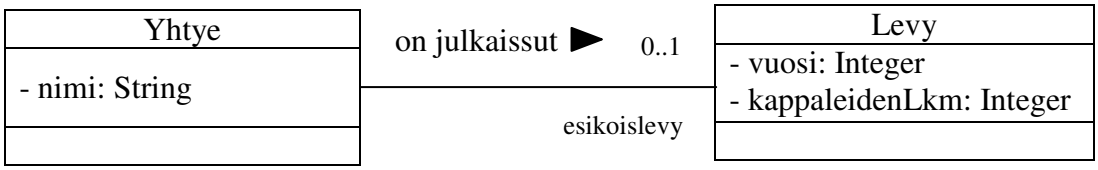
```
Hello! I count some text statistics.
Please, enter text:
this is a test
The number of words: 4.
The sum of word lengths: 11.
Average word length: 2.75.
```

6. Alla on kuvattu UML:n luokkakaaviona suhde "Yhtye on saattanut julkaista tai olla julkaisematta esikoislevyn."

   Kirjoita Java-kielellä *Yhtye*- ja *Levy*-luokat. Kätke kummankin luokan tiedot. Kirjoita kaikille attribuuteille aksessorit. Tee molemmille luokille parametrillinen rakentaja. Levyn rakentajan ensimmäisellä parametrilla välitetään tieto julkaisuvuodesta.

   Tarkista parametrien arvot luokkien rakentajissa ja asetusmetodeissa. Nimi ei ole **null**-arvoinen ja että siinä on vähintään yksi merkki. Julkaisuvuoden on oltava vähintään 1900 ja kappaleiden lukumäärän vähintään yksi. Attribuutille jää Javan antama alkuarvo, jos sen parametriarvo on virheellinen.

   Kiinnitä erityistä huomiota luokkien välisen assosiaation toteuttamiseen.



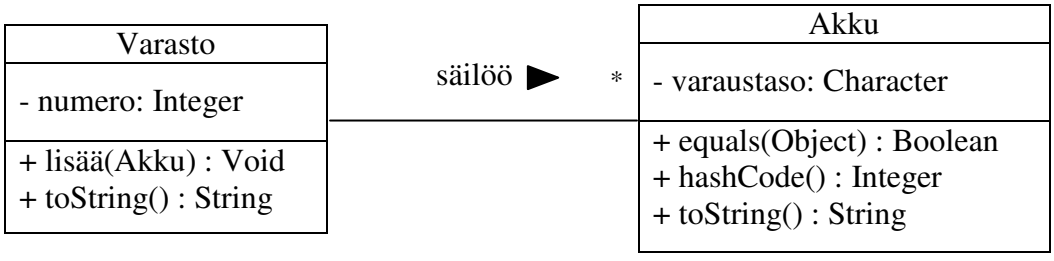
   Testaa listaasi erillisessä *AssosiaatioTesti1*-luokassa.

7. Alla on kuvattu UML:n luokkakaaviona suhde "Varasto säilöö akkuja."

   Kirjoita Java-kielellä kapseloitu *Varasto*-luokka. Kätke *Varasto*-luokan tieto, kirjoita aksessorit ja parametrillinen rakentaja. Luokan attribuutti on **int**-tyyppinen (> 0). Attribuutille jää Javan antama alkuarvo, jos sen parametriarvo on virheellinen.

   Akku-luokkaa käsiteltiin tehtävässä 5.4.

   Kiinnitä erityistä huomiota luokkien välisen assosiaation toteuttamiseen.



   Testaa listaasi erillisessä *AssosiaatioTesti2*-luokassa.

8. Anna seuraavat luokat sisältävä kaupunkiliikennettä kuvaava luokkakaavio: jalankulkija, polkupyöräilijä, polkupyörä, autoilija, auto, jalkakäytävä, pyörätie, katu, suojatie ja liikennevalo. Pyörätien oletetaan olevan yhdistetty jalkakäytävään. Kaikilla jalkakäytävillä ei ole siihen yhdistettyä pyörätietä.

   Voit antaa luokkasymboleissa vain luokkien niemet, koska luokille ei tarvitse keksiä attribuutteja ja metodeja. Sen sijaan jokaisella luokalla on oltava jokin suhde vähintään yhden kaavion muun luokan kanssa. Nimeä tavallisia assosiaatioita ja merkitse mahdollisuuksien mukaan kertautumisia.

   Huomaa, että voit lisätä kaavioon tarvittaessa uusia luokkia. Joillekin edellä luetelluille luokille voi löytyä esimerkiksi yhteinen yliluokka.

   Kaaviota ei ole pakko piirtää ohjelmalla. Hyvälaatuinen valokuva kynällä ja paperilla tehdystä kaaviosta kelpaa myös.