Mallitentti

Lue kukin tehtävänanto huolellisesti ja kokonaisuudessaan ennen kuin aloitat vastaamisen. Kustakin tehtävästä voi saada 0–6 pistettä. Tentin läpipääsyraja on 12 / 24 pistettä. Ratkaisut palautetaan WETO-järjestelmään (https://wetodev.sis.uta.fi/weto5/), jossa on kuvailtu kunkin tehtävän automaattinen testaus. Aksessorit kirjoitetaan samoin kuin harjoituksissa (https://coursepages.uta.fi/tiea2-1a/kevat-2020/harjoitukset/tarkistus/automaattinen/).

1. Tee Javalla konkreettinen *Kone*-luokka koneelle. Luokan attribuutit ovat koneen teho kilowatteina (**double**) ja valmistajan nimi (*String*). Anna attribuuttien nimiksi *teho* ja *valmistaja*. Tehon oikeelliset arvot ovat nollaa suurempia. Valmistajan nimi ei saa olla **null**-arvoinen ja siinä on oltava vähintään yksi merkki. Kätke attribuutit ja kirjoita niille aksessorit.

   Määrittele tuntemattomalle valmistajalle julkinen luokkavakio (**public static final**) TUNTEMATON, jonka arvo on merkkijono "-".

   Tee parametriton oletusrakentaja, jossa tehoksi asetetaan 3,5 kilowattia ja valmistajan nimeksi TUNTEMATON sekä kaksiparametrinen rakentaja, jonka ensimmäinen parametri välittää tehotiedon (**double**) ja jälkimmäinen parametri nimitiedon (*String*).

   Luokan parametrillinen rakentaja ja arvon asettavat aksessorit heittävät *IllegalArgumentException*-poikkeuksen, jos parametrin arvo on virheellinen.

   Testaa luokkaa *KoneTesti1*-luokassa vähintään siten, että a) luot koneolion molemmilla rakentajilla, b) tulostat lukevien aksessoreiden palauttamat arvot näytölle ja c) aiheutat poikkeuksen antamalla rakentajalle tai asettavalle aksessorille virheellisen arvon ja sieppaat poikkeuksen **try**-**catch**-lauseella.

   Laajempi testaus on toivottavaa, vaikka WETO testaakin luokan toiminnallisuutta varsin kattavasti. Älä jätä omaa testausta tekemättä, sillä sen puuttuminen katsotaan virheeksi.

   Luokat on sijoitettava omiin tiedostoihinsa.

2. Kirjoita Javalla konkreettinen *Traktori*-luokka, jolla on **int-**tyyppinen attribuutti renkaiden lukumäärälle. Attribuutin nimi on *renkaita*. Traktorilla on oltava vähintään kaksi rengasta. Kätke attribuutti ja kirjoita sille aksessorit.

   Peri *Traktori*-luokka edellisen tehtävän *Kone*-luokasta.

   Tee parametriton oletusrakentaja, jossa renkaiden lukumääräksi asetetaan neljä. Tee kolmiparametrinen rakentaja, jossa on järjestyksessä luetellen parametrit yliluokan tiedoille (**double** ja *String*) ja traktorin renkaiden lukumäärälle (**int**). Kutsu parametrillisessa rakentajassa yliluokan parametrillista rakentajaa.

   Luokan parametrillinen rakentaja ja arvon asettava aksessori heittävät *IllegalArgumentException*-poikkeuksen, jos parametrin arvo on virheellinen.

   Testaa luokkaa *TraktoriTesti*-luokassa vähintään siten, että luot traktorioliot molemmilla rakentajilla ja tulostat lukuaksessoreiden palauttamat arvot näytölle. Laajempi testaus on toivottavaa, koska WETO testaa vain osan luokan toiminnallisuudesta. Älä jätä omaa testausta tekemättä, sillä sen puuttuminen katsotaan virheeksi.

   Luokat on sijoitettava omiin tiedostoihinsa.

3. Korvaa *Object*-luokan *toString*-, *equals*- ja *hashCode*-metodit edellisissä tehtävissä kirjoittamissasi *Kone*- ja *Traktori*-luokissa. Tee korvaus *Traktori*-luokassa “ketjuttaen” siten, että kutsut aliluokan korvatuissa metodeissa yliluokassa korvattuja metodeja **super**-attribuutin avulla.

   ToString-metodin korvaus palauttaa merkkijonoesityksen, jossa olion tiedot on erotettu toisistaan välilyönnillä. Jos olion attribuuttien arvot ovat esimerkiksi 36.5, "Valmet" ja 4, niin yliluokassa korvattu metodi palauttaa merkkijonon "36.5 Valmet" ja aliluokassa korvattu metodi merkkijonon "36.5 Valmet 4".

   Koneet ovat samoja, jos niiden teho ja valmistaja ovat samat. Traktorit katsotaan samoiksi, jos lisäksi vielä renkaiden lukumäärä on sama.

   HashCode-metodin luoma hajautuskoodi luodaan kurssin esimerkeissä ja harjoituksissa käytetyllä menetelmällä. Lisää tulokseen *Kone*-luokassa ensin teho ja sitten nimi. Saat **double**- ja **int**-tyyppisten arvojen hajautuskoodit selville *Double*- ja *Integer*-luokkien *hashCode*-metodeilla. Esimerkki:

   ```
   tulos = 31 * tulos + Double.hashCode(teho);
   ```

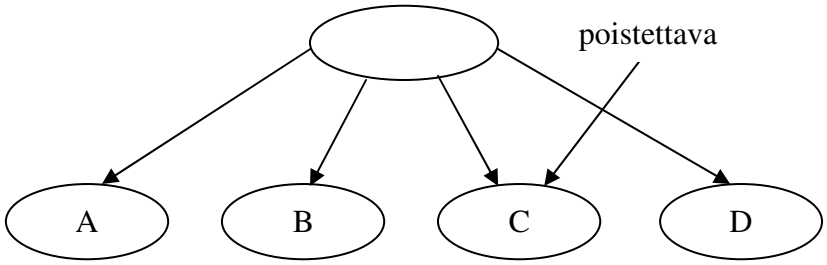
   Testaa korvaamiasi metodeja *ObjectTesti*-luokassa. Laajempi testaus on toivottavaa, koska WETO testaa vain osan luokkien toiminnallisuudesta. Älä jätä omaa testausta tekemättä, sillä sen puuttuminen katsotaan virheeksi.

   Luokat on sijoitettava omiin tiedostoihinsa.

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

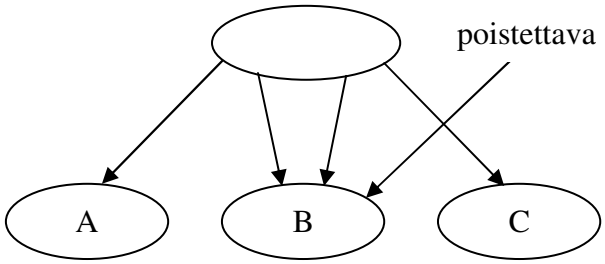
   ```
   public int poista(E poistettava)
   ```

   Metodi poistaa listalta kaikki viitteet, jota liittyvät parametrin viittaamaan tietoon. Tosin sanoen listalta poistetaan kaikki viitteet `x = get(ind)`, joille lauseke `poistettava == get(ind)` on totta. Metodi poistaa useimmiten listalta vain yhden viitteen, koska useimmiten listan viitteet eivät jaa tietoalkioita. Oletetaan esimerkiksi, että listalla on neljä viitettä, jotka liittyvät listan tietoalkioihin seuraavasti:



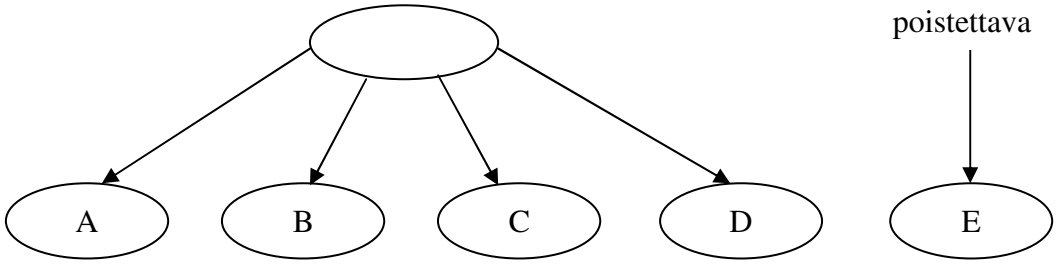
   Jos metodin parametriksi annetaan viite alkioon C, poistetaan kolmas viite ja samalla tietoalkio C. Metodin paluuarvo on yksi.

   Metodi poistaa *n* (> 1) viitettä, jos poistettava viite liittyy alkioon, johon liittyy *n* (> 1) listan viitettä. Näin voi käydä esimerkiksi, kun viite on lisätty listalle *n* kertaa. Tällöin *n* viitettä jakaa yhden tietoalkion. Oletetaan esimerkiksi, että listalla on neljä viitettä, jotka liittyvät kolmeen tietoalkioon siten, että toinen ja kolmas viite liittyvät samaan tietoalkioon:



   Jos operaation parametriksi annetaan viite alkioon B, poistetaan listalta toinen ja kolmas viite sekä B. Metodin paluuarvo on kaksi.

   Lista jää alla olevassa tapauksessa entiselleen, jos metodin parametriksi annetaan E-alkion viite, joka ei liity mihinkään listan tietoalkioista.



   Omaa testiluokkaa ei tarvitse tehdä. Omien testien tekeminen on kuitenkin toivottavaa, koska WETO ei testaa kaikkea mahdollista.