# Harjoitus 4 (viikko 6)

- 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.
- Palauta vastauksesi WETO-järjestelmään (https://wetodev.sis.uta.fi/) viimeistään ensi viikon torstaina 6.2. klo 12.00 (keskipäivä).
- 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.

1. Lisää tehtävässä 3.5 tekemäsi tai mallivastauksena annettu *Karhu*-luokka osoitteessa: https://www.sis.uta.fi/~oope/oope1/kevat-2020/luennot/luento07/esimerkit/pakkauksia/ annettuun *nisakkaat*-pakkaukseen. Älä lisää tehtävän 3.5 testiluokkaa pakkaukseen, vaan sijoita *KarhuTesti*-luokka pakkaushakemiston välittömään ylihakemistoon samalla tavalla kuin luentomateriaalin sivuilla 13.10–13.12 annetussa esimerkissä ja edellä annetussa osoitteessa olevassa esimerkissä on tehty. Ota pakkauksen sisältö käyttöön testiluokassa **import**-lauseella.

   Tiivistä testiluokka ja pakkauksen lähdekoodi zip-muotoiseksi *nisakkaat.zip*-paketiksi siten, että tiedostoon tallentuu myös pakkauksen hakemistotieto, jolloin pakkauksen sisältävä *nisakkaat*-hakemisto muodostuu automaattisesti tiedostoa purettaessa. Oheisessa kuvassa on esitetty pakkauksen ja samalla paketin rakenne. Palauta WETOon vain zip-paketti.

```
KarhuTesti.java
nisakkaat
 |--Ihminen.java
 |--Karhu.java
 |--Kissa.java
 |--Nisakas.java
 |--Tervehtiva.java
```

2. Kokoa tehtävässä 3.2 tehdyt *Henkilo*-, *Poliisi*- ja *Rosvo*-luokat *henkilot*-nimiseen pakkaukseen. Älä lisää *HenkiloTesti*-luokkaa pakkaukseen, vaan sijoita testiluokka pakkaushakemiston välittömään ylihakemistoon. Ota pakkauksen sisältö käyttöön testiluokassa **import**-lauseella.

   Tiivistä testiluokka ja pakkauksen lähdekoodi zip-muotoiseksi *henkilot.zip*-paketiksi siten, että tiedostoon tallentuu myös pakkauksen hakemistotieto. Oheisessa kuvassa on esitetty pakkauksen ja samalla paketin rakenne. Palauta WETOon vain zip-paketti.

```
HenkiloTesti.java
henkilot
 |--Henkilo.java
 |--Poliisi.java
 |--Rosvo.java
```

3. Kurssisivujen *Harjoitukset* | *Tehtävät* -kohdassa on annettu *TiedostonKapistelya*-luokka, jossa luetaan ja tulostetaan tekstitiedosto.

   Korjaa ohjelman *main*-metodi **try**-**catch**-lauseen avulla siten, että saat ohjelman kääntymään ja suoritettua. Varaudu joko samalla tai erillisellä **try**-**catch**-lauseella myös *ArrayIndexOutOfBoundsException*-poikkeukseen, joka tapahtuu, jos ohjelma käynnistetään ilman komentoriviparametria.

   Älä muuta *lueTiedosto*- ja *tulosta*-metodeja millään tavalla äläkä poista *main*-metodissa olevia metodikutsuja.

   Ohjelma tulostaa *ArrayIndexOutOfBoundsException*-poikkeuksen tapahtuessa merkkijonon "Please, remember to give a filename." *IOException*-poikkeuksen tapahtuessa tulostetaan merkkijono "Missing file?" Jonkin muun poikkeuksen merkiksi tulostetaan "Error!" Jos ohjelma ajetaan esimerkiksi komennolla:

   ```
   java TiedostonKapistelya x-file.txt
   ```

   ja hakemistossa ei ole tiedostoa *x-file.txt*, niin ohjelman tuloste on:

   ```
   Missing file?
   ```

4. Lisää *Laite*-luokkaan (tehtävä 3.4) julkinen parametrillinen rakentaja, jolla on **int**-tyyppinen parametri laitteen tilalle. Rakentaja heittää *IllegalArgumentException*-poikkeuksen, jos parametriarvo ei ole *Ohjattava.KÄYNNISSÄ*, *Ohjattava.NUKKUU* tai *Ohjattava.SAMMUTETTU*. Lisää rakentajan otsikkoon **throws**-ilmaus, josta käy ilmi minkä poikkeuksen metodi voi heittää.

   Vihje: luentorungon sivu 14.15.

   Kurssisivujen *Harjoitukset* | *Tehtävät* -kohdassa on annettu *LaiteTesti*-luokka, johon on lisätty uuteen rakentajaan liittyviä testejä. Testaa siksi *Laite*-luokkaasi nimenomaan tällä luokalla.

5. Ajatelma on mietelmä, mietelause tai aforismi. Esimerkiksi "The time is always right to do what is right." on ajaton ajatelma. Kirjoita Java-kielellä *Ajatelma*-luokka, jolla on kätketty *lause*-attribuutti (*String*) ajatelman testille.

   Tee attribuutille julkiset aksessorit. Kirjoita luokalle julkinen yksiparametrinen rakentaja. Tarkista parametrin arvo asetusmetodissa ja rakentajassa. Molemmista metodeista heitetään *IllegalArgumentException*-poikkeus, jos parametrin arvo on **null** tai siinä on vähemmän kuin kaksi merkkiä. Kerro molempien metodien otsikossa **throws**-ilmauksella minkä poikkeuksen metodi voi heittää.

   Huomaa, että voit lyhentää ohjelmaa kutsumalla rakentajassa asetusmetodia, jolloin ei ole tarpeen siepata asetusmetodin heittämää poikkeusta, koska rakentajan tulee heittää sama poikkeus samoissa tilanteissa kuin asetusmetodin.

Kirjoita *AjatelmaTesti*-luokka ja tee sen *main*-metodissa seuraavaa: a) Tee **try**-**catch**-lause, yritä luoda **try**-osassa ajatelma antamalla rakentajalle **null**-arvo ja tulosta **catch**-osassa vapaavalintainen virheilmoitus. b) Tee **try**-**catch**-lause, yritä luoda **try**-osassa ajatelma antamalla rakentajalle liian lyhyt merkkijono ja tulosta **catch**-osassa vapaavalintainen virheilmoitus. c) Tee **try**-**catch**-lause, luo **try**-osassa ajatelma antamalla rakentajalle oikeellinen arvo, kutsu asetusmetodia **null**-arvolla ja tulosta **catch**-osassa vapaavalintainen virheilmoitus. d) Tee **try**-**catch**-lause, luo **try**-osassa ajatelma antamalla rakentajalle oikeellinen arvo, kutsu asetusmetodia liian lyhyellä merkkijonolla ja tulosta **catch**-osassa vapaavalintainen virheilmoitus. e) Luo ajatelma oikeellisella arvolla ja tulosta näytölle lukuaksessorin palauttama arvo.

6. Muokkaa ensimmäisen tehtävän *nisakkaat*-pakkauksesta tämän tehtävän pohjaksi *tehtava6.nisakkaat*-pakkaus. Lisää *Nisakas*-luokkaan **int**-tyyppinen *ikä*-attribuutti ja tee sille aksessorit. Attribuutti ilmaisee nisäkkään iän kokonaisina vuosina. Heitä asettavasta aksessorista *IllegalArgumentException*, jos parametrin arvo on negatiivinen (< 0). Aseta iän arvoksi yksi molemmissa nisäkkään rakentajissa.

   Liitä pakkaukseen kurssisivujen *Harjoitukset | Tehtävät* -kohdassa annettu *Vanhentuva*-rajapinta ja toteuta se *Nisakas*-luokassa siten, että *vanhennu*-metodi kasvattaa ikää yhdellä vuodella. Metodi heittää *IllegalStateException*-poikkeuksen, jos olio ei ole elossa.

   Toteuta *onkoVanha*-metodi nisäkkään aliluokissa. Tässä tehtävässä vanhuus määritellään seuraavasti: kissa on vanha, jos sen ikä on yli 17 vuotta, vanhalla ihmisellä on ikää yli 65 vuotta ja vanha karhu on yli 22 vuoden ikäinen.

   Ota pakkaus käyttöön *VanhentuvaTesti*-luokassa. Luo testiluokassa kissa, ihminen ja karhu. Kutsu iän aksessoreita ja rajapinnan metodeja. Testaa iän asettavaa aksessoria myös negatiivisella arvolla ja sieppaa poikkeus **try**-**catch**-lauseen avulla. Käytä **try**-**catch**-lausetta myös, kun yrität vanhentaa kuollutta oliota.

   Tiivistä testiluokka ja pakkaus zip-paketiksi samalla tavoin kuin ensimmäisessä tehtävässä. Paketin nimi on *vanhentuvat.zip*. Oheisessa kuvassa on esitetty pakkauksen ja samalla paketin rakenne. Palauta WETOon vain zip-paketti.

```
VanhentuvaTesti.java
tehtava6
   |-nisakkaat
        |--Ihminen.java
        |--Karhu.java
        |--Kissa.java
        |--Nisakas.java
        |--Tervehtiva.java
        |--Vanhentuva.java
```

7. Keksi rajapinta ja toteuta se jommassakummassa harjoituksen 3.6. tehtävän luokista. Voit toteuttaa rajapinnan myös osin yli- ja osin aliluokassa, jos yliluokkasi on abstrakti. Rajapinnassa on oltava vähintään yksi metodi. Testaa toteutusta erillisessä testiluokassa. Rajapinnan ja testiluokan tulee olla omissa tiedostoissaan.