Kotitehtävä 3 Deadline 28.9. klo 11:45

Maksimipisteet 10

Valitse tehtäviä niin, että saat niistä maksimissaan 10 pistettä

**Tehtävä 1**: Luokat PlaceToLive ja DisplayPlaces (2p)

Locale, Locale, Locale (Try It Out p. 206)

A PlaceToLive has an address, a number of bedrooms, and an area (in square feet or square meters). Write the PlaceToLive class’s code. Write code for a separate class named DisplayThePlaces. Your DisplayThePlaces class creates a few PlaceToLive instances by assigning values to their address, numberOfBedrooms , and area fields. The DisplayThePlaces class also reads (from the keyboard) the cost of living in each place. For each place, your code displays the cost per square foot (or square meter) and the cost per bedroom.

Kuva, joka sisältää kohteen kuvakaappaus, teksti, Multimediaohjelmisto, ohjelmisto

Kuvaus luotu automaattisesti

**Tehtävä 2**: Tiedoston luku (1p)

On the Record (Try It Out p. 215)

Previously in this chapter, you create instances of your own PlaceToLive class and display information about those instances. Modify the text-based version of your code so that it gets each instance’s characteristics (address, number of bedrooms, and area) from a disk file.

Kuva, joka sisältää kohteen kuvakaappaus, teksti, Multimediaohjelmisto, ohjelmisto

Kuvaus luotu automaattisesti

**Tehtävä 3**: Selitä (2p)

Selitä, mihin sanalla extend viitataan, kun luokka määritellään seuraavasti:

public class FullTimeEmployee extends Employee { …

miten tämä liittyy termeihin subclass ja superclass/child ja parent?

Sana "extend" viittaa siihen, että luodaan uusi luokka, joka perii (inherit) ominaisuudet ja toiminnallisuuden toiselta luokalta. Tässä tapauksessa luokka "FullTimeEmployee" perii (extends) ominaisuudet ja toiminnallisuuden luokalta "Employee". Tämä tarkoittaa, että "FullTimeEmployee"-luokasta tulee "Employee"-luokan aliluokka (subclass), ja "Employee"-luokasta tulee "FullTimeEmployee"-luokan yliluokka (superclass) tai vanhempi luokka (parent class).

Terminologiaa selitettynä:

1. **Subclass (aliluokka)**: "FullTimeEmployee" on aliluokka, mikä tarkoittaa, että se perii "Employee"-luokan ominaisuudet ja toiminnallisuuden. Aliluokka voi laajentaa yliluokan toiminnallisuutta tai lisätä siihen uusia ominaisuuksia.
2. **Superclass (yliluokka) tai Parent Class (vanhempi luokka)**: "Employee" on yliluokka, mikä tarkoittaa, että se antaa perintänä ominaisuudet ja toiminnallisuuden "FullTimeEmployee"-luokalle. Yliluokka voi toimia yleisenä pohjana useammille aliluokille ja määrittää yhteiset piirteet niille.

Joten kun luokkaa "FullTimeEmployee" määritellään tällä tavalla, se laajentaa (extends) "Employee"-luokan toiminnallisuutta ja voi mahdollisesti lisätä siihen omia erityispiirteitään, jotka ovat ainutlaatuisia tämän aliluokan työntekijöille, mutta se perii kaikki perusominaisuudet ja -toiminnallisuuden "Employee"-luokalta. Tämä mahdollistaa ohjelmoinnissa periytyvyyden käytön ja luokkien hierarkian rakentamisen tehokkaasti ja järjestelmällisesti.

**Tehtävä 4**: Uudet alaluokat House ja Apartment ja luokan DisplayThePlaces päivitys (2p)

Buy Or Rent (Try It Out p. 224)

Previously in this chapter, you create instances of your own PlaceToLive class and display information about those instances. Create two subclasses of your PlaceToLive class: a House class and an Apartment class. Each House object has a mortgage cost (a monthly amount) and a property tax cost (a yearly amount). Each Apartment object has a rental cost (a monthly amount).

A separate DisplayThePlaces class creates some houses and some apartments. For each house or apartment, your DisplayThePlaces class displays the total cost per square foot (or square meter) and the total cost per bedroom, both calculated monthly.

**Tehtävä 5**: Luokka Organization\_2, aliluokat ProfitMakingOrganization ja NonProftOrganization (2p)

Tax Breaks (Try It Out p. 225)

In Chapter 7, you create an Organization class. Each instance of your Organization class has a name, an annual revenue amount, and a boolean value indicating whether the organization is or is not a profit-making organization.

Create a new Organization\_2 class. Each instance of this new class has only a name and an annual revenue amount. Create two subclasses: a ProfitMakingOrganization class and a NonProfitOrganization class. A profit-making organization pays 10 percent of its revenue in tax, but a nonprofit organization pays only 2 percent of its revenue in tax.

Make a separate class that creates ProfitMakingOrganization instances and NonProfitOrganization instances while also displaying information about each instance, including the amount of tax the organization pays.

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

**Tehtävä 6**: Selitä (3-5p)

Tutustu Java Documentaatioista löytyvään tutoriaaliin annotaatioista. <https://docs.oracle.com/javase/tutorial/java/annotations/> Kerro, mitkä ovat yleisimmät annotaatiot ja mihin niitä käytetään. Anna myös esimerkkejä, miten annotaatioita käytetään. Tehtävää varten ei tarvitse ymmärtää kaikkea, mitä tutoriaalissa sanotaan eikä tarvitse selittää kaikkea.

Jos selitys on kattava (eli menee jo luennolla käydyn asian ulkopuolelle) ja esimerkit oikein hyviä saat 2 lisäpistettä.

Yleisimmät annotaatiot

Javassa on monia erilaisia annotaatioita, mutta tässä kerrotaan muutamista yleisimmistä.

* @Override: Tämä annotaatio ilmaisee, että metodin on tarkoitus korvata jokin superluokan metodi.
* @Deprecated: Tämä annotaatio ilmoittaa, että metodia tai luokkaa ei pitäisi käyttää enää.
* @SuppressWarnings: Tämä annotaatio estää tiettyjen varoitusten, kuten type safety-varoitusten, näyttämisen.
* @NotNull: Tämä annotaatio kertoo, että muuttujan tai parametrin tulee olla ei-tyhjä.
* @Nullable: Tämä annotaatio kertoo, että muuttujan tai parametrin voi olla myös tyhjä.
* Esimerkkejä annotaatioiden käytöstä

@Override:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, ohjelmisto, Multimediaohjelmisto

Kuvaus luotu automaattisesti

@Deprecated:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

@SuppressWarnings:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

@NotNull:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

Annotaatioiden tyypit

Annotaatioita voidaan jakaa kahteen pääluokkaan: meta-annotaatioihin ja muihin annotaatioihin. Meta-annotaatioita käytetään muiden annotaatioiden määrittämiseen. Muita annotaatioita käytetään ohjelman rakenteeseen, käyttäytymiseen tai tietoihin liittyvien tietojen antamiseen.

Meta-annotaatioita on viisi:

1. @Retention: Määrittää, kuinka kauan annotaatio on voimassa.
2. @Target: Määrittää, mihin elementtiin annotaatio voidaan soveltaa.
3. @Documented: Määrittää, tuleeko annotaatio dokumentoida JavaDoc-dokumentaatioon.
4. @Inherited: Määrittää, periytyvätkö annotaatiot aliluokille.
5. @Repeatable: Määrittää, voidaanko annotaatiota soveltaa samaan elementtiin useita kertoja.

Muita annotaatioita on monia erilaisia, ja niiden käyttötarkoituksia on rajattomasti. Seuraavassa on joitain yleisiä esimerkkejä:

* @Override: Ilmaisee, että metodin on tarkoitus korvata jokin superluokan metodi.
* @Deprecated: Ilmoittaa, että metodia tai luokkaa ei pitäisi käyttää enää.
* @SuppressWarnings: Estää tiettyjen varoitusten, kuten type safety-varoitusten, näyttämisen.
* @NotNull: Kertoo, että muuttujan tai parametrin tulee olla ei-tyhjä.
* @Nullable: Kertoo, että muuttujan tai parametrin voi olla myös tyhjä.

Annotaatioiden käyttöönotto

Annotaatiot lisätään koodiin annotaatiomerkillä. Annotaatiomerkki koostuu annotaation nimestä ja annotaation arvoista. Annotaation nimi on @-merkillä alkava sana. Annotaation arvot ovat avain-arvo-pareja, jotka määrittävät annotaation ominaisuuksia.

Esimerkiksi seuraava koodi lisää @Override-annotaation makeSound()-metodille:

Kuva, joka sisältää kohteen teksti, kuvakaappaus, Fontti

Kuvaus luotu automaattisesti

**Tehtävä 7:** Aliluokka ApartmentWithFees (2p)

Pay More and More (Try It Out p. 230)

In previous sections, you create House and Apartment subclasses of your PlaceToLive class. Create an ApartmentWithFees subclass of your Apartment class. In addition to the monthly rental price, someone living in an ApartmentWithFees pays a fixed amount every quarter (every three months). Create a separate class that displays the monthly cost of living in a House instance, an Apartment instance, and an ApartmentWithFees instance.

**Tehtävä 8**: Vastaa kysymykseen (1p)

Kirjan tehtävä sivu 230, Virtual Methods

Kopioi koodi sivuilta 230-231 ja aja se.

Vastaa seuraavaan kysymykseen:

What output do you see when you run the Main.java file’s code? What does this output tell you about variable declarations and method calling in Java?

Kuva, joka sisältää kohteen teksti, Fontti, kuvakaappaus, Grafiikka

Kuvaus luotu automaattisesti

Tämä tuloste kertoo meille useita asioita muuttujien määrittämisestä ja metodien kutsumisesta Javassa:

1. Polymorfismi: Se, että display-metodia kutsutaan sekä myThing että myThing2 muuttujille ja ne tuottavat erilaisia tulosteita, osoittaa polymorfismia Javassa. Toteutettava metodi riippuu olioiden tyypistä suoritusaikana, ei viittauksen käännösaikaisesta tyypistä.
2. Metodin ylikirjoittaminen: MySubThing- ja MyOtherThing-alaluokat ylikirjoittavat MyThing-yläluokan display-metodin. Tämä mahdollistaa kullekin alaluokalle oman toteutuksen tarjoamisen metodille.
3. Muuttujien julistaminen: Julistukset MyThing myThing, myThing2; määrittelevät kaksi muuttujaa, myThing ja myThing2, tyyppiin MyThing. Ne kuitenkin myöhemmin saavat alaluokkien ilmentymät (MySubThing ja MyOtherThing). Tämä osoittaa, että Javassa voit julistaa muuttujia yläluokan tyypillä ja sijoittaa niihin alaluokkien ilmentymiä (polymorfismi).
4. Metodin kutsuminen: display-metodia kutsutaan sekä myThing että myThing2 muuttujille, mikä näyttää, miten metodeita voidaan kutsua olioihin Javassa. Kutsuttava metodi riippuu varsinaisesta oliotyypistä, johon muuttuja viittaa suoritusaikana.

Yhteenvetona tämä koodi havainnollistaa keskeisiä Java-ohjelmoinnin periaatteita, mukaan lukien polymorfismi, metodien ylikirjoittaminen, muuttujien julistaminen yläluokilla sekä dynaaminen metodien lähettäminen suoritusaikana