Kotitehtävä 4 Deadline 5.10. klo 11:45

Maksimipisteet 10

Valitse tehtäviä niin, että saat niistä maksimissaan 10 pistettä

**Tehtävä 1:** Student luokka, Major enum (2 p)

School Days (Try It Out p. 244)

Create a Student class with a name, an ID number, a grade point average (GPA), and a major area of study. The student’s name is a String. The student’s ID number is an int value. The GPA is a double value between 0.0 and 4.0. The Major is an enum type, with values such as COMPUTER\_SCIENCE, MATHEMATICS , LITERATURE , PHYSICS , and HISTORY.

Every student has a name and an ID number, but a brand-new student might not have a GPA or a major. Create constructors with and without GPA and Major parameters. As usual, create a separate class that makes use of your new Student class.

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

Kuvaus luotu automaattisesti

**Tehtävä 2:** AirplaneFlight luokka, Airport enum (3p)

Tässä tehtävässä tehdään osa kirjan Flight of Fancy tehtävästä. Tässä harjoituksessa käytetään pelkästään suomalaisia lentokenttiä, joten sinun ei tarvitse välittää aikaeroista. Jatketaan tämä tehtävä loppuun myöhemmin. Saat tietysti kokeilla tehdä tehtävää niin kuin se on esitetty kirjassa.

Flight of Fancy (Try It Out p. 244)

Create an AirplaneFlight class with a flight number, a departure airport, the time of departure, an arrival airport, and a time of arrival. The flight number is an int value. The departure and arrival airport fields belong to an Airport enum type, with values corresponding to some of the official IATA airport codes. Create 6 airports, for example, HEL, TMP, JYV, TKU and OUL.

Every flight has a number, a departure airport, and an arrival airport. But some flights might not have departure and arrival times. All the departure and arrival times are given in Finnish time. Create constructors with and without departure and arrival time parameters. Create a separate class that makes use of your new AirplaneFlight class.

Create a subclass that has a method named duration. The duration method, which has no parameters, returns the amount of time between the flight’s departure time and arrival time.

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

Kuvaus luotu automaattisesti

**Tehtävä 3:** getString method inside a subclass (2p)

Student Showcase (Try It Out p. 252)

In the first exercise, you created your own Student class. Create a subclass that has a method named getString. Like the display method in this chapter’s TemperatureNice class, the getString method creates a nice-looking String representation of its object. But unlike the TemperatureNice class’s display method, the getString method doesn’t print that String representation on the screen. Instead, the getString method simply returns that String representation as its result.

(In a way, a getString method is much more versatile than a display method. With a display method, all you can do is show a String representation on the screen. But with a getString method, you can create a String representation and then do whatever you want with it.)

Create a separate class that creates some instances of your new subclass and puts their getString methods to good use.

Tehtävä 4: Painoindeksi (2p)

Tee luokka Henkilo. Luokalla on seuraavat muuttujat: nimi, paino, pituus. Luo Henkilo niminen konstruktori, jolla on parametrina nimi, alusta muut muuttujat arvolla 0.

Luo luokalle setPituus, setPaino ja painoindeksi metodit. Painoindeksi lasketaan paino/pituus2. Pituus ilmoitetaan metreinä ja paino kiloina.

Luo luokalle vielä display metodi, jossa tulostat kunkin henkilön tiedot seuraavasti:

Nimi, painoindeksisi on painoindeksi

Eli esim.

Matti, painoindeksisi on 26,54.

Teen sen jälkeen main-funktio, jossa käytät äsken luotuja metodeja, esim.

Henkilo matti = new Henkilo(”Matti”);

matti.setPituus(180);

matti.setPaino(86);

matti(display);

Tehtävä 5: Selitä (3p selityksistä ja 3p esimerkeistä, yht. 6p)

Selitä omin sanoin seuraavat termit, mitä ne tarkoittavat ja milloin niitä käytetään. Anna käytöstä esimerkki.

1. Konstruktori

Konstruktori on erityinen metodi, joka suoritetaan, kun luot uuden olion tietylle luokalle. Se on vastuussa luokan olioiden alustamisesta ja asettaa niiden alkutilan. Konstruktori on nimetty samalla nimellä kuin luokka, ja se voi ottaa parametreja, jotka määrittelevät sen, miten olio alustetaan.

Esimerkissäni **Auto**-luokassa on seuraava konstruktori:

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

Kuvaus luotu automaattisesti

Tämä konstruktori ottaa kaksi parametria: **merkki** ja **malli**. Kun luon uuden **Auto**-olion, kuten teen Auto**Main**-metodissa:

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

Kuvaus luotu automaattisesti

Nämä parametrit (esimerkiksi "Toyota" ja "Corolla") lähetetään konstruktorille. Konstruktori käyttää sitten näitä parametreja asettaakseen **merkki**- ja **malli**-kentät oikeille arvoille. Tämä tarkoittaa, että jokaiselle luodulle **Auto**-oliolle annetaan omat merkki ja malli, jotka määritellään luonnin yhteydessä.

Konstruktori mahdollistaa siis olioiden alkutilan määrittämisen ja varmistaa, että niiden tietoja ei unohdeta tai jää vahingossa asettamatta.

1. Enum

Enum (lyhenne sanoista "enumerated type") on erityinen tyyppi Java-ohjelmointikielessä, joka koostuu määrätystä joukosta vakioita. Enumit luodaan luokan tavoin, mutta ne voivat sisältää vain ennalta määritettyjä vakioita, joita kutsutaan usein "enumerointiarvoiksi" tai "enum-arvoiksi". Enumit ovat hyödyllisiä, kun halutaan rajoittaa tiettyjen arvojen käyttöä ja tehdä koodista selkeämpää ja ymmärrettävämpää.

Alla on esimerkki enumin käytöstä:

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

Kuvaus luotu automaattisesti

Tässä **tanaan**-muuttujalle annetaan arvoksi **Viikonpaiva.TIISTAI**, joka vastaa tiistain enum-arvoa.

Enumit tekevät koodista helposti luettavaa ja ymmärrettävää, koska ne rajoittavat sallittujen arvojen joukkoa ja tarjoavat selkeät nimet näille arvoille. Ne ovat erityisen hyödyllisiä tilanteissa, joissa on tiettyjä kiinteitä arvoja, jotka eivät saisi muuttua suorituksen aikana, kuten viikonpäivät, kuukaudet tai lippuarvot.

1. Super

Java-ohjelmointikielessä "super" on avainsana, joka viittaa yliluokan (superclass) ominaisuuksiin ja metodeihin. Se käytetään yleensä aliluokassa (subclass) ilmaisemaan, että halutaan kutsua yliluokan konstruktoria tai metodia.

Alla on esimerkki siitä, miten "super" toimii

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

Kuvaus luotu automaattisesti