Metropolia

Oliosovellukset ja tietokannat

TEHTÄVÄSARJA 5

Kevät 2022

SÄIKEET

Lue kurssikirjan luvusta 10 sivut 393-402.

Tavoite

Oppia kirjoittamaan ja käyttämään säikeitä sekä ymmärtämään säikeiden toimintalogiikka.

Käsitteitä

rinnakkaisuus, säie, pääsäie, säikeen määrittely, säieturvallisuus ja volatile-määre, start(), yield(). sleep(), join(), säikeiden vuorottaminen (scheduling), säikeen tilakaavio, prioriteetti

Alkutoimet: Kopioi tehtäväsarjan koodit (**OlsoTehtSarja5**) kurssin repositoriosta omaan repoosi ja luo Eclipse-projekti ensimmäisen tehtäväsarjan ohjeita soveltaen. Projektissa on tehtävässä mainitut luokat, mutta koodi puuttuu. Projektin `tests`-pakkauksessa olevat koodit eivät käänny ennen kuin olet kirjoittanut tehtävien koodit.

Tässä tehtäväsarjassa

- Luokat ovat **saikeet** -pakkauksessa
- Testipääohjelmat ovat projektin oletuspakkauksessa: **default package**
- Luokkien muuttujat on määritelty private näkyvyysmääreellä, joten ne näkyvät vain omaan luokkaansa. Luokkamuuttujalla on määre static.
- Metodeilla on määre public. Luokkamuuttujalle on public static getteri ja setteri.

1. Ohjelmoi luokan Threads aliluokka **OmaSaie**, jossa on private näkyvyydellä instanssimuuttuja `luku` ja luokkamuuttuja (static) `yhteisetAjokerrat`.

   Kirjoita parametriton konstruktori, joka ei tee mitään sekä parametrillinen konstruktori, joka asettaa arvon instanssimuuttujalle `luku`

   Kirjoita muuttujille getterit ja setterit. Voit hyödyntää automaattista koodin generointia: määrittele ensin muuttujat, ja napsauta sitten editori-ikkunassa hiiren oikeaa ja valitse Source... tai käytä suoravalintaa Alt-Shift+S.

   Kirjoita `run()`-metodi, joka kasvattaa kummankin muuttujan arvoa yhdellä.

   Aja testit (OmaSaieTest.java).

   Täydennä oletuspakkauksessa oleva ohjelma (**OmaSaieMain**) siinä olevien kommenttien mukaisesti. Suorituta säiettäsi ja totea ero instanssimuuttujan ja luokkamuuttujan käsittelyssä. Mikä ero niillä on?

2. Kirjoita säieluokka **LuuppaavaSaie**, jossa on pakkaustason näkyvyydellä määritellyt int-tyyppiset instanssimuuttujat `pyydetytKierrokset` ja `kierretytKierrokset`

   Kirjoita vähintään seuraavat metodit
   a) parametrillinen konstruktori, joka asettaa pyydetyt kierrokset
   b) getteri kierretyille kierroksille
   c) `run()`-metodi, jossa säie kiertää toistossa pyydetyt kierrokset kertaa siten, että kierrettyjen kierrosten loppuarvoksi jää toiston päätyttyä kierretyt kierrokset. Sen tulee olla lopuksi siis sama kuin pyydetyt kierrokset. Tulosta jokaisella kierroksella kierroksen numero.

Aja testit (LuuppaavaSaieTest.java).

Täydennä oletuspakkauksessa olevaa ohjelmaa **LuuppaavaSaieMain**: luo säie halutuilla kierroksilla, käynnistä säie ja katso kuinka käy.

3. Kirjoita säieluokka **Juoksija**, jossa juoksija juoksee kilparatta ympäri uudelleen ja uudelleen, kunnes pyydetään lopettamaan. Koodaa kierrosnumeroa varten instanssimuuttuja `kierrokset` ja tee sille getteri.

   Säie tulostaa jokaisella kierroksella kierrosnumeron ja menee sitten nukkumaan 100 ms ajaksi (~yhden kierroksen juoksu): `Thread.sleep(100)`.

   Koodaa lopetusehto `run()`-metodin while-silmukkaan käyttäen boolean-muuttujaa `jatkuu`. Arvo true tarkoittaa, että toisto jatkuu ja false, että pitää lopettaa. Kirjoita metodi `void lopeta()`, joka muuttaa jatkuu-muuttujan epätodeksi. Kun juokseminen halutaan lopettaa, joku toinen säie kutsuu tätä Juoksija-olion lopeta()-metodia. Kirjoita myös metodi `isJuoksemassa()`, joka palauttaa true, jos juoksija jatkaa juoksemistaan.

   Aja testit (JuoksijaTest.java).

   Täydennä oletuspakkauksen ohjelmaa **JuoksijaMain** siten, että käynnistät juoksijasäikeen ja odotat (sleep) jonkun aikaa ennen kuin pysäytät säikeen. Hae vielä säikeeltä juostujen kierrosten määrä ja tulosta se.

4. Kirjoita säieluokka **Kilpajuoksija**, jossa juoksija juoksee 400 metriä. Erona edelliseen on se, että nyt juoksija lopettaa itse, kun 400 metriä on täynnä.

   Kirjoita konstruktori, joka asettaa juoksijalle luontivaiheessa kilpailijanumeron. Pohjassa on tätä tarkoitusta varten staattinen luokkamuuttuja seuraavaNumero. Sen voi alustaa haluamaansa alkuarvoon metodilla setSeuraavaNumero().

   Toteuta säikeen `run()`-metodi siten, että kilpajuoksija etenee silmukassa 10 metriä kerrallaan. Arvo satunnaislukugeneraattorilla 10 metrin juoksuaika realistisesti (Wayde van Niekerkin vuonna 2016 juoksema maailmaennätys on 43,03 s.). Säie nukkuu 10 metrin juoksun ajaksi ja jatkaa sen jälkeen. Tulosta juoksijan numero, juostu matka sekä väliaika 50 m välein.

   Tehtävään on yksi testi (KilpaJuoksijaTest.java), joka edellyttää, että toteutat metodin `double getAika()`. Se palauttaa loppuajan sekunteina (pitää olla välillä 43-60).

   Kirjoita oma pääohjelma **KilpajuoksijaMain** (ei valmista koodipohjaa), jolla voit tsekata, että esimerkiksi tulostukset menevät oikein.

5. Tässä tehtävässä käytetään edellisen tehtävän **Kilpajuoksija**-luokkaa.

   Kirjoita uusi pääohjelma **KilpailuMain** oletuspakkaukseen. Luo siinä 8 kilpajuoksijaa (taulukkoon tai listaan) ja pistä ne juoksemaan kilpaa.

   Tähän ei liity JUnit-testejä.

Jatkotehtävä

6. Pohdiskele ja kokeile, voitko vaikuttaa lopputulokseen prioriteeteilla. Jos et, niin miksi et? Selitä tehtäväpalautuksen tekstikentässä.

   Kopioi Kilpajuoksija-luokka nimelle KilpajuoksijaP, ja muuta ratkaisua siten, että prioriteeteilla on ylipäätään jotain vaikutusta suoritukseen. Vihje: Älä käytä lainkaan sleep()-rutiinia tai säikeen omaa tulostusta. Reaaliaikaa voi mitata metodeilla System.currentTimeMillis() tai System.nanoTime(). Ks. API.

   Tähän ei liity JUnit-testejä.