Metropolia

Oliosovellukset ja tietokannat

TEHTÄVÄSARJA 6

Kevät 2022

SÄIKEIDEN SYNKRONOINTI

Tavoite

Oppia kirjoittamaan, käyttämään ja ymmärtämään synkronoituja tietorakenteita.

Käsitteitä

Yhteiskäyttöisyys, atomisuus, synkronointi, synchronized, wait(),notify(), notifyAll()

Alkutoimet: Kopioi tehtäväsarjan koodit (**OlsoTehtSarja6**) kurssin tietovarastosta omaan repoosi ja luo Eclipse-projekti ensimmäisen tehtäväsarjan ohjeita soveltaen. Olso-repossa on pohja tehtävään, ei muuta. Tähän sarjaan ei liity JUnit-testejä.

1. Kokeile tunnilla esitetyn Kirjoittaja-**IntSailio**-Lukija ohjelman toimintaa (repossa). Aja ohjelma, ja totea, että kirjoitettujen ja luettujen lukujen summa on 100.

   Lisää ohjelmaan kaksi uutta lukijasäiettä. Kukin lukija tulostaa itse lukemiensa lukujen summan. Kun eri lukijoiden summat lasketaan yhteen, yhteissumman tulisi olla jälleen 100. Aja koodia useita kertoja ja totea, että näin ei kuitenkaan aina ole.

   Katso koodista, kuinka Kirjoittaja-säie ilmaisee setInt()-rutiinissa, että se ei enää tuota lukuja. Ongelmana alkuperäisessä ohjelmassa on tapa, jolla Lukija-säie käsittelee getInt()-rutiinissa kirjoittajan viimeiseksi viemän luvun -1. Joudut siis tekemään pienen muutoksen koodiin.
   Vihje: Sopivan kuvan piirtäminen saattaa auttaa ajattelutyötäsi…

2. a) Toteuta **SharedXY**-olio (= tietorakenne), jossa on instanssimuuttujina kaksi kokonaislukumuuttujaa `x` ja `y`. Tee konstruktori, joka asettaa muuttujille alkuarvot sekä metodi `swap()`, joka vaihtaa arvot keskenään sekä tulostaa arvot vaihdon jälkeen. Lisää myös getterit.

   Ohjelmoi säieluokka **Swappaaja**, joka käyttää yllä mainittua tietorakennetta. Swappaaja-olio saa tietorakenteen konstruktorin parametrina. Toteuta säieluokkaan run()-metodi, joka vaihtaa toistuvasti yhteiskäyttöisen tietorakenteen muuttujien arvoja keskenään. Säie lopettaa toimintansa, jos se huomaa, että x- ja y-muuttujien arvot ovat samat. Kun tilanne tulee eteen, lopeta koko ohjelma suoritus kutsumalla System.exit(0). Tilannetta ei pitäisi syntyä, mikäli sekä vaihtaminen että arvojen x ja y tutkiminen toteutuvat kumpikin atomisina (jakamattomina, keskeytymättöminä) operaationa.

   Tee luokka **SwappaajaMain**, jossa luot ensin SharedXY-tietorakenteen ja sen jälkeen useita Swappaaja-säikeitä (jotka tekevät toistuvasti vaihtoja). Totea, että saat aikaan ei-toivotun tilanteen, jossa ilman synkronointia muuttujien arvot ovat lopulta samat.

   b) Korjaa koodia (SharedXY-luokkaa ja Swappaaja-luokkaa) siten, että synkronointi ratkaisee ongelman.

   Mieti tarkkaan, kuinka voit tutkia ovatko muuttujat x ja y jollain ajan hetkellä samat. Toimiva ratkaisu ei ole se, jossa pyydät arvot yhteiskäyttöiseltä SharedXY-oliolta erikseen tyyliin getX() ja getY() ja vertaat saatuja arvoja, koska ne eivät välttämättä palauta minkään tietyn hetken tilannetta. Arvoja pyytävä säie voi nimittäin keskeytyä getX() ja getY() kutsujen välissä, ja joku toinen säie voi käydä siinä välissä swappäämässä.

   [Huomaa, että tässä ei ole tarvetta wait/notify-tekniikan käyttöön**.** Aina on swapättävää.]

3. ja 4.

   Toteuta **Halkovarasto**, johon mahtuu vakiomäärä halkoja. Halkovarasto ”tarjoaa” vain kaksi operaatiota: vieHalkoja() ja haeHalkoja() (älä ohjelmoi muita public-metodeja).

   Halonhakkaajat täyttävät varastoa hakattuaan yhden pölkyn haloiksi (halkojen määrä per pölkky vaihtelee). Jos varastoon ei mahdu kaikki uudet halot, joutuu halonhakkaaja odottamaan, kunnes varasto tyhjenee niin paljon, että kaikki halot mahtuvat sinne. Vähemmän halkoja tuova halonhakkaaja pystyy ohittamaan enemmän halkoja tuovan halonhakkaajan, mikäli hänen halkonsa mahtuvat varastoon. Huomaa, että ohitusta ei tarvitse koodata, koska järjestelmä huolehtii siitä automaattisesti. Mieti, kuinka!

   Partiolaiset paistavat nuotiolla lettuja ja käyvät hakemassa halkoja kukin aina ennalta sovitun määrän, joka vaihtelee partiolaisittain. Ellei halkoja ole haluttua määrää, partiolainen joutuu odottamaan. Se, jolla on pienempi halkojen tarve, voi ohittaa tätä enemmän halkoja tarvitsevan, mikäli halkoja on hänelle riittävästi. Huomaa jälleen, että tämä tapahtuu automaattisesti!

   Kirjoita ohjelma **HalkohommaMain**, jolla simuloit halkovaraston, halonhakkaajien ja partiolaisten toimintaa. Havainnollista toimintaa välitulostuksilla niin hyvin kuin pystyt.

   Tässä Halkovarasto on se yhteyskäyttöinen resurssi, jota käyttää monta säiettä. Ohjelmoi kaikki synkronointiin liittyvä sinne. Luo pääohjelmassa kaikki tarvittavat oliot, tee tarvittavat kytkökset ja käynnistä monta Halonhakkaaja-säiettä ja monta Partiolainen-säiettä.

Jatkotehtävä

5. (Virén Münchenissa 1972, https://www.youtube.com/watch?v=MkXsjfVnG0k) Toteuta edellisessä tehtäväsarjassa olleeseen säikeiden juoksukilpailuun yhdelle säikeelle “kaatuminen”. Pääohjelma arpoo säikeen, jonka se kaataa (keskeyttää valitsemakseen ajaksi) ja jonka se nostaa pystyyn kaatumisen jälkeen.

   Tee tehtäväsarjaan pakkaus `kilpailu`, ja kopioi KilpailuMain.java ja Kilpajuoksija.java sinne. Koodaa Kilpajuoksija-luokkaan wait/notify-tekniikkaan perustuvat kaadu() ja nouse().