2 Tehtävät

2.1 Tehtävä 3-1

Palataan harjoituskerran 2 tehtävään 2-1, jossa aiheena oli 2. asteen yhtälöiden ratkaiseminen. Alkuperäisen ohjelman tarkoitus lienee ollut laskea 2. asteen yhtälön reaaliset ratkaisut, ts. ratkaisut jotka voidaan esittää reaaliluvuilla, tai alkuperäisen ohjelman kontekstissa reaalilukujen approksimaatiolla, `double`-tyyppisellä datalla Javassa.

Aiemmin antamastasi määrittelystä riippuu, millainen semantiikka ohjelmalle lopulta muodostui. Tai jos et tehnyt tehtävää, käytä tehtävän pohjana tehtävän 2-1 malliratkaisua (ilmestyy Moodleen). Huomaa, että jos määrittelemäsi ohjelma on ristiriidassa seuraavan ohjeistuksen kanssa, määritelmää on muutettava (esim. virheilmoitukset ja virheiden käsittelyt pois) niin, että ohjelman uusi toiminnallisuus on käytettävissä.

Ohjelman toiminnan uusi tehtävänanto: Harjoituksen 2-1 ohjelmaa (https://gitlab.utu.fi/jmjmak/oom2019-harj2-1) halutaan laajentaa niin, että se laskee reaalisten ratkaisujen sijaan yleiset ratkaisut myös silloin, kun $b^2 < 4ac$. Nämä ratkaisut ovat ilmaistavissa kompleksiluvuilla (https://fi.wikipedia.org/wiki/Kompleksiluku). Käytännössä tehtävässä riittää huomioida $\sqrt{-x} = \sqrt{-1} \times \sqrt{x} = i \times \sqrt{x}$, kun $x \geq 0$ ja käänteisesti: $i^2 = (\sqrt{-1})^2 = -1$. Saatua lukuparia (reaaliosa ja kompleksiosa kerrottuna i:llä) voidaan käsitellä kompleksilukujen laskusääntöjen mukaan (vertaa yhteen- ja vähennyslaskuissa polynomiin). Esimerkiksi $(i+1)+(-2i+10) = (-i+11)$ ja $(i+1)\times(i+1) = i^2+2i+1 = -1+2i+1 = 2i$.

Tehtävän kannalta voidaan olettaa, että käyttäjän antamat syötteet ovat aina reaalilukuja (jäsennettävissä double-tyyppiseksi). Eli pelkästään ohjelman tuloksena voi olla kompleksilukuja.

a) Anna kompleksiluvuille toteutus olio-ohjelmoinnin menetelmin. Toteutus riittää tehdä neljän perusoperaation suhteen (plus-, miinus-, kerto- ja jakolasku). Luvun arvon voi tulostaa esimerkiksi `System.out.println` -menetelmällä osa kerrallaan. Totea ohjelman graafisen käyttöliittymän kautta, että ratkaisu toimii muutamalla esimerkkiarvolla. Esim. kun $a = 1, b = 3, c = 2$ ja $a = 1, b = 2, c = 5$. Vinkki: idea on etsiä Javan rakenteista mielivaltaisen kompleksiluvun $xi + y$ mallintava keino – ratkaisua ei kannata hakea Javan perustieto- ja luokkatyppien ulkopuolelta.

   Tehtävä painottaa tiedon hallintaa algoritmiikan sijaan. Huomaa mahdolliset tulkinnanvaraisuudet ja moniselitteisyys datan koodaamisessa Javan tyypeiksi.

b) Määrittele kompleksilukujen toteutus kurssin olio-ohjelmoinnin menetelmin (lähinnä alku- ja loppuehdot sekä luokkainvariantti) ja laadi yksikkö- tai ominaisuustestejä, jotka testaavat yhdellä testillä luokan kutakin laskuoperaatiota lukuarvolla, joka koostuu sekä kompleksi- että reaaliosasta. Huom, tätä testaamista varten toteutuksen pitää mahdollisesti olla laajemmin toteutettu kuin a)-kohdan graafinen ohjelma edellyttäisi. Kokeile, saatko rajattua testin syötettä niin, että esim. nollalla jakamista ei vahingossa testata.

c) Jos et käyttänyt a)- ja b)-kohdissa pohjana tehtävän 2-1 projektirunkoa, sovita ratkaisusi runkoon, jolloin ratkaisu on palautettavissa gitlabiin. Tarkista, että gitlabin CI-testi suoriutuu onnistuneesti kuten myös Maven-testi paikallisesti – komentoriviltä (`mvn clean compile test`) ja Eclipsestä ja että Gitlabiin ladattuna projektiin ilmestyy symboli onnistuneesta CI-ajosta (vihreä ruksi ympyrän sisällä ja seliteteksti `Commit: passed`).