Harjoitus 1 (viikko 36)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (*jorma.laurikkala@uta.fi*).
- Kolmesta symbolista (aloitus- ja lopetussymbolit ja yksi toimintosymboli) koostuvia vuokaavioita ei hyväksytä vastaukseksi mihinkään alla olevista tehtävistä. Kaavioissa on siis oltava ainakin neljä symbolia.
- Operaatioita sisältävissä vuokaavioissa näppäimistöltä lukeminen hoidetaan esimerkiksi luennolla käsitellyllä **lue**-operaatiolla tai jollakin vastaavalla itse määritellyllä operaatiolla. Samoin näytölle tulostamiseen voi käyttää luentokalvoilla esiintynyttä **tulosta**-operaatiota tai omaa operaatiota. Tehtävissä ei ole tärkeintä se, miten luetaan ja tulostetaan vaan algoritmien hahmottelu vuokaavion avulla.
- Löydät tietoa vuokaavioiden piirrosta kurssisivujen *Linkit* | *Vuokaavioiden piirto* -kohdasta.
- Ensi viikolla pidettävissä mikroharjoituksissa ja Luupin koodauspajassa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään torstaina 8.9. **klo 23.55**. Takaraja on ensi viikolla normaalia myöhemmin yliopiston avajaisten vuoksi.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon **perjantaina 9.9. klo 14–16 (B1083)**. Aika on vaihtunut, jotta avajaisten alta torstaille siirretyt mikroharjoitusryhmät saadaan pidettyä ennen luentosaliharjoituksia. Luentosaliharjoitusryhmä palaa alkuperäiseen aikaan ja paikkaan toisista viikkoharjoituksista alkaen.

1. Selvitä mitä liitteessä 1 vuokaaviona esitetyn algoritmin mukaan tapahtuu, kun

   a) ei laiskota pahasti,
   b) laiskottaa pahasti ja koira ei ole paikalla ja
   c) laiskottaa pahasti ja koira on paikalla, mutta koira onkin megalomaanikko.

2. Selitä kuinka elämää eletään liitteessä 2 annetun vuokaavion mukaan.

3. Aivoinfarkti on vakava tila, jossa nopea hoitoon pääseminen on tärkeää. Yhdysvalloissa aivoinfarktin tunnistamiseen ja siihen reagointiin käytetään FAST-muistisääntöä. Face: Toimivatko kasvojen lihakset normaalisti? Esimerkiksi onko hymy toispuoleinen? Arms: Onko toinen käsivarsi heikko tai tunnoton molempia käsivarsia nostettaessa? Speech: Onko puhe epäselvää tai puheentuotto vaikeaa? Time: Soita heti hätänumeroon ja laita soittoaika ylös, jos yksi tai useampi edellä kuvatuista testeistä viittaa aivoinfarktiin.

   Piirrä vapaamuotoinen vuokaavio (ei muuttujia ja operaatioita), jossa esittää kuinka FAST-säännön mukaan toimitaan. Vuokaaviossa ei tarvitse olla silmukkaa. Yllä annettuja kysymyksiä ei tarvitse kirjoittaa päätössymboleiden ehtoihin sellaisenaan. Voit ilmaista Face-, Arms- ja Speech-päätökset lyhemmin sinulle sopivammalla tavalla. Esimerkiksi Arms-päätöksen voisi lausua lyhyesti “Avuton raaja?”.

4. Kuvaa jokin algoritmi vapaamuotoisesti (ilman muuttujia ja operaatioita) vuokaaviona. Esitä myös mahdollisesti tarvittavat oletukset. Huomaa, että algoritmisi ei saa olla luentomateriaalissa esitetty esimerkki.

5. Oletetaan, että eräässä kännykässä on herätyskellosovellus, joka soittaa aluksi hälytyksen ja kysyy sitten haluaako käyttäjä torkkua. Jos käyttäjä vastaa “kyllä”, sovellus soittaa torkkuhälytyksen viiden minuutin päästä ja kysyy uudelleen haluaako käyttäjä torkkua. Sovellus antaa torkkuhälytyksen korkeintaan 10 kertaa. Sovellus pysähtyy, jos käyttäjä vastaa joko alkuperäisen hälytyksen tai minkä tahansa torkkuhälytyksen jälkeiseen kysymykseen “ei” tai käyttäjä on vastannut 10 kertaa peräkkäin “kyllä”. Piirrä vapaamuotoinen vuokaavio (ei muuttujia ja operaatioita), joka kuvaa kuinka sovellus toimii. Vuokaaviossa kannattaa käyttää silmukkaa, koska ilman silmukkaa kaaviosta tulee järkyttävän suuri.

6. Esitä vuokaaviona muuttujia ja operaatioita sisältävä algoritmi, jonka laskee tuotteelle alennetun hinnan. Algoritmi kysyy käyttäjältä tuotteen hinnan euroina ja alennusprosentin, laskee uuden hinnan ja tulostaa sen näytölle. Voit olettaa, että hinta on nollaa suurempi ja että alennusprosentti on nollaa suurempi ja toisaalta alle 100 prosenttia.

7. Esitä vuokaaviona muuttujia ja operaatioita sisältävä algoritmi lukujono positiivisten (≥ 0, oli alun perin >, korjattu 19.9.) lukujen lukumäärän laskemiseen. Algoritmi lukee aluksi käyttäjältä lukujen lukumäärän. Tämän jälkeen jonon luvut luetaan yksi kerrallaan ja päätellään positiivisten lukujen lukumäärä. Algoritmi tulostaa lopuksi positiivisten lukujen lukumäärän näytölle, jos lukujen lukumäärä oli vähintään yksi. Muussa tapauksessa tulostetaan sopiva virheilmoitus. Jos käyttäjä antaa jonon lukujen lukumääräksi esimerkiksi 4 ja luvut ovat 42, 6, 71 ja -1, on tulos 3, koska kolme ensimmäistä lukua olivat positiivisia.

   Tarvitset algoritmiisi laskurin ohjaaman silmukan, joka muistuttaa luentorungon sivulla 3.22 annetun algoritmin silmukka. Huomaa, että tämän tehtävän silmukan täytyy sisältää päätös, jotta voit kasvattaa positiivisten lukujen lukumäärän sisältävän muuttujan arvoa.

8. Java-ohjelmointi aloitetaan ensi viikolla. Tarvitset ohjelmoinnissa komentoikkunaa ja Java JDK -ohjelmistoa. Näistä on kerrottu kurssisivujen *Ohjelmointivälineitä*-kohdan alla.

   Kirjaudu ensin joko omalle tai mikroluokan koneelle. Asenna omalle koneellesi tarvittaessa Java JDK kurssisivujen ohjeiden mukaan, koska tässä tehtävässä pitää käynnistää tutustumismielessä Java-kääntäjä. Java JDK on mikroluokkien koneilla valmiina.

   Tee kurssille oma kansio. On suositeltavaa, teet kurssikansioon kansiot harjoituksille ja tähän kansioon vielä kansiot kullekin viikkoharjoituskerralle. Esimerkiksi tämän harjoituksen tiedostot voisivat olla kansiossa *laki1* | *harjoitukset* | *harjoitus1*. Mikroluokan koneella kansiot kannattaa tehdä kotihakemistoon (X-asemalle), koska C-asemalle tehdyt tiedostot ja hakemistot tuhoutuvat, kun kirjaudut koneelta ulos.

   Avaa koneella komentoikkuna ja siirry siinä *cd*-komennolla johonkin luomistasi kansioista. Voit vaihtaa Windowsissa asemaa kirjoittamalla aseman kirjaimen ja heti sen perään kaksoispisteen ja painamalla sitten *Enter*. Esimerkiksi mikroluokan koneilla kotihakemistoon pääsee kirjoittamalla *x:* ja painamalla *Enter*.

   Pyydä lopuksi Java-kääntäjää kertomaan versionsa kirjoittamalla komento:

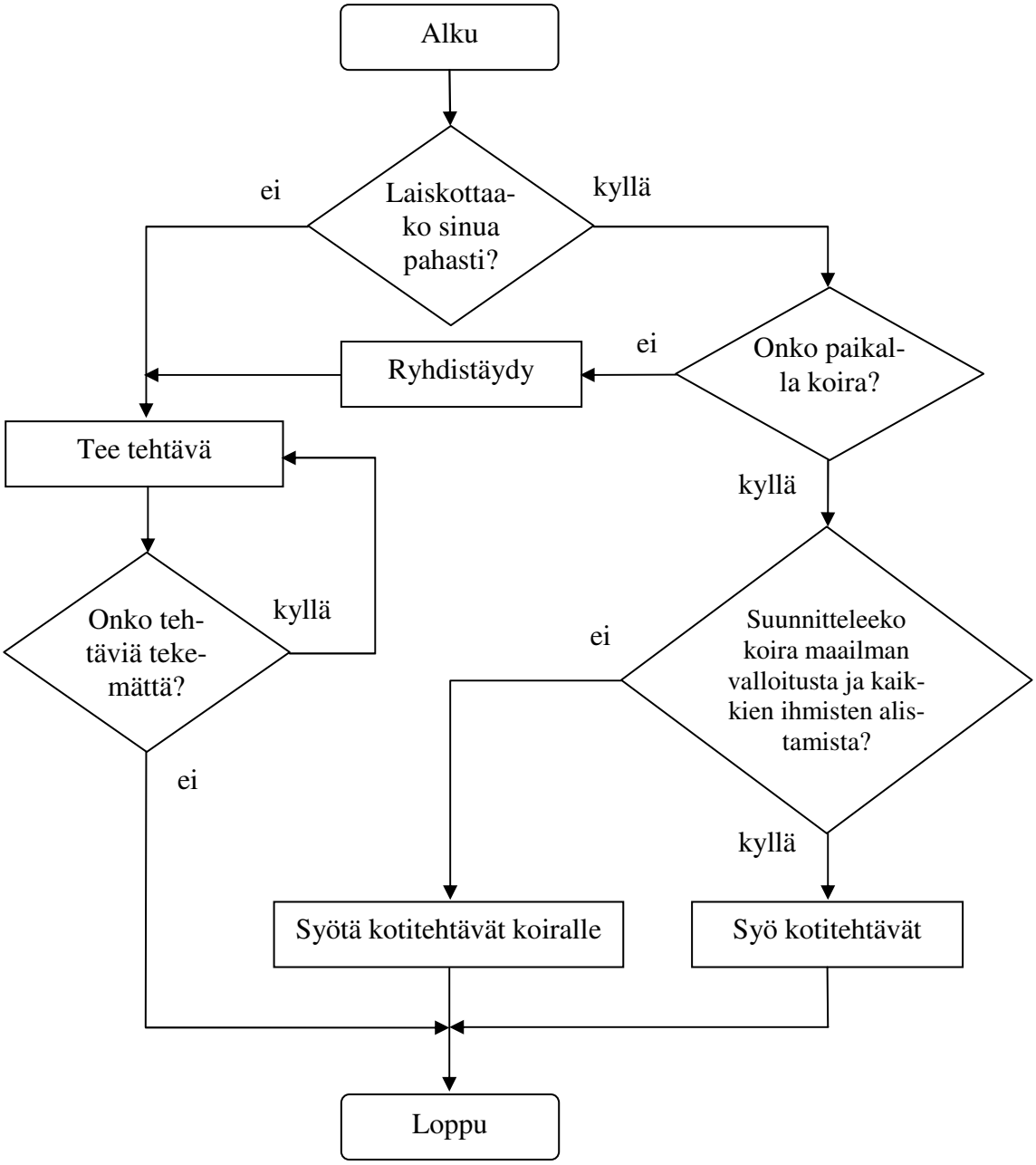
   ```
   javac –version
   ```

   ja painamalla *Enter*-näppäintä. Ota komentoikkunasta kuvankaappaus ja palauta kaappaus WETO järjestelmään, jos näytölle tulostui Java-kääntäjän versio. Harkitse Java JDK:n päivittämistä uudempaan, jos kääntäjän versio on kovin vanha (esimerkiksi 1.6).

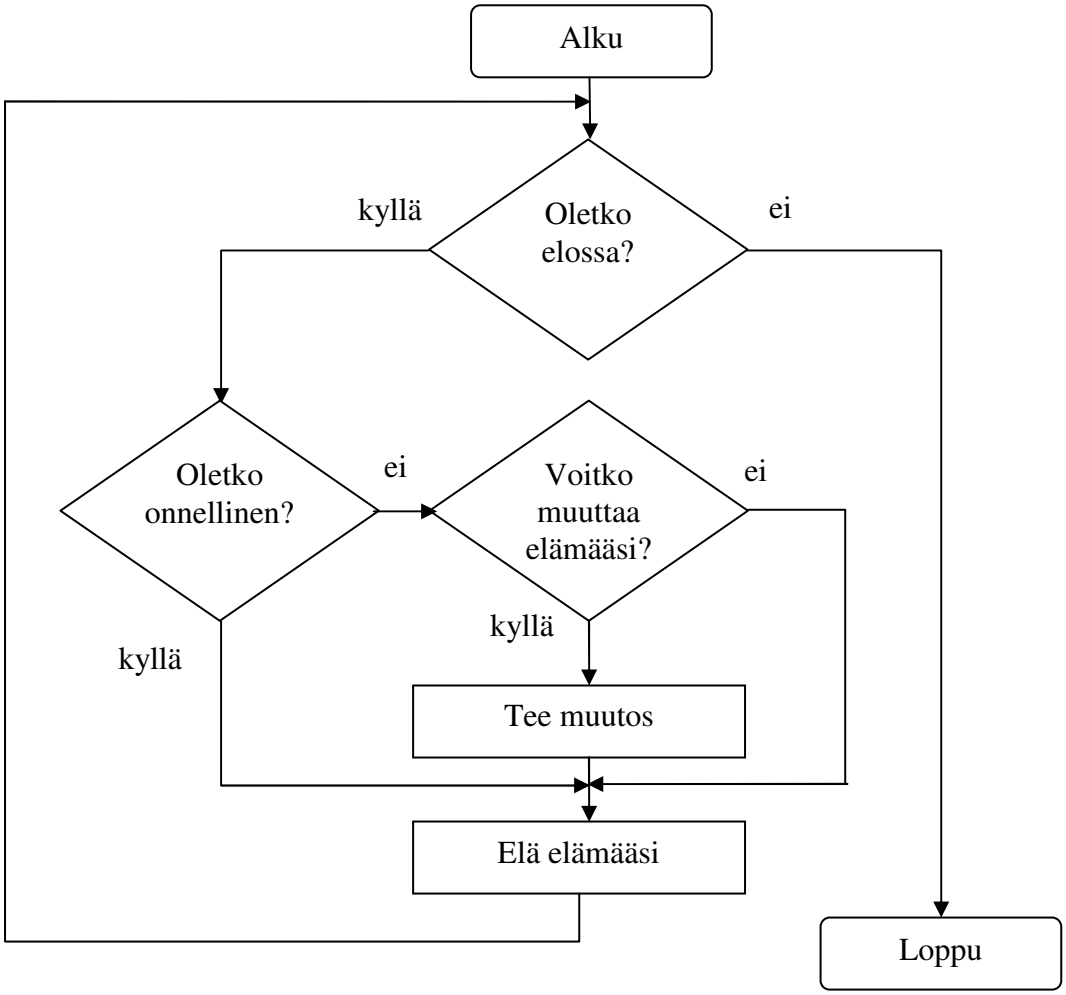
   Tarkista virhetilanteessa, että kirjoitit komennon oikein ja että koneellasi on varmasti Java JDK. (Java JRE ei sisällä kääntäjää.) Mac-järjestelmissä vanhan JDK:n käyttö saattaa olla tietoturvasyistä estetty.

   Löydät ohjeita kuvan kaappaamiseen yksittäisestä ikkunasta eri käyttöjärjestelmissä esimerkiksi täältä: https://en.wikipedia.org/wiki/Screenshot. Liitteessä 3 on annettu esimerkkisuoritus vastuuopettajan työkoneella.

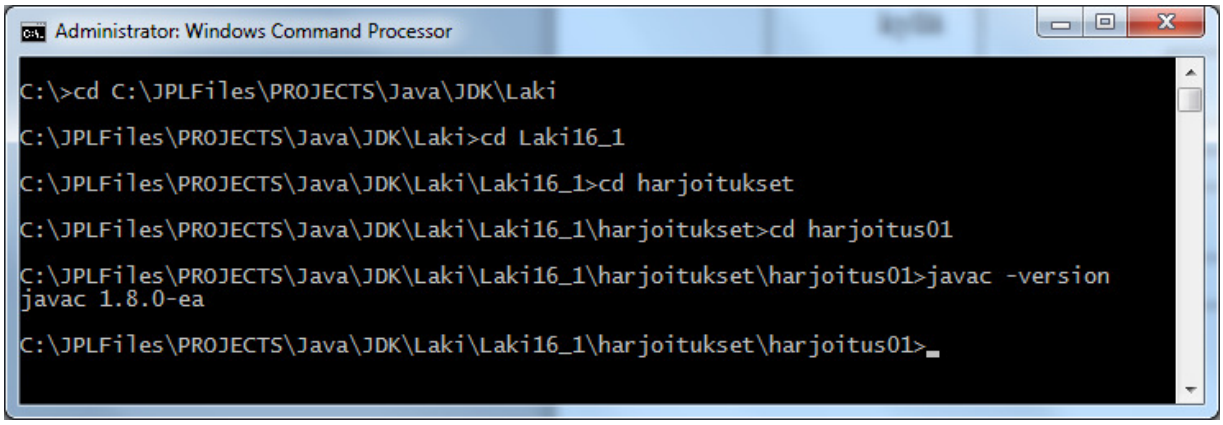
Liite 1: Algoritmi kotitehtävien tekoon ja välttelyyn. Oletetaan, että tehtävät ovat paperilla ja että todella huono tuuri on mahdollista (http://dilbert.com/strip/1991-03-27).



Liite 2: Naiivi algoritmi onnellisen elämän elämiseen. Oletetaan helppo elämä.



Liite 3: Hakemiston vaihtoja *cd*-komennolla ja Java-kääntäjän version tarkistus Windows 7 -käyttöjärjestelmän komentoikkunassa.



Harjoitus 2 (viikko 37)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista nimetä muuttujat hyvin sekä kommentoida ja sisentää koodisi.
- Ohjelmointitehtävien osalta palautetaan vain ratkaisun lähdekoodi (java-päätteinen tiedosto). *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (B1084) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 15.9. klo 12.00.
- **WETO tarkistaa lähdekoodia automaattisesti. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Tehtävät ja vastaukset* -kohdassa. Harjoitusten palautus avautuu normaalia myöhemmin automaatin säädön vuoksi.**
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 15.9. klo 14–16 (B0020).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.
- Java-kääntäjä ja -tulkki löytyvät yliopiston mikroluokkien koneilta. Nämä ohjelmat voi asentaa myös omalle koneelle. Lisätietoja osoitteesta: http://www.uta.fi/sis/tie/laki1/ohjelmointivalineita/java_jdk.html. Ota yhteyttä kurssin vastuuopettajaan, jos Java ei suostu asentumaan koneellesi.

1. Kirjoita luennolla esitelty *HelloWorld*-ohjelma (luentorungon sivu 5.3) jollakin tekstieditorilla. Windowsissa esimerkiksi Notepad (Muistio) sopii pienten ohjelmien tekoon. Microsoft Word -ohjelmaa ei ole tarkoitettu ohjelmointiin. Tallenna lähdekoodi tiedostoon (*HelloWorld.java*), käännä se Java-kääntäjällä (javac-ohjelma) ja suorita tavukooditiedosto Java-tulkilla (java-ohjelma). Älä leikkaa ja liimaa ohjelmaa luentorungosta vaan tee ohjelma aidosti tekstieditoriin kirjoittamalla. Opit ohjelmoimaan vain ohjelmia itse tekemällä.

   Ohjelmoinnissa on tapana kertoa lähdekoodin alussa, kuka ohjelman on tehnyt. Kirjoita siksi ohjelman alussa olevaan lohkokommenttiin (/*...*/) nimesi ja sähköpostisoitteesi. Omien tietojen antaminen helpottaa myös opettajien työtä; tarkistettavia tiedostoja on vaikeampi sekoittaa keskenään, jos niissä on tekijöidensä nimet.

   Ole tarkkana sisennyksen kanssa: tarkista ettet ole käyttänyt sisennykseen välilyöntejä ja tabulaattoreita. Käytä vain jompaakumpaa tapaa sisentää. Kurssin vastuuopettaja suosittelee aloittelijoille välilyöntien käyttöä, koska näin sisentäen koodisi näyttää aina täsmälleen siltä kuin haluat sen näyttävän.

2. Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus02/ on annettu *Bugeja*-niminen ohjelma, jossa on kielioppivirheitä. Lue luentorungon sivulta 5.14 kuinka kielioppivirheitä korjataan. Korjaa virheet yksi kerrallaan ja kirjoita ohjelmaan kommentit, joista selviää mitä virheitä koodissa oli ja kuinka korjasit ne. Palauta korjattu lähdekoodi (*Bugeja.java*-tiedosto) WETOon.

   Muista tallentaa muokkaamasi lähdekoodi aina ennen kääntämistä, koska Java-kääntäjä (javac-ohjelma) ei lue lähdekoodia editorista vaan lähdekoodin sisältävästä tiedostosta.

   Windowsin komentoikkunassa ä-, ö-, å-, Ä-, Ö-, Å-kirjaimet voivat näyttää hieman omituisilta. (Tästä ei tarvitse huolestua.) Komennon *chcp 1252* pitäisi muuttaa komentoikkunan merkistön tutummaksi. Jos komennolla ei ole vaikutusta, niin tarkista, että komentoikkunan fontti on Lucinda Console.

3. Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus02/ on annettu *Bugeja2*-niminen ohjelma, jossa on kielioppivirheitä. Korjaa virheet yksi kerrallaan ja kirjoita ohjelmaan kommentit, joista selviää mitä virheitä koodissa oli ja kuinka korjasit ne. Palauta korjattu lähdekoodi (*Bugeja2.java*-tiedosto) WETOon.

4. Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus02/ on annettu *Tyyliton*-niminen ohjelma, jossa ei ole kielioppivirheitä, mutta joka on tehty pitkälti hyvän ohjelmointitavan vastaisesti. Nimeä muuttujat paremmin, sisennä ja lisää ohjelmaan kommentit, myös ohjelman alkuun tuleva yleinen kommentti. Palauta korjattu lähdekoodi (*Tyyliton.java*-tiedosto) WETOon.

5. Kirjoita Java-ohjelma, jossa esittelet kolme **erityyppistä** muuttujaa, annat muuttujille arvot suoraan niitä käyttäjältä lukematta ja tulostat kunkin muuttujan arvon omalle rivilleen. Kerro kommenteissa minkä tiedon kukin muuttuja sisältää. Muista nimetä muuttujat siten, että myös nimistä käy ilmi mistä tiedosta on kyse.

   Esimerkki muuttujan esittelystä, alustamisesta ja muuttujan arvon tulostamisesta

   ```
   ...
   // Tällä hetkellä luettavana olevan kirjan nimi.
   String kirjanNimi = "Matka toiseen maailmaan";
   ...
   // Tulostetaan muuttujan arvo selityksen kera näytölle.
   System.out.println("Nyt luettavana on " + kirjanNimi + ".");
   ...
   ```

   Yllä ... tarkoittaa lähdekoodista pois jätettyä osuutta.

6. Liitteessä 1 on annettu vuokaavio, jossa tulostetaan näytölle tietoja alkuaineesta rauta. Muunna vuokaavio Java-ohjelmaksi.

7. Kirjoita Java-ohjelma, joka tulostaa näytölle vapaavalintaisen kuvion (hyvän maun rajoissa) niin sanottuna ASCII-grafiikkana, jossa grafiikka koostuu näppäimistöltä löytyvistä “tavallisista” merkeistä.

   Alla oleva esimerkkikuvio on tehty alaviivamerkin ('_') ja putkimerkin ('|') avulla.

   ```
    _____   _____   ____   _   _
   |  _  | |  ___| | ___| | | | | |
   | |_| | | |___  | |    | | | | |
   |  _  | |___  | | |    | | | | |
   | | | |  ___| | | |__  | | | | |
   |_| |_| |_____| |____| |_| |_|
   ```

   Kenoviiva ('\') ei valitettavasti tulostu sellaisenaan, koska sen avulla esitetään Javan erikoismerkit. Kenoviivan tulostamiseksi sitä täytyy edeltää toinen kenoviiva. Esimerkki:

   ```
   // Tulostetaan kenoviiva näytölle.
   System.out.println("\\");
   ```

8. Muunna 1. harjoituksen 6. tehtävän (alennetun hinnan laskeminen) vuokaavio Java-ohjelmaksi. Voit käyttää ohjelmasi pohjana joko mallivastausta tai omaa vuokaaviotasi. Joudut luultavasti jättämään omaan ratkaisuusi mahdollisesti lisäämäsi virheenkäsittelyn pois, koska Javan valintarakenteita ei ole vielä opetettu.

9. Kirjoita Javalla ohjelma, joka lukee käyttäjältä viikonpäivän, järjestysluvut päivälle, kuukaudelle ja vuodelle sekä nimipäiväänsä viettävän henkilön nimen. Jos nimiä on useampia, niin käyttäjä antaa nimet pilkulla erotettuna.

   Ohjelma tulostaa lukemansa tiedot yhdelle riville siten, että ensin tulostetaan viikonpäivä ja tämän jälkeen välilyönti ja sitten päivän, kuukauden ja vuoden järjestysnumerot toisistaan pisteellä erotettuna. Lopuksi tulostetaan välilyönti ja nimi tai nimet.

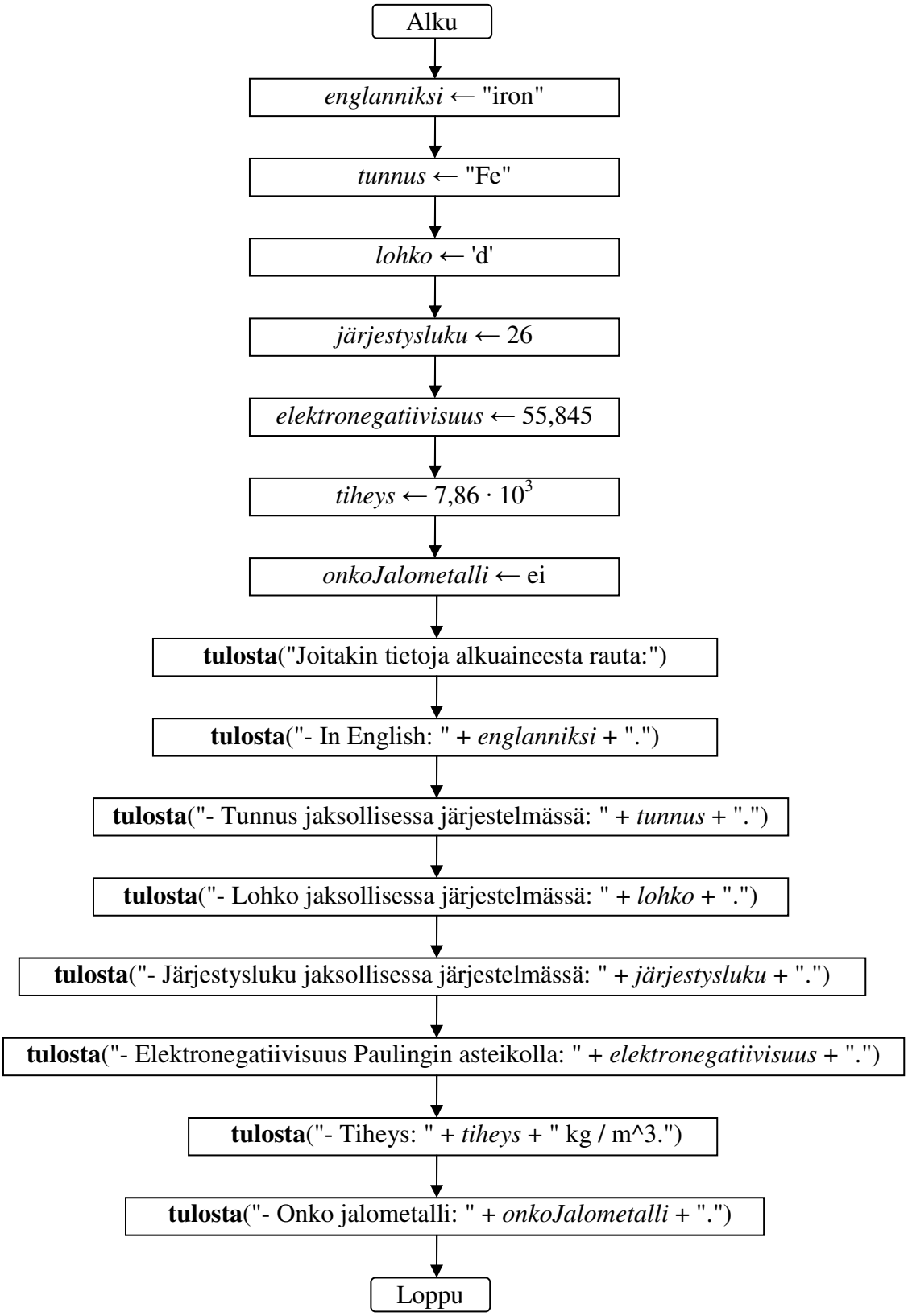
   Hyvään ohjelmointitapaan kuuluu pitää lähdekoodit rivit järkevän mittaisina. Pitkän lauseen lukeminen on vaikeaa. Lauseen rivi ei välttämättä mahdu editorin ikkunaan ja ennen kaikkea pitkän lauseen sisältöä on hankala hahmottaa. Tulosta siksi kahta lausetta käyttäen, jos tulostavan lauseen rivi venähtää kovin pitkäksi (yli 100 merkkiä). Saat tekstin tulostumaan yhdelle riville käyttämällä ensimmäisessä tulostuslauseessa *System.out.print*-operaatiota.

   Oletetaan, että käyttäjä antaa järkeviä syötteitä.

   Esimerkki ohjelman toiminnasta:

```
Moi! Muodostan päivän tiedoista tulosteen.
Anna viikonpäivä:
keskiviikko
Anna päivä:
7
Anna kuukausi:
9
Anna vuosi:
2016
Anna nimipäiväsankarit:
Miro, Milo, Arho, Arhippa
keskiviikko 7.9.2016 Miro, Milo, Arho, Arhippa
```

Liite 1: Vuokaaviona esitetty algoritmi rautaan liittyvien tietojen tulostamiseen.



Harjoitus 3 (viikko 38)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista nimetä muuttujat hyvin sekä kommentoida ja sisentää koodisi. Vältä liian pitkiä rivejä.
- Ohjelmointitehtävien osalta palautetaan vain ratkaisun lähdekoodi (java-päätteinen tiedosto). *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (B1084) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 22.9. klo 12.00.
- **Viidennessä tehtävässä testataan WETOn automaattista ohjelman tarkistusta. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* -kohdassa.**
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 22.9. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.
- Java-kääntäjä ja -tulkki löytyvät yliopiston mikroluokkien koneilta. Nämä ohjelmat voi asentaa myös omalle koneelle. Lisätietoja osoitteesta: http://www.uta.fi/sis/tie/laki1/ohjelmointivalineita/java_jdk.html. Ota yhteyttä kurssin vastuuopettajaan, jos Java ei suostu asentumaan koneellesi.

1. Kirjoita Java-ohjelma, jossa testaat aritmeettiset operaatiot +, -, *, / ja %. Esittele **int**-tyyppiset muuttujat *ekaluku* ja *tokaluku* ja lue muuttujille kokonaislukuarvot käyttäjältä. Sijoita kunkin operaation tulos omaan tulosmuuttujaansa, jotka ovat jako-operaation tulosmuuttujaa lukuun ottamatta **int**-tyyppisiä. Osamäärä sijoitetaan **double**-tyyppiseen muuttujaan. Tulosta lopuksi tulosmuuttujien arvot näytölle. Kun syötteet ovat 7 ja 4, niin ohjelma voisi toimia seuraavasti.

```
Moi! Testaan aritmeettisia operaatioita.
Anna 1. luku:
7
Anna 2. luku:
4
7 + 4 = 11
7 - 4 = 3
7 * 4 = 28
7 / 4 = 1.75
7 % 4 = 3
```

   Huomaa, että tarvitset tyyppimuunnoksen, jotta Java ei hävitä desimaaleja jakolausekkeessa *ekaluku* / *tokaluku*.

2. Tee Java ohjelma, joka onnittelee sinua **if**-lauseen avulla. Ohjelma kysyy sinulta tavoitteen (kokonaisluku) ja saavutuksen (toinen kokonaisluku) ja onnittelee, jos saavutus on yhtä suuri tai suurempi kuin tavoite. Ohjelma ei sano muussa tapauksessa mitään.

3. Yhdysvaltalainen teknologiayhtiö Google perustettiin vuonna 1998. Kirjoita Javalla ohjelma, joka lukee käyttäjänsä syntymävuoden ja kertoo **if-else**-lauseen avulla onko käyttäjä Googlea vanhempi (syntymävuosi < 1998) vai ei (syntymävuosi ≥ 1998).

4. Liuoksen happamuutta mitataan pH-arvolla. Jos pH on alle 7, sanotaan liuoksen olevan hapan. Jos pH on yli 7, on liuos emäksinen. Liuosta, jonka pH on 7, sanotaan neutraaliksi. Kirjoita Java-ohjelma, joka tulostaa lukee käyttäjältä pH-arvon liukulukuna ja tulostaa saamansa arvon mukaan joko "Liuos on hapan.", "Liuos on neutraali." tai "Liuos on emäksinen." Esimerkki ohjelman toiminnasta:

```
Moi! Analysoin liuosta.
Anna pH-arvo:
6.5
Liuos on hapan.
```

5. 12 tunnin kellossa aika esitetään käyttäen tunteina lukuja 0–12. Aamupäivän ja iltapäivän kellonajat erotetaan toisistaan lisäämällä aikaan lyhenne. Kirjoita Java-ohjelma, joka lukee käyttäjältään 24 tunnin kellon mukaiset tunnit ja tulostaa tunnit 12 tunnin kellon mukaan **if-else**-lauseen avulla. 24 tunnin kellon tunnit väliltä 0–12 tulostetaan tässä tehtävässä muodossa "*x* a.m.", missä *x* on tuntien määrä. 24 tunnin kellon tunnit väliltä 13–23 tulostetaan muodossa "*x* p.m.", missä *x* on tuntien määrä väliltä 1–11. Esimerkiksi syötteellä 6 tuloste on "6 a.m." ja syötteellä 16 tuloste on "4 p.m." Voit olettaa, että käyttäjä antaa tuntien määräksi aina kokonaisluvun väliltä 0–23.

   Esimerkki ohjelman toiminnasta, kun syöte on 12:

   ```
   Hello! I convert times between clocks.
   Please, enter hours:
   12
   12 a.m.
   ```

   Esimerkki ohjelman toiminnasta, kun syöte on 13:

   ```
   Hello! I convert times between clocks.
   Please, enter hours:
   13
   1 p.m.
   ```

   Tässä tehtävässä testataan tulosteiden vertailuun perustuvaa WETOn automaattista testausta. Ohjelman on tulostettava tästä syystä tervehdysrivi ja kehote täsmälleen edellä annettujen esimerkkien tapaisesti englannin kielellä. Ohjelman on myös annettava tuloksensa merkilleen yllä kuvatussa muodossa. Jo yhden merkin ero malliohjelman ja opiskelija ratkaisun tulosteissa riittää hylkäykseen. Ole erityisen tarkkana ettei ohjelmasi tulosta välilyönnin tai tabulaattorin tapaisia näkymättömiä merkkejä rivin loppuun.

   19.9.: Anna ohjelmasi nimeksi *Hours* ja palauta ohjelma *Hours.java*-tiedostossa.

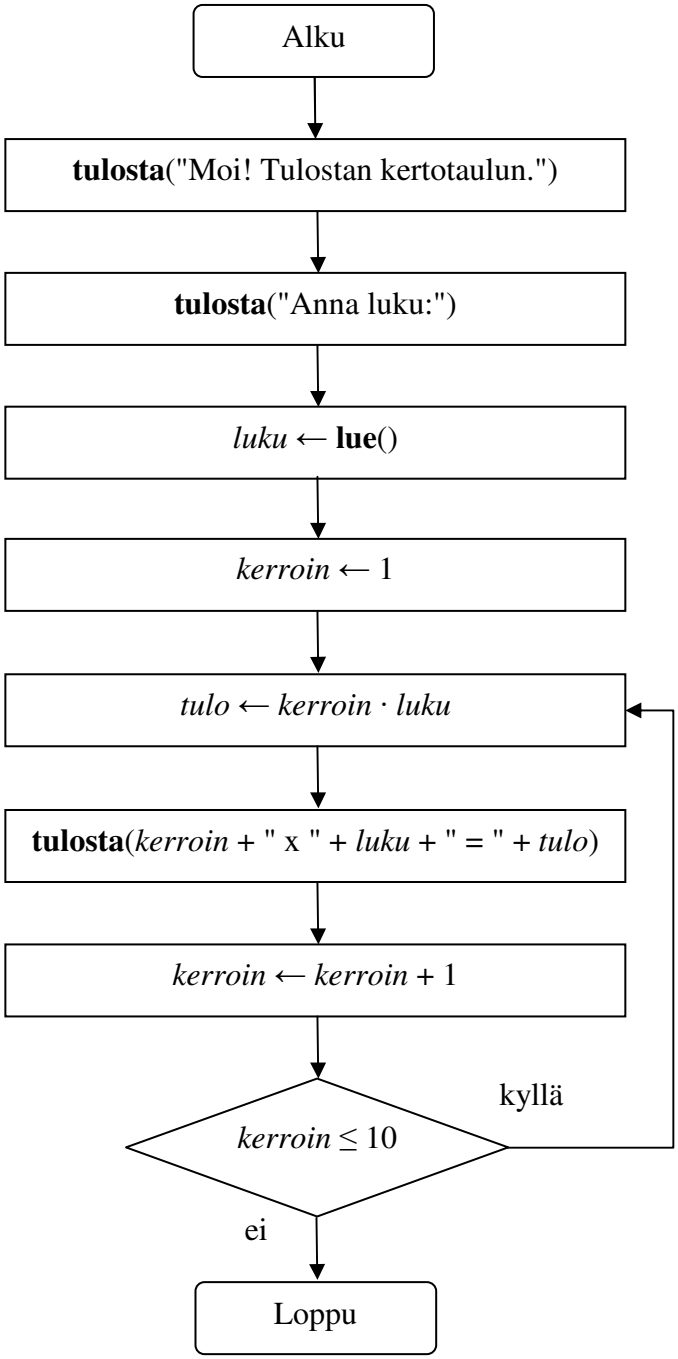
6. Toteuta liitteessä 1 vuokaaviona esitetty algoritmi Java-kielellä.

7. Tee Java-ohjelma, joka tulostaa yhdelle riville kokonaislukujen jonon. Lukujono alkaa käyttäjän antamasta luvusta ja siinä on lukuja käyttämän antama määrä. Kutakin lukujonon lukua a_i seuraava luku $a_{i+1} = a_i + 2$. Jos käyttäjä antaa esimerkiksi jonon ensimmäiseksi luvuksi 1 ja lukujen lukumääräksi 5, niin ohjelma tulostaa luvut 1, 3, 5, 7 ja 9. Oletetaan, että ohjelmalle syötetään vain kokonaislukuja ja että lukujen määrä on aina nollaa suurempi. Esimerkki ohjelman toiminnasta:

   ```
   Moi! Tulostan lukujonon.
   Anna 1. luku:
   -4
   Anna lukujen määrä:
   7
   -4 -2 0 2 4 6 8
   ```

8. Toteuta 1. harjoituksen 7. tehtävän (positiivisten lukujen laskeminen) Java-kielellä. Voit käyttää apuna joko tehtävän mallivastausta tai omaa vuokaaviotasi. Ohjelman on toimittava määrittelyn mukaan. Esimerkiksi virheilmoitus on tulostettava tarvittaessa.

Liite 1: Vuokaaviona esitetty algoritmi kertotaulun tulostamiseen. Oletetaan syötteeksi kokonaisluku.



Harjoitus 4 (viikko 39)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista noudattaa hyvää ohjelmointi tapaa muun muassa koodia kommentoimalla ja sisentämällä. Katso lisää ohjeita luentomateriaalin 14. luvusta.
- Ohjelmointitehtävien osalta palautetaan vain ratkaisun lähdekoodi (java-päätteinen tiedosto). *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (B1084) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 29.9. klo 12.00.
- **Osa tehtävistä testataan automaattisesti. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus | Harjoitukset* -kohdassa.**
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 29.9. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.
- Java-kääntäjä ja -tulkki löytyvät yliopiston mikroluokkien koneilta. Nämä ohjelmat voi asentaa myös omalle koneelle. Lisätietoja osoitteesta: http://www.uta.fi/sis/tie/laki1/ohjelmointivalineita/java_jdk.html. Ota yhteyttä kurssin vastuuopettajaan, jos Java ei suostu asentumaan koneellesi.

1. Kirjoita Javalla *SawTeeth1*-niminen ohjelma, joka tulostaa yhdelle riville sahahammas-kuvion **while**-silmukkaa käyttäen. Rivin pituus luetaan käyttäjältä. Kuviota tulostetaan siten, että ensin tulostetaan jakomerkki (/) ja sitten kenoviiva (\). Näitä merkkejä tulostetaan edellä annetussa järjestyksessä, kunnes rivi on annetun pituinen. Ohjelma tulostaa merkkijonon "Error!", jos käyttäjän syöte on pienempi tai yhtä suuri kuin nolla.

   Kenoviiva ('\') ei valitettavasti tulostu sellaisenaan, koska sen avulla esitetään Javan erikoismerkit. Kenoviivan tulostamiseksi sitä täytyy edeltää toinen kenoviiva. Esimerkki:

   ```
   // Tulostetaan kenoviiva näytölle.
   System.out.println("\\");
   ```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syöte on kuusi:

   ```
   Hello! I print saw teeth.
   Please, enter the number of characters:
   6
   /\/\/\
   ```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syöte on yhdeksän:

   ```
   Hello! I print saw teeth.
   Please, enter the number of characters:
   9
   /\/\/\/\/
   ```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syöte on nolla:

   ```
   Hello! I print saw teeth.
   Please, enter the number of characters:
   0
   Error!
   ```

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

2. Tee *SawTeeth2*-niminen ohjelma, joka toteuttaa 1. tehtävän **for**-silmukkaa käyttäen.

   Tämäkin tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

3. Tee Java-ohjelma, joka lukee käyttäjältä kaksi merkkiä ja päättelee onko kyseessä kaari-, aalto-, haka- tai kulmasulkeiden pari. Ohjelma kertoo onko kyseessä pari vai ei. Käyttäjälle tulostetaan siten viesti myös, kun kyseessä ei ole sulkeiden pari. Sulkeet katsotaan pariksi, jos parin ensimmäinen merkki on “avaava” sulje ja jälkimmäinen merkki on “sulkeva” sulje. Näin syötteet '(' ja ')', '{' ja '}', '[' ja ']' ja '<' ja '>' ovat pareja. Toisaalta esimerkiksi syötteet ')' ja '(' eivät muodosta paria, koska sulkeiden järjestys on väärä. **Älä käytä loogisia operaatioita**.

   Esimerkki ohjelman toiminnasta:

   ```
   Moi! Tunnistan sulkeiden parin.
   Anna 1. merkki:
   {
   Anna 2. merkki:
   }
   Merkit '{' ja '}' ovat pari.
   ```

   Toinen esimerkki ohjelman toiminnasta:

   ```
   Moi! Tunnistan sulkeiden parin.
   Anna 1. merkki:
   X
   Anna 2. merkki:
   X
   Merkit 'X' ja 'X' eivät ole pari.
   ```

   Vinkki: 7. luennon *esimerkit*-kansio.

4. Toteuta 3. tehtävä **loogisia operaatioita käyttäen**. Vinkki: 7. luennon *esimerkit*-kansio.

5. Tee Javalla ohjelma, joka lukee käyttäjältä merkin ja ilmoittaa mihin kolmesta merkkien joukosta merkki kuuluu. Joukot ovat numeromerkit ('0'–'9'), Javan aritmeettisia operaatioita symboloivat merkit ('+', '- ', '*', '/' ja '%') ja muut merkit.

   Käytä ohjelmassa loogisia operaattoreita.

   Ohjelmaa voi lyhentää numeromerkkien tunnistamisen osalta hyödyntämällä järjestyksen tunnistavia vertailuoperaattoreita (<, <=, >, >=) ja tietoa siitä, että numeromerkit ovat ASCII-merkistössä peräkkäin kasvavassa järjestyksessä. Javassa on sallittua vertailla merkkejä esimerkiksi näin:

   ```
   // Alla merkki-muuttuja on char-tyyppiä.
   if ('0' <= merkki) {
   ```

6. Tee Javalla ohjelma, joka lukee käyttäjältä numeromerkin. Jos käytät *In*-luokkaa, niin valintasi on tässä tehtävässä *readChar*-operaatio. *Scanner*-luokan käyttäjät huomaavat, että luokassa ei ole operaatiota yksittäisen merkin lukemiseen. Tähän ongelmaan löytyy ratkaisu. Voit katsoa mallia esimerkiksi *In*-luokasta. Oli lukemisen tekniikka mikä tahansa, syötettä tiedustellaan uudelleen niin pitkään kuin käyttäjä antaa syötteeksi jotain muuta kuin numeromerkin.

   Esimerkki ohjelman toiminnasta:

```
Moi! Tinkaan sinulta numeromerkin.
Anna numeromerkki:
S
Virhe!
Anna numeromerkki:
5
Kiitokset!
```

   Vinkki: 7. luennon *esimerkit*-kansio.

7. Kirjoita Java-kielellä ohjelma, joka laskee positiivisten (≥ 0) lukujen summan. Lukujen lukumäärä (> 0) ja kukin luku luetaan käyttäjältä.

   Lukumäärää tiedustellaan **do**-**while**-silmukassa kunnes käyttäjä antaa laillisen syötteen. Luvut lukeva **while**-silmukka pysähtyy, kun kaikki luvut on luettu tai käyttäjä antaa negatiivisen luvun. Negatiivista lukua ei lasketa summaan. Älä pysäytä kumpaakaan silmukkaa **break**-lauseen avulla, vaan käytä lippumuuttujia. Huomaa, että tarvitset luvut lukevaa silmukkaa ohjaavaan lausekkeeseen loogisen operaattorin. Ohjelman lopuksi tulostetaan näytölle lukujen summa.

   Esimerkki ohjelman toiminnasta:

```
Moi! Lasken lukujen summan.
Anna lukujen lukumäärä:
-1
Lukuja oltava vähintään yksi!
Anna lukujen lukumäärä:
4
Anna luku:
10
Anna luku:
20
Anna luku:
-999
Summa on 30.
```

8. BizzBuzz-peliä pelataan siten, että pelaajat istuvat piirissä ja joku aloittaa pelin sanomalla "1". Seuraava pelaaja sanoo joko seuraavan luvun, "bizz", "buzz" tai "bizz buzz". Luku korvautuu sanalla "bizz", jos luku on kolmella jaollinen. Sana "buzz" korvaa luvun, jos luku on viidellä jaollinen. Luvun sijasta sanotaan "bizz buzz", kun luku sekä kolmella että viidellä jaollinen. Peli jatkuu kunnes joku takeltelee tai sanoo väärän vastauksen.

   Esimerkki: 1, 2, bizz, 4, buzz, bizz, 7, 8, bizz, buzz, 11, bizz, 13, 14, bizz buzz, 16, ...

   Kirjoita Java-kielellä *BizzBuzz*-niminen ohjelma, joka kysyy käyttäjältä pelivuorojen lukumäärän ja tulostaa näytölle oikeiden vastauksien sarjan. Voit olettaa, että syöte on aina oikeellinen (> 0).

   Esimerkki ohjelman toiminnasta, kun käyttäjä antaa vuorojen määräksi viisi:

```
Hello! This is a BizzBuzz simulation.
Please, enter the number of turns:
5
1, 2, bizz, 4, buzz.
```

   Esimerkki ohjelman toiminnasta, kun käyttäjä antaa vuorojen määräksi 15:

```
Hello! This is a BizzBuzz simulation.
Please, enter the number of turns:
15
1, 2, bizz, 4, buzz, bizz, 7, 8, bizz, buzz, 11, bizz, 13, 14, bizz buzz.
```

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

Harjoitus 5 (viikko 40)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista noudattaa hyvää ohjelmointi tapaa muun muassa koodia kommentoimalla ja sisentämällä. Katso lisää ohjeita luentomateriaalin 14. luvusta.
- Ohjelmointitehtävien osalta palautetaan vain ratkaisun lähdekoodi (java-päätteinen tiedosto). *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (B1084) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 6.10. klo 12.00.
- Osa tehtävistä testataan automaattisesti. Lisätietoja tarkistuksesta on kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 6.10. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus05/ on annettu sisentämätön *QuadraticEquationSolver*-ohjelma. Lisää ohjelmaan sisennys. Kerro kommenteissa mitä sisennystyyliä käytät ja anna viite (esimerkiksi verkko-osoite) dokumenttiin, jossa tyylisi on määritelty, jos käytät sisentämiseen tyyliä, joka poikkeaa selvästi kurssilla opetetusta tyylistä. Ohjelma on englanninkielinen, koska kurssilla on ulkomaalaisia opiskelijoita. Palauta WETOon sisennetty ohjelma *QuadraticEquationSolver.java*-tiedostossa.

2. Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus05/ on annettu *OddCounter*-ohjelma, jossa on sekä kielioppivirheitä että pieniä loogisia virheitä. **Muista ajaa ja testata ohjelmaa, jotta löydät varmasti kaikki loogiset virheet**. Kerro kommenteissa mistä virheistä oli kyse ja kuinka korjasit virheet. Lisää ohjelman yleisiin kommentteihin omat tietosi. Kommentit voi kirjoittaa suomeksi. Ohjelma on englanninkielinen, koska kurssilla on ulkomaalaisia opiskelijoita. Palauta WETOon korjattu ohjelma *OddCounter.java*-tiedostossa.

   Vinkki: Ehto-operaattori on yhteenlaskua heikompi. Oletuslaskujärjestystä voi muuttaa koulusta opitulla tavalla sulkeilla.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki ohjelman toiminnasta:

```
Hello! I count odd integers.
Please, enter the number of integers:
0
The number must be at least one!
Please, enter the number of integers:
-1
The number must be at least one!
Please, enter the number of integers:
3
Please, enter an integer:
42
Please, enter an integer:
13
Please, enter an integer:
5
Found 2 odd integers.
```

3. Lausekielinen ohjelmointi I -kurssilla pitää kerätä harjoituspisteitä siten, että tehtävien ratkaisuista ja läsnäoloista saatujen pisteiden summa on vähintään 40 % kaikkien annettujen tehtävien ja harjoituskertojen (6) summasta. Jos tehtäviä annetaan esimerkiksi 51 kappaletta, niin saatavilla on yhteensä 51 + 6 = 57 pistettä. Huomaa, että harjoitustehtävien lopullinen lukumäärä selviää vasta viimeisen harjoituksen julkaisun jälkeen.

   Ahkerasta harjoitusten ratkaisusta palkitaan edellä määritellyn suhdeluvun mukaan seuraavasti: vähintään 60 % → 1 hyvityspiste, vähintään 70 % → 2 hyvityspistettä ja vähintään 80 % → 3 hyvityspistettä. Hyvityspisteet lisätään tenttipisteisiin, jos tentistä saa vähintään 12 pistettä. Hyvityksiä laskettaessa prosenttilukua ei pyöristetä ylöspäin, vaan prosenttiluvun on oltava rajalla tai sen yli. Näin esimerkiksi 59,6 % tuottaa 0 hyvityspistettä, koska 1 hyvityspisteen saa vasta, kun prosenttiluku on vähintään 60 %.

   Osoitteessa: http://www.sis.uta.fi/~laki1/harjoitukset/harjoitus05/ on annettu *Points*-niminen ohjelma, joka lukee ensin valistuneen arvauksesi tehtävien kokonaislukumäärästä (49–51) ja tavoitteeksesi asettamasi prosentin (kokonaisluku väliltä 40–100) ja kertoo sitten paljonko pisteitä tarvitset tavoitteesi saavuttamiseen. Ohjelma tulostaa virheilmoituksen, jos tehtäviä on alle 49 tai yli 51 tai jos prosentti on alle 40 tai yli 100.

   Tee vakiot virheilmoitukselle "Does not compute!" ja luvuille 49, 51, 40 ja 100. Ota vakiot käyttöön ohjelmassa.

   Kiinnitä huomiota vakioiden nimeämiseen. Vakiot kirjoitetaan **isoin kirjaimin** ja nimen tulee olla kuvaava. Erityisesti on vältettävä vakion nykyisen arvon ilmaisua nimessä, koska vakioita käytetään nimenomaan helpottamaan ohjelman ylläpitoa. Esimerkiksi HUNDRED ei ole hyvä nimi, koska se perustuu suoraan vakion nykyiseen arvoon. Sen sijaan MAXPERCENTAGE on paljon parempi. Vakiot esitellään aina aivan ohjelman alussa ennen tavallisia muuttujia.

   Palauta muokattu ohjelma WETOon *Points.java*-tiedostossa.

4. Lausekielinen ohjelmointi I -kurssin arvosana määräytyy tenttipisteiden (0–24 kpl) ja tenttipisteisiin lisättävien hyvityspisteiden (0–3 kpl) summan perusteella seuraavasti: 12–14 pistettä → 1 (välttävä), 15–17 pistettä → 2 (tyydyttävä), 18–20 pistettä → 3 (hyvä), 21 tai 22 pistettä → 4 (kiitettävä) ja vähintään 23 pistettä → 5 (erinomainen). Hyvityspisteet huomioidaan vasta, kun opiskelija on suorittanut tentin hyväksyttävästi eli saanut tentistä vähintään 12 pistettä.

   Kirjoita *Grade*-niminen Java-ohjelma, joka lukee käyttäjältä tenttipisteiden ja hyvityspisteiden lukumäärät sekä laskee ja tulostaa pisteiden summaa vastaavan arvosanan. Ohjelma tulostaa virheilmoituksen "I cannot give a grade.", jos tenttipisteet eivät ole välillä 12–24 tai hyvityspisteet eivät ole välillä 0–3. Hyvityspisteet kysytään, vaikka tenttipisteet olisivat epäkelvot.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

Esimerkki ohjelman toiminnasta, kun syötteet ovat 22 ja 1:

```
Hello! I am a grader.
Please, enter exam points:
22
Please, enter bonus points:
1
Your grade is 5.
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 15 ja 3:

```
Hello! I am a grader.
Please, enter exam points:
15
Please, enter bonus points:
3
Your grade is 3.
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 11 ja 1:

```
Hello! I am a grader.
Please, enter exam points:
11
Please, enter bonus points:
1
I cannot give a grade.
```

5. Ilmatieteen laitos luokittelee tuulen voimakkuuden sen nopeuden 10 minuutin keskiarvona seuraavasti: 0 m/s = tyyntä, 1–3 m/s = heikkoa tuulta, 4–7 m/s = kohtalaista tuulta, 8–13 m/s = navakkaa tuulta, 14–20 m/s = kovaa tuulta, 21–32 m/s = myrskyä ja yli 32 m/s = hirmumyrskyä.

   Tee Javalla ohjelma, joka lukee käyttäjältä tuulen nopeuksia kokonaislukuina ja tulostaa vastaavia voimakkuuden kuvauksia kunnes käyttäjä antaa negatiivisen arvon (< 0).

   Esimerkki ohjelman toiminnasta:

```
Moi! Luokittelen tuulta.
Anna tuulen nopeus (m/s):
10
Tuuli on navakkaa.
Anna tuulen nopeus (m/s):
32
Myrskytuulta.
Anna tuulen nopeus (m/s):
0
Tyyntä.
Anna tuulen nopeus (m/s):
-1
```

6. Kirjoita ohjelma, joka lukee merkkijonon ja kaksi indeksiarvoa ja tutkii ovatko annetuissa paikoissa olevat merkit samat. Ohjelma tulostaa virheilmoituksen, jos jompikumpi tai molemmat indeksiarvot ovat virheelliset. Laillinen indeksiarvo on välillä [0, *n* - 1], missä *n* on merkkijonon pituus. Esimerkki ohjelman toiminnasta:

```
Moi! Vertailen kahta merkkijonon merkkiä.
Anna merkkijono:
Java
Anna 1. merkin indeksiarvo:
1
Anna 2. merkin indeksiarvo:
3
Merkit 'a' ja 'a' ovat samat.
```

7. Tee Java-ohjelma, joka tutkii alkaako sille annettu merkkijono suurella kirjaimella.

   Huomaa, että voit hyödyntää tässäkin tehtävässä ASCII-koodeja. Esimerkki:

```
// Alla merkki on char-tyyppinen muuttuja.
if ('A' <= merkki && merkki <= 'Z'){ ...
```

   Huomaa myös, että å-, ä- ja ö-kirjaimet täytyy tunnistaa erikseen.

   Voi olla, että Windowsin komentoikkunassa on merkistö, jonka vuoksi ohjelmasi ei tunnista skandinaavisia kirjaimia. Muuta tällöin komentoikkunan fontiksi Lucinda Console ja anna komento *chcp 1252*.

8. Kirjoita Javalla *Zorro*-ohjelma, joka tulostaa alla olevien esimerkkien mukaisen kuvion. Ohjelma lukee käyttäjältä tulostuksessa käytettävän merkin ja koon (kuvion rivien ja sarakkeiden lukumäärä). Ohjelma piirtää kuvion, jos koko on vähintään kolme. Ohjelma tulostaa "No comprendo.", jos koko on virheellinen.

   Vinkki: Sisäkkäiset silmukat.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että kuvion keskirivien alkuun tulostetaan poikkeuksellisesti välilyöntejä. Rivien näkyvien merkkien jälkeen ei tulosteta välilyöntejä. Kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat tähtimerkki ja 3:

```
Hola! I am Zorro.
Please, enter mark:
*
Please, enter size:
3
***
 *
***
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat pieni o-kirjain ja 4:

```
Hola! I am Zorro.
Please, enter mark:
o
Please, enter size:
4
oooo
  o
 o
oooo
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat pieni x-kirjain ja 5:

```
Hola! I am Zorro.
Please, enter mark:
x
Please, enter size:
5
xxxxx
   x
  x
 x
xxxxx
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat suuri z-kirjain ja 0:

```
Hola! I am Zorro.
Please, enter mark:
Z
Please, enter size:
0
No comprendo.
```

Harjoitus 6 (viikko 41)

Nämä ovat kurssin viimeiset harjoitukset. Muista, että hyväksytyistä ratkaisuista ja läsnäoloista kerättyjen pisteiden summan tulee olla vähintään 40 % (22 pistettä) tehtävien ja läsnäolopisteiden kokonaislukumäärien summasta (49 + 6 = 55). Hyvityspisteiden rajat: 60 % (33 pistettä), 70 % (39 pistettä) ja 80 % (44 pistettä).

Voit tehdä lisätehtäviä (korkeintaan 8 kpl), jos aiot suorittaa kurssin, mutta pisteesi ovat alle 40 % -rajan tämän harjoituksen jälkeen. Kurssin vastuuopettaja ottaa yhteyttä opiskelijoihin, jotka voivat saavuttaa 40 %:n rajan lisätehtäviä tekemällä.

Tentti ja Lausekielinen ohjelmointi II -kurssin ensimmäisen harjoitustyön ohjelmointi sujuvat todennäköisesti paremmin, jos ratkaiset alla olevia tehtäviä ja tulet myös luentoharjoituksiin.

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista noudattaa hyvää ohjelmointi tapaa muun muassa koodia kommentoimalla ja sisentämällä. Katso lisää ohjeita luentomateriaalin 14. luvusta.
- Ohjelmointitehtävien osalta palautetaan vain ratkaisun lähdekoodi (java-päätteinen tiedosto). *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (B1084) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 13.10. klo 12.00.
- Osa tehtävistä testataan automaattisesti. Lisätietoja tarkistuksesta on kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 13.10. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Kirjoita Java ohjelma, joka lukee sinulta juomapullon ja hörpyn tilavuuden millilitroina (litran tuhannesosa). Ohjelma tulostaa näytölle hörppyä simuloivan merkkijonon niin monta kertaa, kuin pullosta voidaan ottaa hörppy. Ohjelma kertoo lopuksi paljonko pulloon jäi juomaa. Jos pullon tilavuus on esimerkiksi 330 ml (0,33 litraa) ja hörppysi on tilavuudeltaan 4,5 ml, jää pulloon juomaa jäljelle 1,5 ml. Ohjelma tarkistaa, että pullon ja hörpyn tilavuudet ovat nollaa suurempia ja että hörpyn tilavuus ei ole pullon tilavuutta suurempi ja tulostaa näytölle virheilmoituksen, jos jompikumpi tai molemmat syöteistä olivat virheellisiä.

2. Alkuluku on lukua yksi suurempi kokonaisluku, joka on jaollinen vain luvulla yksi ja itsellään. Kymmenen ensimmäistä alkulukua ovat 2, 3, 5, 7, 11, 13, 17, 19, 23 ja 29. Kirjoita Javalla *Primes*-ohjelma, joka lukee käyttäjältä kokonaisluvun ja ilmoittaa onko luku alkuluku vai ei. Ohjelma tulostaa virheilmoituksen, jos käyttäjä antaa syötteeksi luvun, joka on pienempi tai yhtä suuri kuin yksi.

   Vihje: Tässä tehtävässä ratkaisuksi riittää raaka laskentavoima silmukan muodossa.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki ohjelman toiminnasta, kun syöte on 997.

```
Hello! I am a prime detective.
Please, enter an integer:
997
997 is a prime number.
```

Esimerkki ohjelman toiminnasta, kun syöte on 10.

```
Hello! I am a prime detective.
Please, enter an integer:
10
10 is not a prime number.
```

Esimerkki ohjelman toiminnasta, kun syöte on 1.

```
Hello! I am a prime detective.
Please, enter an integer:
1
Error!
```

3. Tee Javalla ohjelma, joka esittää käyttäjälle kysymyksen ja hyväksyy vastaukseksi vain jommankumman kahdesta vastauksia vastaavasta kirjaimesta. Käyttäjälle tulostetaan virheilmoitus ja kysymys esitetään uudestaan niin pitkään kuin käyttäjä antaa vastaukseksi jonkin muun kuin ohjelman tunnistaman kirjaimen.

   Esimerkki ohjelman toiminnasta, kun ohjelma hyväksyy vastaukseksi joko pienen k-kirjaimen tai pienen e-kirjaimen:

```
Moi! Tinkaan sinulta merkin.
Vastaa (k)yllä tai (e)i:
K
Virhe!
Vastaa (k)yllä tai (e)i:
x
Virhe!
Vastaa (k)yllä tai (e)i:
k
Kiitokset!
```

4. Kirjoita Java-ohjelma, joka lukee käyttäjältä merkkijonon. Ohjelma tulostaa hylkäävän viestin ja pysähtyy, jos merkkijono alkaa välilyönnillä tai loppuu välilyöntiin. Ohjelma vastaavasti tulostaa hyväksyvän viestin ja pysähtyy, kun merkkijonon ensimmäinen ja viimeinen merkki eivät ole välilyöntejä. Esimerkiksi merkkijonot "a ", "ab " ja " abc ", " d " hylätään ja merkkijonot "a b" ja "abc" hyväksytään. Voit olettaa, että ohjelmalle annetaan aina vähintään yhden merkin mittainen merkkijono.

5. Tee Javalla *Concatenator*-ohjelma, joka lukee merkkijonoja ja yhdistää ne yhdeksi merkkijonoksi siten, että merkkijonot on erotettu toisistaan yhdellä välilyönnillä. Välilyöntiä ei lisätä viimeisen merkkijonon loppuun. Merkkijonojen lukeminen loppuu, kun syötteeksi annetaan "stop". Ohjelma ei lisää "stop"-merkkijonoa yhdistettyyn merkkijonoon. Voit olettaa, että ohjelmalle annetaan vähintään yksi merkkijono ennen "stop"-merkkijonoa. Ohjelma tulostaa yhdistetyn merkkijonon näytölle ennen pysähtymistään. Huomaa, että merkkijonojen vertailu on varminta *String*-luokan *equals*-metodilla.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

Esimerkki ohjelman toiminnasta:

```
Hello! I concatenate strings.
Please, enter a string:
After
Please, enter a string:
a
Please, enter a string:
while
Please, enter a string:
crocodile!
Please, enter a string:
stop
After a while crocodile!
```

6. Tee Javalla *RepetitiveCharacter*-ohjelma, joka lukee käyttäjältä merkkijonon ja merkin ja tutkii onko merkkijonossa merkin peräkkäisiä esiintymiä vai ei. Jos syötteet ovat esimerkiksi "abba" ja 'b', niin ohjelma toteaa merkin toistuvan, koska b-kirjainta seuraa toinen b-kirjain. Sen sijaan syötteillä "abba" ja 'a' ohjelma ilmoittaa ettei merkki toistu, koska pientä a-kirjainta ei seuraa yksi tai useampi pieni a-kirjain. Ohjelma katsoo pienet ja suuret kirjaimet eri merkeiksi. Voit olettaa, että ohjelmalle annetaan aina vähintään yhden merkin mittainen merkkijono.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat merkkijono "abba" ja merkki on pieni a-kirjain:

```
Hello! I say if a character repeats itself.
Please, enter a string:
abba
Please, enter a character:
a
Character 'a' is non-repetitive.
```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat merkkijono "Three, two, one... Lift off!" ja merkki on välilyönti:

```
Hello! I say if a character repeats itself.
Please, enter a string:
Three, two, one...   Lift off!
Please, enter a character:

Character ' ' is repetitive.
```

7. Tee Javalla ohjelma merkkijonojen merkkien limittämiseen. Käyttäjältä luetaan kaksi merkkijonoa, joista muodostetaan limitetty merkkijono siten, että ensimmäisen merkkijonon viimeinen kirjain on limitetyn jonon viimeinen kirjain, toisen merkkijonon viimeinen kirjain on limitetyn jonon toiseksi viimeinen kirjain, ensimmäisen merkkijonon toiseksi viimeinen kirjain on limitetyn merkkijonon kolmanneksi viimeinen kirjain ja niin edelleen. Merkkijonot voivat olla eripituisia. Merkit valitaan yksin pitemmästä jonosta sen jälkeen, kun pienemmästä jonosta ei voida enää valita. Jos merkkijonot ovat esimerkiksi "abc" ja "12", niin limitetty merkkijono on "a1b2c".

8. Tee Javalla *ShortestPart*-ohjelma, joka lukee merkkijonon ja päättelee sen lyhimmän osan pituuden. Merkkijonon osat on aina erotettu toisistaan yhdellä välilyönnillä. Voit olettaa, että ohjelmalle annetaan vähintään yhden merkin mittainen merkkijono ja että merkkijono ei ala välilyönnillä tai lopu välilyöntiin. Jos syöte on esimerkiksi "I can reboot you, but I won’t lie: It’s going to hurt.", niin tulos on yksi, koska merkkijonon lyhin osa on "I".

   Älä käytä taulukkoa tai muita tietorakenteita.

   Tämä tehtävä tarkistetaan automaattisesti. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki ohjelman toiminnasta:

   ```
   Hello! I find the length of the shortest substring.
   Please, enter a string:
   I can reboot you, but I won’t lie: It’s going to hurt.
   The length is 1.
   ```

   Esimerkki ohjelman toiminnasta:

   ```
   Hello! I find the length of the shortest substring.
   Please, enter a string:
   Open the pod bay doors, HAL.
   The length is 3.
   ```

   Esimerkki ohjelman toiminnasta:

   ```
   Hello! I find the length of the shortest substring.
   Please, enter a string:
   foobar
   The length is 6.
   ```