Harjoitus 1 (viikko 44)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@tuni.fi).
- Muista nimetä muuttujat hyvin sekä kommentoida ja sisentää koodisi. Tee ohjelman alkuun kommentti, jossa on harjoituksen ja tehtävän numero sekä nimesi ja sähköpostisoitteesi.
- Lue syötteet `Scanner`-luokalla. Käytä ohjelmassa vain yhtä `Scanner` -muuttujaa. Tietovirrassa tulee olla samanaikaisesti vain yksi lukija, jotta ohjelma toimii oikein kaikissa tilanteissa. Ohjelmasi pysähtyy ajonaikaiseen virheeseen WETOssa, jos käytät siinä useampaa kuin yhtä lukijaa.
- Ensi viikon maanantaina klo 12–14 pidettävissä Zoom-harjoituksissa saa apua ongelmakohtiin.
- Palautusaikaa on jatkettu. Palauta vastauksesi WETO-järjestelmään (https://wetodev.sis.uta.fi/) viimeistään ensi viikon sunnuntaina 1.11. klo 23.55.
- Tarvitset tehtävien tekoon Java-kääntäjän ja -tulkki. Löydät asennusohjeet kurssisivuilta. Harjoituksissa autetaan Javan asennuksessa omalle kannettavalle tietokoneelle.
- WETO tarkistaa kaikki tämän harjoituskerran tehtävät automaattisesti. Varmista, että ohjelmasi toimii täsmälleen esimerkkien mukaisesti, koska tarkistus tehdään opiskelijan ja malliohjelman tulosteita vertailemalla. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit – viimeinen rivi mukaan lukien – päätetään rivinvaihtoon. Kurssisivujen *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa annetaan lisäohjeita. Ota yhteyttä harjoitusryhmän vetäjään, jos et keksi järjellisessä ajassa miksi WETO hylkää palautuksesi.

1. Kirjoita Javalla *Adage*-niminen ohjelma, joka lukee käyttäjältä ajatelman ja tulostaa sen siten, että ennen ajatelmaa tulostuu merkkijono "<< " ja ajatelman jälkeen tulostuu merkkijono " >>". Ajatelma ja sitä edeltävä ja seuraava merkkijono tulostetaan samalle riville.

   Esimerkki ohjelman toiminnasta:

   ```
   Hello! I read and pretty print your thoughts.
   Please, say something:
   Anything that can go wrong will go wrong.
   << Anything that can go wrong will go wrong. >>
   ```

2. Osoitteessa: https://coursepages.uta.fi/tiep5-1/syksy-2020/harjoitukset/tehtavat/ on annettu *Bugeja.java*-tiedostossa ohjelma, jossa on kielioppivirheitä. Tiedostosta on annettu eri versiot Windows-tyyppisillä rivinvaihdoilla ja Mac- ja Linux-tyyppisillä rivinvaihdoilla. Tiedostojen ainoa ero on rivinvaihdoissa; molemmat tiedostot ovat utf-8 merkistössä.

   Lue luentorungon sivulta 1.1.16 kuinka kielioppivirheitä korjataan. Korjaa virheet yksi kerrallaan ja kirjoita ohjelmaan kommentit, joista selviää mitä virheitä koodissa oli ja kuinka korjasit ne. Älä poista tai lisää lauseita, koska virheet ovat lauseissa, eivät ohjelman rakenteessa.

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

   Palauta korjattu lähdekoodi WETOon. WETO tarkistaa, että korjattu ohjelma tulostaa:

   ```
   ------------------------
   | Bugit on nitistetty! |
   ------------------------
   ```

3. Kirjoita Javalla *NameDay*-niminen ohjelma, joka lukee käyttäjältä viikonpäivän, järjestysluvut päivälle, kuukaudelle ja vuodelle sekä nimipäiväänsä viettävän henkilön nimen. Jos nimiä on useampia, niin käyttäjä antaa nimet pilkulla ja välilyönnillä 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, koska 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`-metodia. Weto hylkää ohjelman, jos siinä on yli 120 merkin mittaisia rivejä.

   Oletetaan, että käyttäjä antaa järkeviä syötteitä.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Friday", 27, 10, 2019 ja "Helle, Helli, Hellin, Hellä":

   ```
   Hello! I read and print date information.
   Name of day:
   Sunday
   Day:
   27
   Month:
   10
   Year:
   2019
   Given names of day:
   Helle, Helli, Hellin, Hellä
   Sunday 27.10.2019 Helle, Helli, Hellin, Hellä
   ```

4. Kirjoita Javalla *Arithmetics*-niminen ohjelma, jossa testaat aritmeettiset Javan operaattorit `+`, `-`, `*`, `/` 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. Huomaa, että tarvitset tyyppimuunnoksen, jotta Java ei hävitä desimaaleja jakolausekkeessa *ekaLuku* / *tokaLuku*. Esimerkiksi:

   ```
   osamäärä = (double)ekaLuku / tokaLuku;
   ```

   Tulosta lopuksi tulosmuuttujien arvot näytölle. Tulosta osamäärä kahden desimaalin tarkkuudella *System.out.printf*-metodia käyttäen esimerkiksi näin:

   ```
   System.out.printf("%d / %d = %.2f%n", ekaLuku, tokaLuku, osamäärä);
   ```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 7 ja 4:

```
Hello! I do some basic arithmetic.
Please, enter the first integer:
7
Please, enter the second integer:
4
7 + 4 = 11
7 - 4 = 3
7 * 4 = 28
7 / 4 = 1.75
7 % 4 = 3
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 2 ja 1:

```
Hello! I do some basic arithmetic.
Please, enter the first integer:
2
Please, enter the second integer:
1
2 + 1 = 3
2 - 1 = 1
2 * 1 = 2
2 / 1 = 2.00
2 % 1 = 0
```

5. Kirjoita Javalla *StringLengthComparator*-niminen ohjelma merkkijonojen pituuksien vertailuun. Ohjelma lukee käyttäjältä kaksi merkkijonoa ja kertoo alla olevien esimerkkien mukaisesti onko ensimmäinen merkkijono lyhempi kuin toinen merkkijono, ovatko merkkijonot samanpituiset tai onko ensimmäinen merkkijono pitempi kuin toinen merkkijono.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "summer" ja "you think":

   ```
   Hello! I compare the lengths of two strings.
   Please, enter the first string:
   summer
   Please, enter the second string:
   you think
   "summer" is shorter than "you think".
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "pain" ja "life":

   ```
   Hello! I compare the lengths of two strings.
   Please, enter the first string:
   pain
   Please, enter the second string:
   life
   "pain" is as long as "life".
   ```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "metre" ja "yard":

```
Hello! I compare the lengths of two strings.
Please, enter the first string:
metre
Please, enter the second string:
yard
"metre" is longer than "yard".
```

6. Kirjoita Javalla *CharacterComparator*-niminen ohjelma, joka lukee merkkijonon ja kaksi indeksiarvoa ja tutkii ovatko annetuissa paikoissa olevat merkit samat. Pienet ja suuret kirjaimet katsotaan eri merkeiksi. Ohjelma tulostaa virheilmoituksen, jos jompikumpi indeksiarvo tai molemmat indeksiarvot ovat virheelliset. Jälkimmäinen indeksiarvo luetaan, vaikka ensimmäinen indeksiarvo olisikin virheellinen. Javassa merkkijonon merkin laillinen indeksiarvo on välillä [0, *n* - 1], missä *n* on merkkijonon pituus.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Java", 1 ja 3:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Java
   Please, enter the first position:
   1
   Please, enter the second position:
   3
   'a' is equal to 'a'.
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Python", 5 ja 0:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Python
   Please, enter the first position:
   5
   Please, enter the second position:
   0
   'n' is different from 'P'.
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Tis but a scratch", -1 ja 4:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Tis but a scratch
   Please, enter the first position:
   -1
   Please, enter the second position:
   4
   Error!
   ```

7. Tee Javalla *Congratulator*-niminen 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.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 7 ja 8:

   ```
   Hello! I may congratulate you.
   Please, enter your target:
   7
   Please, enter your achievement:
   8
   You made it! Congratulations!
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 7 ja 7:

   ```
   Hello! I may congratulate you.
   Please, enter your target:
   7
   Please, enter your achievement:
   7
   You made it! Congratulations!
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 8 ja 7:

   ```
   Hello! I may congratulate you.
   Please, enter your target:
   8
   Please, enter your achievement:
   7
   ```

8. Tee Javalla *Glass*-niminen ohjelma, joka tiedustelee onko käyttäjä optimisti vai pessimisti ja tulostaa vastauksen mukaan `if-else`-lausetta käyttäen joko "The glass is half full." tai "The glass is half empty." Voit olettaa, että käyttäjän antama syöte on aina joko pieni o- tai p-kirjain.

   Esimerkki ohjelman toiminnasta, kun syöte on "o":

   ```
   Hello! I tell about glasses.
   Are you an (o)ptimist or a (p)essimist?
   o
   The glass is half full.
   ```

   Esimerkki ohjelman toiminnasta, kun syöte on "p":

   ```
   Hello! I tell about glasses.
   Are you an (o)ptimist or a (p)essimist?
   p
   The glass is half empty.
   ```