Harjoitus 1 (viikko 44)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 3.11. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 3.11. klo 14–16 (paikka ilmoitetaan myöhemmin).
- 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. Osoitteesta http://www.sis.uta.fi/~laki2/harjoitukset/harjoitus01/ on annettu *HelloWorld*-ohjelma. Kerro kommenteissa miksi ohjelma ei käänny vaikka *main*-operaatiossa on esitelty muuttuja, jonka nimi on sama kuin *say*-operaation parametrin nimi. Korjaa ohjelmaa siten, että se kääntyy (eli *say*-operaation kutsu onnistuu). Älä muuta *say*-operaatiota äläkä varsinkaan käytä attribuuttia ongelman ratkaisuun!

2. Kirjoita Javalla operaatio, jossa valitaan satunnaisesti kortti 52 kortin pakasta. Operaatio tulostaa valintansa merkkijonona, joka ensimmäinen merkki on maan kirjain (S = pata, C = risti, H = hertta ja D = ruutu) ja loput merkit ovat muodostuvat kortin numeroarvosta (1–13). Jos operaatio arpoo esimerkiksi herttakuningattaren, näytölle tulostuu merkkijono "H12". Operaatiolla ei ole paluuarvoa (**void**-määre) eikä operaatiolle välitetä parametreja (parametrilista on tyhjä). Kutsu operaatiotasi 10 kertaa *main*-operaatiosta.

   Voit arpoa kokonaisluvun väliltä [1, YLARAJA] esimerkiksi näin:

```
// Math.random-operaatio tuottaa satunnaisen liukuluvun väliltä [0, 1[.
// Satunnainen kokonaisluku saadaan aikaiseksi kertomalla arvottu
// liukuluku sopivasti ja poistamalla desimaalit tyyppimuunnoksella.
int arvottu = (int)(YLARAJA * Math.random()) + 1;
```

3. The Javalla maaginen kasipallo. Ohjelma lukee käyttäjältä kysymyksen ja tulostaa satunnaisesti jokin 20 vastauksesta. Ohjelmassa tulee olla paluuarvoton (**void**) ja parametriton operaatio (tyhjä parametrilista), joka valitsee ja tulostaa vastauksen. Kysymys luetaan *main*-operaatiossa, josta käsin vastauksen tuottavaa operaatiota kutsutaan. Edellisessä tehtävässä on annettu eräs menetelmä satunnaisluvun tuottamiseen. Löydät kasipallon kuvauksen ja pallon antamat vastaukset täältä: https://en.wikipedia.org/wiki/Magic_8-Ball.

   Esimerkki ohjelman toiminnasta:

```
Hello! I am a Magic 8 Ball. Ask me.
Enter a question:
Are you telling the truth?
Better not tell you now.
```

4. Kirjoita Javalla *HelloYou*-ohjelma, jossa on paluuarvoton (**void**) operaatio, joka tervehtii ohjelman käyttäjää nimellä. Käyttäjän nimi välitetään operaatiolle *String*-tyyppisen parametrin avulla. Parametrin arvoa ei tarvitse tarkistaa. Lue käyttäjän nimi *main*-operaatiossa ja kutsu kirjoittamaasi operaatiota *main*-operaatiosta siten, että annat käyttäjältä lukemasi syötteen operaatiolle parametrina.

   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 shall say hello to you.
   Enter your name:
   Conan the Librarian
   Hello Conan the Librarian!
   ```

5. Kirjoita Javalla operaatio, joka saa parametrinaan viikonpäivän järjestysluvun (**int**) ja tulostaa näytölle lukua vastaavan viikonpäivän nimen. Luku 1 vastaa maanantaita, luku 2 vastaa tiistaita ja niin edelleen. Koska operaatio on tyypitön (**void**), se ei palauta paluuarvoa. Voit olettaa, että operaation parametrin arvo on aina välillä 1–7. Lue järjestysnumero käyttäjältä *main*-operaatiossa ja kutsu operaatiotasi antaen käyttäjän syötteen operaation parametrin arvoksi.

6. Kirjoita tyypitön (**void**) kolmiparametrinen Java-operaatio, jossa tutkitaan onko parametrina (**int**) saatu luku *a* välillä, jonka rajat *b* ja *c* ovat luvun tapaan parametreja (**int**). Päättelyn tulos tulostetaan näytölle. Väli on suljettu: luvun katsotaan kuuluvan väliin, kun luvun ja rajan arvot ovat samat.

   Rajoille ei ole määrätty keskinäistä järjestystä. Luvun katsotaan kuuluvan väliin, jos se kuuluu väliin jommallakummalla rajojen keskinäisellä järjestyksellä. Operaation tulee näin tunnistaa tilanteet, joissa *b* on välin alaraja ($b \leq a \leq c$) tai yläraja ($c \leq a \leq b$). Jos tarkasteltava luku on esimerkiksi 7 ja rajat ovat 1 ja 10, niin ohjelma katsoo luvun kuuluvan välille, koska riittää, että järjestyksen $1 \leq 7 \leq 10$ havaitaan pitävän paikkansa.

   Lue luku ja välin rajat käyttäjältä *Interval*-ohjelman *main*-operaatiossa ja anna syötteet parametrien arvoiksi, kun kutsut operaatiotasi *main*-operaatiosta.

   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 determine whether an integer belongs to an interval.
   Enter the first endpoint:
   7
   Enter the second endpoint:
   1
   Enter the integer:
   1
   The integer belongs to the interval.
   ```

Toinen esimerkki ohjelman toiminnasta:

```
Hello! I determine whether an integer belongs to an interval.
Enter the first endpoint:
-1
Enter the second endpoint:
10
Enter the integer:
11
The integer does not belong to the interval.
```

7. Kirjoita Java-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. Merkkien vertailu ja tuloksen tulostaminen tapahtuu tyypittömässä (**void**) operaatiossa, jolle merkkijono (*String*) ja indeksiarvot (**int**) välitetään parametreina. Operaatiota kutsutaan *main*-operaatiossa, jossa luetaan käyttäjän syötteet.

   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.
```

   Vinkki: Lausekielinen ohjelmointi I -kurssin viidennen harjoituksen kuudes tehtävä.

8. Tee Javalla tyypitön (**void**) operaatio, joka tutkii moniko ensimmäisen merkkijonon merkeistä esiintyy jälkimmäisessä merkkijonossa. Merkkijonot välitetään operaatiolle *String*-tyyppisinä parametreina. Operaatio tulostaa esiintymien lukumäärän näytölle. Isot ja pienet kirjaimet katsotaan eri merkeiksi. Oletetaan, että kukin ensimmäisen merkkijonon merkeistä esiintyy jonossaan vain kerran. Näin ollen ensimmäinen merkkijono ei voi olla esimerkiksi "aa".

   Tulos on yksi, jos ensimmäinen merkkijono on esimerkiksi "ab" ja toinen merkkijono "Big data", koska merkki 'a' esiintyy jälkimmäisessä merkkijonossa, mutta merkillä 'b' ei ole esiintymiä. Huomaa, että esiintymät lasketaan kyllä/ei-asteikolla. Tästä syystä edellisessä esimerkissä tulos on yksi, vaikka merkillä 'a' on kaksi esiintymää jälkimmäisessä merkkijonossa.

   Molemmat merkkijonot luetaan käyttäjältä *main*-operaatiossa. Kutsu kirjoittamaasi operaatiota *main*-operaatiosta siten, että annat käyttäjältä lukemasi syötteen operaatiolle parametrina. Anna ohjelmasi nimeksi *Occurrences*.

   Vihje: Sisäkkäiset silmukat.

   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 character occurrences between strings.
Enter the first string:
em
Enter the second string:
Tampere
The number of occurrences is 2.
```

   Toinen esimerkki ohjelman toiminnasta:

```
Hello! I count character occurrences between strings.
Enter the first string:
k
Enter the second string:
Kuopio
The number of occurrences is 0.
```

Harjoitus 2 (viikko 45)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 10.11. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 10.11. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Osoitteessa http://www.sis.uta.fi/~laki2/harjoitukset/harjoitus02/ on annettu *Dice*-ohjelma. Ohjelma heittää operaatiossa kuusisivuista noppaa ja yrittää välittää tuloksen takaisin operaation kutsupaikkaan. Kerro kommenteissa miksi *sideNumber*-muuttujan arvo ei muutu, vaikka *main*-operaation muuttuja (*sideNumber*) on samanniminen kuin noppaa heittävän operaation parametri. Korjaa ohjelmaa siten, että saat *sideNumber*-muuttujalle operaatiossa arvotun arvon. Älä siirrä arvontaa pois *rollDice*-operaatiosta äläkä käytä attribuuttia ongelman ratkaisuun.

   Vinkki: paluuarvo.

2. Muokkaa 1. harjoituksen 5. tehtävässä tehtyä operaatiota siten, että operaatio palauttaa *String*-tyyppisenä paluuarvona viikonpäivän järjestyslukua vastaavan viikonpäivän nimen. Lue järjestysnumero käyttäjältä *main*-operaatiossa ja kutsu operaatiotasi antaen käyttäjän syötteen operaation parametrin arvoksi. Sijoita operaation palauttama arvo muuttujaan ja tulosta muuttujan arvo näytölle. Huomaa, että operaatio ei tulosta mitään, vaan kaikki tulostus tapahtuu nyt *main*-operaatiossa.

3. Kirjoita Javalla **int**-tyyppinen operaatio, joka tulostaa käyttäjälle ohjeen, lukee käyttäjältä luvun ja palauttaa käyttäjän syötteen paluuarvonaan. Ohje välitetään operaatiolle *String*-tyyppisenä parametrina. Kutsu kirjoittamaasi operaatiota *main*-operaatiosta. Tulosta *main*-operaatiossa näytölle operaation paluuarvo. Esimerkki ohjelman toiminnasta, kun ohje on "Anna kengännumero:":

   ```
   Moi! Luen satunnaisen tiedon.
   Anna kengännumero:
   42
   Tiedoksi: 42.
   ```

4. Tee Javalla **boolean**-tyyppinen operaatio, joka tarkistaa kellonajan, jossa tunnit ovat välillä 0–23, minuutit välillä 0–59 ja sekunnit välillä 0–59. Operaatiolla on kolme parametria: tunnit, minuutit ja sekunnit välitetään operaatiolle **int**-tyyppisinä parametreina. Paluuarvo on **true**, jos kellonaika on laillinen. Lue tunnit, minuutit ja sekunnit käyttäjältä *main*-operaatiossa. Kutsu operaatiota *main*-operaatiosta ja anna syötteet operaatiosi parametrien arvoiksi kutsussa. Sijoita operaation palauttama arvo muuttujaan. Tulosta päättelyn tulos näytölle muuttujan arvon avulla *main*-operaatiossa. Ohjelman nimi on *TimeInspector*.

Tämä tehtävä tarkistetaan automaattisesti toiminnallisuuden osalta ja lisäksi opettajan toimesta hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma läpäisee WETO-testit.

Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon. Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista

Esimerkki ohjelman toiminnasta:

```
Hello! I inspect a 24-hour time.
Please, enter hours:
23
Please, enter minutes:
59
Please, enter seconds:
60
Time is invalid.
```

Toinen esimerkki ohjelman toiminnasta:

```
Hello! I inspect a 24-hour time.
Please, enter hours:
23
Please, enter minutes:
59
Please, enter seconds:
59
Time is valid.
```

5. Kirjoita Javalla *StringInspector1*-ohjelma, jolla on **boolean**-tyyppinen operaatio, joka tutkii alkaako merkkijono välilyönnillä tai loppuuko se välilyöntiin. Merkkijono välitetään operaatiolle *String*-tyyppisenä parametrina. Operaatiolla ei ole muita parametreja. Paluuarvo on **true**, jos merkkijonon alussa tai lopussa on välilyönti. Paluuarvo on **false**, jos merkkijonon alussa ja lopussa ei ole välilyöntiä. Oletetaan, että operaatiolle annetaan aina vähintään yhden merkin mittainen merkkijono.

   Lue merkkijono käyttäjältä *main*-operaatiossa ja kutsu operaatiotasi antaen käyttäjän syöte operaation parametrin arvoksi. Sijoita operaation palauttama arvo muuttujaan. Tulosta päättelyn tulos *main*-operaatiossa näytölle muuttujan arvon avulla. Huomaa, että operaatio ei tulosta mitään, vaan kaikki tulostus tapahtuu nyt *main*-operaatiossa.

   Vinkki: Lausekielinen ohjelmointi I -kurssin 6. harjoituksen 4. tehtävä.

   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 " ab":

```
Hello! I inspect a string.
Please, enter the string:
 ab
Starts or ends with a space.
```

Esimerkki ohjelman toiminnasta, kun syöte on "a b":

```
Hello! I inspect a string.
Please, enter the string:
a b
Does not start or end with a space.
```

6. Tee Javalla *StringInspector2*-ohjelma, jonka **boolean**-tyyppinen operaatio tutkii onko välilyöntimerkillä peräkkäisiä esiintymiä operaatiolle parametrina välitetyssä *String*-tyyppisessä merkkijonossa. Merkkijonon oletetaan olevan aina vähintään yhden merkin mittainen. Operaatiolla ei ole muita parametreja. Operaation paluuarvo on **true**, jos merkkijonosta löytyy välilyönnin toistoja. Muussa tapauksessa palautetaan **false**. Paluuarvo on **false** myös, jos merkkijono on yhden merkin mittainen, koska tällöin merkki ei voi toistua.

   Lue *main*-operaatiossa käyttäjältä merkkijono ja kutsu kirjoittamaasi operaatiota *main*-operaatiosta siten, että annat käyttäjältä lukemasi syötteen operaatiolle parametrina. Sijoita operaation palauttama arvo *main*-operaatiossa muuttujaan ja tulosta *main*-operaatiossa muuttujan arvon avulla näytölle tieto välilyönnin toistumisesta tai toistumattomuudesta. Huomaa, että operaatio ei tulosta mitään, vaan kaikki tulostus tapahtuu nyt *main*-operaatiossa.

   Vinkki: Lausekielinen ohjelmointi I -kurssin 6. harjoituksen 6. tehtävä.

   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 merkkijono "a b c":

```
Hello! I say if a space repeats itself.
Please, enter a string:
a b c
No repetitive occurrences.
```

   Esimerkki ohjelman toiminnasta, kun syöte on merkkijono "Three, two, one... Lift off!":

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

7. Tee Javalla **boolean**-tyyppinen operaatio, joka tarkistaa parametrinaan saamansa merkkijonon (*String*) välilyöntien osalta. Merkkijono on operaation ainut parametri. Operaation paluuarvo on **true**, jos merkkijono alkaa välilyönnillä, loppuu välilyöntiin tai jos merkkijonossa on peräkkäisiä välilyönnin esiintymiä. Muuten palautetaan **false**. Voit olettaa merkkijonon olevan aina vähintään yhden merkin mittainen.

   Tarkista merkkijonon alku ja loppu 5. tehtävän operaatiota käyttäen ja tutki toistuuko merkkijono 6. tehtävän operaatiolla. Kopioi 5. ja 6. tehtävissä tekemäsi omat operaatiot ohjelmaasi *main*-operaation ja tässä tehtävässä tekemäsi oman operaation seuraksi. Kutsu kopioituja operaatiota omasta operaatiostasi antaen molempien kopioitujen operaatioiden parametriksi tässä tehtävässä toteuttamasi operaation parametriarvo.

   Esimerkki:

```
...
public ... tarkistaValit(String tutkittava) {
   ...
   // Kutsutaan 5.ja 6. tehtävän operaatioita antamalla tämän
   // operaation parametrin arvo kutsuttavien operaatioiden
   // parametriarvoksi. Paluuarvo sijoitetaan apumuuttujaan.
   boolean valiAlussaTaiLopussa = onkoValiaPaissa(tutkittava);
   boolean toistuukoVali = onkoValinToistoa(tutkittava);
   ...
}
...
```

   Yllä ... tarkoittaa pois jätettyä ohjelman osuutta.

   Lue *main*-operaatiossa käyttäjältä merkkijono ja kutsu kirjoittamaasi operaatiota *main*-operaatiosta siten, että annat käyttäjältä lukemasi syötteen operaatiolle parametrina. Sijoita operaation palauttama arvo *main*-operaatiossa muuttujaan ja tulosta muuttujan arvon avulla *main*-operaatiossa näytölle tieto tarkistuksen tuloksesta. Huomaa, että operaatio ei tulosta mitään, vaan kaikki tulostus tapahtuu *main*-operaatiossa.

Harjoitus 3 (viikko 46)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 17.11. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 17.11. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Esittele, luo ja alusta yksiulotteinen taulukko, jonka alkiot ovat **boolean**-tyyppiä. Tulosta taulukon alkiot näytölle. Käytä loogisten AND-, OR- ja XOR-operaatioiden operandeina taulukon ensimmäistä ja viimeistä alkiota ja tulosta operaatioiden tulokset näytölle. Jos taulukon alkiot ovat esimerkiksi { **true**, **false**, **true** }, niin operadit ovat **true** ja **true**. Voit kirjoittaa koodisi *main*-operaation sisään. Tämäkin ohjelma tehdään Javalla. Käyttäjältä ei tarvitse lukea syötteitä.

2. Kirjoita Javalla ohjelma, joka lukee aluksi käyttäjältä taulukon koon ja alkuarvon taulukon alkioille. Tämän jälkeen ohjelma luo pyydetyn kokoisen yksiulotteisen **int**-tyyppisten alkioiden taulukon, alustaa kunkin alkion käyttäjän antamalla arvolla ja tulostaa taulukon alkiot näytölle. Ohjelma ei luota käyttäjään sokeasti, vaan tarkistaa käyttäjän antaman koon ja ei tee mitään, jos koko on nolla tai negatiivinen. Voit kirjoittaa kaiken koodisi *main*-operaation sisään.

3. Lisää 2. harjoituksen 2. tehtävän operaatioon yksiulotteinen taulukko, joka sisältää viikon seitsemän päivän nimet. Taulukon alkioiden arvot ovat "maanantai", "tiistai", ... , "lauantai" ja "sunnuntai". Huomaa, että voit poistaa operaatiossa aiemmin olleen pitkän valintarakenteen ja lukea järjestyslukua vastaavan nimen suoraan taulukosta, koska väliltä 1–7 olevan järjestysluvun voi muuttaa helposti väliltä 0–6 olevaksi indeksiarvoksi.

   Lue järjestysnumero käyttäjältä *main*-operaatiossa ja kutsu operaatiotasi antaen käyttäjän syötteen operaation parametrin arvoksi. Sijoita operaation palauttama arvo muuttujaan ja tulosta muuttujan arvo näytölle. Huomaa, että operaatio ei tulosta mitään, vaan kaikki tulostus tapahtuu *main*-operaatiossa.

4. Kirjoita Javalla **int**-tyyppinen operaatio, joka laskee montako kertaa **int**-tyypinen luku esiintyy yksiulotteisessa **int**-tyyppisten alkioiden muodostamassa taulukossa. Luku ja taulukko annetaan operaatiolle parametreina. Operaatiolla ei ole muita parametreja. Tarkista operaatiossa että, taulukolle on varattu muistia. Operaatio ei tee muuta kuin palauttaa negatiivisen arvon, jos taulukkoparametri on **null**-arvoinen.

   Luo *main*-operaatiossa yksiulotteinen **int**-tyyppisten alkioiden taulukko ja alusta sen alkiot satunnaisilla luvuilla. Lue taulukosta haettava luku käyttäjä ja kutsu operaatiota antamalla parametriarvoiksi käyttäjän syöte ja taulukko. Sijoita operaatiosi palauttama arvo *main*-operaatiossa muuttujaan ja kerro muuttujan arvon perusteella laskennan tulos. Tulosta myös taulukon sisältö näytölle.

Voit käyttää taulukon täyttämiseen ja tulostamiseen luentorungon sivuilla 2.22 ja 2.23 annettuja operaatioita joko sellaisenaan tai muokattuna.

Tämä tehtävä tarkistetaan opettajan toimesta sekä toiminnallisuuden että hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma toimii oikein.

5. Kirjoita Javalla **void**-tyyppinen operaatio, joka tulostaa yksiulotteisen **char**-tyyppisistä alkioista koostuvan taulukon alkiot näytölle alla annettujen esimerkkien tapaan. Taulukko on operaation ainoa parametri. Varaudu operaatiossa **null**-arvoiseen parametriin **if**-lauseella. Operaatio ei tulosta mitään, jos parametri on **null**-arvoinen.

   Kutsu operaatiotasi *main*-operaatiossa, jossa tulostettava taulukko myös luodaan. Käyttäjältä ei tarvitse lukea syötteitä.

   Vihje: luentorungon sivu 2.23.

   Esimerkki operaation tulosteesta, kun taulukossa on yksi alkio, jonka arvo on 'a':

   ```
   { 'a' }
   ```

   Esimerkki operaation tulosteesta, kun taulukon alkioiden arvot ovat 'x' ja 'y':

   ```
   { 'x', 'y' }
   ```

   Esimerkki operaation tulosteesta, kun taulukon alkioiden arvot ovat '1', '2' ja '3':

   ```
   { '1', '2', '3' }
   ```

6. Kirjoita Javalla *ArrayFiller*-ohjelma, jolla on **void**-tyyppinen operaatio taulukon täyttämiseen. Operaatio saa parametrinaan yksiulotteisen **char**-tyyppisistä alkioista koostuvan taulukon ja lukee käyttäjältä arvon kullekin taulukon alkiolle. Voit olettaa, että taulukolle on varattu aina muistia. Taulukko on operaation ainoa parametri.

   Lue *main*-operaatiossa käyttäjältä taulukon koko ja luo annetun kokoinen yksiulotteinen **char**-tyyppisten alkioiden taulukko. Voit olettaa, että syöte on aina vähintään yksi. Kutsu taulukon luomisen jälkeen kirjoittamaasi operaatiota siten, että annat luodun taulukon operaation parametriksi.

   Tulosta lopuksi taulukon sisältö näytölle. Tulostus tapahtuu 5. tehtävässä tehdyllä operaatiolla, jonka voi kopioida ja liittää sellaisenaan tässä ohjelmassa tehtyyn ohjelmaan. Kutsu tulostavaa operaatiota *main*-operaatiosta, kun taulukko on täytetty tässä tehtävässä tehdyssä operaatiossa.

   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 create and fill an array.
Please, enter the size of array:
4
Enter value:
J
Enter value:
a
Enter value:
v
Enter value:
a
{ 'J', 'a', 'v', 'a' }
```

7. Kirjoita Javalla *ElementSwapper*-niminen ohjelma. Ohjelmalla on **boolean**-tyyppinen operaatio, joka vaihtaa kahden alkion arvot päikseen **char**-tyyppistä alkioista koostuvassa yksiulotteisessa taulukossa. Taulukko ja vaihdettavien alkioiden indeksiarvot (**int**) välitetään operaatiolle parametrina. Jos parametrit ovat esimerkiksi { 'r', 'e ', 'n', 'o' }, 0 ja 2, niin taulukon sisältö on operaation suorittamisen jälkeen { 'n ', 'e ', 'r', 'o' }.

   Tarkista, että taulukolle on varattu muistia ja että indeksiarvot ovat järkeviä. Indeksiarvojen on oltava välillä [0, *n* - 1], missä *n* on taulukon alkioiden lukumäärä. Operaatio palauttaa **true**-arvon, jos parametrit olivat kunnossa. Muuten palautetaan **false**.

   Lue *main*-operaatiossa käyttäjältä taulukon koko ja luo annetun kokoinen yksiulotteisen **char**-tyyppisten alkioiden taulukko. Voit olettaa, että syöte on aina vähintään yksi. Täytä taulukko 6. tehtävässä tehdyllä operaatiolla, jonka voi kopioida ja liittää sellaisenaan tässä ohjelmassa tehtyyn ohjelmaan. Tulosta taulukko sen täytön jälkeen 5. tehtävän operaatiolla, jonka saa käyttöön tässäkin tehtävässä kopioimalla.

   Lue käyttäjältä indeksiarvot *main*-operaatiossa. Kutsu sitten arvot vaihtavaa operaatiota *main*-operaatiosta, sijoita operaation palauttama arvo *main*-operaatiossa muuttujaan ja kerro muuttujan arvon perusteella voitiinko arvot vaihtaa. Tulosta lopuksi taulukon sisältö näytölle 5. tehtävän operaatiolla.

   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 swap element values.
Please, enter the size of array:
3
Enter value:
n
Enter value:
e
Enter value:
d
{ 'n', 'e', 'd' }
Enter the first index value:
0
Enter the second index value:
1
The element values were swapped.
{ 'e', 'n', 'd' }
```

Toinen esimerkki ohjelman toiminnasta:

```
Hello! I swap element values.
Please, enter the size of array:
4
Enter value:
n
Enter value:
e
Enter value:
r
Enter value:
o
{ 'n', 'e', 'r', 'o' }
Enter the first index value:
0
Enter the second index value:
4
The element values were not swapped.
{ 'n', 'e', 'r', 'o' }
```

Harjoitus 4 (viikko 47)

- **Kaikki tämän harjoituksen tehtävät liittyvät joko suoraan tai epäsuorasti kurssin toiseen harjoitustyöhön. Saa hyvän alun harjoitustyön tekoon, kun ratkaiset mahdollisimman monta tehtävää.**
- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 24.11. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 24.11. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Kirjoita Javalla *Underliner*-ohjelma, joka tulostaa ensimmäisenä komentoriviparametrina saamansa merkkijonon toisena komentoriviparametrina saamallaan merkillä alustettuna. Ohjelma tulostaa alleviivatun tekstin vain, jos komentoriviparametreja on kaksi. Muissa tapauksissa tulostetaan virheilmoitus. Voit kirjoittaa kaiken koodisi *main*-operaatioon.

   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 komentoriviparametrit ovat "The secret to getting ahead is getting started." ja "-":

   ```
   The secret to getting ahead is getting started.
   -----------------------------------------------
   ```

   Esimerkki ohjelman toiminnasta, kun komentoriviparametrit ovat "See", "you" ja "later".

   ```
   Error!
   ```

   Komentoikkunassa testattaessa välilyöntejä sisältävä merkkijono tulee sulkea lainausmerkkeihin, jotta merkkijonon osia ei pidettäisi erillisinä parametreina. Esimerkki:

   ```
   java Underliner "The secret to getting ahead is getting started." -
   ```

2. Kirjoita Javalla **void**-tyyppinen operaatio, joka tulostaa kaksiulotteisen **char**-alkioiden taulukon näytölle siten, että jokaisen rivin merkit tulostetaan peräkkäin ilman erottimia. Myös viimeinen rivi päätetään rivinvaihtoon. Taulukko on operaatio ainoa parametri.

   Varaudu operaatiossa **null**-arvoiseen parametriin **if**-lauseella. Kutsu operaatiotasi *main*-operaatiossa, jossa käsiteltävä taulukko myös luodaan. Käyttäjältä ei tarvitse lukea syötteitä.

   Esimerkki ohjelman toiminnasta, kun taulukon alkiot ovat { { 'a', 'b', 'c' }, { 'd', 'e', 'f' } }:

   ```
   abc
   def
   ```

3. Alla olevassa taulukossa on annettu 16 merkkiä lukuvastineineen. Taulukon viimeinen merkki on välilyönti. Kirjoita Javalla **char**-tyyppinen operaatio, joka palauttaa lukua vastaavan merkin. Luku välitetään operaatiolle parametrina. Sijoita merkit yksiulotteiseen taulukkoon ja valitse palautettava merkki taulukosta parametrin arvon avulla. Tarkista ennen taulukosta lukemista, että parametri on oikeellinen. Paluuarvo on ~~negatiivinen~~ jokin muu kuin alla olevassa taulukossa määritelty merkki, jos luku ei ole välillä 0–15. Virhekoodina voi käyttää myös *Character*-luokan vakioita. Esimerkiksi *Character.MAX_VALUE* kävisi virhearvoksi.

#	**@**	**&**	**$**	**%**	**x**	*****	**o**	**l**	**!**	**;**	**:**	**'**	**,**	**.**	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Kutsu operaatiotasi *Judge*-ohjelman *main*-operaatiosta antamalla käyttäjän syöte operaatiosi parametriksi. Sijoita operaation palauttama arvo muuttujaan ja muodosta tuloste muuttujan avulla.

Tämä tehtävä tarkistetaan automaattisesti toiminnallisuuden osalta ja lisäksi opettajan toimesta hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma läpäisee WETO-testit.

Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon. Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.

Esimerkki ohjelman toiminnasta:

```
Hello! I am a judge of character.
Please, enter an integer:
5
I judge 5 as 'x'.
```

Esimerkki ohjelman toiminnasta:

```
Hello! I am a judge of character.
Please, enter an integer:
16
I cannot judge 16.
```

4. Tee Javalla *ShortestPart*-ohjelma, jolla on **int**-tyypinen operaatio, joka päättelee parametrina saamansa merkkijonon lyhimmän osan pituuden. Merkkijonon osat on aina erotettu toisistaan yhdellä välilyönnillä. Voit lisäksi olettaa, että merkkijono on aina 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".

   Pilko merkkijono ensin osiin *String*-luokan *split*-operaatiolla, joka sijoittaa osat yksiulotteiseen *String*-tyyppisten alkioiden taulukkoon. Päättele sitten taulukkoa tutkimalla mikä osista on lyhin.

   Lue merkkijono käyttäjältä *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriksi. Sijoita operaation palauttama arvo muuttujaan ja käytä muuttujaa, kun tulostat tuloksen näytölle. Esimerkki *split*-operaation käytöstä:

   ```
   // Merkkijonon osia erottava merkki on annettava hakasulkeissa,
   // koska split-operaation parametri on säännöllinen ilmaus.
   String[] osat = rivi.split("[ ]");
   ```

   Vinkki: Lausekielinen ohjelmointi I -kurssin 6. harjoituksen 8. tehtävä.

   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.
   ```

5. Kirjoita Javalla **boolean**-tyyppinen operaatio, joka lajittelee parametrinaan saamansa yksiulotteisen **int**-tyyppisten alkioiden taulukon arvot nousevaan järjestykseen. Operaatio palauttaa arvon **true**, jos taulukolle on varattu muistia. Paluuarvo on muuten **false**. Jos taulukon arvot ovat esimerkiksi { 1, 0, 2, 5 }, niin operaation kutsun jälkeen taulukon arvot ovat { 0, 1, 2, 5 }.

   Lajittele alkiot valintalajittelulla (selection sort), joka on yksinkertainen ja hidas algoritmi. Algoritmi pseudokoodina:

```
Algoritmi Valintalajittelu {
   // Ensimmäisen lajittelemattoman alkion indeksiarvo.
   i ← 0;
   // Lajitellaan kunnes kaikki alkiot on käsitelty. Kunkin
   // kierroksen jälkeen välin [0, i – 1] alkiot on lajiteltu
   // ja välin [i, n – 1] alkiot ovat lajittelematta.
   while (i < n – 1) {
      // Haetaan pienin alkio väliltä [i, n – 1]
      // ja asetetaan tämän alkion indeksiarvo muuttujaan j.
      j ← valitse(i);
      // Vaihdetaan ensimmäinen lajittelematon alkio
      // ja pienin alkio keskenään.
      vaihda(i, j);
      // Pienin alkio on nyt oikeassa paikassa. Suljetaan se pois
      // lajittelusta laskuria kasvattamalla.
      i ← i + 1;
   }
}
```

   Anna ohjelman nimeksi *SelectionSorter*. Lue *main*-operaatiossa käyttäjältä taulukon koko ja luo yksiulotteinen **int**-tyyppisten alkioiden taulukko. Tiedustele taulukon alkioiden arvot käyttäjältä joko *main*-operaatiossa tai omassa operaatiossa (3. harjoituksen 6. tehtävän operaatio muokattuna).

   Kutsu lajittelevaa operaatiota antamalla parametriarvoksi käyttäjän täyttämä taulukko. Tulosta taulukon sisältö näytölle operaation kutsun jälkeen. Taulukon tulostus tehdään operaatiolla, jonka voi muokata esimerkiksi luentomateriaalin sivulla 2.23 annetusta operaatiosta tai kopioida 3. harjoituksen 4. tehtävän mallivastauksesta.

   Tämä tehtävä tarkistetaan automaattisesti toiminnallisuuden osalta ja lisäksi opettajan toimesta hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma läpäisee WETO-testit.

   Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon. Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista

Esimerkki ohjelman toiminnasta:

```
Hello! I sort an array.
Please, enter the size of array:
4
Enter value:
1
Enter value:
0
Enter value:
2
Enter value:
5
{ 0, 1, 2, 5 }
```

6. Mediaani on järjestetyn joukon keskimmäinen alkio. Esimerkiksi joukon { 0, 0, 0, 0, 6, 15, 15, 15, 15 } mediaani on 6. Mediaani on kahden keskimmäisimmän alkion keskiarvo, jos joukossa on parillinen määrä alkioita. Esimerkiksi joukon { 0, 1, 2, 5 } mediaani on (1 + 2) / 2 = 1,5.

   Kirjoita Javalla **double**-tyyppinen operaatio, joka saa parametrinaan yksiulotteisen **int**-tyyppisten alkioiden taulukon ja palauttaa paluuarvonaan taulukon alkioiden mediaanin. Paluuarvo on Javan oma vakio *Double.NaN*, jos taulukolle ei ole varattu muistia. Lajittele taulukko 5. tehtävässä tekemälläsi metodilla.

   Anna ohjelman nimeksi *Median*. Lue *main*-operaatiossa käyttäjältä taulukon koko ja luo yksiulotteinen **int**-tyyppisten alkioiden taulukko. Tiedustele taulukon alkioiden arvot käyttäjältä joko *main*-operaatiossa tai omassa operaatiossa (3. harjoituksen 6. tehtävän operaatio muokattuna). Kutsu lajittelevaa operaatiota antamalla parametriarvoksi käyttäjän täyttämä taulukko ja sijoita paluuarvo muuttujaan. Tulosta muuttujan arvo näytölle *printf*-operaatiolla seuraavasti:

```
// Alla md paluuarvon sisältävä double-tyyppinen muuttuja.
System.out.printf("Median is %.1f.%n", md);
```

   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 determine median.
Please, enter the size of array:
4
Enter value:
1
Enter value:
0
Enter value:
2
Enter value:
5
Median is 1.5.
```

Harjoitus 5 (viikko 48)

- **Moni tämän harjoituksen tehtävistä liittyy joko suoraan tai epäsuorasti kurssin toiseen harjoitustyöhön. Harjoitustyö edistyy sitä paremmin, mitä enemmän tehtäviä ratkaiset.**
- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta julkaistaan kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 1.12. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 1.12. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Löydät osoitteesta http://www.sis.uta.fi/~laki2/harjoitukset/harjoitus05/ *Characters*-ohjelman. Ohjelma on toteutettu erittäin huonosti, koska parametreihin ja paluuarvoihin perustuva tiedonvälitys on korvattu attribuuteilla. Attribuutit on tarkoitettu käsitettä mallintavan luokan keskeisimpien tietojen säilömiseen olio-ohjelmoinnissa. Pelkästään tiedonvälitykseen käytettynä attribuuteista on lähinnä haittaa, koska modulaarisuusperiaatetta rikkovat ohjelmat ovat vaikeaselkoisia. Poista attribuutit ja kirjoita ohjelmasta versio, jossa tiedot välitetään operaatioiden välillä parametreja ja paluuarvoja käyttäen. Huomaa, että tällä kurssilla ei saa käyttää attribuutteja elleivät ne ole vakiomuotoisia (**final**-määreellä määriteltyjä).

2. Paljon nolla-arvoisia alkioita sisältävää matriisia kutsutaan harvaksi. Matriisin harvuutta voi mitata nolla-alkioiden ja kaikkien alkioiden suhteena.

   Kirjoita Javalla **double**-tyyppinen operaatio, joka saa parametrinaan kaksiulotteisena taulukkona esitetyn matriisin. Taulukon alkiot ovat **int**-tyyppisiä. Operaatio palauttaa paluuarvonaan matriisin harvuutta kuvaavan arvon. Paluuarvo on negatiivinen, jos parametri on **null**-arvoinen. Jos parametri on esimerkiksi arvoiltaan { { 0, 0 }, { 1, 1 }, { 0, 2 } }, on operaation paluuarvo 3 / 6 = 0,5.

   Esittele ja alusta *main*-operaatiossa taulukko. Kutsu operaatiotasi *main*-operaatiosta antamalla taulukko operaatiosi parametriksi, sijoita operaation palauttama arvo muuttujaan ja kerro tämän muuttujan avulla tulos käyttäjälle. Tulosta taulukon arvot näytölle. Käyttäjältä ei tarvitse lukea syötteitä.

3. Matriisin alkioiden kertominen skalaarilla on usein käytetty matriisioperaatio.

   Ohjelmoi Javalla **void**-tyyppinen operaatio, jolle välitetään parametrina kaksiulotteisena taulukkona esitetty matriisi ja luku (**double**), jolla alkioiden arvot kerrotaan. Taulukko koostuu **double**-tyyppisistä alkioista. Operaatio ei muuta alkioita, jos taulukolle ei ole varattu muistia. Jos parametrit ovat arvoiltaan esimerkiksi { { 0.5, 0.1 }, { 0.2, 1.5 } } ja 2, niin taulukon sisältö on { { 1.0, 0.2 }, { 0.4, 3.0 } } operaation suorittamisen jälkeen.

   Esittele ja alusta *main*-operaatiossa taulukko. Kutsu operaatiotasi *main*-operaatiosta antamalla taulukko ja kerroin operaatiosi parametreiksi. Tulosta taulukon arvot ennen ja jälkeen operaatiokutsun. Käyttäjältä ei tarvitse lukea syötteitä.

4. Kirjoita Java-kielellä **double**-tyyppinen operaatio, joka laskee ja palauttaa tekstitiedostossa olevien kokonaislukujen summan. Kullakin tiedoston rivillä on yksi tai useampi kokonaisluku. Luvut on erotettu toisistaan yhdellä pilkulla. Tiedoston voi olettaa olevan kunnossa. Siinä ei esimerkiksi ole kirjaimia tai välilyöntejä. Tiedoston nimi (*String*) välitetään operaatiolle parametriarvona. Paluuarvo on Javan vakio *Double.NaN*, mikäli tapahtuu poikkeus.

   String-luokan *split*-operaatio on hyödyllinen lukujen erottelussa toisistaan. Merkkijonona esitetyn luvun voi muuttaa kokonaisluvuksi esimerkiksi *Integer*-luokan *parseInt*-operaatiolla. Esimerkki *parseInt*-operaation käytöstä:

```
String lukuMerkkijonona = "42";
int luku = Integer.parseInt(lukuMerkkijonona);
```

   Double.NaN-vakion tunnistamisen on käytettävä *Double.isNaN*-operaatiota. Esimerkki:

```
// Summa voitiin laskea.
if (!Double.isNaN(summa))
```

   Lue tiedoston nimi ja merkki *IntegerAdder*-ohjelman *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriarvoksi. Sijoita operaatiosi palauttama arvo muuttujaan. Kerro tämän muuttujan arvon avulla *main*-operaatiossa laskennan tulos.

   Huomaa, että tätä ohjelmaa tai mitä tahansa tiedostoja käsittelevää ohjelmaa ei kannata kokeilla tärkeällä tiedostolla ellei ole täysin varma, että testitiedosto ei esimerkiksi tuhoudu ohjelmointivirheen seurauksena.

   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, kun tiedoston *data.txt* rivit ovat "11,2", "32" ja "678,-12,13":

```
Hello! I add integers.
Please, enter file name:
data.txt
Sum is 724.0.
```

   Esimerkki, kun hakemistossa ei ole tiedostoa *x-files.txt*:

```
Hello! I add integers.
Please, enter file name:
x-files.txt
I could not add.
```

5. Kirjoita Java-kielellä **int**-tyyppinen operaatio, joka laskee ja palauttaa parametrina annetun merkin (**char**) esiintymien lukumäärän tekstitiedostossa. Myös tiedoston nimi (*String*) välitetään operaatiolle. Paluuarvo on negatiivinen, mikäli tapahtuu poikkeus. Pienet ja isot kirjaimet katsotaan eri merkeiksi.

   Lue tiedoston nimi ja merkki *CharacterCounter*-ohjelman *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriarvoksi. Sijoita operaatiosi palauttama arvo muuttujaan. Kerro tämän muuttujan arvon avulla *main*-operaatiossa laskennan tulos.

   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:

```
Hello! I count characters.
Please, enter file name:
LineBreaker.java
Please, enter character:
e
I found 409 occurrences.
```

   Esimerkki, kun hakemistossa ei ole tiedostoa *x-files.txt*:

```
Hello! I count characters.
Please, enter file name:
x-files.txt
Please, enter character:
x
I could not count.
```

6. Kirjoita Javalla **int**-tyyppinen operaatio, joka laskee kuinka monta riviä tekstitiedostossa on. Tiedoston nimi (*String*) annetaan operaatiolle parametrina. Operaation paluuarvo on negatiivinen, jos operaatiossa tapahtuu poikkeus.

   Lue tiedoston nimi *LineCounter*-ohjelman *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriarvoksi. Sijoita operaatiosi palauttama arvo muuttujaan. Kerro tämän muuttujan arvon avulla *main*-operaatiossa laskennan tulos.

   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:

```
Hello! I count lines.
Please, enter file name:
LineBreaker.java
There are 302 lines.
```

Esimerkki, kun hakemistossa ei ole tiedostoa *x-files.txt*:

```
Hello! I count lines.
Please, enter file name:
x-files.txt
I could not count.
```

7. Kirjoita Javalla operaatio, joka luo kaksiulotteisen **char**-tyyppisten alkioiden taulukon ja lataa tekstitiedoston merkit tähän taulukoon siten, että tekstitiedoston ensimmäisen rivin merkki on taulukon ensimmäisessä paikassa (0, 0) ja viimeisen rivin viimeinen merkki on taulukon viimeisessä paikassa (*n* - 1, *m* - 1), missä *n* on taulukon rivien lukumäärä ja *m* on taulukon sarakkeiden lukumäärä.

   Saat selville taulukon rivien lukumäärän ennen taulukon luomista tehtävässä 6. kirjoitetulla operaatiolla. Taulukon sarakkeiden lukumäärän voi selvittää ensimmäisen rivin lukemisen jälkeen. Oletetaan, että tiedoston rivit ovat samanmittaisia.

   Tiedoston nimi välitetään operaatiolle parametrina (*String*). Operaation tyyppi on **char**[][], koska operaatio palauttaa viitteen merkit sisältävään taulukkoon. Paluuarvo on **null**, jos operaatiossa tapahtuu poikkeus.

   Lue tiedoston nimi *FileLoader*-ohjelman *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriarvoksi. Sijoita paluuarvo muuttujaan, jonka tyyppi on **char**[][], jotta operaatiosi luoma taulukko-olio ei häviä. Tulosta taulukko kutsumalla 4. harjoituksen 2. tehtävän tulostusoperaatiota *main*-operaatiosta siten, että annat muuttujan operaation parametriksi.

   Tämä tehtävä tarkistetaan automaattisesti toiminnallisuuden osalta ja lisäksi opettajan toimesta hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma läpäisee WETO-testit.

   Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon. Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja, joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.

   Esimerkki, kun tiedoston *letters.txt* rivit ovat "abc" ja "def":

```
Hello! I load files.
Please, enter file name:
letters.txt
abc
def
```

   Esimerkki, kun hakemistossa ei ole tiedostoa *x-files.txt*:

```
Hello! I load files.
Please, enter file name:
x-files.txt
I could not load.
```

Harjoitus 6 (viikko 49)

Nämä ovat kurssin viimeiset harjoitukset. Muista, että hyväksytyistä ratkaisuista ja läsnäoloista kerättyjen pisteiden summan tulee olla vähintään 60 % (29 pistettä) tehtävien ja läsnäolopisteiden kokonaislukumäärien summasta (41 + 6 = 47).

Voit tehdä lisätehtäviä (korkeintaan 8 kpl), jos aiot suorittaa kurssin, mutta pisteesi ovat alle 60 % -rajan tämän harjoituksen jälkeen. Kurssin vastuuopettaja ottaa yhteyttä opiskelijoihin, jotka voivat saavuttaa 60 %:n rajan lisätehtäviä tekemällä.

Tiistaina 6.12. ei järjestetä opetusta itsenäisyyspäivän vuoksi. Korvaavat ajat ja paikat ovat 2. ryhmälle ma 5.12. klo 10–12 (B1084), 3. ryhmälle ma 5.12. klo 14–16 (ML51) ja 4. ryhmälle ke 7.12. klo 14–16 (B1084). Voit vierailla toisessa ryhmässä, jos et pääse ryhmääsi korvaavaan aikaan.

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@uta.fi).
- Muista lisätä **static**-määre operaatioidesi otsikoihin, jotta ohjelmasi kääntyvät.
- Muista noudattaa hyvän ohjelmointitavan perusteita (Lausekielinen ohjelmointi I -kurssin luentomateriaalin 14. luku) ja uusia hyviä tapoja (luentomateriaalin 7. luku), joihin kuuluu operaation otsikkoon liittyvä yleisluonteinen kommentti operaation tarkoituksesta sekä operaation mahdollisesti saamista ja palauttamista tiedoista.
- Ratkaisut palautetaan WETO-järjestelmään, joka tarkistaa lähdekoodia automaattisesti. *In*-luokkaa (katso alla) ei tarvitse eikä tule palauttaa. Lisätietoja tarkistuksesta löytyy kurssisivujen *Opetus* | *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa.
- Ensi viikolla pidettävissä mikroharjoituksissa saa apua ongelmakohtiin. Keskiviikon klo 12–14 -ryhmässä (ML51) avustetaan hieman enemmän kuin muissa ryhmissä.
- Palauta vastauksesi WETO-järjestelmään viimeistään ensi viikon torstaina 8.12. klo 12.00.
- Mallivastaukset esitellään luentosaliharjoituksissa ensi viikon torstaina 8.12. klo 14–16 (B1083).
- Syötteiden lukemiseen tarvittava *In*-luokka löytyy kurssin kotisivuilta *Koodit*-kohdasta.

1. Kirjoita Javalla **boolean**-tyyppinen operaatio, joka lukee ensin käyttäjältä merkkijonojen lukumäärän ja lukee sitten yhden merkkijonon kerrallaan ja kirjoittaa kunkin lukemansa merkkijonon omalle rivilleen *lines.txt*-nimiseen tiedostoon. Jokaisen tiedoston rivin tulee päättyä rivinvaihtoon. Voi olettaa, että käyttäjä antaa lukumääräksi aina vähintään yhden merkkijonon. Huomaa, että WETO ei hyväksy ohjelmaasi, jos tiedoston nimi on jokin muu kuin *lines.txt*. Tulosta tervehdys *StringSaver*-ohjelman *main*-operaatiossa ja kutsu sitten omaa operaatiotasi. Sijoita operaation palauttama arvo muuttujaan.

   Ole hyvin varovainen, kun kirjoitat tiedostoon. Ohjelmasi hävittää tiedoston aiemman sisällön, jos hakemistossa on jo tiedosto nimellä, jonka määrittelet ohjelmassa.

   Tämä tehtävä tarkistetaan automaattisesti sekä tulosteita että kirjoitettuja tiedostoja vertailemalla. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että näytölle tulostuvien rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki:

```
Hello! I save strings.
Enter the number of strings:
3
Enter a string:
Hofstadter's Law:
Enter a string:
It always takes longer than you expect,
Enter a string:
even when you take into account Hofstadter's Law.
```

2. Kirjoita Javalla **boolean**-tyyppinen operaatio, joka lukee tiedoston rivit ja kirjoittaa ne samassa järjestyksessä, mutta käännettynä, *reverse.txt*-nimiseen tiedostoon. Luettavan tiedoston nimi (*String*) annetaan operaatiolle parametrina. Voi olettaa, että parametri ei ole "reverse.txt". Kunkin tiedoston rivin tulee päättyä rivinvaihtoon. Paluuarvo on **false**, mikäli lukiessa tai tallentaessa tapahtuu poikkeus. Huomaa, että WETO ei hyväksy ohjelmaasi, jos tiedoston nimi on jokin muu kuin *reverse.txt*.

   Lue tiedoston nimi *LineReverser*-ohjelman *main*-operaatiossa ja kutsu operaatiotasi antamalla käyttäjän syöte operaatiosi parametriarvoksi. Sijoita operaatiosi palauttama arvo muuttujaan. Kerro tämän muuttujan arvon avulla *main*-operaatiossa laskennan tulos.

   Voit käyttää halutessasi *StringBuilder*-luokan *reverse*-operaatiota rivin merkkien kääntämiseen. Esimerkki:

```
// Luodaan muuttuva merkkijono tavallisesta String-tyyppisestä
// merkkijonosta "rivi" ja käännetään muuttuvan merkkijonon merkit.
StringBuilder uusiRivi = new StringBuilder(rivi);
uusiRivi.reverse();
```

   Muuttuvan merkkijonon voi antaa sellaisenaan ilman tyyppimuunnoksia *PrintWriter*-luokan *println*-operaatiolle.

   Tämä tehtävä tarkistetaan automaattisesti sekä tulosteita että kirjoitettuja tiedostoja vertailemalla. Varmista, että ohjelmasi toimii esimerkkien mukaisesti. Huomaa, että näytölle tulostuvien rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit päätetään rivinvaihtoon.

   Esimerkki, jossa *in.txt*-tiedoston rivit ovat "this", "is a" ja "test":

```
Hello! I reverse files.
Please, enter file name:
in.txt
File reversed.
```

   Yllä annetun esimerkin jälkeen *reverse.txt*-tiedoston rivit ovat "siht", "a si" ja "tset".

   Esimerkki, kun hakemistossa ei ole tiedostoa *x-files.txt*:

```
Hello! I reverse files.
Please, enter file name:
x-files.txt
I could not reverse.
```

3. Kirjoita Javalla operaatio taulukon kopiointiin. Operaatio saa parametrinaan kaksiulotteisen **int**-alkioiden taulukon, joka on operaation ainoa parametri. Operaatio luo uuden taulukon, joka on samansuuruinen kuin kopioitava taulukko ja kopioi parametrina saamansa taulukon alkioiden arvot uuteen taulukoon. Operaation tyyppi on **int**[][], koska operaatio palauttaa viitteen uuteen taulukkoon. Paluuarvo on **null**, jos kopioitavalle taulukolle ei ole varattu muistia.

   Jos kopioitavan taulukon alkioiden arvot ovat esimerkiksi { { 1, 2, 3 }, { 4, 5, 6 } }, niin uuden taulukon alkioiden arvot ovat täysin samat kuin vanhassa taulukossa: { { 1, 2, 3 }, { 4, 5, 6 } }.

   Esittele ja alusta *main*-operaatiossa taulukko. Kutsu operaatiotasi *main*-operaatiosta antamalla taulukko operaatiosi parametriksi. Sijoita paluuarvo muuttujaan, jonka tyyppi on **int**[][]. Tulosta vanhan taulukon arvot ennen operaatiokutsua erillistä operaatiota käyttäen. Tulosta uuden taulukon arvot samalla operaatiolla operaation kutsun jälkeen. Löydät taulukon tulostavan operaation esimerkiksi edellisen viikkoharjoituksen mallivastauksista. Käyttäjältä ei tarvitse lukea syötteitä.

   Älä käytä *Arrays*-luokkaa.

   Tämä tehtävä tarkistetaan opettajan toimesta sekä toiminnallisuuden että hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma toimii oikein.

4. Tee Java-kielellä **int**[]-tyyppinen operaatio, joka palauttaa viitteen taulukoon, jossa ovat operaation parametrinaan saaman yksiulotteisen **int**-alkioiden taulukon alkiot ensimmäinen alkio pois lukien. Yleisemmin luonnehtien operaatio "leikkaa" parametrinaan saamaansa taulukkoa siten, että tuloksesta on pudonnut pois ensimmäinen alkio. Operaatiolla ei ole muita parametreja taulukon lisäksi. Paluuarvo on viite nollan alkion kokoiseen tyhjään taulukkoon, jos parametritaulukossa on vain yksi alkio. Paluuarvo on **null**, jos parametrina saadulle taulukolle ei ole varattu muistia (parametri on **null**-arvoinen).

   Jos parametrina saadun taulukon alkioiden arvot ovat esimerkiksi { 1, 2, 3, 4 }, niin uuden taulukon sisältö on { 2, 3, 4 }.

   Esittele ja alusta *main*-operaatiossa yksiulotteinen **int**-tyyppisten alkioiden taulukko. Kutsu operaatiotasi *main*-operaatiosta antamalla taulukko operaation parametriksi. Sijoita operaation luomaan taulukko-olioon viite *main*-operaatiossa. Tulosta vanhan ja uuden taulukon arvot näytölle erillisellä operaatiolla, jota kutsutaan *main*-operaatiosta. Käyttäjältä ei tarvitse lukea syötteitä.

   Älä käytä *Arrays*-luokkaa.

   Tämä tehtävä tarkistetaan opettajan toimesta sekä toiminnallisuuden että hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys voi tuottaa nollan, vaikka ohjelma toimii oikein.

5. Kirjoita Javalla **double**-tyyppinen operaatio, joka saa parametrinaan yksiulotteisen **int**-tyyppisten alkioiden taulukon ja palauttaa paluuarvonaan taulukon alkioiden summan. Taulukko on operaation ainoa parametri. Summa lasketaan silmukkaa käyttäen. Paluuarvo on Javan oma vakio *Double.NaN*, jos taulukolle ei ole varattu muistia. **Null**-tarkistus on tehtävä, vaikka tässä tehtävässä operaatiolle annetaan aina olemassa oleva taulukko.

   Lue käyttäjältä taulukon koko ja luo yksiulotteinen **int**-tyyppisten alkioiden taulukko ~~*RecursiveAdder*~~ *IterativeAdder*-ohjelman *main*-operaatiossa. Tiedustele taulukon alkioiden arvot käyttäjältä erillisessä operaatiossa (katso 4. harjoituksen 6. tehtävän mallivastaus), jota kutsutaan *main*-operaatiosta. Kutsu summan laskevaa operaatiota *main*-operaatiosta antamalla parametriarvoksi käyttäjän täyttämä taulukko ja sijoita paluuarvo muuttujaan.

   Tulosta muuttujan arvo näytölle *println*-operaatiolla ilman nolladesimaalia. Esimerkki:

```
// Alla summa on paluuarvon sisältävä double-tyyppinen muuttuja.
System.out.println("Sum is " + (int)summa + ".");
```

   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 add array element values.
Please, enter the size of array:
4
Enter value:
1
Enter value:
0
Enter value:
2
Enter value:
5
Sum is 8.
```

6. Toteuta viidennen tehtävän summan laskeva operaatio **rekursiivisesti**. **Operaatiossa ei saa olla silmukoita**.

   Rekursio syntyy kenties luontevimmin, kun huomataan, että esimerkiksi taulukon { 1, 0, 2, 5 } alkioiden summa on 1 + taulukon { 0, 2, 5 } alkioiden summa ja vain yhden alkion sisältävä taulukko on perustapaus, jossa summa saadaan "suoraan".

   Ohjelmaan saa lisätä operaatioita. Lisäoperaatioissa saa olla silmukoita. Operaation otsikkoon saa halutessaan lisätä parametreja. Neljännessä tehtävässä tehdystä operaatiosta on apua, jos ongelmaa lähestyy yllä hahmotellulla tavalla, jolloin operaation otsikon voi jättää entiselleen.

   RecursiveAdder-ohjelma toimii muuten kuten viidennessä tehtävässä on määritelty.

   Älä käytä *Arrays*-luokkaa.