Harjoitus 5 (viikko 48)

- Osa tämän harjoituksen tehtävistä liittyy joko suoraan tai epäsuorasti kurssin harjoitustyöhön. Harjoitustyö edistyy sitä paremmin, mitä enemmän tehtäviä ratkaiset.
- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@tuni.fi).
- Muista nimetä muuttujat hyvin sekä kommentoida ja sisentää koodisi. Kirjoita metodin otsikkoon liittyvä yleisluonteinen kommentti metodin tarkoituksesta sekä metodin mahdollisesti saamista ja palauttamista tiedoista. Tee ohjelman alkuun kommentti, jossa on harjoituksen ja tehtävän numero sekä nimesi ja sähköpostiosoitteesi.
- Lue syötteet `Scanner`-luokalla. Käytä vain yhtä `Scanner`-oliota. Esittele lukija vakiomuotoisena attribuuttina, jos syötteitä on luettava kahdessa tai useammassa metodissa (katso luentorungon luku 2.2).
- Ensi viikon maanantaina klo 12–14 pidettävissä Zoom-harjoituksissa saa apua ongelmakohtiin.
- Palauta vastauksesi WETO-järjestelmään (https://wetodev.sis.uta.fi/) viimeistään ensi viikon torstaina 26.11. klo 23.55.
- WETO tarkistaa kaikki tämän harjoituskerran tehtävät automaattisesti. Kurssisivujen *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa on annettu lisäohjeita. Ota yhteyttä harjoitusryhmän vetäjään, jos et keksi järjellisessä ajassa miksi WETO hylkää palautuksesi.
- Kaikki opettajien tarkistamat tehtävät arvioidaan hyvän ohjelmointitavan osalta. Näin esimerkiksi huono sisennys tai puutteellinen kommentointi voi tuottaa nollan, vaikka ohjelma läpäisee WETOn testit. Muista erityisesti metodin otsikkoon liittyvä yleisluonteinen kommentti. Opettaja antaa paluuarvollisen oman metodin sisältävälle ratkaisulle pisteen vain, jos metodin paluuarvo sijoitetaan muuttujaan. Oman metodin kutsuja on oltava kussakin tehtävässä annettu määrä, jotta pisteen voi saada.

1. Tee Javalla *ShortestPart*-ohjelma, jolla on `int`-tyyppinen *mittaaLyhinOsa*-metodi, joka päättelee parametrina saamansa merkkijonon (`String`) lyhimmän osan pituuden ja palauttaa päättelyn tuloksen paluuarvonaan. Paluuarvo on -1, jos parametri on `null`-arvoinen tai parametrin arvo on tyhjä merkkijono "". Metodi ei lue syötteitä eikä tulosta mitään.

   Merkkijonon osat on aina erotettu toisistaan yhdellä välilyönnillä. Lisäksi voidaan olettaa, että merkkijono ei ala välilyönnillä tai lopu välilyöntiin ja että välilyönnit eivät toistu. 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`-metodilla, joka sijoittaa osat yksiulotteiseen `String`-tyyppisten alkioiden taulukkoon. Päättele sitten taulukkoa tutkimalla mikä osista on lyhin. Esimerkki `split`-metodin käytöstä:

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

   Lue merkkijono käyttäjältä pääohjelmassa ja kutsu metodiasi yhden kerran antamalla käyttäjän syöte metodisi parametriksi. Sijoita metodin palauttama arvo muuttujaan ja käytä muuttujaa, kun tulostat tuloksen näytölle.

   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.
```

2. Kirjoita Javalla `void`-tyyppinen *tulosta2d*-metodi, joka tulostaa kaksiulotteisen `char`-tyyppisten alkioiden muodostan taulukon (`char[][]`) sisällön näytölle siten, että jokaisen rivin merkit tulostetaan peräkkäin ilman erottimia. Taulukko on metodin ainoa parametri. Varaudu metodissa `null`-arvoiseen parametriin `if`-lauseella: metodi ei tee mitään, jos taulukolle ei ole varattu muistia.

   Kutsu metodiasi vähintään viidesti *CharacterArray2DPrinter*-ohjelman pääohjelmassa erilaisilla parametriarvoilla. Anna yhdessä kutsussa parametrin arvoksi `null`.

   Metodin tuloste, kun metodin parametriarvo on taulukko {{'a', 'b', 'c'}, {'d', 'e', 'f'}}:

   ```
   abc
   def
   ```

   Metodin tuloste, kun metodin parametriarvo on taulukko {{ 'a', 'b', 'c' }}:

   ```
   abc
   ```

   Metodin tuloste, kun metodin parametriarvo on taulukko {{'a'}, {'b'}, {'c'}}:

   ```
   a
   b
   c
   ```

   Metodin tuloste, kun metodin parametriarvo on taulukko {{ 'a' }}:

   ```
   a
   ```

3. Kirjoita Javalla *CharacterArray2DFiller*-niminen ohjelma, jolla on `char[][]`-tyyppinen *luo2d*-niminen metodi, jossa luodaan kaksiulotteinen `char`-tyyppisen alkioiden taulukko ja alustetaan sen alkiot. Taulukon rivien ja sarakkeiden lukumäärä välitetään metodille `int`-tyyppisinä parametreina. Rivien lukumäärä on metodin ensimmäinen parametri. Metodin kolmas parametri (`char`) on merkki, joka asetetaan taulukon kaikkien alkioiden arvoksi. Metodilla ei ole muita parametreja. Metodi palauttaa viitteen alustettuun taulukkoon, jos molemmat parametriarvot ovat nollaa suurempia. Paluuarvo on muussa tapauksessa `null`. Metodi ei lue syötteitä eikä tulosta mitään.

Kopioi toisen tehtävän tulostusmetodi oman metodisi kaveriksi tämän tehtävän ohjelmaan. Lue taulukon rivien ja sarakkeiden lukumäärät sekä alustusmerkki käyttäjältä pääohjelmassa ja kutsu metodiasi antamalla käyttäjän syötteet metodisi parametriksi. Sijoita metodin palauttama arvo muuttujaan. Anna paluuarvon sisältämä muuttuja tulostusmetodin parametriksi pääohjelmassa, jos muuttujan arvo ei ole `null`. Tulosta muussa tapauksessa virheilmoitus.

Älä käytä *Arrays*-luokkaa.

Esimerkki ohjelman toiminnasta, kun syötteet ovat 1, 1 ja 'x':

```
Hello! I am an array filler.
Please, enter the number of rows:
1
Please, enter the number of columns:
1
Please, enter a character:
x
x
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 2, 3 ja '+':

```
Hello! I am an array filler.
Please, enter the number of rows:
2
Please, enter the number of columns:
3
Please, enter a character:
+
+++
+++
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat 0, 5 ja '-':

```
Hello! I am an array filler.
Please, enter the number of rows:
0
Please, enter the number of columns:
5
Please, enter a character:
-
Error!
```

4. Tässä tehtävässä kuormitetaan kolmannessa tehtävässä tehty *luo2d*-metodi. Kopioi siis aluksi se ja *tulosta2d*-metodi *CharacterArray2DFiller2*-ohjelmaan. Kirjoita sitten *luo2d*-metodista parametriton muoto, joka palauttaa aina viitteen 3×4 -kokoiseen taulukkoon, jonka alkiot on alustettu ristikkomerkillä #. Tee lopuksi *luo2d*-metodi, jolla on `int`-tyyppiset parametrit taulukon rivin ja sarakkeiden lukumäärille. Tämäkin metodi alustaa taulukon ristikkomerkillä.

   Älä monista koodia, vaan kutsu uusista metodeista alkuperäistä kolmiparametrista metodia ja palauta uusista metodeista alkuperäisen metodin palauttama arvo. Uudet metodit eivät lue syötteitä eivätkä tulosta mitään.

Kutsu ensimmäistä uutta metodia kerran ja toista uutta metodia vähintään kolmesti pääohjelmassa. Käytä toisen uuden metodin kutsussa erilaisia parametriarvoja. Anna yhdessä toisen uuden metodin kutsussa yhdelle tai kahdelle parametrille virheellinen arvo. Sijoita kutsujen palauttamat arvot muuttujiin ja tulosta taulukot antamalla muuttujat *tulosta2d*-metodin kutsujen parametriarvoiksi.

Tulosta2d-metodin tuloste on:

```
####
####
####
```

kun metodin parametriksi on annettu muuttuja, johon on sijoitettu parametrittoman *luo2d*-metodin palauttama arvo esimerkiksi näin:

```
char[][] oletustaulu = luo2d();
tulosta2d(oletustaulu);
```

Tulosta2d-metodin tuloste, kun metodin parametriksi on annettu muuttuja, johon on sijoitettu kaksiparametrisen *luo2d*-metodin palauttama arvo:

```
#
#
```

kun *luo2d*-metodia on kutsuttu parametriarvoilla 2 ja 1.

5. Kirjoita Javalla `boolean`-tyyppinen *vaihdaMerkit*-metodi, joka saa parametrinaan kaksiulotteisen `char`-tyyppisen alkioiden muodostaman taulukon (`char[][]`) sekä kaksi merkkiä (`char`) ja korvaa kaikki ensimmäisen merkin esiintymät toisella merkillä ja päinvastoin. Parametrit ovat tässä järjestyksessä eikä muita parametreja ei ole. Paluuarvo on `true`, jos parametrina saadulle taulukolle on varattu muistia ja taulukon koko on vähintään 1×1 -alkiota. Muussa tapauksessa paluuarvo on `false`. Metodi ei lue syötteitä eikä tulosta mitään.

   Kopioi toisen tehtävän tulostusmetodi *CharacterSwitcher*-ohjelmaan. Kutsu *vaihdaMerkit*-metodia vähintään kolmesti pääohjelmassa. Käytä kutsuissa erilaisia parametriarvoja. Anna yhdessä kutsussa parametrin arvoksi `null`. Sijoita kutsujen palauttamat arvot muuttujiin ja tulosta taulukot antamalla muuttujat *tulosta2d*-metodin kutsujen parametriarvoiksi.

   Muistatahan tässä ja seuraavassa tehtävässä, että loogisten AND- ja OR-operaattoreiden ehdottomat versiot (& ja |) pakottavat Javan arvioimaan molemmat operandinsa, vaikka tulos tiedettäisiin jo ensimmäisen operandin arvon perusteella. Ota siksi käyttöön ehdolliset operaattorit (&& ja ||) myös tässä tehtävässä, jos loogiset operaattorit ovat tarpeen.

   Tulosta2d-metodin tuloste:

```
+|-
-|+
```

   sen jälkeen, kun *vaihdaMerkit*-metodia on kutsuttu parametriarvoilla {{'-', '|', '+'}, {'+', '|', '-'}}, '+' ja '-'.

6. Tee Javalla `int[]`-tyyppinen *laskeFrekvenssit*-metodi, joka laskee montako esiintymää kullakin yksiulotteisessa taulukossa (`char[]`) olevalla merkillä on kaksiulotteisessa merkkien taulukossa (`char[][]`). Metodi luo ja palauttaa yksiulotteisen taulukon (`int[]`), joka sisältää laskettavien merkkien lukumäärät samassa järjestyksessä kuin merkit ovat parametrina saadussa yksiulotteisessa merkkien taulukossa.

   Paluuarvo on `null`, jos jommallekummalle parametrina saadulle taulukolle tai molemmille näistä taulukoista ei ole varattu muistia tai jos jompikumpi tai kumpikin taulukko on tyhjä.

   Jos laskettavat merkit sisältävän taulukon sisältö on esimerkiksi {'x', 'o'} ja kaksiulotteisessa taulukossa on yksi x-merkin esiintymä ja kolme o-merkin esiintymää, metodi palauttaa taulukon, jonka sisältö on {1, 3}.

   Kutsu metodiasi vähintään kolmesti *CharacterCounter*-ohjelman pääohjelmassa. Käytä kutsuissa erilaisia parametriarvoja. Anna yhdessä kutsussa molempien taulukkoparametrien arvoiksi `null`. Sijoita kutsujen palauttamat arvot muuttujiin. Metodin testaaminen muuttujien arvot tulostamalla on suositeltavaa ennen palautusta. Tulostukseen voi käyttää vaikkapa luentorungossa annettua valmista metodia.

7. Kirjoita Javalla *Indenter*-ohjelma, joka tulostaa komentoriviparametreina saamansa merkkijonot riveittäin siten, että ne on sisennetty välilyönneillä. Sisennyksen syvyys määritellään viimeisenä komentoriviparametrina. Ohjelma tulostaa merkkijonot vain, jos komentoriviparametreja on vähintään kaksi, sisennyksen syvyys voidaan muuntaa kokonaisluvuksi ja sisennyksen syvyys on positiivinen (≥ 0). Muissa tapauksissa tulostetaan virheilmoitus. Voit kirjoittaa kaiken koodisi pääohjelman sisään.

   Muunna sisennyksen koko merkkijonosta kokonaisluvuksi `Integer`-luokan `parseInt`-metodilla, joka heittää `NumberFormatException`-poikkeuksen. Käsittele mahdollinen poikkeus `try-catch`-lauseella.

   Esimerkki:

```
try {
   ...
   // Yritetään muuntaa viimeinen komentoriviparametri luvuksi.
   int syvyys = Integer.parseInt(args[args.length - 1]);
   ...
}
catch (NumberFormatException e) {
   ...
}
```

   Yllä … tarkoittaa esimerkistä pois jätettyjä osuuksia.

Esimerkki ohjelman toiminnasta:

```
see
you
later
```

kun ohjelma ajetaan komennolla:

```
java Indenter see you later 7
```

Esimerkki ohjelman toiminnasta:

```
Error!
```

kun ohjelma ajetaan komennolla:

```
java Indenter this is a test
```