Koitehtävä 6

Deadline 9.11. kello 11:45

Tee tehtäviä niin, että saat yhteensä 10 pistettä

1. **Selitä (1 p)**
    a. Mitä eroa on Arraylla ja ArrayListilla

Java-ohjelmointikielessä on kaksi erilaista tietorakennetta, jotka voivat tallentaa joukon arvoja: Array ja ArrayList. Näillä kahdella tietorakenteella on useita eroja:

1. **Kiinteä koko vs. dynaaminen koko**:
    - **Array**: Taulukko (Array) on kiinteän koon tietorakenne. Kun taulukko luodaan, sen koko määräytyy, eikä sitä voi muuttaa. Jos tarvitset suuremman tai pienemmän taulukon, sinun on luotava uusi taulukko ja kopioitava tiedot vanhasta uuteen.
    - **ArrayList**: ArrayList on dynaamisesti kasvava tietorakenne. Se alustetaan tietyn kokoisena, mutta se voi automaattisesti kasvaa, kun siihen lisätään lisää alkioita. ArrayList tarjoaa monia käteviä toimintoja, kuten lisääminen, poistaminen ja hakeminen, ilman, että sinun tarvitsee huolehtia taulukon koon hallinnasta.
2. **Tietotyypit**:
    - **Array**: Taulukko voi sisältää yhden tietotyypin arvoja, ja sen tietotyyppi määritetään taulukon luomisen yhteydessä. Esimerkiksi voit luoda int-taulukon tai String-taulukon.
    - **ArrayList**: ArrayList voi säilyttää mitä tahansa tietotyyppiä olevia arvoja. Se on geneerinen tietorakenne, joka voi sisältää erilaisia olioiden instansseja. Voit luoda ArrayListin, joka sisältää esimerkiksi String-olioita, Integer-olioita tai muita objekteja.
3. **Suorituskyky**:
    - **Array**: Taulukot voivat olla tehokkaampia, kun tiedät tarkalleen, kuinka monta alkioita tarvitset ja niiden tietotyypin. Ne ovat yksinkertaisempia ja vaativat vähemmän muistia.
    - **ArrayList**: ArrayListit ovat joustavampia mutta voivat vaatia enemmän muistia ja aiheuttaa hieman suorituskykykustannuksia dynaamisen koon hallinnasta.

Yhteenvetona, jos tiedät tarkalleen, kuinka monta alkioita tarvitset ja et tarvitse dynaamista koon muutosta, taulukko voi olla hyvä vaihtoehto. Jos sen sijaan tarvitset joustavuutta ja haluat lisätä ja poistaa alkioita helposti, ArrayList on parempi valinta. Usein ArrayList on monipuolisempi ja helpommin käytettävä vaihtoehto monimutkaisempiin ohjelmiin.

    b. Mitä tarkoittaa LinkedList? Miten se eroaa ArrayLististä?

LinkedList on toinen Java-ohjelmointikielessä käytettävä tietorakenne, joka voi tallentaa joukon arvoja. Se eroaa ArrayLististä monissa suhteissa, erityisesti tavassa, jolla se hallitsee ja tallentaa tietoja.

Tärkeimmät erot LinkedListin ja ArrayListin välillä ovat seuraavat:

1. **Tietorakenteen perusrakenne:**
    - **ArrayList**: ArrayList perustuu taulukkoon ja säilyttää alkionsa peräkkäisesti muistissa. Tämä tarkoittaa, että ArrayList voi nopeasti hakea alkioita niiden indeksin perusteella, mutta lisääminen ja poistaminen keskeltä listaa voi olla hidasta, koska kaikki seuraavat alkiot on siirrettävä.
    - **LinkedList**: LinkedList on linkitetty rakenne, joka koostuu solmuista. Jokainen solmu sisältää arvon ja viittauksen seuraavaan solmuun. Tämä mahdollistaa nopeat lisäykset ja poistot sekä listan alusta että keskeltä. Haku vaatii kuitenkin yleensä enemmän aikaa, koska sinun on käytävä läpi solmut yksi kerrallaan.
2. **Haku ja indeksöinti**:
    - **ArrayList**: ArrayList tarjoaa nopean indeksipohjaisen haun, koska voit saada suoraan minkä tahansa alkion sen indeksin perusteella. Tämä tekee ArrayListista hyödyllisen, kun sinun on nopeasti päästävä tiettyyn alkioon.
    - **LinkedList**: LinkedListin haku voi olla hidasta, koska sinun on aloitettava ensimmäisestä solmusta ja käytävä läpi solmut yksi kerrallaan etsittäessä tiettyä arvoa. Haku voi olla tehokkaampaa, jos tiedät, että etsittävä alkio on lähellä listan alussa.
3. **Lisäys ja poisto:**
    - **ArrayList**: ArrayListin lisääminen tai poistaminen keskeltä listaa voi olla hidasta, koska se vaatii elementtien siirtämistä. Lisäys tai poisto listan lopusta voi olla nopeampaa.
    - **LinkedList**: LinkedList mahdollistaa nopeat lisäykset ja poistot missä tahansa kohdassa listaa, koska sinun tarvitsee vain päivittää solmujen viittauksia. Tämä tekee siitä tehokkaamman, kun sinun on usein muokattava listan sisältöä.

Yhteenvetona, ArrayList on usein parempi valinta, kun tarvitset nopeita indeksipohjaisia hakuja ja tiedät, että listan sisältö ei muutu usein. LinkedList on hyödyllisempi, kun sinun on usein lisättävä, poistettava tai muokattava listan sisältöä, erityisesti keskeltä tai alusta. Valinta riippuu käyttötarkoituksesta ja suorituskykytarpeista.

    c. Mitä tarkoittaa Stack eli Pino?

Stack, suomeksi "pino", on abstrakti tietorakenne, joka noudattaa LIFO (Last-In, First-Out) -periaatetta. Tämä tarkoittaa, että viimeksi lisätty alkio on se, joka poistetaan ensimmäisenä. Pinon toimintaperiaate muistuttaa pinon, kuten lautapelin palikoiden tai kirjojen, järjestämistä päällekkäin.

Stackissa voit suorittaa kaksi perustoimintoa:

1. **Push**: Tämä toiminto lisää uuden alkion pinon päälle. Uusi alkio menee aina ylimmäksi pinossa.
2. **Pop**: Tämä toiminto poistaa ylimmän alkion pinosta. Ylimmäksi lisätty alkio poistetaan, ja pino pienenee yhdellä alkioilla.

Lisäksi stack tarjoaa yleensä operaation nimeltä "peek", joka mahdollistaa ylimmän alkion tarkastelun ilman sen poistamista.

Stackilla on monia käyttötarkoituksia ohjelmoinnissa. Esimerkiksi sitä voidaan käyttää seuraavissa tilanteissa:

- Kutsupinona (call stack): Ohjelman suorituksen aikana kutsutut funktiot ja niiden paikalliset muuttujat tallennetaan pinoon, jotta suoritusjärjestys voidaan palauttaa oikein.
- Undo-toimintojen hallinta: Voit käyttää pinoa tallentamaan toimintoja (esim. käyttäjän tekemiä muutoksia) siten, että voit peruuttaa ne viimeisestä alkaen.
- Syntaksipuiden evaluointi: Pinoja käytetään usein matemaattisten lausekkeiden syntaksipuiden arvioinnissa, kun on tarpeen noudattaa oikeaa operaatioiden suoritusjärjestystä.
- Muut tilanteet, joissa LIFO-järjestys on merkityksellinen.

Java-ohjelmointikielessä ja monissa muissa ohjelmointikielissä on valmiita stack-tietorakenteita tai luokkia, jotka helpottavat pinon toteuttamista ja käyttöä. Esimerkiksi Java tarjoaa "Stack" -luokan, mutta usein "Deque"

(Double-ended queue) -luokkaa käytetään myös pinona.

    d. Selitä mitä tarkoittavat käsitteet FIFO ja LIFO kun puhutaan listoista

Kun puhutaan listoista tai tietorakenteista, käsitteet FIFO ja LIFO viittaavat siihen, millä tavalla alkioita lisätään ja poistetaan listalta sekä missä järjestyksessä ne käsitellään.

1. **FIFO (First-In, First-Out):**
    - FIFO tarkoittaa, että ensimmäisenä listalle lisätty alkio on myös ensimmäisenä poistettava. Tämä periaate muistuttaa jonon toimintaa, kuten jonottamista kaupassa, jossa ensimmäisenä jonoon tullut asiakas palvellaan ensimmäisenä.
    - Listaa käytetään kuin putkea, jossa uudet alkiot liikkuvat putken läpi ja vanhat poistetaan ensimmäisinä.
    - FIFO-toimintaperiaatetta noudattavia tietorakenteita ovat esimerkiksi jono (queue) ja jonopuskuri.
2. **LIFO (Last-In, First-Out):**
    - LIFO tarkoittaa, että viimeisenä listalle lisätty alkio on ensimmäisenä poistettava. Tämä periaate muistuttaa pinon toimintaa, kuten kirjojen tai lautapelin palikoiden asettamista päällekkäin, jolloin ylimmäisenä oleva palikka poistetaan ensin.
    - Listaa käytetään kuin pinoa, jossa uudet alkiot lisätään päälle ja ylimmäinen alkio poistetaan ensin.
    - LIFO-toimintaperiaatetta noudattavia tietorakenteita ovat esimerkiksi pino (stack) ja rekursiivinen pino (call stack).

Esimerkkejä tilanteista, joissa käytetään FIFO- ja LIFO-periaatteita:

- FIFO: Voit käyttää FIFO-jonoa tilanteissa, joissa tulee käsittelyyn tehtäviä tai pyyntöjä, ja ne on hoidettava järjestyksessä saapumisjärjestyksessä. Esimerkiksi tulostinjonossa tulostustehtävät käsitellään FIFO-periaatteen mukaisesti.
- LIFO: LIFO-pinon avulla voit hallita ohjelman suoritusta, kun useita funktiokutsuja tehdään. Viimeisintä funktiota kutsutaan ensin, ja sen suoritus päättyy ennen kuin aiemmin kutsuttuja funktioita suoritetaan. Tämä auttaa palauttamaan suoritusjärjestyksen oikein ja mahdollistaa rekursiivisten funktioiden suorituksen.

2. **Selitä koodi (2p**)

Beginnersbook.com sivuilla on koodi HashMap Example to demonstrate various methods https://beginnersbook.com/2013/12/hashmap-in-java-with-example/
Selitä omin sanoin, miten koodi toimii. Sisällä selitykseesi myös selitys siitä, miten HashMap toimii.

3. **Koodaa ja pohdi (2 p)**

Sinulla on lista [2, 76, 4, 8, 55, 4, 34, 7, 2, 22, 55, 4, 98]

a. Sinun tulee selvittää, sisältääkö lista luvun 4. Kirjoita koodi ja selitä omin sanoin ohjelman logiikka. Miten käyt listan elementtejä läpi?

Lista.java

```
1  package kotitehtävät6;
2
3  import java.util.Arrays;
4
5  public class Lista {
6
7      public static void main(String[] args) {
8        // Luo lista
9        int[] lista = {2, 76, 4, 8, 55, 4, 34, 7, 2, 22, 55, 4, 98};
10
11       // Selvitä, sisältääkö lista luvun 4
12       int indeksi = Arrays.binarySearch(lista, 4);
13
14       // Tulosta tulos
15       if (indeksi >= 0) {
16         System.out.println("Lista sisältää luvun 4.");
17       } else {
18         System.out.println("Lista ei sisällä luvun 4.");
19       }
20     }
21 }
22
23
```

Console

<terminated> Lista [Java Application] C:\Users\jouni\.p2\pool\plugins\org.ecl

```
Lista sisältää luvun 4.
```

b. Sinun tulee selvittää, sisältääkö lista kaksi instanssia luvusta 4. Kirjoita koodi ja selitä omin sanoin ohjelman logiikka. Miten käyt listan elementtejä läpi?

c. Sinun tulee selvittää kuinka monta lukua 4 on listassa. Kirjoita koodi ja selitä omin sanoin ohjelman logiikka. Miten käyt listan elementtejä läpi?

d. Järjestä lista, selvitä sitten onko listassa lukua 9. Kirjoita koodi ja selitä omin sanoin ohjelman logiikka. Miten käyt listan elementtejä läpi?

e. Selitä omin sanoin, mitä eroja tekemilläsi algoritmeilla on, minkä uskot olevan nopein tapa selvittää, onko jokin luku listassa?

4. ArrayList listan käsittelyä (1p) a. **Tulosta viimeksi luettu arvo**

Tee ohjelma, joka kysyy käyttäjältä syötteitä ja lisää syötteet listalle. Syötteen lukeminen lopetetaan, kun käyttäjä syöttää tyhjän merkkijonon. Kun syötteiden lukeminen lopetetaan, ohjelma tulostaa viimeksi luetun arvon. Käytä tässä tehtävässä hyödyksi listan koon kertovaa metodia.

b. **Tulosta listan suurin luku**

Tee ohjelma, joka kysyy käyttäjältä positiivisia lukuja listalle. Syötteiden lisääminen lopetetaan, kun käyttäjä syöttää negatiivisen luvun. Lukujen lukemisen jälkeen ohjelma etsii listalta suurimman luvun ja tulostaa sen arvon sekä tuon suurimman luvun neliön.
Ohjelman pitäisi toimia seuraavasti:
Syötä lukuja:
72
2
8
93
11
-2

Listan suurin luku oli 93 ja sen neliö on 8649.

c. **Löytyykö listalta**

Tee ohjelma, joka lukee käyttäjältä syötteitä, kunnes tämä syöttää tyhjän merkkijonon. Kun merkkijono on valmis, ohjelma kysyy, mitä etsitään. Tämän jälkeen ohjelma kertoo, löytyykö etsittävä merkkijono listalta.

5. **Vertaile kahta listaa (1p)**

Tee ohjelma, joka sisältää kaksi merkkijonolistaa, jossa on vähintään viisi merkkijonoa. Ohjelma vertailee näitä listoja keskenään ja tulostaa “Erit”, jos listoissa on eri elementtejä ja “Samat”, jos niissä on samat elementit. Listan elementtien järjestyksellä ei ole väliä. Älä käytä sort metodia. Voit lopussa vielä vertailla ovatko listat täsmälleen samat.

6. **Oracle Java Documentointi (2p)**

Käytä Oraclen Java dokumentointia hyväksesi ja selitä, miten HashMap toimii. Käytä myös muuta materiaalia hyväksesi ja selitä, mitä hashillä tarkoitetaan ja miten sillä talletetaan tietoa tietokoneen muistiin. Miksi käyttäisit hashmapia, etkä esim. ArrayListia? Tee myös koodiesimerkit listan peruskäytöstä (lisääminen, poistaminen, etsiminen). Miten järjestät listan arvot, jos käyttössäsi on Hashmap tyyppinen lista?
https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

7. LinkedList Iteraattorin Tila (1p)

a. Miten tarkistat, onko Iteraattorilla vielä elementtejä luettavana?
b. Mitä etua on Iteraattorin käytöstä?
c. Mitä haittaa voi olla Iteraattorin käytöstä?

8. **Koodaustehtävä HashSet (1p)**

Kirjoita ohjelma, jossa luot HashSet tyyppisen listan, jossa elementteinä on merkkijonoja (esim. mausteita, autojen rekisterinumeroita, teelaatuja, värejä, yms.). Koodissa pitää lisätä elementtejä listaan, poistaa elementtejä listasta, tarkistaa onko lista tyhjä vai ei, käydä läpi listan jokainen elementti ja tulostaa se, sekä etsiä jokin tietty elementti.

9. **Selitä**, miten HashSet ja HashMap eroavat toisistaan, missä tilanteessa käyttäisit HashSetia ja missä taas HashMapia? **(1p)**