Harjutustund 8

Ülesannete kontroll. Kontrolltöö vigade analüüs. Andmekogumid : Collections: ArrayList.

Andmekogumid: Collections: ArrayList

Dünaamilised andmekogumid – **Collections -** pikkust saab programmi töö käigus vabalt muuta, elemente saab erinevatesse kohtadesse lisada ja neid eemaldada.

Liigid: nimistu - *list*, magasin - *stack*, järjekord - *queue*, kuhi - *heap*, "andmebaas" - *map*)

Listi (kui andmestruktuuri) jaoks Javas valmis klassi pole, on hoopis liides (*interface*).

Liste on võimalik realiseerida erineval moel. Siin kasutame klassi `ArrayList`. Nii seda kui teisi dünaamiliste andmestruktuure käsitleme põhjalikumalt semestri teisel poolel. Praegu lihtsalt kasutame paindlikumaid võimalusi kui pakuks massiiv.

Näide:

```
import java.util.ArrayList;
...
ArrayList list1 = new ArrayList();
```

Põhilised toimingud, mida saab listiga teha:

- elemendi võtmine indeksi järgi, nt. `list.get(2)` (NB! indeksid algavad 0-st)
- elemendi lisamine listi lõppu (`list.add(780)`) või määratud positsioonile (`list.add(3, 780)`)
- mitme elemendi lisamine (`list.addAll(mingiTeineList)`)
- elemendi eemaldamine indeksi järgi (`list.remove(3)`)
- sisalduvuse kontroll (`list.contains(780)`)
- sisalduvuse kontroll koos indeksi tagastamisega (`list.indexOf(780)`)
- pikkuse küsimine (`list.size()`)

Listi koostamine

```
import java.util.ArrayList;
...
ArrayList list1 = new ArrayList();
list1.add(1);
list1.add(2);
list1.add(3);
```

Alternatiivne variant sama tulemuse jaoks:

```
import java.util.ArrayList;
import java.util.Arrays;
...

ArrayList list2 = new ArrayList();
list2.addAll(Arrays.asList(1,2,3));
```

Listi väljastamine ekraanile

Erinevalt massiivist ei pea (aga võib) listi väljastamiseks kasutada tsüklit:

```
import java.util.ArrayList;

import java.util.Arrays;

...

ArrayList list1 = new ArrayList();
list1.add(1);
list1.add(2);
list1.add(3);
System.out.println(list1);

ArrayList list2 = new ArrayList();
list2.addAll(Arrays.asList(1,2,3));
for (int i = 0; i < list2.size(); i++)
 System.out.print(list2.get(i)+" ");
System.out.println();
```

Listi ümberpööramine

```
public static ArrayList reverse(ArrayList list) {
 ArrayList tulemus = new ArrayList();

 for (int i = list.size()-1; i >= 0; i--) {
 tulemus.add(list.get(i));
 }

 return tulemus;
}
```

Meetodi kasutamine:

```
ArrayList intList = new ArrayList();
intList.addAll(Arrays.asList(1,2,3));
System.out.println(reverse(intList));
```

Elemendi lisamine listi keskele

```
ArrayList list3 = new ArrayList();
list3.addAll(Arrays.asList(1,2,3));
list3.add(2, 36); // '36' saab 3. elemendiks (indeksiga 2)

System.out.println(list3);
```

Listi elemendi võrdlemine (koos tüübiteisendusega)

Kui listis on täisarvud ja tahame võrrelda listi elementi mingi arvuga (nt. 0), siis on vajalik tüübiteisendus:

```
ArrayList list4 = new ArrayList();
list4.addAll(Arrays.asList(1,2,3));
if ((int)list4.get(2) > 0){

    System.out.println(list4.get(2));

}
```

Ülesanded

Koostada klass *Student*, klassi väljad on:

```
String name; String studentId; String personalId;
```

Lisadada konstruktor, getterid ja setterid, toString();

Koostada klass *StudentGroup*, klassi väljad on :

```
String groupName; ArrayList <Student> students;
```

Lisadada konstruktor, getterid ja setterid, toString()

StudentGroup klassi meetodid:

1. `addStudent(student)` – uue tudengi lisamine. Veenduda, et iga lisatud tudeng on unikaalne.
2. `printStudentsList()` – tudengi listi kuvamine.
3. `removeStudent(studentID)` – tudengi kustutamine matriklinumbri järgi.
4. `Student findStudent(studentID)` – tudengi (objekti) leidmine matriklinumbri järgi.