# Seminar 11

## 1 Partea teoretica

### 1.1 Array

Un *array* reprezinta o colectie ordonata de elemente de acelasi tip, cu dimensiune fixa. Indexarea elementelor incepe de la 0. Odata creat, un array nu isi poate modifica dimensiunea.

Exemplu de declarare si initializare:

```
1        int[] numere = new int[5]; // creaza un array de 5 elemente
2        numere[0] = 10;
3        numere[1] = 20;
4        // ...
5        // Sau direct:
6        int[] valori = {10, 20, 30, 40, 50};
```

Caracteristici:

- Acces la element prin index (acces in O(1))
- Dimensiune fixa
- Poate stoca tipuri primitive sau obiecte

### 1.2 ArrayList

*ArrayList* este o clasa din Java Collections Framework care ofera o structura de date dinamica, redimensionabila. Se bazeaza intern pe un array, dar dimensiunea lui se poate schimba pe masura ce adaugam sau stergem elemente.

Exemplu:

```
1        import java.util.ArrayList;
2
3        ArrayList<String> listaCuvinte = new ArrayList<>();
4        listaCuvinte.add("abc");
5        listaCuvinte.add("def");
6        listaCuvinte.remove("abc");
7        System.out.println(listaCuvinte.get(0)); // afiseaza "def"
```

Caracteristici:

- Creste sau scade dinamic in functie de necesitati
- Metode utile: add(), remove(), get(), size()
- Acces la element in O(1), inserare si stergere in O(n) in general

### 1.3 HashMap

*HashMap* este o implementare a interfetei *Map* ce stocheaza perechi cheie-valoare. Accesul la elemente se face pe baza cheii, iar in general operatiile de inserare, cautare si stergere se realizeaza in timp O(1) amortizat. Cheile sunt unice.

Exemplu:

```
import java.util.HashMap;

HashMap<String, Integer> scoruri = new HashMap<>();
scoruri.put("Ana", 10);
scoruri.put("Ion", 15);
System.out.println(scoruri.get("Ana")); // afiseaza 10
```

Caracteristici:

- Stocare sub forma de perechi cheie-valoare
- Cheile sunt unice, valorile nu neaparat
- Acces rapid la date pe baza cheii

## 2 Partea de aplicatii

### 2.1 Problema rezolvata 1 (cu array)

**Problema:** Se cere realizarea unui program care citeste un array de numere intregi, apoi afiseaza suma tuturor elementelor si elementul de valoare minima.

Rezolvare:

1. Citim un array de intregi, de exemplu direct din cod.
2. Initializam variabilele suma cu 0 si min cu Integer.MAX_VALUE.
3. Iteram prin array, adunam fiecare element la suma, iar daca elementul curent este mai mic decat min, actualizam min.
4. Afisam suma si min.

```
public class ProblemaRezolvataArray {
    public static void main(String[] args) {
        int[] valori = {5, 2, 9, -1, 10};

        int suma = 0;
        int min = Integer.MAX_VALUE;

        for (int val : valori) {
            suma += val;
            if (val < min) {
                min = val;
            }
        }

        System.out.println("Suma elementelor: " + suma);
        System.out.println("Elementul minim: " + min);
    }
}
```

La rulare, pentru array-ul dat: Suma elementelor: 25 Elementul minim: -1

### 2.1.1 Probleme propuse (bazate pe prima problema cu array)

1. Cititi un array de numere intregi si afisati valoarea maxima si a doua cea mai mare valoare.

2. Cititi un array de numere intregi si determinati frecventa fiecarui numar folosind un HashMap, apoi afisati numarul si frecventa sa.

3. Cititi o lista de cuvinte intr-un ArrayList si eliminati duplicatele, pastrand doar prima aparitie a fiecarui cuvant.

4. Cititi un array de numere intregi si rearanjati elementele astfel incat toate valorile pare sa fie plasate la inceput, urmate de cele impare.

## 2.2 Problema rezolvata 2 (cu HashMap)

**Problema:** Se cere realizarea unui program care citeste un sir de cuvinte si construieste un HashMap in care cheia este cuvantul, iar valoarea este numarul de aparitii ale acelui cuvant. La final, se afiseaza fiecare cuvant impreuna cu numarul de aparitii.

Rezolvare:

1. Definim un array de cuvinte ca exemplu.

2. Cream un HashMap¡String,Integer¿ numarAparitii.

3. Iteram prin array si pentru fiecare cuvant incrementam contorul in HashMap.

4. Afisam rezultatele.

```
        import java.util.HashMap;

        public class ProblemaRezolvataHashMap {
            public static void main(String[] args) {
                String[] cuvinte = {"ana", "are", "mere", "ana", "ion", "are"};

                HashMap<String, Integer> numarAparitii = new HashMap<>();

                for (String c : cuvinte) {
                    if (numarAparitii.containsKey(c)) {
                        numarAparitii.put(c, numarAparitii.get(c) + 1);
                    } else {
                        numarAparitii.put(c, 1);
                    }
                }

                for (String key : numarAparitii.keySet()) {
                    System.out.println(key + ": " + numarAparitii.get(key));
                }
            }
        }
```

Pentru sirul dat, output-ul va fi: ana: 2 are: 2 mere: 1 ion: 1

### 2.2.1 Probleme propuse (bazate pe problema cu HashMap)

1. Cititi un sir de cuvinte si folosind un HashMap aflati cuvantul cu cele mai multe aparitii.
2. Cititi un sir de numere intregi si folosind un HashMap creati o mapa care stocheaza pentru fiecare numar cate valori mai mici decat el au aparut in sir.
3. Folositi un HashMap pentru a transforma un array de cuvinte intr-un array de frecvente, in care fiecare element este frecventa cuvantului in array.
4. Avand un HashMap care asociaza nume de studenti cu media lor, determinati care sunt studentii cu media cea mai mare.