Laborator 2 - Programare Avansata pe Obiect

Tablouri

- Un tablou este o structură de date care contine mai multe valori de acelaşi tip, numite elemente.
- Fiecare valoare poate fi accesată prin specificarea unui indice.
- Lungimea unui tablou este stabilita la momentul creării tabloului.

Nota:

- Lungimea tabloului poate fi accesată prin intermediul variabilei membru **length**, care este de tip primitiv **int**.
- Nu se specifică dimensiunea cand declari tabloul. Spre deosebire de alte limbaje, nu se pune niciodată dimensiunea tabloului în declaraţie, deoarece o declaraţie de tablou specifică doar tipul elementelor şi numele variabilei.

Declarare

- declarea unui tablou in java se face cu paranteze pătrate închise(**[]**)
- exista doua moduri de declarare a variabilelor de tip tablou care o sa conţină elemente de tip primitiv
- dimensiunea unui tablou nu poate fi negativa

```
//Exista doua moduri de a declara un tablou primitiv

// tipPrimitiv[] numeTablou
int[] a;
// tipPrimitiv numeTablou[]
int b[];


// Initializarea se poate face separat de declarare
a = new int[100];
// sau se pot face declararea si initializarea pe aceasi linie
int []c = new int[100];
```

O cale alternativa de a initializa un tablou, este de a specifica o lista cu elementele tabloului în momentul declarării.

```
int[] e = { 1, 2, 3, 4, 5 };
char[] f = { 'a', 'b', 'c' };
float[] g = { 5f, 6f, 7f };
```

Lungimea unui tablou se determina folosind **.length**:

```
int[] d;
d = new int[5];
// lungimea o sa fie 5
System.out.println("Lungimea este: " + d.length);
d = new int[0];
// lungimea o sa fie 0
System.out.println("Lungimea este: " + d.length);
d = new int[-5];
// lungimea o sa fie ??
System.out.println("Lungimea este: " + d.length);
```

Accesarea elementelor din tablou

- elementele dintr-un tablou sunt numerotate de la 0 la n-1, unde n este numărul de elemente dintr-un tablou
- accesarea unui tablou în afara limitelor sale va genera o excepție în timpul rulării programului, ArrayIndexOutOfBoundsException

```
public static void main(String[] args) {
    int[] unTablou = new int[10];
    for (int i = 0; i < unTablou.length; i++) {
        unTablou[i] = i;
        System.out.print(unTablou[i] + " ");
    }
    System.out.println();
}
```

Tablouri cu elemente de tip referinta

- tablouri de obiecte
- tablouri de tablouri

Nota: Elementele tabloului sunt referințe la structurile propriu-zise, ale obiectelor sau ale tablourilor.

```
// tablou de elemente referinta la obiecte String
String[] tablouDeSiruri;
// tablou de tablouri de intregi
int[][] tablouDeTablouriDeIntregi;

tablouDeSiruri = new String[] { "fsafsa", "fsa", "assa" };
tablouDeTablouriDeIntregi = new int[][] { { 1, 2, 3 }, { 4, 5, 6 }, { 1, 2 } };
```

Clasa Arrays

Clasa java.util.Arrays oferă diverse metode utile în lucrul cu vectori.

Metode:

- **sort** - sortează ascendent un vector

```
public static void main(String[] args) {
    int[] a = { 1, 24, 6, 2, 123, 65 };
    Arrays.sort(a);
    for (int i = 0; i < a.length; i++) {
        System.out.print(a[i] + " ");
    }
    // Output: 1 2 6 24 65 123
}
```

- **equals** - testarea egalității valorilor a doi vectori (au aceleaşi număr de elemente şi pentru fiecare indice valorile corespunzătoare din cei doi vectori sunt egale)

```java
int[] b = { 1, 2, 3, 4 };
int[] c = { 1, 2, 3, 4 };
System.out.println(Arrays.equals(b, c));

int[] d = { 1, 2, 3, 4 };
int[] e = { 4, 3, 2, 1 };

System.out.println(Arrays.equals(d, e));
```

- **fill** - atribuie fiecărui element din vector o valoare specificată

```java
int[] f = { 1, 2, 3, 4 };
Arrays.fill(f,  val: 4);
for (int i = 0; i < f.length; i++) {
    System.out.print(f[i] + " ");
}
```

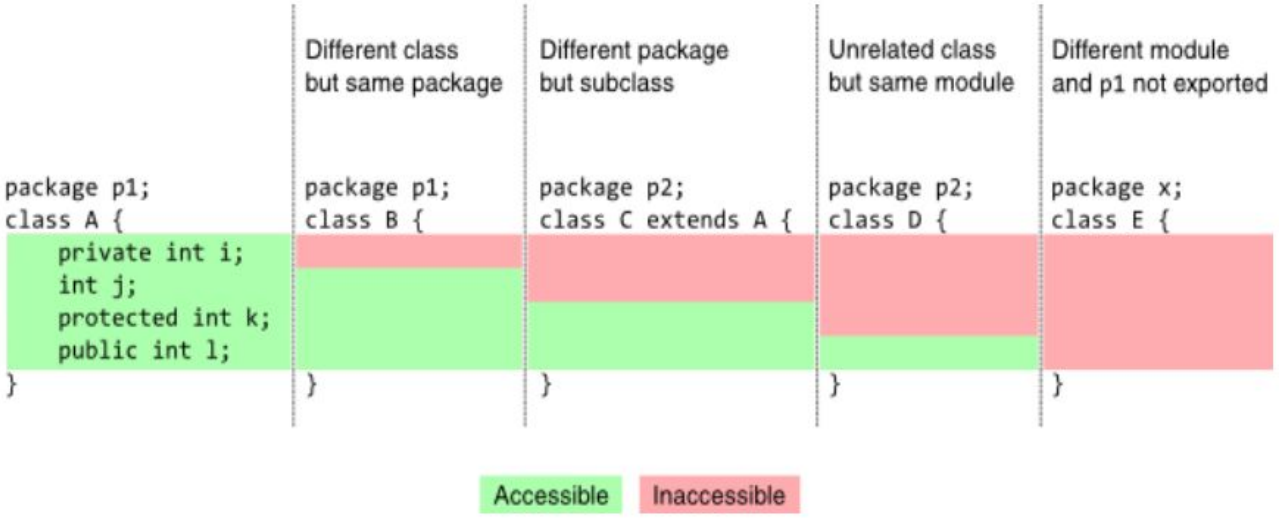
- **binarySearch** - căutarea binară a unei anumite valori într-un vector sortat;

```java
int[] f = { 1, 3, 4, 5 };
Arrays.binarySearch(f,  key: 4);
System.out.println(Arrays.binarySearch(f,  key: 4));
```

Modificatori de acces

- Modificatorii de acces sunt cuvinte rezervate ce controlează accesul celorlalte clase la membrii unei clase.
- Specificatorii de access pentru variabilele și metodele unei clase sunt:
    - **private**
    - **protected**
    - **public**
    - **default**

Specificator	Clasa	Subclasă	Pachet	Oriunde
private	X			
protected	X	X*	X	
public	X	X	X	X
implicit	X		X	



Încapsulare

- În general, este recomandat ca field-urile unei clase sa nu poata sa fie accesate direct de alte clase.
- Incapsularea ajuta la ascunderea detaliilor legate de implementare și la controlul accesului la campurile clasei.

```java
private int varsta;

public int getVarsta() {
    return varsta;
}

public void setVarsta(int varsta) {
    this.varsta = varsta;
}
```

Exerciţii:

1. Scrieţi o aplicaţie in Java care sa va permita sa cititi de la tastatura **n** note, numere întregi într-un vector. Cand utilizatorul citeşte de la tastatură valoarea -1, citirea notelor se opreşte şi programul afişează media notelor.

2. Scrieţi o aplicaţie in Java care sa citeasca n numere de la tastatura. elementele citite de la tastatura vor fi organizate în doi vectori diferiţi, în funcţie de paritate (Ex. numerele impare vor fi puse într-un vector x, numerele pare intr-un vector y). Afişaţi care vector are mai multe elemente.

3. Scrieţi o aplicaţie in Java care sa declare un obiect (entitate din viata reala, alegerea voastra care), cu doua atribute (de tipuri diferite) private, implementaţi getteri/setteri, un constructor, şi o metoda care sa definească un comportament al entitatii (sa afisati/ sau sa incrementati un atribut).

4. Se citesc de la tastatura n linii de forma:

   Nume1 nota1

   Nume2 nota2

   Nume3 nota3

   Declaraţi o clasa Student cu atributele/constructorii/metodele necesare ca sa mapati liniile citite de la tastatura. Stocaţi toţi studenţii într-un tablou. După afişaţi-le în fluxul de ieşire.