Lab2: Lucrul cu tablouri și cu stringuri

Cuprins

1. Recapitulare laborator 1

În primul laborator, am făcut o primă introducere în limbajul Java şi am văzut cum se compilează un program simplu, cum se rulează şi cum se pot citi argumentele pasate din linia de comandă. De asemenea, am descoperit tipurile primitive, alături de clasele "înfăşurătoare" şi am consultat unele clase (`Integer, String, Scanner`) pentru a observa metodele puse la dispoziţie şi a le folosi în rezolvarea unor probleme simple.

Încercaţi să răspundeţi la următoarele întrebări:

- Care sunt principalele caracteristici ale limbajului Java ?
- Ce fel de limbaj este Java ?
- Care este rolul funcţiei `main` şi care este signatura acesteia ?
- Cum se realizează compilarea unui cod sursă? Dar rularea?
- Ce înseamnă clasă "înfăşurătoare" şi la ce sunt folosite? Daţi exemple.
- Ce categorii de variabile există în Java?

2. Tablouri în Java

Un tablou este o secvență de **componente de acelaşi tip**. Acest tip poate fi un tip primitiv sau un tip referință (putem lucra cu tablouri de obiecte). Un tablou unidimensional `a` poate fi declarat folosind una dintre următoarele modalități:

```
tip[] a;
tip a[];
```

unde `tip` este tipul componentelor tabloului (!nu se specifică un număr de elemente).

Declararea unui tablou **nu** are drept consecință crearea sa (alocarea memoriei necesare).

Crearea (instanțierea) tabloului a declarat mai sus trebuie făcută explicit, prin:

```
a = new tip[n];
```

unde `n` poate fi o constantă sau o variabilă întreagă ce a primit o valoare strict pozitivă.

Un tablou este un **tip referință.** Prin creare se obține un obiect de tip tablou (obiect numit prin abuz de limbaj tot tablou).

Componentele tabloului pot fi referite prin `a[i]`, cu i luând valori în intervalul `0..n-1;` dacă i nu este în acest interval, va fi semnalată o eroare la executare (excepția `IndexOutOfBoundsException`). Lungimea tabloului poate fi referită prin `a.length`.

Evident, declararea şi crearea pot fi făcute şi simultan prin:

```
tip[] a = new tip[n];
```

sau printr-o inițializare efectivă, ca de exemplu:

`int[] a = {0,1,2,3,4};` prin care, evident, `a.length` devine 5.

Exemplu:

```
public class ArrayTest {

   public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int[] a = new int[n];
        for(int i = 0; i < n; ++i) {
            a[i] = sc.nextInt();
        }

        // Parcurgerea tabloului - Var 1.
        for(int i = 0; i < n; ++i) {
            // Code goes here!...
        }

        // Parcurgerea tabloului - Var 2.
        for(int i : a) {
            // Code goes here!...
        }

   }
}
```

Câmpul length al unui (obiect de tip) tablou este un **câmp constant** (cu modificatorii public şi final) de tip int; deci, odată creat, un obiect tablou nu îşi poate schimba dimensiunea (numărul de componente).

Copierea elementelor unui vector a într-un alt vector b se poate face fie element cu element, fie cu ajutorul metodei `System.arraycopy`, ca în exemplele următoare (`b` trebuie să fie deja alocat). Metoda primeşte ca parametri vectorul sursă, indicele de la care începe copierea, vectorul destinație, indicele din vectorul destinație începând de la care sunt puse elementele copiate şi numărul de elemente copiate.

O atribuire de genul `b = a` are altă semnificație decât copierea elementelor lui a în b şi nu poate fi folosită în acest scop. Este o **atribuire de referințe**, în urma acestei atribuiri variabilele b şi a vor referi acelaşi obiect (vector). Dacă modificăm un element al lui a se modifică şi b şi invers.

Pentru toate exemplele şi exerciţiile legate de lucrul cu vectori, creaţi un pachet numit `tablouri`. În acest pachet, veţi crea clasele corespunzătoare pentru rezolvarea fiecărei probleme.

Amintim că un pachet este o grupare logică de clase, cu rolul de a facilita organizarea fişierelor sursă, a evita conflictele de nume şi a controla accesul la clase.

Exemplu: Copierea unui vector în alt vector

```
 int a[] = {1, 2, 3, 4};
 int b[] = new int[4]; // Atentie, b trebuie alocat

// Varianta 1
for(int i = 0; i < a.length; i++) {
    b[i] = a[i];
}

// Varianta 2
System.arraycopy(a, 0, b, 0, a.length);

// Varianta 3 - Nu are efectul dorit
b = a;

System.out.println(a[0] + " " + b[0]);
b[0] = 5;
System.out.println(a[0] + " " + b[0]);
a[0] = 6;
System.out.println(a[0]+" "+b[0]);
```

Încercaţi să folosiţi metoda care verifică primalitatea unui număr scrisă în primul laborator pentru a determina suma numerelor prime dintr-un tablou de lungime n.

Scrieţi o metodă care sortează crescător un şir de numere reale. Ca bonus, încercaţi să folosiţi o metodă de sortare diferită de bubble sort.

Pentru lucrul mai uşor cu tablouri, Java pune la dispoziţie clasa `Arrays` din pachetul `java.util`. Acesta trebuie importat pentru a putea fi folosit.

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

Consultaţi clasa şi observaţi metodele puse la dispoziţie.

Fie un array `a` de numere întregi de lungime `n` dat de la tastatură şi un array `b`, tot de lungime `n`, format numai din 5. Să se creeze un nou vector, `c`, format din elementele lui `a` printre

care s-au interpus elemente din `b`. Copiaţi într-un alt vector `d`, elementele din `c` între indicii `[c.length / 3, 2 * c.length / 3]`. Sortaţi vectorul `d`.

Ex: `a = {1, 2, 3, 4}`
`b = {5, 5, 5, 5}`
`c = {1, 5, 2, 5, 3, 5, 4, 5}`

Reamintim că dimensiunea unui tablou este una constantă, o dată creat un astfel de obiect acesta nu poate fi redimensionat. Totuşi, Java pune la dispoziţie implementări de vectori cu număr variabil de elemente, prin intermediul claselor `Vector` sau `ArrayList`, asupra cărora vom reveni.

3. Observaţii asupra citirii de la tastatură - Clasa Scanner

- O clasă care se poate folosi pentru citirea datelor de tipuri primitive sau String este clasa `Scanner` din pachetul `java.util`. Această clasă se poate folosi pentru citirea din diferite surse: de la tastatură, din fişier, din obiecte de tip String, în funcție de tipul obiectul trimis ca parametru constructorului clasei: `InputStream`, `File`, `String`.
- Deoarece clasa `Scanner` se află în pachetul `java.util`, acesta trebuie importat.

```
import java.util.Scanner;
```

- În mod predefinit un obiect de tip `Scanner` citeşte entități delimitate prin caractere albe şi apoi încearcă să le interpreteze în modul cerut. Pentru tipurile primitive de date există metodele `nextByte()`, `nextShort()`, `nextInt()`, `nextLong()`, `nextFloat()`, `nextDouble()`, `nextBoolean()`.
- Pentru a testa dacă sunt disponibile valori de anumit tip există metode ca `hasNextInt()`, `hasNextDouble()` etc. Există şi metodele `hasNext()` şi `next()` pentru a testa existența unei următoare entități (fără un tip specificat), respectiv pentru citirea următoarei entități (tipul rezultatului întors de metoda next() este String). Mai menționăm metodele `nextLine()` şi `hasNextLine()` (utile de exemplu dacă dorim să citim un şir de caractere care conține şi spații).
- Atunci când se testează existența unei entități de un anumit tip, dacă următoarea entitate nu are tipul dorit ea rămâne în bufferul de intrare (poate fi citită cu metoda `next()`).

 Să presupunem că dorim să citim un întreg. În caz că valoarea introdusă de utilizator nu reprezintă un întreg, îi permitem utilizatorului să reintroducă valoarea. Utilizatorul va putea reintroduce valoarea de maximum 3 ori.

3. Lucrul cu stringuri

 Pentru exerciţiile legate de stringuri, creaţi un nou pachet, numit `stringuri`, unde să adaugaţi sursele pentru rezolvarea problemelor.

În Java, şirurile de caractere sunt obiecte ale clasei `String`, din pachetul `java.lang`. Un literal de tip `String` este o secvenţă de caractere între ghilimele:

```
String s="Un sir de caractere";
```

Consultaţi clasa String şi observaţi metodele puse la dispoziţie. Dintre acestea, amintim:

`length`	Lungimea şirului de caractere
`charAt`	Accesarea caracterului de pe poziţia position
`substring`	Accesarea unui substring.
`indexOf`	Căutarea unui caracter sau a unui şir de caractere într-un string.

 **Exemplu:** Să se determine substringul unui string aflat între prima şi cea de-a doua apariţie a unui caracter în stringul respectiv.

```
public static String solve(String s, char c) {
  int pos = s.indexOf(c); /* daca nu este gasit caracterul,
      metoda returneaza -1, altfel returneaza prima pozitie
      pe care este gasit*/

  if(pos >= 0){
      int pos2 = s.indexOf(c, pos + 1); /* cauta caracterul c de la
          pozitia poz+1 */
      if(pos2 >= 0)
          return s.substring(pos, pos2 + 1);
      else
          return s.substring(pos);
  }
  return s;
}
```

 Se citeşte de la tastarură un şir de stringuri de lungime n. Să se afişeze numai acele stringuri care conţin ca substring un alt cuvânt dat de la tastatură.

Egalitatea stringurilor

 Operatorul == testează dacă dacă două referinţe indică acelaşi obiect. În particular, dacă s1 şi s2 sunt două variabile de tip String, atunci s1==s2 este true doar dacă s1 şi s2 sunt referinţe către acelaşi şir (nu dacă şirurile sunt egale în sens lexicografic).

 **Exemplu:** Ce va afişa secvenţa următoare de cod?

```
String s1 = "ab";
char c1 = 'a';
char c2 = 'b';
String s2 = c1 + "" + c2;

System.out.println("s1 = " + s1);
System.out.println("s2 = " + s2);
System.out.println(s1 == s2);
```

Pentru a testa egalitatea se foloseşte metoda `equals` (sau `equalsIgnoreCase` pentru a nu diferenţia literele mari de mici):

```
System.out.println(s1.equals(s2));
```

 Scrieţi o aplicaţie Java care afişează în ordine lexicografică şirurile primite de la tastatură. Hint: Consultaţi metodele `compareTo` sau `compareToIgnoreCase` din clasa `String`.

4. Exerciţii

 Scrieţi un program Java care să interclaseze doi vectori ordonaţi crescător într-un alt vector.

 Scrieţi o aplicaţie Java care, pentru un cuvânt dat de la tastatură afişează numărul de vocale din cuvânt şi cea mai lungă subsecvenţă de consoane (consecutive) din cuvânt.

 Fie a un array de lungime n. Scrieţi o metodă care "shiftează" elementele vectorului nu p poziţii la dreapta, unde p este dat de la tastatură.

 *Scrieţi o aplicaţie Java care împarte o propoziţie primită de la tastatură în cuvinte (cuvintele se pot separa prin spaţiu, virgulă şi punct). Hint: Consultaţi metoda `split` a clasei `String`. O altă soluţie ar fi metoda `useDelimiter` din clasa `Scanner`.

```
// Varianta 1
String[] siruri = s.split("[ .,]"); /* Observaţi [] cu semnificaţia sau pentru
         lista de separatori.*/

// Varianta 2
Scanner sc=new Scanner(s);
sc.useDelimiter("[ .,]");
while(sc.hasNext()){
   String x = sc.next();
   if(x.length() != 0)
   System.out.println(x);
}
```

 Valoarea unui şir de caractere de tip `String` nu poate fi modificată după creare (de exemplu nu se poate modifica un caracter, sau un subşir al şirului). Dacă este necesară şi modificarea şirului de caractere se poate utiliza clasa `StringBuffer` din pachetul `java.lang`. Consultaţi documentaţia pentru a aprofunda metodele clasei StrinBuffer. Ce afişează următorul program ?

```
class ExempluString {

   public static void main(String arg[]) {
       String s = "abcdefgh";
       System.out.println("lungime " + s.length());
       System.out.println(s.charAt(2));

       StringBuffer sb = new StringBuffer(s);
       System.out.println(sb);
       sb.setCharAt(2, 'x');
       System.out.println(sb);
       sb.append('y');
       sb.append(1234);
       System.out.println(sb);
       sb.replace(1, 6, "yz");
       System.out.println(sb);
       System.out.println("lungime " + sb.length());
       System.out.println(sb.substring(1, 4));
       s = sb.toString();
       System.out.println(s);
   }
}
```