Programare orientată pe obiecte

Laboratorul 2

Mihai Nan
mihai.nan.cti@gmail.com

Facultatea de Automatică şi Calculatoare
Universitatea Politehnica din Bucureşti
Anul universitar 2015 - 2016

1 Operaţii cu şiruri de caractere

1.1 Introducere

În limbajele de programare, majoritatea tipurilor de date utilizate pentru variabile sunt ***valorile booleene***, ***şirurile de caractere*** (sau vectori de caractere terminaţi cu **'\0'**) şi ***valorile numerice***. În contrast cu limbajele C sau C++, în Java modul de gestiune a şirurilor de caractere este unul diferit, deoarece:

- fiecare char reprezintă o valoare Unicode pe 16 biţi şi nu 1 octet (ca în C / C++);
- valorile de tip şiruri de caractere sunt gestionate de obiecte ***String***;
- sintaxa vă permite să utilizaţi ***String*** asemenea unui tip primitiv de date (puteţi folosi operatorul = pentru a le iniţializa);
- ***String*** sunt obiecte ***imutabile*** (***immutable***), în sensul că odată ce sunt create, ele nu îşi pot schimba valoarea.

1.2 Clasa String

În Java, ***String*** este o clasă elementară care oferă funcţiile de bază pentru manipularea şirurilor de caractere. Ca utilizare, această clasă se foloseşte pentru compararea a două şiruri de caractere, extragerea de subşiruri, determinarea lungimii unui şir sau alte operaţii care vor fi descrise mai jos.

După cum vom vedea, orice şir este, de fapt, o instanţă a clasei ***String*** definită în pachetul ***java.lang***.

1.2.1 Instanţierea unui obiect String

Cod sursă Java

```
1 class Test {
2     String str1 = "text";
3     String str2 = new String("text");
4     char data[] = {'t', 'e', 'x', 't'};
5     String str3 = new String(data);
6 }
```

1.2.2 Operaţii uzuale cu obiecte de tip String

Cod sursă Java

```
class Test {
    public static void main(String args[]) {
        String str = "Laborator POO";
        String str1 = " 2015-2016";
        String str2 = "POO";
        String str3;
        char c = str.charAt(1); //caracterul de pe pozitia 1
        str3 = str.concat(str1); //concatenarea de siruri
        System.out.println(str3 + " != " + str);
        if(str.equals(str1)) {
            System.out.println("Sirurile sunt egale");
        }
        //pozitia de inceput a subsirului str2 in sirul str1
        int poz = str.indexOf(str2);
        int lungime = str.length();
        //operatorul + introduce posibilitatea concatenarii
        str3 = str + str1;
    }
}
```

Observaţie

În Java, operatorul de concatenare + este extrem de flexibil, în sensul că permite concatenarea şirurilor cu obiecte de orice tip care au o reprezentare de tip şir de caractere.
Mai jos, sunt prezentate câteva exemple pentru a lămuri ceea ce execută compilatorul atunci când întâlneşte o secvenţă care conţine operatorul de concatenare.

Cod sursă Java

```
class Test {
    public static void main(String args[]) {
        String x = "a" + 1 + "b";
        x = 1 + 2 + "c" + 1 + 2;
        x = "" + 1 + 2 + "c";
        x = 1.5 + " " + 2.3;
    }
}
```

⚠ IMPORTANT !

⚠ Atenţie însă la ordinea de efectuare a operaţiilor. şirul ***s*** va avea valoarea ***"3a12"***, primul + fiind operatorul matematic de adunare, iar al doilea + este cel de concatenare a şirurilor.

```
1 s = 1 + 2 + "a" + 1 + 2;
```

1.3 Clasa StringBuffer

În Java, un şir de caractere poate fi reprezentat printr-un vector format din elemente de tip ***char***, un obiect de tip ***String*** sau un obiect de tip ***StringBuffer***. Dacă un şir de caractere este constant (nu se doreşte schimbarea conţinutului său pe parcursul execuţiei programului) atunci el va fi declarat de tipul ***String***, altfel, va fi declarat de tip ***StringBuffer***.

⚠ IMPORTANT !

⚠ Diferenţa principală între aceste clase este că ***StringBuffer*** pune la dispoziţie metode pentru modificarea conţinutului şirului, cum ar fi: ***append***, ***insert***, ***delete***, ***reverse***.

Observaţie

Uzual, cea mai folosită modalitate de a lucra cu şiruri este prin intermediul clasei ***String***, care are şi unele particularităţi faţă de restul claselor menite să simplifice cât mai mult folosirea şirurilor de caractere. Clasa ***StringBuffer*** va fi utilizată predominant în aplicaţii dedicate procesării textelor cum ar fi editoarele de texte.

1.3.1 Instanţierea unui obiect StringBuffer

Cod sursă Java

```
1 StringBuffer str = new StringBuffer("abc");
2 StringBuffer str1 = new StringBuffer();
```

1.3.2 Operaţii uzuale cu obiecte de tip StringBuffer

Cod sursă Java

```
class Test {
    public static void main(String args[]) {
        StringBuffer str = new StringBuffer("abc");
        StringBuffer str1 = new StringBuffer();
        //se adauga str la sbuf
        str1.append(str);
        System.out.println("str1 = " + str1.toString());
        //se adauga reprezentarea ca sir de caractere a nr 2
        str1.append(1);
        str1.append("xyz");
        //se insereaza sirul "bc" in str1
        str1.append("abc", 1, 3);
        System.out.println(str1);
        //se inlocuiesc caracterele de la pozitiile 1-3
        //cu sirul s => aPOO1xyzbc
        String s = "POO";
        str1.replace(1, 3, s);
        //se sterg caracterele de la pozitiile 1-3
        str1.delete(1, 3);
    }
}
```

1.4 Clasa Vector

Un vector este o structura de date dinamica in Java. Se pot adauga elemente in orice pozitie, se pot sterge elemente, se pot inlocui elemente, se poate determina pozitia pe care se afla un element, se poate determina numarul de elemente dintr-un vector. In Java, un obiect de tip ***Vector*** lucreaza cu elemente de tip ***Object***, aceasta insemnand ca permite stocarea oricaror tipuri de elemente.

Pentru o intelegere mai buna a importantei utilizarii acestei clase, este recomandata parcurgerea exemplului de mai jos.

Cod sursă Java

```java
public class Exemplu {
    public static void main(String args[]) {
        //Instantierea obiectului
        Vector vect = new Vector();
        //Adaugarea in vector a unor numere intregi
        vect.add(20);
        vect.add(5);
        vect.add(7);
        //Determinarea pozitie lui 5
        System.out.println(vect.indexOf(5));
        //Determinarea primului element
        System.out.println(vect.firstElement());
        //Modificarea elementului de pe pozitia 2
        vect.set(2, 10);
        //Afisarea vectorului
        System.out.println(vect);
        for(int i = 0; i < vect.size(); i++) {
            System.out.println(vect.get(i));
        }
        //Determinarea primului element
        int first = (int) vect.get(0);
        //Afisarea vectorului - recomandat
        System.out.println(vect);
        //Afisarea vectorului - nerecomandat
        for(int i = 0 ; i < vect.size(); i++) {
            System.out.println(vect.get(i));
        }
    }
}
```

2 Probleme de laborator

Observaţie

Datele de intrare se vor da fie ca parametrii în linia de comandă, fie ca valori fixe ale aplicaţiilor!

2.1 Probleme standard

Problema 1 - 1 punct

Să se scrie un program pentru căutarea tuturor aparitiilor unui şir dat s1 într-un alt şir s. Se va afişa numărul de aparitii.

Exemplu:

String s1 = "si";

String s = "sir1 si cu sir2 fac un sir3";

Rezultat: 4.

Se poate folosi metoda ***indexOf()*** din clasa ***String*** cu două argumente: şirul căutat şi poziţia din şirul destinaţie de unde începe căutarea.

```
int pos = s2.indexOf(s1, p);
```

Problema 2 - 1 punct

Să se scrie un program pentru extragerea şi afişarea cuvintelor dintr-un text introdus ca un şir constant. Un cuvânt este un şir de caractere delimitat prin spaţii (blancuri) de alte cuvinte.

```
int indexOf(char c, int p);
String substring(int p1, int p2);
```

Problema 3 - 1 punct

Să se scrie un program pentru extragerea şi afişarea cuvintelor dintr-un text introdus ca un şir constant. Se va folosi un obiect de tip ***StringTokenizer***. Să se adauge un alt text care conţine şi alţi separatori de cuvinte: virgulă (','), punct('.'), punct şi virgulă (';'), două puncte (':'). Se va utiliza un alt constructor pentru obiectul analizator (***StringTokenizer***).

```
String text = "Operatii cu siruri de caractere!";
StringTokenizer st = new StringTokenizer(text);
String cuvant = st.nextToken();
```

Problema 4 - 2 puncte

Să se implementeze o clasă executabilă, avand ca nume ***Problema4***, care conţine, pe lângă metoda ***main***, o metodă care primeşte două argumente: un şir de caractere constant şi un vector de cuvinte (şiruri de caractere).

1. Dacă textul va conţine cel puţin o apariţie a unui cuvânt din vectorul primit ca parametru, se va afişa mesajul **"Text suspect"**, altfel, afisându-se mesajul **"Nimic suspect"**.

2. Metoda va returna un şir de caractere în care fiecare apariţie a unui cuvânt, din vectorul de cuvinte, este cenzurată. De exemplu, apariţiile cuvântului **terorist** se vor înlocui cu **t******t**.

```
String text = "Un terorist avea o bomba";
String cuvinte[] = new String[2];
cuvinte[0] = "terorist";
cuvinte[1] = "bomba";
Problema4 prb4 = new Problema4();
String rezultat;
rezultat = prb4.cenzurare(text, cuvinte);
```

Problema 5 - 2 puncte

Implementaţi o clasă executabilă care să conţină, în metoda ***main***, un obiect de tip ***Vector*** în care se adaugă 20 de numere reale, folosind metoda ***random*** din clasa ***Math***. Să se determine valoarea componentei maxime, poziţia componentei minime şi media aritmetică a elementelor din vector. Să se elimine toate elementele mai mici decât media aritmetică determinată.

```
Vector v = new Vector(20);
v.add(2.7);
double first = (double) v.get(0);
int index = v.indexOf(2.7);
v.remove(2.7);
```

Problema 6 - 3 puncte

Implementaţi o clasă executabila care să conţină, în metoda ***main***, două obiecte de tip ***Vector***, reprezentând două mulţimi cu numere intregi. Să se verifice dacă există elemente duplicate în aceşti vectori, dacă există să se elimine duplicatele şi să se realizeze apoi operaţiile elementare cu mulţimi: reuniunea, intersecţia şi diferenţa dintre prima mulţime şi a doua.

Pentru testare, se vor introduce în vectori minimum zece elemente. Pentru

fiecare operaţie elementară cu mulţimi, se va folosi un obiect de tip ***Vector*** pentru a se reţine rezultatul.

```
boolean ok = v.contains(2);
```

2.2 Problema bonus

Problema 7 - 2 puncte

Să se realizeze o clasă executabilă care să conţina, în metoda ***main***, un obiect de tip ***Vector*** în care să se introducă mai multe tipuri de obiecte (***int***, ***double***, ***float***, ***String***, ***char***, ***boolean***). Pentru fiecare tip, să se determine câte elemente de acest tip există în vector.

Cod sursă Java

```
1 Vector v = new Vector();
2 v.add(7.5);
3 v.add("String");
4 System.out.println(v.get(0).getClass());
5 System.out.println(v.get(1).getClass());
```

3 Interviu

Observaţie

Această secţiune este una opţională şi încearcă să vă familiarizeze cu o serie de întrebări ce pot fi adresate în cadrul unui interviu tehnic. De asemenea, această secţiune poate fi utilă şi în pregătirea pentru examenul final de la această disciplină.

Întrebări interviu

1. De ce este considerată clasa ***String*** imutabilă?
2. De ce este considerată clasa ***StringBuffer*** mutabilă?
3. Care este diferenţa esenţială între ***StringBuffer*** şi ***StringBuilder***?
4. Se pot compara două obiecte de tip ***String***? Argumentaţi!
5. De ce un vector de caractere este mai potrivit decât un obiect de tip ***String*** pentru reţinerea unei parole?
6. Ce se întâmplă atunci când folosim operatorul == pentru a compara obiecte de tip ***String***?
7. Există vreo modalitate prin care un obiect se poate converti în obiect de tip ***String***?

Feedback

Pentru îmbunatăţirea constantă a acestui laborator, vă rog să completaţi formularul de feedback disponibil aici.

De asemenea, vă rog să semnalaţi orice greşeală / neclaritate depistată în laborator pentru a o corecta.

Vă mulţumesc anticipat!