POO – Laborator 1

Dezvoltarea programelor. Scrierea functiilor membru (metodelor). Introducere in programarea Java

1.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- Instalarea *kitului* de dezvoltare Java (*jdk*) (include si configurarea variabilelor de mediu *PATH* si *CLASSPATH* si a tipurilor de fisiere *.java*) – **link extern**

- Etapele dezvoltarii unui program Java (conceperea, editarea – scrierea comentariilor si a codului, compilare, lansarea programului in interpretor, depanarea) – **link extern**

- **Studiu de caz**: programul ***Salut.java***

  - **Utilizarea *script*-urilor (fisiere *batch*) pentru simplificarea dezvoltarii programelor Java**

  - **Depanarea programelor Java** (creare de erori, urmarirea efectului, interpretarea mesajelor)

- **Modificarea programelor Java** (modificarea functionalitatii, adaugarea unor noi functionalitati)

  - Studiu de caz: programul *Salut.java* – **link extern**

  - **Program de calcul al unui polinom**

- **Teme de casa**

1.2. Dezvoltarea unui program Java. Studiu de caz: programul Salut.java

1.2.1. Conceperea si editarea programului *Salut.java*

O cale de a incepe conceperea programelor simple este de a crea un asa numit ***pseudocod***, adica o **descriere intr-un limbaj apropiat de cel natural** a ceea ce trebuie sa faca programul. De exemplu:

clasa *care ilustreaza elementele esentiale ale unui program Java, contine:*
metoda principala *(punctul de intrare in program):*
afiseaza pe ecran textul *"Buna ziua"*

Urmatorul pas poate fi transformarea acestor texte in comentarii.

```
1    /** Clasa care ilustreaza elementele esentiale ale unui program Java.
2     *  Trebuie sa fie publica pentru ca are metoda principala.
3     */
4
5      /** Metoda principala (punct de intrare in program).
6       *  Este prima metoda executata de JVM (Java Virtual Machine).
7       *  Primeste ca parametri argumentele din lina de comanda.
8       *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
9       */
10
11       // Corpul metodei afiseaza textul "Buna ziua" pe ecran
```

In continuare, **se pot utiliza pasi succesivi in care se adauga codul Java**, pornind de la nivelul cel mai inalt (urmand o **strategie *top-down*** – de la nivel inalt la nivel jos – prin detaliere):

- declaratia clasei:

```
1   /** Clasa care ilustreaza elementele esentiale ale unui program Java.
2    *  Trebuie sa fie publica pentru ca are metoda principala.
3    */
4   public class Salut {
5
6     /** Metoda principala (punct de intrare in program).
7      *  Este prima metoda executata de JVM (Java Virtual Machine).
8      *  Primeste ca parametri argumentele din lina de comanda.
9      *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
10     */
11
12      // Corpul metodei afiseaza textul "Buna ziua" pe ecran
13  }
```

- declaratiile atributelor si metodelor (in cazul nostru metoda main()):

```
1   /** Clasa care ilustreaza elementele esentiale ale unui program Java.
2    *  Trebuie sa fie publica pentru ca are metoda principala.
3    */
4   public class Salut {
5     /** Metoda principala (punct de intrare in program).
6      *  Este prima metoda executata de JVM (Java Virtual Machine).
7      *  Primeste ca parametri argumentele din lina de comanda.
8      *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
9      */
10    public static void main(String[] args) {
11      // Corpul metodei afiseaza textul "Buna ziua" pe ecran
12    }
13  }
```

- si corpurile metodelor (in cazul nostru metoda main()):

```
1   /** Clasa care ilustreaza elementele esentiale ale unui program Java.
2    *  Trebuie sa fie publica pentru ca are metoda principala.
3    *  @author  Eduard C. Popovici
4    */
5   public class Salut {
6     /** Metoda principala (punct de intrare in program).
7      *  Este prima metoda executata de JVM (Java Virtual Machine).
8      *  Primeste ca parametri argumentele din lina de comanda.
9      *  Nu returneaza nici o valoare. Trebuie sa fie 'public static'
10     */
11    public static void main(String[] args) {
12      System.out.println("Buna ziua"); // Afisarea unui text pe ecran
13    }
14  }
```

Daca inlaturam comentariile, codul ramas este urmatorul:

```
1   public class Salut {
2     public static void main(String[] args) {
3       System.out.println("Buna ziua");
4     }
5   }
```

In laborator: Se va deschide editorul **Notepad**, si **se va edita codul de mai sus (fara comentarii)** intr-un fisier salvat cu numele **`Salut.java`** in directorul cu numele **D:\Software2006\NumarulGrupeiSiSeria**. (exemplu: 421E sau 454D sau 454 E).

1.2.2. Compilarea programului *Salut.java*

Daca **se presupune utilizarea compilatorului Java din linia de comanda** (consola de intrare), atunci trebuie executata urmatoarea comanda, in directorul in care se afla fisierul *Salut.java*:

```
directorcurent> javac Salut.java
```

In urma acestei comenzi, **compilatorul Java va crea genera codul de octeti** corespunzator **intr-un fisier** cu numele ***Salut.class***, in directorul in care se afla si fisierul *Salut.java*.

In laborator: Se va compila codul sursa in directorul cu numele **D:\Software2006\NumarulGrupeiSiSeria**.

1.2.3. Lansarea programului *Salut.java*

Pentru **executia programului**, acesta trebuie **lansat in interpretor**, folosind comanda:

```
directorcurent> java Salut
```

Rezultatul va fi aparitia mesajului **`Buna ziua!`** pe ecranul monitorului.

In laborator: Se va lansa in executie codul de octeti in directorul cu numele **D:\Software2006\NumarulGrupeiSiSeria**.

In final, pe ecran poate fi urmatoarea secventa de informatii:

```
directorcurent> javac Salut.java

directorcurent> java Salut
Buna ziua!

directorcurent>
```

1.2.4. Automatizarea dezvoltarii prin utilizarea unui fisier script

O simplificare a lansarii in executie a programului este **crearea unui fisier script cu extensia .bat** (in acelasi directorul cu sursa java), numit de exemplu **`Run_Salut.bat`**, **cu urmatorul continut:**

```
javac Salut.java
java Salut
pause
```

si lansarea lui in executie. Pe ecran se va obtine:

```
directorcurent> Run_Salut.bat
directorcurent> javac Salut.java
directorcurent> java Salut
Buna ziua!
directorcurent> pause
Press any key to continue...
```

In laborator: Se va crea fisierul script **`Run_Salut.bat`** cu continutul de mai sus. **Se va executa scriptul** prin dublu click, in Windows Explorer, pe numele fisierului.

1.2.5. Depanarea programului *Salut.java*

In laborator: Se vor produce urmatoarele erori si modificari in codul sursa (varianta de 5 linii, fara comentarii), se vor urmari efectele, si se vor face interpretari ale acestora:

Dupa fiecare modificare codul va fi recompilat (iar **in caz de succes executat din nou**), iar **inaintea unei noi modificari va fi mai intai restabilit programul initial**.

Raspunsurile la urmatoarele intrebari vor usura interpretarea rezultatelor :

- ***Ce se afiseaza pe ecran?***
- ***Ce fel de problema apare (eroare de compilare, exceptie in timpul executiei, niciuna)?***
- ***Care este cauza probabila?***
- ***Cat de sugestiv este mesajul care apare?***
- ***Ce concluzii se pot trage?***

I. Se va elimina prima acolada (din linia 1)

Exemplu de rezolvare: La compilare, pe ecran se afiseaza:

```
Salut.java:1: '{' expected
public class Salut
                ^
1 error
```

Problema: eroare de compilare. **Cauza**: compilatorul sesizeaza inexistenta acoladei deschise dupa numele clasei, acolada care marcheaza inceputul corpului clasei. Mesajul e **destul de sugestiv** incat sa ne permita corectarea rapida a erorii. **Concluzii**: …

II. Se va elimina acolada din linia 2

Exemplu de rezolvare: La compilare, pe ecran se afiseaza:

```
Salut.java:2: ';' expected
    public static void main(String[] args)
                                           ^
Salut.java:5: 'class' or 'interface' expected
}
^
Salut.java:6: 'class' or 'interface' expected
^
3 errors
```

Problema: eroare de compilare. **Cauza**: compilatorul sesizeaza inexistenta acoladei deschise dupa numele metodei main, acolada care marcheaza inceputul corpului metodei. **Prima parte a mesajului este utila pentru a corecta eroarea**, dar **restul mesajului poate produce confuzie in primul** moment. Dupa corectarea erorii din linia doi, "erorile" 2 si 3 "dispar". **Concluzii**: Apare un fenomen de "**propagare a erorilor**", lipsa acoladei respective putand avea mai multe interpretari.

III. Se va elimina simbolul punct si virgula (`;`) din linia 3

IV. Se vor elimina parantezele drepte, **`[]`**, din linia 2

V. Se va elimina cuvantul cheie **`void`** (din linia 2)

VI. Se va elimina cuvantul cheie **`static`** (din linia 2)

Exemplu de rezolvare: Dupa compilare, in momentul executiei, pe ecran se afiseaza:

```
Exception in thread "main" java.lang.NoSuchMethodError: main
```

Problema: situatie exceptionala (**`java.lang.NoSuchMethodError`**) in timpul executiei. **Cauza**: Masina Virtuala Java (JVM), cea care sta in spatele interpretorului **`java`** sesizeaza inexistenta unei metode Java cu semnatura completa: `public static void main(String[] args)` Mesajul **produce confuzie in primul** moment, deoarece da impresia ca metoda nu exista, si nu sugereaza cauza situatiei exceptionale. **Concluzii**: ***Este strict necesara declararea metodei `main()` ca fiind de tip `static`. Este recomandabila retinerea simptomelor acestei situatii exceptionale!***

VII. Se vor inlocui ghilimelele cu apostroafe in linia 3

VIII. Se va inlocui numele clasei **`System`** cu **`system`** in linia 3

IX. Se va inlocui numele clasei **`Salut`** cu **`salut`** in linia 1

X. Se va inlocui **`args`** din linia 2 cu cuvantul **`argumente`**

1.3. Scrierea si modificarea programelor Java. Studiu de caz: Program de calcul al unui polinom

1.3.1. Specificatia initiala (varianta cu valori prestabilite)

Sa se scrie un program Java numit **`Polinom0`**, care:
- **creeaza un polinom** cu grad prestabilit si coeficienti de **valori prestabilite** (1, 2, 3, 2, 1),
- **afiseaza polinomul**,
- **calculeaza valoarea polinomului** pentru **o anumita valoare prestabilita** a necunoscutei (2), si
- **afiseaza aceasta valoare**.

```
1    public class Polinom0 {
2
3      public static void main(String[] args) {
4        System.out.println("\nProgramul Polinom0 a fost lansat...\n");
5
6        // Declararea si initializarea variabilei intregi numita gradPolinom,
7        // care contine gradul polinomului, N=4
8
9        // Crearea tabloului coeficientilor (de dimensiune N+1),
10       // numit coeficienti, folosind valorile prestabilite : 1, 2, 3, 4, 5
11
12       // Afisarea polinomului P(X)
13       // - mai intai termenul liber Co
14       // - apoi termenii Ci*X^i, unde i=1,N
15
16       // Declararea si initializarea variabilei intregi numita necunoscuta,
17       // care contine valoarea necunoscutei, X=2
18
19       // Afisarea valorii necunoscutei (X)
20
21       // Declararea si initializarea variabilei intregi numita polinom,
22       // care contine valoarea polinomului, P(X)
23
24       // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
25          // - actualizarea valorii polinomului
26          // - actualizarea valorii X^i, unde i=1,N
27
28       // Afisarea valorii polinomului P(X)
29     }
30   }
```

Observatii:

- **pentru a crea un tablou** se foloseste sintaxa:

```
// Crearea unui tablou de 7 valori intregi, varianta scurta
int[] tab = { 1, 2, 3, 4, 3, 2, 1 };
```

- **pentru a afisa valorile din tabloul `tab`** poate fi folosita urmatoarea secventa de cod:

```
System.out.print("elementele tabloului sunt: " + tab[0]);
for (int i=1; i<tab.length; i++) {
  System.out.print(", " + tab[i]);
}
System.out.println("\n");
```

- **formatul pentru afisarea polinomului** (pentru **gradul 4** si **coeficientii 1**, **2**, **3**, **4**, **5**) va fi:

```
P(X) = 1 + 2*X^1 + 3*X^2 + 4*X^3 + 5*X^4
```

In laborator: Se va completa codul cu instructiuni Java, se va salva intr-un fisier cu numele `Polinom0.java`, se va compila si executa programul. Link catre **`Polinom0.class`**.

In aceasta forma, **la fiecare executie programul va genera aceleasi iesiri.**

1.3.2. Program de calcul al unui polinom cu valori obtinute de la utilizator

Pornind de la programul **`Polinom0`** sa se scrie un program numit **`Polinom1`**, care:
- **creeaza un polinom** cu grad si coeficienti avand **valorile obtinute de la utilizator,**
- **afiseaza polinomul**,
- **calculeaza valoarea polinomului** pentru o **valoare** a necunoscutei **specificata de utilizator**, si
- **afiseaza aceasta valoare**.

Programul **`Polinom1`** reprezinta o **generalizare a programului anterior** (versiune mai flexibila), deoarece gradul polinomului, valorile coeficientilor si necunoscutei sunt **obtinute de la utilizator**.

```
 1    import javax.swing.JOptionPane;
 2
 3    public class Polinom1 {
 4
 5      public static void main(String[] args) {
 6        System.out.println("Programul Polinom1 a fost lansat...");
 7
 8        // Declararea variabilei intregi gradPolinom pt. gradul polinomului, N
 9        // Obtinerea gradului polinomului de la utilizator, conversie String-int
10
11        // Declararea si crearea tabloului coeficientilor, numit coeficienti
12
13        // Obtinerea de la utilizator a coeficientilor Ci, unde i=0,N
14
15        // Afisarea polinomului P(X)
16        // - mai intai termenul liber Co
17        // - apoi termenii Ci*X^i, unde i=1,N
18
19        // Declararea variabilei intregi necunoscuta
20        // Obtinerea valorii necunoscutei de la utilizator, conversie String-int
21
22        // Afisarea valorii necunoscutei (X)
23
24        // Declararea si initializarea variabilei intregi numita polinom,
25        // care contine valoarea polinomului, P(X)
26
27        // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
28           // - actualizarea valorii polinomului
29           // - actualizarea valorii X^i, unde i=1,N
30
31        // Afisarea valorii polinomului P(X)
32
33        System.exit(0); // Inchiderea interfetei grafice
34      }
35    }
```

Observatii:

- **pentru a obtine de la utilizator o valoare de tip sir de caractere (`String`)** se foloseste sintaxa:

```
String nume = JOptionPane.showInputDialog("Introduceti numele"));
```

- **pentru a o valoare de la `String` la `int`** se foloseste sintaxa:

```
int numarStudenti = Integer.parseInt("25");
```

- evident, combinand cele sintaxe se poate scrie:

```
int numarStudenti = Integer.parseInt(JOptionPane.showInputDialog(
                              "Introduceti numarul studentilor din grupa"));
```

- **pentru a declara si crea (aloca memorie pentru) un tablou de intregi** se foloseste sintaxa:

```
int[] noteStudenti = new int[numarStudenti];
```

- **pentru a popula (atribui valori initiale) un tablou de intregi** se foloseste sintaxa:

```
for (int i=0; i<noteStudenti.length; i++) {
   noteStudenti[i] = Integer.parseInt(JOptionPane.showInputDialog(
                              "Introduceti nota studentului cu numarul " + i));
}
```

In laborator: Se va completa codul cu instructiuni Java, se va salva intr-un fisier cu numele `Polinom1.java`, se va compila si executa programul. Link catre **`Polinom1.class`**.

In aceasta forma, **la fiecare executie programul poate genera alte iesiri, in functie de valorile de intrare**.

1.3.3. Program de calcul al unui polinom (varianta cu delegare functionala)

Pornind de la programul **`Polinom1`** sa se scrie codul unei clase Java numita **`Polinom2`**, a carei structura interna contine:

- **o metoda** (declarata `public static`) numita **`obtineGrad()`**, care obtine de la utilizator valoarea gradului polinomului, si o returneaza ca intreg de tip **`int`**,

- **o metoda** (declarata `public static`) numita **`stabilesteCoeficienti()`**, care primeste un parametru intreg de tip **`int`**, numit **`gradPolinom`**, reprezentand gradului polinomului, creaza un nou tablou al coeficientilor (cu **`gradPolinom +1`** elemente), obtine de la utilizator valori pentru coeficientii polinomului si populeaza cu ei tabloul nou creat, apoi returneaza taboul de tip **`int[]`** creat,

- **o metoda** (declarata `public static`) numita **`obtineNecunoscuta()`**, care obtine de la utilizator valoarea necunoscutei, si o returneaza ca intreg de tip **`int`**,

- **o metoda** (declarata `public static`) numita **`afisarePolinom()`**, care primeste un parametru intreg de tip **`int`**, numit **`gradPolinom`**, reprezentand gradului polinomului, si un parametru de tip **`int[]`**, numit **`coeficienti`**, reprezentand coeficientii polinomului, si afiseaza polinomul corespunzator valorilor primite,

- **o metoda** numita **`valoarePolinom()`**, care primeste un parametru intreg de tip **`int`**, numit **`gradPolinom`**, reprezentand gradului polinomului, un parametru de tip **`int[]`**, numit **`coeficienti`**, reprezentand coeficientii polinomului, si un parametru intreg de tip **`int`**, numit **`necunoscuta`**, reprezentand necunoscuta, si calculeaza valoarea polinomului corespunzatoare valorilor primite si o returneaza ca intreg de tip **`int`**,

- o **metoda principala, de test**, care:
  - deleaga catre metoda **`obtineGrad()`** obtinerea valorii gradului polinomului,
  - deleaga catre metoda **`stabilesteCoeficienti()`** stabilirea valorilor coeficientilor polinomului,
  - deleaga catre metoda **`afisarePolinom()`** afisarea polinomului,
  - deleaga catre metoda **`obtineNecunoscuta()`** obtinerea valorii necunoscutei,
  - afiseaza valoarea necunoscutei,
  - deleaga catre metoda **`valoarePolinom()`** calculul valorii polinomului,
  - afiseaza valoarea polinomului.

In laborator: Se va completa codul sursa cu instructiuni Java, se va salva intr-un fisier cu numele `Polinom2.java`, se va compila si executa programul. Link catre **`Polinom2.class`**.

```
1   import javax.swing.JOptionPane;
2   public class Polinom2 {
3
4     // Metoda care obtine de la utilizator gradul polinomului
5     public int obtineGrad() {
6       // Obtinerea de la utilizator a gradului polinomului
7
8       // Returnarea valorii gradului polinomului
9     }
10
11    // Metoda care obtine de la utilizator coeficientii polinomului
12    public int[] stabilesteCoeficienti(int gradPolinom) {
13      // Declararea si crearea tabloului coeficientilor, numit coeficienti
14
15      // Obtinerea de la utilizator a coeficientilor Ci, unde i=0,N
16
17      // Returnarea tabloului coeficientilor
18    }
19
20    // Metoda care afiseaza polinomul P(X)
21    public void afisarePolinom(int gradPolinom, int[] coeficienti) {
22      // Afisarea polinomului P(X)
23      // - mai intai termenul liber Co
24      // - apoi termenii Ci*X^i, unde i=1,N
25    }
26
27    // Metoda care obtine de la utilizator valoarea necunoscutei
28    public int obtineNecunoscuta() {
29      // Obtinerea de la utilizator a valorii necunoscutei
30
31      // Returnarea valorii necunoscutei
32    }
33
34    // Metoda care calculeaza valoarea polinomului pt o valoare a necunoscutei
35    public int valoarePolinom(int gradPolinom, int[] coeficienti,
36                              int necunoscuta) {
37      // Declararea si initializarea variabilei intregi numita polinom,
38      // care contine valoarea polinomului, P(X)
39
40      // Calculul polinomului P(X) = suma(Ci * X^i), unde i=0,N
41         // - actualizarea valorii polinomului
42         // - actualizarea valorii X^i, unde i=1,N
43
44      // Returnarea valorii polinomului
45    }
46
47    // Metoda principala. Utilizata pentru testarea celorlalte metode.
48    public static void main(String[] args) {
49      // Apelul metodei care obtine de la utilizator gradul polinomului
50
51      // Apelul metodei care obtine de la utilizator coeficientii polinomului
52
53      // Apelul metodei care afiseaza polinomul
54
55      // Apelul metodei care obtine o valoare a necunoscutei
56      // Afisarea valorii necunoscutei
57
58      // Apelul metodei care calculeaza polinomul pentru necunoscuta data
59      // Afisarea valorii polinomului
60      System.exit(0); // Inchiderea interfetei grafice
61    }
62  }
```

De reflectat:

1. Ce alte metode ar putea fi utile pentru lucrul cu polinomul?

2. Cum ar trebui modificat programul **pentru a putea efectua scenariul in mod repetat**?

In laborator:

1. Se va adauga o metoda (declarata `public static`) numita **`toString()`**, care primeste un parametru intreg de tip **`int`**, numit **`gradPolinom`**, reprezentand gradului polinomului, si un parametru de tip **`int[]`**, numit **`coeficienti`**, reprezentand coeficientii polinomului, si care **returneaza polinomul** sub forma de **sir de caractere** (obiect de tip **`String`**) in formatul (pentru **grad 4** si **coeficienti 1**,**2**,**3**,**4**,**5**):

```
P(X) = 1 + 2*X^1 + 3*X^2 + 4*X^3 + 5*X^4
```

2. Se va modifica programul **`Polinom2`** astfel incat **sa utilizeze** metoda **`toString()`** **pentru a afisa forma generala a polinomului** (**in loc sa utilizeze** metoda **`afisarePolinom()`** in acest scop).

3. Se va crea o clasa noua, `UtilizarePolinom2` care sa utilizeze metodele clasei **`Polinom2`** pentru calculul unui polinom (**delegare catre metodele altei clase**).

4. Se va modifica programul **`Polinom2`** **pentru a putea efectua scenariul in mod repetat**.

1.4. Referinte

Bruce Eckel, **Thinking in Java, 3rd ed. Rev. 4.0**, **http://www.faqs.org/docs/think_java/TIJ3_c.htm**
(PDF: http://www.planetpdf.com/codecuts/pdfs/eckel/TIJ3.zip (beta), Zipped HTML and source code: http://www.mindviewinc.com/downloads/TIJ-3rd-edition4.0.zip)
Kevin Taylor, **Java Progr. Tutorial**, http://java.about.com/od/beginningjava/a/beginjavatutor.htm
Sun Microsystems, **The Java™ Tutorials**, **http://java.sun.com/docs/books/tutorial/java/TOC.html**
(download : http://java.sun.com/docs/books/tutorial/information/download.html)
Sun Microsystems, **J2SE 5.0 API Specification, http://java.sun.com/j2se/1.5.0/docs/api/index.html**
Sun Microsystems, **JDK™ 5.0 Documentation,** http://java.sun.com/j2se/1.5.0/docs/index.html

1.5. Teme pentru acasa

Tema de casa pentru data viitoare:

I. Sinteza (interpretarea) raporturilor de eroare obtinute prin modificarea programului `Salut.java` (scrisa pe hartie, nu tiparita la imprimanta).

II. Codurile sursa ale programelor `Polinom0`, `Polinom1` si `Polinom2` completate.

De reflectat, pentru examen, asupra programelor din paragrafele urmatoare:

1. Cum trebuie modificat programul pentru a lucra cu **valori specificate de utilizator** in timpul executiei?

2. Cum poate fi realizata delegarea functionala?

3. Cum poate fi realizata delegarea catre o alta clasa (metodele altei clase)?

4. Cum trebuie modificat programul pentru a putea efectua scenariul **in mod repetat**?

1.5.1. Program de calcul al numarului de aparitii ale fiecarei valori intr-o multime (histograma)

Programul **`NumarAparitii0.java`** creaza un tablou care contine o multime de valori **prestabilite** (cea minima este 1, cea maxima **prestabilita**), afiseaza elementele tabloului, **calculeaza si stocheaza intr-un tablou numarul de aparitii ale fiecarei valori in multime** (**histograma**), si afiseaza elementele acestui tablou.

```
1    public class NumarAparitii0 {
2
3      public static void main(String[] args) {
4
5        System.out.println("\n  Program NumarAparitii (histograma) \n");
6
7        // Valoarea maxima (M) (valoarea minima este 1)
8        int maxim = 5;
9
10       // Numarul de valori (N)
11       int numarValori = 20;
12
13       // Crearea tabloului (de dimensiune N) valorilor
14       int[] valori = { 1, 2, 3, 4, 5, 4, 3, 2, 1, 2,
15                       3, 4, 5, 4, 3, 2, 1, 2, 3, 4 };
16
17       // Afisarea tabloului initial
18       System.out.print("\nValorile testate: ");
19
20       for (int i=0; i<numarValori; i++) {
21         System.out.print(valori[i] + " ");
22       }
23       System.out.println("\n");
24
25       // Crearea tabloului (de dimensiune M) cu numarul de aparitii
26       int[] numarAparitii = new int[maxim];
27
28       // Determinarea numarul de aparitii ale fiecarei valori de la 1 la M
29       for (int i=0; i<numarValori; i++) {
30         numarAparitii[valori[i]-1]++;
31       }
32
33       // Afisarea tabloului cu numarul de aparitii
34       for (int i=0; i<maxim; i++) {
35         System.out.println("Valoarea  " + (i+1) +
36                            " apare de " + numarAparitii[i] + " ori");
37       }
38       System.out.println();
39     }
40   }
```

1.5.2. Program de cautare a unor cuvintelor cheie in texte (siruri de caractere) date

Programul **`CautareCuvantCheie0.java`** creaza un tablou care contine siruri de caractere **prestabilite** si un tablou care contine cuvinte cheie **prestabilite**, afiseaza elementele tabloului, pentru fiecare sir de caractere si fiecare cuvant cheie determina daca si unde se afla cuvantul in sir, si informeaza utilizatorul cu privire la cuvintele gasite si pozitia lor in siruri.

```
 1    import java.io.*;
 2    import java.util.*;
 3
 4    public class CautareCuvantCheie0 {
 5
 6      public static void main(String[] args) throws IOException {
 7
 8        String[] texteAnalizate = {
 9        "The string tokenizer class allows an application" ,
10        "to break a string into tokens." ,
11        "The tokenization method is much simpler than" ,
12        "the one used by the StreamTokenizer class. " ,
13        "The StringTokenizer methods do not distinguish" ,
14        "among identifiers, numbers, and quoted strings," ,
15        "nor do they recognize and skip comments. " ,
16        "The set of delimiters (the characters that separate tokens)" ,
17        "may be specified either at creation time or on a per-token basis." };
18
19        String[] cuvinteCheie = { "string" , "token" };
20
21        // Pentru toate textele analizate
22        for (int i=0; i<texteAnalizate.length; i++) {
23
24          // Pentru toate cuvintele cheie cautate
25          for (int j=0; j<cuvinteCheie.length; j++) {
26
27            // Daca un anumit cuvant cheie este gasit intr-un anumit text
28            // Varianta cu String.indexOf()
29            if ( texteAnalizate[i].indexOf(cuvinteCheie[j]) > -1 ) {
30
31              // Informeaza utilizatorul (indicand si pozitia)
32              System.out.println("Cuvantul cheie '" + cuvinteCheie[j] +
33                 "' a fost gasit in textul '" + texteAnalizate[i] +
34                 "' pe pozitia " +
35                 (texteAnalizate[i].indexOf(cuvinteCheie[j]) + 1) + "\n");
36            }
37
38            // Daca un anumit cuvant cheie este gasit intr-un anumit text
39            // Varianta cu String.split()
40            if ( texteAnalizate[i].split(cuvinteCheie[j]).length != 1 ) {
41
42              // Informeaza utilizatorul
43              System.out.println("Cuvantul cheie '" + cuvinteCheie[j] +
44                 "' a fost gasit in textul '" + texteAnalizate[i] + "'\n");
45            }
46          }
47        }
48      }
49    }
```

1.5.3. Program de calcul al unor statistici asupra unor texte (siruri de caractere) date

Programul **`StatisticiText0.java`** creaza un tablou care contine siruri de caractere **prestabilite** si un tablou care contine cuvinte cheie **prestabilite**, afiseaza elementele tabloului, pentru fiecare sir de caractere si fiecare cuvant cheie determina daca si unde se afla cuvantul in sir, si informeaza utilizatorul cu privire la cuvintele gasite si pozitia lor in siruri.

```
 1    import java.io.*;
 2    import java.util.*;
 3
 4    public class StatisticiText0 {
 5
 6      public static void main(String[] args) throws IOException {
 7
 8        String textAnalizat =
 9         "The string tokenizer class allows an application to break a string" +
10         " into tokens. The tokenization method is much simpler than the one" +
11         " used by the StreamTokenizer class. The StringTokenizer methods do" +
12         " not distinguish among identifiers, numbers, and quoted strings," +
13         " nor do they recognize and skip comments. The set of delimiters" +
14         " (the characters that separate tokens) may be specified either at" +
15         " creation time or on a per-token basis.";
16
17        String[] cuvinte = textAnalizat.split(" ");
18
19        // Numarul de cuvinte
20        System.out.println("  Textul:\n\n" + textAnalizat + "\n\n  contine " +
21                           cuvinte.length + " cuvinte:");
22
23        // Cuvintele
24        for (int i=0; i<cuvinte.length; i++) {
25          System.out.print("\t" + cuvinte[i]);
26        }
27        System.out.println();
28
29        // Propozitiile
30        System.out.println("\n  Textul contine urmatoarele propozitii:\n");
31
32        String propozitie = null;
33
34        int p = 0;
35
36        String text = textAnalizat;
37
38        for (int f = 0; (f = text.indexOf(".")) > -1; p++) {
39          propozitie = text.substring(0, f);
40          System.out.println(propozitie + ".\n");
41          text = text.substring(f+1, text.length());
42          if (text.indexOf(" ") == 0) {
43            text = text.substring(1, text.length());
44          }
45        }
46
47        // Numarul de propozitii
48        System.out.println("\n  In total sunt " + p + " propozitii:\n");
49      }
50    }
```