SwRTc – supliment proiect

Fluxuri de intrare-iesire (IO) Java

IO.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:

- **Lucrul cu fluxurile de intrare-iesire (IO) Java**
- **Programe ecou** pentru consola standard si fisiere
- **Programe care combina diferite tipuri de fluxuri IO**

IO.2. Clase Java pentru operatii de intrare-iesire (IO)

IO.2.1. Definitii si clasificari ale fluxurilor IO Java

Programele pot avea nevoie sa **preia** informatii **de la surse** externe, sau sa **trimita** informatii **catre destinatii** externe. Sursa si/sau destinatia pot fi: ***fisier pe disc, retea, memorie (alt program), dispozitive IO standard (ecran, tastatura)***. Tipurile informatiilor sunt diverse: ***caractere, obiecte, imagini, sunete.***

Pentru **preluarea** informatiilor programul **deschide un flux de la o sursa** de informatii si **citeste serial** (secvential, FIFO) informatiile, astfel:

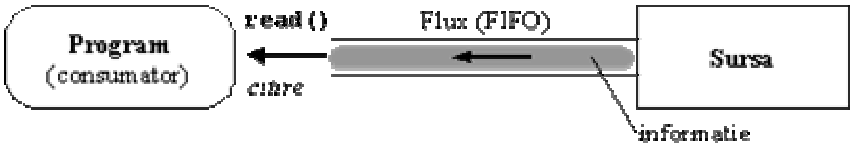

Figura 1. Fluxuri de intrare (citire) Java

Pentru **trimiterea** informatiei programul **deschide un flux catre o destinatie** de informatie si **scrie serial** (secvential, FIFO) informatiile, astfel:

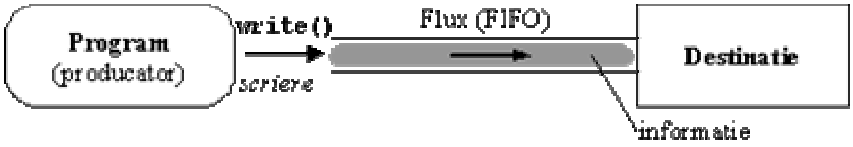

Figura 2. Fluxuri de iesire (scriere) Java

In functie de **tipul de date** transferate, clasele din pachetul **`java.io`** se impart in **doua ierarhii:**

- fluxurile **de caractere (pe 16b)**, cu radacini ale arborilor de clase derivate superclasele abstracte:
  - **`Reader`** (pentru fluxuri de intrare) si
  - **`Writer`** (pentru fluxuri de iesire)

- fluxurile **de octeti (pe 8b)**, avand ca radacini ale arborilor de clase derivate superclasele abstracte:
  - **`InputStream`** (pentru fluxuri de intrare) si
  - **`OutputStream`** (pentru fluxuri de iesire)

In functie de **specializarea pe care o implementeaza**, subclasele claselor de mai sus se impart in **doua categorii:**

- **fluxuri terminale** (*data sink*), care **au o sursa / destinatie propriu-zisa**, **nu un alt flux**, avand ca rol **interfatarea cu acea sursa / destinatie**, ***fisierele, memoria (tablourile), reteaua (socketurile), sirurile de caractere (String), alte programe (prin conducte - pipes).***

- **fluxuri de prelucrare** (*processing*), care **au ca sursa / destinatie un alt flux**, avand ca rol **prelucrarea informatiilor care trec prin flux**: ***buffer-are (stocare temporara), filtrare de diferite tipuri (conversie, contorizare, etc.), tiparire.***

Fluxuri terminale (*data sink streams*)

Tip de Terminal	Utilizare	Fluxuri de caractere	Fluxuri de octeti
Memorie	*Accesul secvential la tablouri*	**CharArray**Reader	**ByteArray**InputStream
		CharArrayWriter	**ByteArray**OutputStream
	Accesul secvential la siruri de caractere	**String**Reader	**StringBuffer**InputStream
		StringWriter	**StringBuffer**OutputStream
Canal / conducta (*pipe*)	*Conducte intre programme*	**Piped**Reader	**Piped**InputStream
		PipedWriter	**Piped**OutputStream
Fisier	*Accesul la fisiere*	**File**Reader	**File**InputStream
		FileWrite	**File**OutputStream

Fluxuri de prelucrare (*processing streams*)

Tip de Prelucrare	Utilizare	Fluxuri de Caractere	Fluxuri de octeti
Buffer-are	*Stocare temporară*	**Buffered**Reader	**Buffered**InputStream
		BufferedWriter	**Buffered**OutputStream
Filtrare	*Prelucrare*	**Filter**Reader	**Filter**InputStream
		FilterWriter	**Filter**OutpuStream
Conversie octet/caracter	*Bridge byte-char*	**InputStream**Reader	
		OutputStreamWriter	
Concatenare	*Prelucrare*		**Sequence**InputStream
Serializarea obiectelor			**Object**InputStream
			ObjectOutputStream
Conversia datelor	*Acces la tip date primitiv Java*		**Data**InputStream
			DataOutputStream
Numararea (contorizarea)	*Numarare linii*	**LineNumber**Reader	**LineNumber**InputStream
Testare	*Buffer de 1 byte/char*	**PushBock**Reader	**Pushback**InputStream
Imprimare	*Tiparire*	**Print**Writer	**Print**Stream

In continuare sunt prezentate cateva exemple de utilizare a fluxurilor de prelucrare care pot fi utilizate de programe pentru citirea de la consola standard de intrare sau pentru scrierea la consola standard de iesire.

IO.2.2. Citirea sirurilor de caractere de la consola prin fluxurile `BufferedReader` si `InputStreamReader`

Constructorul tipic al clasei **`BufferedReader`** este:

`BufferedReader``(Reader in)` Creaza un flux de intrare a caracterelor cu stocare temporara (si posibilitate de citire a caracterelor sub forma de linii) din fluxul de intrare a caracterelor primit ca parametru (`in`), utilizand dimensiunea implicita a tabloului intern.

Metoda oferita de clasa **`BufferedReader`** pentru **citirea liniilor de text** este:

`String`	**`readLine`**`()` Citeste o linie de text (citeste din bufferul intern pana la intalnirea caracterului care semnaleaza terminarea liniei). Se blocheaza in asteptarea caracterului care semnaleaza terminarea liniei.

Constructorul tipic al clasei **`InputStreamReader`** este:

`InputStreamReader``(InputStream in)` Creaza un flux de intrare a caracterelor obtinute prin conversia octetilor primiti de la fluxul de intrare primit ca parametru (`in`) utilizand setul de caractere implici.

O utilizare tipica a acestor doua fluxuri de prelucrare, **in cascada**, este cea care permite **citirea sirurilor de caractere de la consola standard de intrare** (tastatura, care este **incapsulata in obiectul `System.in`**, al carui tip este **`InputStream`** - flux de intrare a octetilor).

Exemplu: citirea unui nume de la tastatura:

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Introduceti numele: ");
String nume = in.readLine();
```

IO.2.3. Afisarea sirurilor de caractere la consola prin fluxul `PrintStream`

Metodele oferite de clasa **`PrintStream`** pentru afisarea sirurilor de caractere (scrierea intr-un flux de iesire a octetilor a sirurilor de caractere care cuprind si caractere *escape*) sunt:

`void`	**`print`**`(boolean b)` Afiseaza valoarea booleana primita ca parametru.
	… (Similar pentru celelalte tipuri de date primitive Java: char, double, float, int, …)
`void`	**`print`**`(String s)` Afiseaza sirul de caractere primit ca parametru.
`void`	**`println`**`()` Termina linia curenta, **adaugand un separator de linie**.
`void`	**`println`**`(boolean x)` Afiseaza valoarea booleana primita ca parametru si apoi termina linia.
	… (Similar pentru celelalte tipuri de date primitive Java: char, double, float, int, …)
`void`	**`println`**`(String x)` Afiseaza sirul de caractere primit ca parametru si apoi termina linia.

O utilizare tipica a acestui flux de prelucrare este cea care permite **scrierea sirurilor de caractere la consola standard de iesire** (monitorul, care este **incapsulat intr-un `OutputStream` ce serveste ca destinatie obiectului `System.out`**, al carui tip este **`PrintStream`**).

Exemplu: afisarea argumentelor programului curent:

```
PrintStream ps = System.out;
ps.println("Argumentele programului: ");
for (int i=0; i<args.length; i++) {
  ps.print(args[i] + " ");
}
ps.println();
```

IO.2.4. Citirea sirurilor de caractere de la consola prin fluxul `DataInputStream`

Constructorul clasei **`DataInputStream`** este:

`DataInputStream``(InputStream in)` Creaza un flux de intrare a octetilor care permite citirea datelor formatate (ca tipuri primitive) de la fluxul de intrare a octetilor primit ca parametru (`in`).

Metodele oferite de clasa **`DataInputStream`** pentru citirea datelor formatate (ca tipuri primitive) de la fluxul de intrare a octetilor primit ca parametru in momentul constructiei sunt:

`boolean`	**`readBoolean`**`()` Citeste un octet din fluxul de intrare a octetilor primit ca parametru in momentul constructiei (din amonte), si returneaza `true` daca este nenul si `false` daca este nul.
`byte`	**`readByte`**`()` Citeste si returneaza un octet.
`char`	**`readChar`**`()` Citeste si returneaza un `char`.
`double`	**`readDouble`**`()` Citeste 8 octeti si returneaza un `double`.
`float`	**`readFloat`**`()` Citeste 4 octeti si returneaza un `float`.
`void`	**`readFully`**`(byte[] b)` Citeste octetii disponibili si ii stocheaza in tabloul primit ca parametru. Metoda se blocheaza pana cand `b.length` octeti sunt disponibili.
`void`	**`readFully`**`(byte[] b, int off, int len)` Citeste `len` octeti si ii stocheaza in tabloul primit ca parametru incepand de la indexul `off`. Metoda se blocheaza pana cand `len` octeti sunt disponibili.
`int`	**`readInt`**`()` Citeste 4 octeti si returneaza un `int`.
`long`	**`readLong`**`()` Citeste 8 octeti si returneaza un `long`.
`short`	**`readShort`**`()` Citeste 2 octeti si returneaza un `short`.
`String`	**`readUTF`**`()` Citeste un sir de caractere care a fost codat utilizand un format UTF-8 modificat.
`String`	**`readLine()`** **Metoda nerecomandata (*deprecated*) pentru citirea sirurilor de caractere terminate cu separator de linie.**

Exemplu: citirea unui nume de la tastatura:

```
DataInputStream in = new DataInputStream(new BufferedInputStream(System.in));
System.out.println("Introduceti numele: ");
String nume = in.readLine();
```

IO.2.5. Afisarea sirurilor de caractere la consola prin fluxul `DataOutputStream`

Constructorul clasei **`DataOutputStream`** este:

`DataOutputStream``(OutputStream out)` Creaza un flux de iesire a octetilor care permite scrierea datelor formatate (ca tipuri primitive) catre fluxul de iesire a octetilor primit ca parametru (`out`).

Metodele oferite de clasa **`DataOutputStream`** pentru scrierea datelor formatate (ca tipuri primitive) in fluxul de iesire a octetilor primit ca parametru in momentul constructiei (din aval) sunt:

`void`	**`flush`**`()` Forteaza trimiterea datelor scrise in acest flux de iesire catre fluxul de iesire (din aval) primit ca parametru in momentul constructiei.
`void`	**`writeBoolean`**`(boolean v)` Scrie valoarea booleana primita ca parametru in fluxul de iesire (din aval) primit ca parametru in momentul constructiei, ca octet.
`void`	**`writeByte`**`(int v)` Scrie octetul specificat ca parametru (cei mai putin semnificativi 8 biti ai argumentului `v`) in fluxul de iesire (din aval) primit ca parametru in momentul constructiei.
`void`	**`writeBytes`**`(String s)` Scrie ca secventa de octeti sirul de caractere parametru, in fluxul de iesire din aval.
`void`	**`writeChar`**`(int v)` Scrie cei doi octeti ai caracterului Unicode specificat ca parametru (cei mai putin semnificativi 16 biti ai argumentului `v`) in fluxul de iesire din aval.
`void`	**`writeDouble`**`(double v)` Converteste argumentul `double` la un `long` utilizand metoda `doubleToLongBits()` din clasa `Double`, apoi scrie valoarea rezultata ca 8 octeti in fluxul de iesire din aval, cel mai semnificativ octet primul.
`void`	**`writeFloat`**`(float v)` Converteste argumentul `float` la un `int` utilizand `floatToIntBits()` din `Float`, apoi scrie valoarea rezultata ca 4 octeti in fluxul de iesire din aval, cel mai semnificativ octet primul.
`void`	**`writeInt`**`(int v)` Scrie valoarea specificata ca parametru `int` ca 4 octeti in fluxul de iesire din aval, cel mai semnificativ octet primul.
`void`	**`writeLong`**`(long v)` Scrie valoarea specificata ca parametru `long` ca 8 octeti in fluxul de iesire din aval, cel mai semnificativ octet primul.
`void`	**`writeShort`**`(int v)` Scrie valoarea specificata ca parametru `short` ca 2 octeti in fluxul de iesire din aval, cel mai semnificativ octet primul.
`void`	**`writeUTF`**`(String str)` Scrie sirul de caractere specificat ca parametru in fluxul de iesire (din aval) primit ca parametru in momentul constructiei utilizand codarea UTF-8 modificata a Java, intr-o forma independenta de masina de calcul.

Exemplu: afisarea argumentelor programului curent:

```
DataOutputStream dos = new DataOutputStream(System.out);
dos.writeBytes("Argumentele programului: \n");
for (int i=0; i<args.length; i++) {
  dos.writeBytes(args[i] + " ");
}
dos.writeChar('\n');
dos.flush();
```

IO.2.6. Citirea si scrierea fisierelor prin fluxuri de caractere

In continuare sunt ilustrate:

- **citirea dintr-un fisier, caracter cu caracter,** prin intermediul unui **flux de caractere** (Unicode!):

```
1    // Crearea unui obiect referinta la fisier pe baza numelui fisierului
2    File inputFile = new File("nume1.txt");
3
4    // Crearea unui flux de intrare a caracterelor dinspre fisierul dat
5    FileReader in = new FileReader(inputFile);
6
7    // Citirea unui caracter din fisier
8    int c = in.read();          // Input
9
10   // Inchiderea fisierului
11   in.close();
```

- **scrierea intr-un fisier, caracter cu caracter,** prin intermediul unui **flux de caractere**:

```
1    // Crearea unui obiect referinta la fisier pe baza numelui fisierului
2    File outputFile = new File("nume2.txt");
3
4    // Crearea unui flux de iesire a caracterelor spre fisierul dat
5    FileWrite out = new FileWriter(outputFile);
6
7    char c = 'x';
8    // Scrierea unui caracter in fisier
9    out.write(c);          // Output
10
11   // Inchiderea fisierului
12   out.close();
```

IO.3. Programe de lucru cu fluxuri IO Java

IO.3.1. Programe ecou consola

In continuare sunt prezentate cateva variante de programe Java care folosesc **diferite fluxuri de intrare-iesire** pentru a lucra cu **consolele standard de intrare si de iesire**.

De reflectat:

Care sunt diferentele intre urmatoarele doua programe?

Programul **EcouConsola_BR_PS** **citeste linii de text de la tastatura** (consola standard de intrare) si **le afiseaza in consola standard de iesire**, folosind clasele **BufferedReader** si **PrintStream**. **(script pentru lansare)**

```
1  import java.io.*;
2  /**
3    * Ecou consola folosind BufferedReader si PrintStream
4    */
5  public class EcouConsola_BR_PS {
6
7    public static void main (String args[]) throws IOException {
8      // Intrare - consola prin BufferedReader (si InputStreamReader)
9      BufferedReader inConsola =
10           new BufferedReader(new InputStreamReader(System.in));
11
```

```
12      // Iesire - consola prin PrintStream
13      PrintStream outConsola= System.out;
14
15      String sirCitit;
16      outConsola.println("\nPentru terminare introduceti '.' si <Enter>");
17
18      while (true) {
19        outConsola.print("\n?> ");
20
21        // Citirea unei linii de text de la consola
22        sirCitit = inConsola.readLine();
23
24        // Conditie terminare program
25        if (sirCitit.equals(new String("."))) break;
26
27        // Citirea unei linii de text de la consola
28        outConsola.println("!> " + sirCitit);
29      }
30    }
31  }
```

Programul **EcouConsola_DIS_DOS** **citeste linii de text de la tastatura** (consola standard de intrare) si **le afiseaza in consola standard de iesire**, folosind clasele **DataInputStream** si **DataOutputStream**. (**script pentru lansare**)

```
1   import java.io.*;
2   /**
3     * Ecou consola folosind DataInputStream si DataOutputStream
4     */
5   public class EcouConsola_DIS_DOS {
6
7     public static void main (String args[]) throws IOException {
8       // Intrare - consola prin DataInputStream (si BufferedInputStream)
9       DataInputStream inConsola
10            = new DataInputStream(new BufferedInputStream(System.in));
11      // Iesire - consola prin DataOutputStream
12      DataOutputStream outConsola = new DataOutputStream(System.out);
13
14      String sirCitit;
15      outConsola.writeBytes("Pentru oprire introduceti '.' si <Enter>\n");
16      outConsola.flush();
17
18      while (true) {
19        outConsola.writeBytes("\n?> ");
20        outConsola.flush();
21
22        // Citirea unei linii de text de la consola
23        sirCitit = inConsola.readLine(); // metoda nu e recomandata!!!
24
25        // Conditie terminare program
26        if (sirCitit.equals(new String("."))   break;
27
28        // Scrierea liniei de text la consola
29        outConsola.writeBytes("!> " + sirCitit + "\n");
30        // Fortarea trimiterii (golirii bufferului)
31        outConsola.flush();
32      }
33    }
34  }
```

IO.3.2. Programe de copiere a fisierelor

In continuare sunt prezentate cateva variante de programe Java care folosesc **diferite fluxuri de intrare-iesire** pentru a lucra cu **fisiere pe disc**.

De reflectat:

Care sunt diferentele intre urmatoarele doua programe?

Programul **CopiereFisier1** **citeste linii de text dintr-un fisier** de intrare si **le scrie intr-un fisier (copie) de iesire**, folosind clasele **BufferedReader** si **PrintWriter**. (**script pentru lansare**)

```
1   import java.io.*;
2
3   public class CopiereFisier1 {
4
5     public static void main(String[] args) throws IOException {
6
7       System.out.println("\nProgramul CopiereFisier a fost lansat...\n");
8
9       // Intrare consola
10      BufferedReader inConsola = new
11                          BufferedReader(new InputStreamReader(System.in));
12
13      System.out.print("Introduceti numele fisierului de copiat: ");
14      String numeFisier = inConsola.readLine();
15
16      // Intrare fisier
17      BufferedReader inFisier = null;
18      try {
19        inFisier = new BufferedReader(new FileReader(numeFisier));
20      }
21      catch (FileNotFoundException ex) {
22        System.out.println("Fisierul nu exista in acest director");
23        System.exit(0);
24      }
25
26      numeFisier = "Copie_" + numeFisier;
27
28      // Iesire fisier
29      PrintWriter outFisier = new PrintWriter(
30                          new BufferedWriter(new FileWriter(numeFisier)));
31
32      String linieText;
33
34      // Cat timp mai sunt linii de text de citit din fisier
35      while ((linieText = inFisier.readLine())!=null) {
36
37        // Scrierea liniei de text in fisier
38        outFisier.println(linieText);
39        outFisier.flush();
40      }
41
42      // Inchiderea fisierelor
43      inFisier.close();
44      outFisier.close();
45    }
46  }
```

Programul **CopiereFisier_BR_PS** **citeste linii de text dintr-un fisier** de intrare si **le scrie intr-un fisier (copie) de iesire**, folosind clasele **BufferedReader**, **InputStreamReader** si **PrintWriter**. (**script pentru lansare**)

```
1  import java.io.*;
```

```
 2
 3  /**
 4    * Copiere fisier folosind BufferedReader si PrintStream
 5    */
 6  public class CopiereFisier_BR_PS {
 7    public static void main (String args[]) throws IOException {
 8      BufferedReader inConsola  = new BufferedReader
 9                                  (new InputStreamReader(System.in));
10
11      System.out.print("\nIntroduceti numele fisierului de copiat: ");
12      String sirCitit = inConsola.readLine();
13
14      // Intrare -fisier prin BufferedReader,InputStreamReader,FileInputStream
15      BufferedReader inFisier = new BufferedReader(new InputStreamReader
16                                  (new FileInputStream(sirCitit)));
17
18      System.out.print("Introduceti numele fisierului copie: ");
19      sirCitit = inConsola.readLine();
20
21      // Iesire - fisier prin PrintStream (si FileOutputStream)
22      PrintStream outFisier = new PrintStream(new FileOutputStream(sirCitit));
23
24      while (true) {
25        // Citirea unei linii de text din fisier
26        sirCitit = inFisier.readLine();
27
28        // Conditie terminare program
29        if (sirCitit == null)   break;
30
31        // Scrierea liniei de text in fisier
32        outFisier.println(sirCitit);
33      }
34    }
35  }
```

IO.4. Modificarea programelor Java pentru a lucra cu fluxuri IO

IO.4.1. Programe de lucru cu fisiere

In laborator:

1. Pornind de la programul **`CopiereFisier1`** **se va crea** un program **`AfisareFisier1`** care sa **citeasca linii de text dintr-un fisier** de intrare si **sa le afiseze in consola standard de iesire**, folosind clasele **`BufferedReader`** si **`PrintWriter`.**

2. Pornind de la programul **`CopiereFisier1`** **se va crea** un program **`ScriereInFisier1`** care sa **citeasca linii de text de la tastatura** (consola standard de intrare) si **sa le scrie intr-un fisier de iesire**, folosind clasele **`BufferedReader`** si **`PrintWriter`.**

3. Pornind de la programul **`CopiereFisier_BR_PS`** **se va crea** un program **`ScriereInFisier_BR_PS`** care sa **citeasca linii de text de la tastatura** (consola standard de intrare) si **sa le scrie intr-un fisier de iesire**, folosind clasele **`BufferedReader`**, **`InputStreamReader`** si **`PrintWriter`.**

4. Se va compara efectul utilizarii programului **`ScriereInFisier1`** **cu efectul utilizarii** programului **`ScriereInFisier_BR_PS`.**

IO.4.2. Program de calcul al unui polinom (varianta fara fisiere)

Programul **Polinom3.java** reprezinta o varianta a programului **Polinom2.java** in care:

- **initializarea atributelor** este realizata de un **constructor** fara parametri, **Polinom3()**,

- **afisarea polinomului** este realizata de o **metoda** fara parametri si care nu returneaza nimic, numita **afisarePolinom()**,

- **obtinerea atributelor** in format **String** este realizata de **o metoda** cu numele **toString()**, care **nu primeste parametru** si **returneaza polinomul** sub forma de **sir de caractere** (obiect de tip **String**) in formatul:
  P(X) = 1 **+** 2***X^**1 **+** 3***X^**2 **+** 4***X^**3 **+** 5***X^**4
daca se presupune ca **gradul** este **4** iar **coeficientii** sunt {**1**, **2**, **3**, **4**, **5**}.

- **returnarea primului coeficient** este realizata de o metoda fara parametri, **termenLiber()**,

- **returnarea sumei coeficient** este realizata de o metoda fara parametri, **sumaCoeficienti()**,

- **calculul valorii polinomului** este realizat de o **metoda** fara parametri, care returneaza o valoare de tip **int**, **valoarePolinom()**.

Metoda **main()** a clasei **UtilizarePolinom3.java** contine **scenariul principal** (obtinere valori, calcule, afisare rezultate).

```
 1    import javax.swing.JOptionPane;
 2
 3    public class UtilizarePolinom3 {
 4      public static void main(String[] args) {
 5        System.out.println("Programul UtilizarePolinom3 a fost lansat...");
 6
 7        // Constructor - initializeaza elementele polinomului
 8        Polinom3 pol = new Polinom3();
 9
10        // Apelul metodei care afiseaza polinomul
11        pol.afisarePolinom();
12
13        // Obtinerea unei valori a necunoscutei (X)
14        int X = Integer.parseInt(JOptionPane.showInputDialog(
15                                    "Introduceti valoarea necunoscutei"));
16        // Afisarea valorii necunoscutei (X)
17        System.out.println("X = " + X);
18
19        // Apelul metodei care calculeaza polinomul pentru necunoscuta X
20        int P_X = pol.valoarePolinom(X);
21
22        // Afisarea valorii P(X)
23        System.out.println("P(" + X + ") = " + P_X);
24
25        // Apelul metodei care calculeaza polinomul pentru necunoscuta = 0
26        int P_0 = pol.termenLiber();
27
28        // Afisarea valorii P(0)
29        System.out.println("P(0) = " + P_0);
30
31        // Apelul metodei care calculeaza polinomul pentru necunoscuta = 1
32        int P_1 = pol.sumaCoeficienti();
33
34        // Afisarea valorii P(1)
35        System.out.println("P(1) = " + P_1);
36
37        System.exit(0);      // Inchiderea interfetei grafice
38      }
39    }
```

```
 1    import javax.swing.JOptionPane;
 2    public class Polinom3 {
```

```
3       protected int N;         // gradul polinomului
4       protected int[] C;       // coeficientii polinomului
5
6       // Constructor - initializeaza elementele polinomului
7       public Polinom3() {
8         // Obtinerea gradului polinomului (N)
9         N = Integer.parseInt(JOptionPane.showInputDialog(
10                                  "Introduceti gradul polinomului"));
11        // Crearea tabloului coeficientilor (de dimensiune N+1)
12        C = new int[N+1];
13
14        // Obtinerea coeficientilor Ci, unde i=0,N
15        for (int i=0; i<=N; i++) {
16          C[i] = Integer.parseInt(JOptionPane.showInputDialog(
17                                       "Coeficientul de grad " + i));
18        System.out.println("A fost creat un obiect Polinom...");
19        }
20      }
21      // Metoda care afiseaza polinomul
22      public void afisarePolinom() {
23        // Termenul liber Co
24        System.out.print("P(X) = " + C[0]);
25
26        // Termenii Ci*X^i, unde i=1,N
27        for (int i=1; i<=N; i++) {
28          System.out.print(" + " + C[i]  + "*X^" + i);
29        }
30        System.out.println();
31      }
32      // Metoda care calculeaza valoarea polinomului pentru o necunoscuta
33      public int valoarePolinom(int X) {
34        int P_X = 0;
35        int X_i = 1;
36
37        // Termenii +Ci*X^i, unde 0=1,N
38        for (int i=0; i<=N; i++) {
39          P_X = P_X + C[i]*X_i;
40          X_i = X_i * X;
41        }
42        return (P_X);
43      }
44      // Metoda care calculeaza valoarea polinomului pentru necunoscuta = 0
45      public int termenLiber() {
46        return (C[0]);
47      }
48      // Metoda care calculeaza valoarea polinomului pentru necunoscuta = 1
49      public int sumaCoeficienti() {
50        int S_C = C[0];
51        for (int i=1; i<=N; i++) {
52          S_C = S_C + C[i];
53        }
54        return (S_C);
55      }
56      // Metoda care returneaza tabloul coeficientilor polinomului
57      public int[] coeficienti() {
58        int[] coef = new int[N+1];
59        System.arraycopy(C, 0, coef, 0, N+1);
60        return (coef);
61      }
62      // Metoda care returneaza coeficientii polinomului sub forma de String
63      public String toString() {
64        String polinom = "" + C[0];
65
66        // Termenii Ci*X^i, unde i=1,N
67        for (int i=1; i<=N; i++) {
68          polinom = polinom + " + " + C[i]  + "*X^" + i;
69        }
70        return (polinom);
71      }
72    }
```

IO.4.3. Program de calcul al unui polinom (varianta de lucru cu fisiere)

Programul **Polinom4.java** reprezinta o varianta a programului **Polinom3.java** in care **initializarea atributelor** este realizata de:

- un **constructor cu parametru de tip `String`**, care obtine gradul polinomului si valorile coeficientilor de la utilizator, initializeaza atributele cu aceste valori, si apoi **le scrie intr-un fisier pe disc** (al carui nume este pasat ca parametru),
- un **constructor cu un parametru de tip `String` si un parametru de tip `int[]`**, care foloseste tabloul primit pentru a initializa valorile atributelor, apoi **le scrie intr-un fisier pe disc** (al carui nume este pasat ca parametru),
- un **constructor cu doi parametri de tip `String`**, care formeaza din ei numele unui **fisier de pe disc, din care citeste** valorile coeficientilor si cu ele initializeaza atributele.

Metoda **`main()`** a clasei **Utilizare1Polinom4** contine **un scenariu de utilizare a clasei `Polinom4`** bazat pe apelul **primului constructor (cel care obtine valorile coeficientilor de la utilizator). (script pentru lansare)**

```
 1    import javax.swing.JOptionPane;
 2    import java.io.*;
 3
 4    public class Utilizare1Polinom4 {
 5      public static void main(String[] args) throws IOException {
 6        System.out.println("Programul Utilizare1Polinom4 a fost lansat...");
 7
 8        // Apelul constructorului care initializeaza elementele polinomului
 9        // cu valori obtinute de la utilizator, si le scrie in fisier
10        Polinom4 pol = new Polinom4("coef.txt");
11
12        // Apelul metodei care afiseaza polinomul
13        pol.afisarePolinom();
14
15        // Obtinerea unei valori a necunoscutei (X)
16        int X = Integer.parseInt(JOptionPane.showInputDialog(
17                                      "Introduceti valoarea necunoscutei"));
18
19        // Afisarea valorii necunoscutei (X)
20        System.out.println("X = " + X);
21
22        // Apelul metodei care calculeaza polinomul pentru necunoscuta X
23        int P_X = pol.valoarePolinom(X);
24
25        // Afisarea valorii P(X)
26        System.out.println("P(" + X + ") = " + P_X);
27
28        // Apelul metodei care calculeaza polinomul pentru necunoscuta = 0
29        int P_0 = pol.termenLiber();
30
31        // Afisarea valorii P(0)
32        System.out.println("P(0) = " + P_0);
33
34        // Apelul metodei care calculeaza polinomul pentru necunoscuta = 1
35        int P_1 = pol.sumaCoeficienti();
36
37        // Afisarea valorii P(1)
38        System.out.println("P(1) = " + P_1);
39
40        System.exit(0);      // Inchiderea interfetei grafice
41      }
42    }
```

Primul constructor asigura:
- obtinerea valorii atributelor de la utilizator,
- persistenta atributelor (scrierea lor pe disc).

Al doilea constructor asigura:
- obtinerea valorii atributelor de la un alt obiect,
- persistenta atributelor (scrierea lor pe disc).

Al treilea constructor asigura:
- obtinerea valorii atributelor persistente de pe disc.

Cele trei variante pot fi utile in cazul sistemelor client-server in care se doreste realizarea persistentei datelor (pe disc) la server.

Metoda **`main()`** a clasei **`Utilizare2Polinom4`** contine **un scenariu de utilizare a clasei `Polinom4`** bazat pe apelul celui de-**al treilea constructor (cel care obtine valorile dintr-un fisier). (script pentru lansare)**

```
1    import javax.swing.JOptionPane;
2    import java.io.*;
3
4    public class Utilizare2Polinom4 {
5      public static void main(String[] args) throws IOException {
6        System.out.println("Programul Utilizare2Polinom4 a fost lansat...");
7
8        // Apelul constructorului care initializeaza elementele polinomului
9        // cu valori citite din fisier
10       Polinom4 pol = new Polinom4("coef", "txt");
11
12       // Apelul metodei care afiseaza polinomul
13       pol.afisarePolinom();
14
15       // Obtinerea unei valori a necunoscutei (X)
16       int X = Integer.parseInt(JOptionPane.showInputDialog(
17                                      "Introduceti valoarea necunoscutei"));
18
19       // Afisarea valorii necunoscutei (X)
20       System.out.println("X = " + X);
21
22       // Apelul metodei care calculeaza polinomul pentru necunoscuta X
23       int P_X = pol.valoarePolinom(X);
24
25       // Afisarea valorii P(X)
26       System.out.println("P(" + X + ") = " + P_X);
27
28       // Apelul metodei care calculeaza polinomul pentru necunoscuta = 0
29       int P_0 = pol.termenLiber();
30
31       // Afisarea valorii P(0)
32       System.out.println("P(0) = " + P_0);
33
34       // Apelul metodei care calculeaza polinomul pentru necunoscuta = 1
35       int P_1 = pol.sumaCoeficienti();
36
37       // Afisarea valorii P(1)
38       System.out.println("P(1) = " + P_1);
39
40       System.exit(0);      // Inchiderea interfetei grafice
41     }
42   }
```

```
1    import javax.swing.JOptionPane;
2    import java.io.*;
```

```
 3
 4   public class Polinom4 {
 5     protected int N;         // gradul polinomului
 6     protected int[] C;       // coeficientii polinomului
 7
 8     // Constructor - initializeaza elementele polinomului
 9     // cu valori obtinute de la utilizator, si le scrie in fisier
10     public Polinom4(String numeFisier) throws IOException {
11
12       PrintWriter outFisier = new PrintWriter(new
13                           BufferedWriter(new FileWriter(numeFisier)));
14
15       // Obtinerea gradului polinomului (N)
16       N = Integer.parseInt(JOptionPane.showInputDialog(
17                                  "Introduceti gradul polinomului"));
18
19       // Stocarea numarului de coeficienti
20       outFisier.println(N+1);
21
22       // Crearea tabloului coeficientilor (de dimensiune N+1)
23       C = new int[N+1];
24
25       // Obtinerea coeficientilor Ci, unde i=0,N
26       for (int i=0; i<=N; i++) {
27         C[i] = Integer.parseInt(JOptionPane.showInputDialog(
28                                       "Coeficientul de grad " + i));
29         // Stocarea coeficientilor
30         outFisier.println(C[i]);
31       }
32       outFisier.close();
33
34       System.out.println("A fost creat un obiect Polinom...");
35     }
36
37     // Constructor - initializeaza elementele polinomului
38     // cu valori obtinute ca parametru, si le scrie in fisier
39     public Polinom4(String numeFisier, int[] coeficienti) throws IOException {
40
41       PrintWriter outFisier = new PrintWriter(new
42                           BufferedWriter(new FileWriter(numeFisier)));
43
44       // Obtinerea gradului polinomului (N)
45       N = coeficienti.length - 1;
46
47       // Stocarea numarului de coeficienti
48       outFisier.println(N+1);
49
50       // Crearea tabloului coeficientilor (de dimensiune N+1)
51       C = new int[N+1];
52
53       // Obtinerea coeficientilor Ci, unde i=0,N
54       for (int i=0; i<=N; i++) {
55         C[i] = coeficienti[i];
56
57         // Stocarea coeficientilor
58         outFisier.println(C[i]);
59       }
60       outFisier.close();
61
62       System.out.println("A fost creat un obiect Polinom...");
63     }
64
65     // Constructor - initializeaza elementele polinomului
66     // cu valori citite din fisier
67     public Polinom4(String nume, String extensie) throws IOException {
68
69       String numeFisier = nume + "." + extensie;
70
71       BufferedReader inFisier = new BufferedReader(new FileReader(numeFisier));
72
73       // Obtinerea gradului polinomului (N)
74       N = Integer.parseInt(inFisier.readLine()) - 1;
75
76       // Crearea tabloului coeficientilor (de dimensiune N+1)
77       C = new int[N+1];
```

```
78
79        // Obtinerea coeficientilor Ci, unde i=0,N
80        for (int i=0; i<=N; i++) {
81          C[i] = Integer.parseInt(inFisier.readLine());
82        }
83        inFisier.close();
84
85        System.out.println("A fost creat un obiect Polinom...");
86      }
87
88      // Metoda care afiseaza polinomul
89      public void afisarePolinom() {
90
91        // Termenul liber Co
92        System.out.print("P(X) = " + C[0]);
93
94        // Termenii Ci*X^i, unde i=1,N
95        for (int i=1; i<=N; i++) {
96          System.out.print(" + " + C[i]  + "*X^" + i);
97        }
98        System.out.println();
99      }
00
01      // Metoda care calculeaza valoarea polinomului pentru o necunoscuta
02      public int valoarePolinom(int X) {
03        int P_X = 0;
04        int X_i = 1;
05
06        // Termenii +Ci*X^i, unde 0=1,N
07        for (int i=0; i<=N; i++) {
08          P_X = P_X + C[i]*X_i;
09          X_i = X_i * X;
10        }
11        return (P_X);
12      }
13
14      // Metoda care calculeaza valoarea polinomului pentru necunoscuta = 0
15      public int termenLiber() {
16        return (C[0]);
17      }
18
19      // Metoda care calculeaza valoarea polinomului pentru necunoscuta = 1
20      public int sumaCoeficienti() {
21        int S_C = C[0];
22        for (int i=1; i<=N; i++) {
23          S_C = S_C + C[i];
24        }
25        return (S_C);
26      }
27
28      // Metoda care returneaza tabloul coeficientilor polinomului
29      public int[] coeficienti() {
30        int[] coef = new int[N+1];
31        System.arraycopy(C, 0, coef, 0, N+1);
32        return (coef);
33      }
34
35      // Metoda care returneaza coeficientii polinomului sub forma de String
36      public String toString() {
37        String polinom = "" + C[0];
38
39        // Termenii Ci*X^i, unde i=1,N
40        for (int i=1; i<=N; i++) {
41          polinom = polinom + " + " + C[i]  + "*X^" + i;
42        }
43        return (polinom);
44      }
45    }
```