Fluxuri

Programare Orientată pe Obiecte

Fluxuri

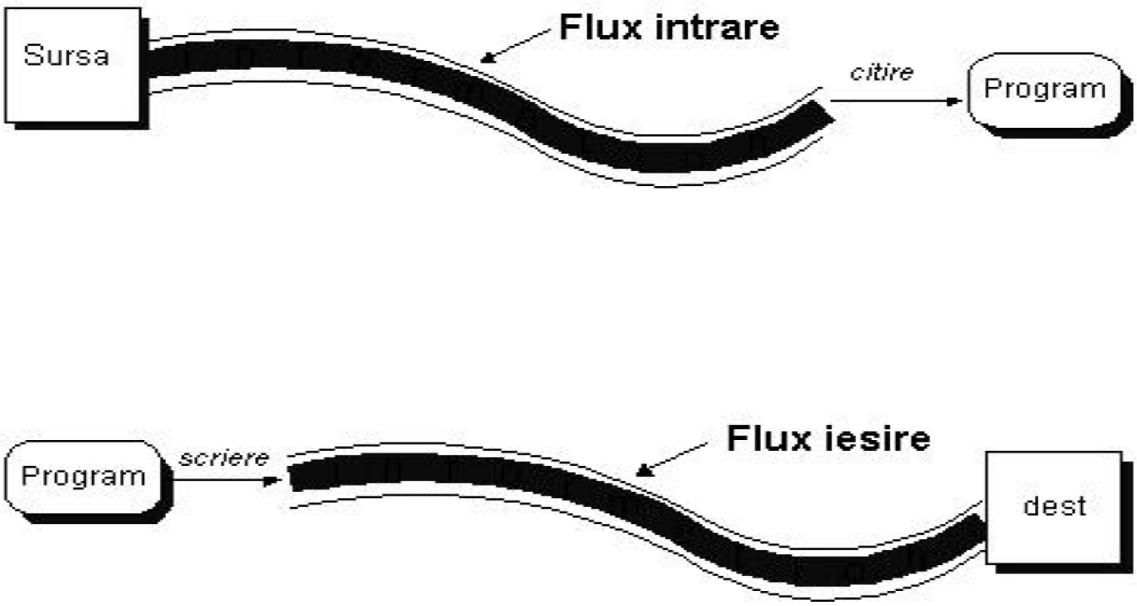
- Ce sunt fluxurile ?
- Clasificare, ierarhie
- Fluxuri primitive
- Fluxuri de procesare
- Intrări şi ieşiri formatate
- Fluxuri standard de intrare şi ieşire
- Analiza lexicală
- Clase independente
  (RandomAccessFile, File)

Ce sunt fluxurile? (1)

- Schimb de informații cu mediul extern
- Canal de comunicație între două procese

- Seriale, pe 8 sau 16 biți
- Producător: proces care descrie o sursă externă de date
- Consumator: proces care descrie o destinație externă pentru date
- Unidirecționale, de la producător la consumator
- Un singur proces producător
- Un singur proces consumator

Ce sunt fluxurile? (2)



Un flux care citeşte date se numeşte flux de intrare.
Un flux care scrie date se numeşte flux de ieşire.

Ce sunt fluxurile? (3)

- Pachetul care oferă suport pentru operațiile de intrare/ieşire: java.io

- Schema generală de utilizare a fluxurilor:

```
deschide canal comunicatie
while (mai sunt informatii) {
   citeste/scrie informatie;
}
inchide canal comunicatie;
```

Clasificarea fluxurilor

- După direcția canalului de comunicație:
  - fluxuri de intrare (citire)
  - fluxuri de ieşire (scriere)
- După tipul de date:
  - fluxuri de octeți (8 biți)
  - fluxuri de caractere (16 biți)
- După acțiunea lor:
  - fluxuri primare (se ocupă efectiv cu citirea/scrierea datelor)
  - fluxuri pentru procesare

Ierarhia claselor pentru lucrul cu fluxuri

Caractere:

- Reader
  - FileReader, BufferedReader,...
- Writer
  - FileWriter, BufferedWriter,...

Octeți:

- InputStream
  - FileInputStream, BufferedInputStream..
- OutputStream
  - FileOutputStream, BufferedOutputStream..

Metode comune fluxurilor

- Metodele comune sunt definite în superclasele abstracte corespunzătoare:

Reader	**Writer**
int read()	void write()
int read(char buf[])	void write(char buf[])
…	...

InputStream	**OutputStream**
int read()	void write()
int read(byte buf[])	void write(byte buf[])
	void write(String str)

- Inchiderea oricărui flux: close.
- Excepții: IOException sau derivate.

Fluxuri primitive

In funcție de tipul sursei datelor:

- Fişier
  - FileReader, FileWriter,
  - FileInputStream, FileOutputStream
- Memorie
  - CharArrayReader, CharArrayWriter,
  - ByteArrayInputStream,
  - ByteArrayOutputStream,
  - StringReader, StringWriter
- Pipe
  - PipedReader, PipedWriter,
  - PipedInputStream, PipedOutputStream
  - folosite pentru a canaliza ieşirea unui program sau fir de execuție către intrarea altui program sau fir de execuție.

Fluxuri de procesare (1)

- responsabile cu preluarea datelor de la un flux primitiv şi procesarea acestora pentru a le oferi într-o altă formă, mai utilă dintr-un anumit punct de vedere.

- ”Bufferizare”
  - BufferedReader, BufferedWriter
  - BufferedInputStream, BufferedOutputStream
- Conversie octeți-caractere/caractere-octeți
  - InputStreamReader
  - OutputStreamWriter
- Concatenare
  - SequenceInputStream
- Serializare
  - ObjectInputStream, ObjectOutputStream

Fluxuri de procesare (2)

- Conversie tipuri de date de tip primitiv într-un format binar, independent de maşina pe care se lucrează
  - DataInputStream, DataOutputStream
- Numărare
  - LineNumberReader, LineNumberInputStream
- Citire în avans
  - PushbackReader, PushbackInputStream
- Afişare
  - PrintWriter, PrintStream

Crearea unui flux primitiv

FluxPrimitiv numeFlux =new FluxPrimitiv (dispozitivExtern);

- **crearea unui flux de intrare pe caractere**

FileReader in = new FileReader("fisier.txt");

- **crearea unui flux de iesire pe caractere**

FileWriter out = new FileWriter("fisier.txt");

- **crearea unui flux de intrare pe octeti**

FileInputStream in =new FileInputStream("fisier.dat");

- **crearea unui flux de iesire pe octeti**

FileOutputStrem out =new FileOutputStream("fisier.dat");

Crearea unui flux de procesare

FluxProcesare numeFlux = new FluxProcesare(fluxPrimitiv);

- **crearea unui flux de intrare printr-un buffer**

```
BufferedReader in = new BufferedReader(new FileReader("fisier.txt"));

//echivalent cu:
FileReader fr = new FileReader("fisier.txt");
BufferedReader in = new BufferedReader(fr);
```

- **crearea unui flux de iesire printr-un buffer**

```
BufferedWriter out = new BufferedWriter(new FileWriter("fisier.txt")));

//echivalent cu:
FileWriter fo = new FileWriter("fisier.txt");
BufferedWriter out = new BufferedWriter(fo);
```

Fluxuri pentru lucrul cu fişiere

FileReader, FileWriter - caractere

FileInputStream, FileOutputStream - octeti

Listing 1: Copierea unui fisier

```
import java.io.*;
public class Copiere {
  public static void main(String[] args)  {

    try {
      FileReader in  = new FileReader("in.txt");
      FileWriter out = new FileWriter("out.txt");

      int c;
      while ((c = in.read()) != -1)
        out.write(c);

      in.close();
      out.close();

    } catch(IOException e) {
      System.err.println("Eroare la operatiile cu fisiere!");
      e.printStackTrace();
    }
  }
}
```

Citirea şi scrierea cu buffer

- BufferedReader, BufferedWriter
- BufferedInputStream, BufferedOutputStream

- Introduc un buffer (zonă de memorie) în procesul de citire/scriere a informațiilor.

- BufferedOutputStream out = new BufferedOutputStream (new FileOutputStream("out.dat"), 1024);

  //1024 este dimensiunea bufferului

- Scopul: eficiența
  - Scade numărul de accesări ale dispozitivului extern
  - Creşte viteza de execuție

Metoda flush

- goleşte explicit bufferul

```
BufferedWriter out = new BufferedWriter(new
    FileWriter("out.dat"), 1024);
     //am creat un flux cu buffer de 1024 octeti
for(int i=0; i<1000; i++)   out.write(i);
     //bufferul nu este plin, in fisier nu s-a scris nimic
out.flush();
     //bufferul este golit, datele se scriu in fisier
```

Metoda readLine

```
BufferedReader br = new
  BufferedReader(new FileReader("in"));
String linie;
while ((linie = br.readLine()) != null) {
        ...
        //proceseaza linie
 }
br.close();
```

DataInputStream şi DataOutputStream

- un flux nu mai este văzut ca o însiruire de octeți, ci de date primitive.
- scrierea datelor se face în format binar, ceea ce înseamnă că un fişier în care au fost scrise informații folosind metode writeX nu va putea fi citit decât prin metode readX.
- transformarea unei valori în format binar - serializare.
- permit serializarea tipurilor primitive şi a şirurilor de caractere.

DataInputStream	DataOutputStream
readBoolean	writeBoolean
readByte	writeByte
readChar	writeChar
readDouble	writeDouble
readFloat	writeFloat
readInt	writeInt
readLong	writeLong
readShort	writeShort
readUTF	writeUTF

Intrări formatate: java.util.Scanner

```
Scanner s=new Scanner(System.in);

String nume = s.next();
int varsta = s.nextInt();
double salariu = s.nextDouble();

s.close();
```

Ieşiri formatate

PrintStream şi PrintWriter :

- print, println
- format, printf

```
System.out.printf("%s %8.2f %2d %n", nume,
    salariu, varsta);
```

- Formatarea şirurilor de caractere se bazează pe clasa java.util.Formatter.

Fluxuri standard de intrare şi ieşire

- System.in - InputStream
- System.out - PrintStream
- System.err - PrintStream

- Afişarea informațiilor pe ecran:

```
System.out.print (argument);
System.out.println(argument);
System.out.printf (format, argumente...);
System.out.format (format, argumente...);
```

- Afişarea erorilor:

```
catch(Exception e) {
System.err.println("Exceptie:" + e);
}
```

Citirea datelor de la tastatură (1)

- Clasa BufferedReader

```
BufferedReader stdin = new BufferedReader(
    new InputStreamReader(System.in));
System.out.print("Introduceti o linie:");
String linie = stdin.readLine()
System.out.println(linie);
```

- Clasa DataInputStream

```
DataInputStream stdin = new DataInputStream(
    System.in);
System.out.print("Introduceti o linie:");
String linie = stdin.readLine()
System.out.println(linie);
```

- Clasa java.util.Scanner (1.5)

```
Scanner s=Scanner.create(System.in);
```

Citirea datelor de la tastatură (2)

- Redirectarea fluxurilor standard: stabilirea unei alte surse decât tastatura pentru citirea datelor, respectiv alte destinații decât ecranul pentru cele două fluxuri de ieşire.

- setIn(InputStream) - redirectare intrare
- setOut(PrintStream) - redirectare ieşire
- setErr(PrintStream) - redirectare erori

```
PrintStream fis = new PrintStream( new
    FileOutputStream("rezultate.txt")));
System.setOut(fis);

PrintStream fis = new PrintStream(new
    FileOutputStream("erori.txt")));
System.setErr(fis);
```

Analiza lexicală pe fluxuri: StreamTokenizer

Listing 2: Citirea unor atomi lexicali dintr-un fisier

```
/* Citirea unei secvente de numere si siruri
   dintr-un fisier specificat
   si afisarea tipului si valorii lor
*/

import java.io.*;
public class CitireAtomi {
  public static void  main(String args[]) throws IOException{

    BufferedReader br = new BufferedReader(new FileReader("
       fisier.txt"));
    StreamTokenizer st = new StreamTokenizer(br);

    int tip = st.nextToken();
    //Se citeste primul atom lexical

    while (tip != StreamTokenizer.TT_EOF) {
        switch (tip) {
          case StreamTokenizer.TT_WORD:
            System.out.println("Cuvant: " + st.sval);
            break;
          case StreamTokenizer.TT_NUMBER:
            System.out.println("Numar: " + st.nval);
        }

        tip = st.nextToken();
        //Trecem la urmatorul atom
    }
  }
}
```

Clasa RandomAccessFile

- permite accesul nesecvențial (direct) la conținutul unui fişier;
- este o clasă de sine stătătoare, subclasă directă a clasei Object;
- se găseşte în pachetul java.io;
- oferă metode de tipul readX, writeX;
- permite atât citirea cât şi scriere din/în fişiere cu acces direct;
- permite specificarea modului de acces al unui fişier (read-only, read-write).

```
RandomAccessFile f1 = new
   RandomAccessFile("fisier.txt", "r");
//deschide un fisier pentru citire

RandomAccessFile f2 =new
   RandomAccessFile("fisier.txt", "rw");
//deschide un fisier pentru scriere si citire
```

Clasa RandomAccessFile (2)

- Program pentru afişarea pe ecran a liniilor dintr-un fişier text, fiecare linie precedată de numărul liniei şi de un spațiu.

```
import java.io.*;
class A{
  public static void main(String arg[]) throws
    IOException{
    RandomAccessFile raf=new RandomAccessFile(
                                    arg[0],"r");
    String s;
    int i=1;
    while( (s=raf.readLine())!=null) {
          System.out.println(i+" "+s);
          i++;
    }
    raf.close();
  }
}
```

Clasa File (1)

- **Clasa File nu se referă doar la un fişier ci poate reprezenta fie un fişier anume, fie mulțimea fişierelor dintr-un director.**
- **Utilitatea clasei File constă în furnizarea unei modalități de a abstractiza dependențele căilor şi numelor fişierelor față de maşina gazdă**
- **Oferă metode pentru testarea existenței, ştergerea, redenumirea unui fişier sau director, crearea unui director, listarea fişierelor dintr-un director, etc.**
- **Constructorii fluxurilor pentru fişiere acceptă ca argument obiecte de tip File:**

  File f = new File("fisier.txt");

  FileInputStream in = new FileInputStream(f);

- **String[ ] list()**
- **File[ ] listFiles()**
- **String getAbsolutePath()**

Clasa File (2)

- **Listare fişiere dintr-un director dat, cu indentare, recursiv, în subdirectoare**

```
import java.io.*;
import java.util.*;
class Dir{
  public void dirlist (File d, String sp) throws
    IOException {
    String [] files=d.list();
         //lista numelor din obiectul d
    if (files ==null ) return;
    String path =d.getAbsolutePath();
         //calea completa spre obiectul d
```

Clasa File (3)

```
    for(int i=0;i<files.length;i++){
      File f = new File(d+"\\"+files[i]);
      if (f.isDirectory()) {
        System.out.println (sp+path+"\\"+files[i]);
        dirlist (f,sp+"  ");
      }
      else System.out.println (sp+ files[i]);
    }
  }
  public static void main (String arg[]) throws
    IOException {
      String dname =".";
      if (arg.length ==1)   dname=arg[0];
      Dir d= new Dir();
       d.dirlist(new File(dname)," ");
  }
}
```