Programare avansată

Fluxuri de date

Probleme

- Cum citesc/scriu un fișier
  - text, linie cu linie sau caracter cu caracter sau ...?
  - binar, octet cu octet sau ...?
- Cum trimit/primesc
  - obiecte serializate în rețea?
  - date către/de la un port serial?
- Cum comunică asincron două fire de execuție?
- Cum scriu date voluminoase într-o BD?
- ...

Fluxuri de date

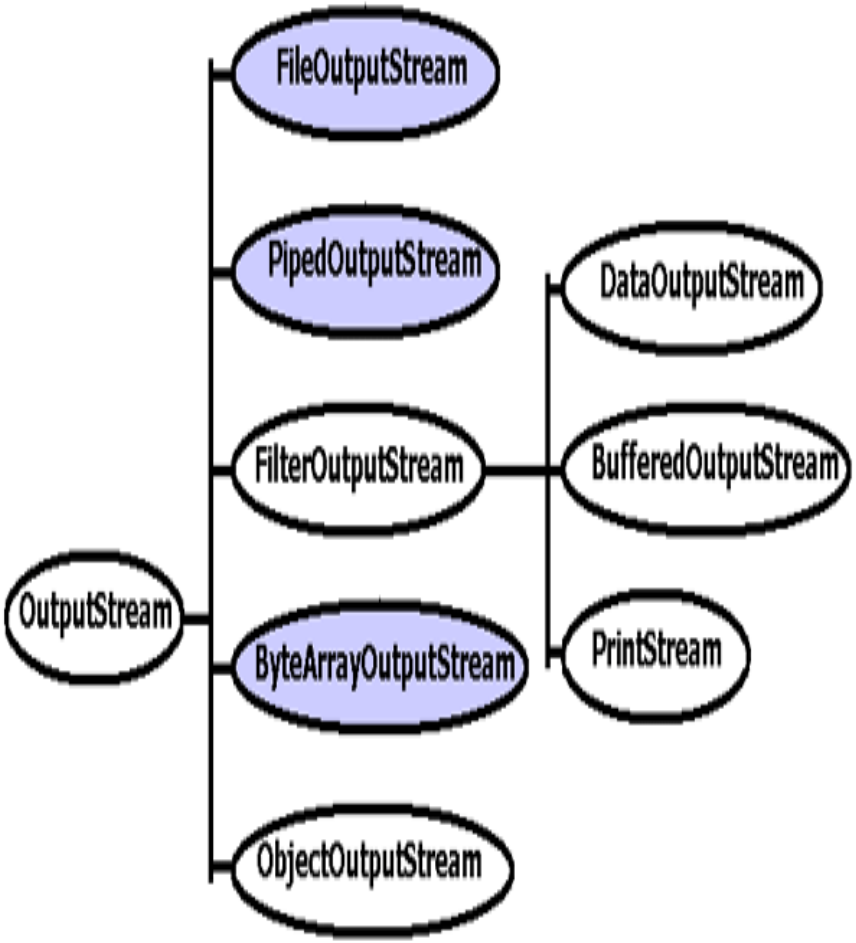
- Canale de comunicație seriale, unidirecționale între două procese
- Pe 8 biți (binare) sau 16 biți (caractere)
- *Producător (scriere)*

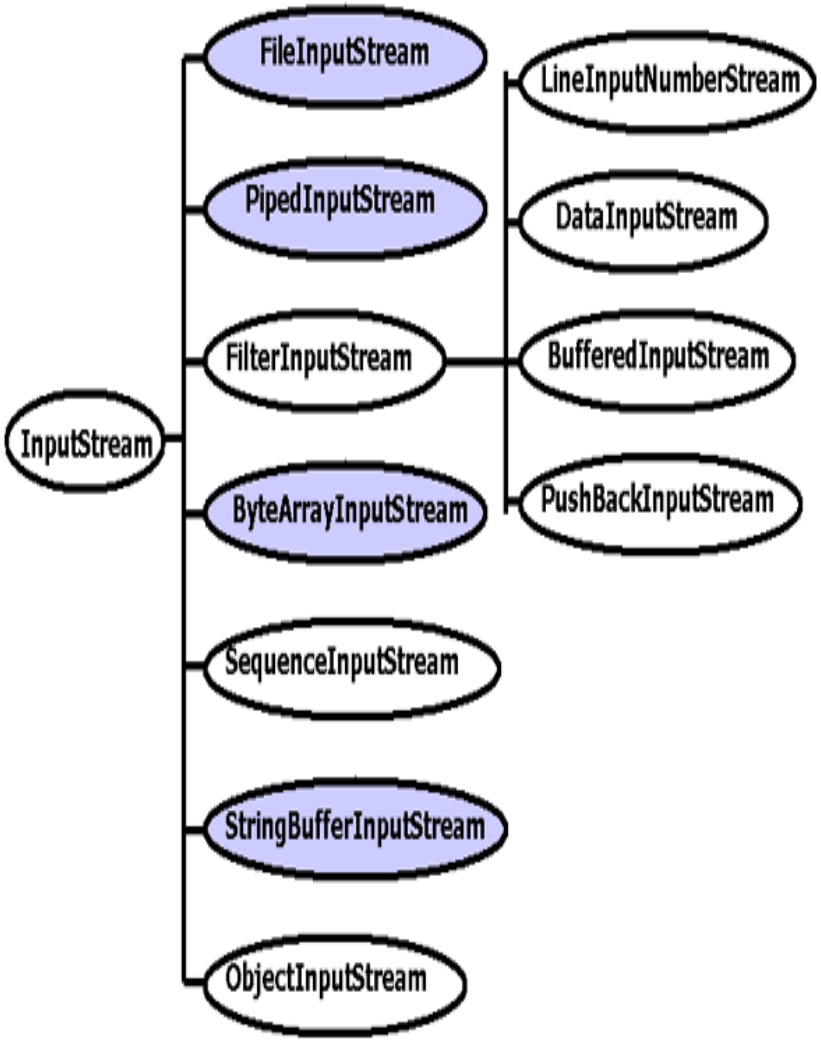
- *Consumator (citire)*

Schema de utilizare a unui flux

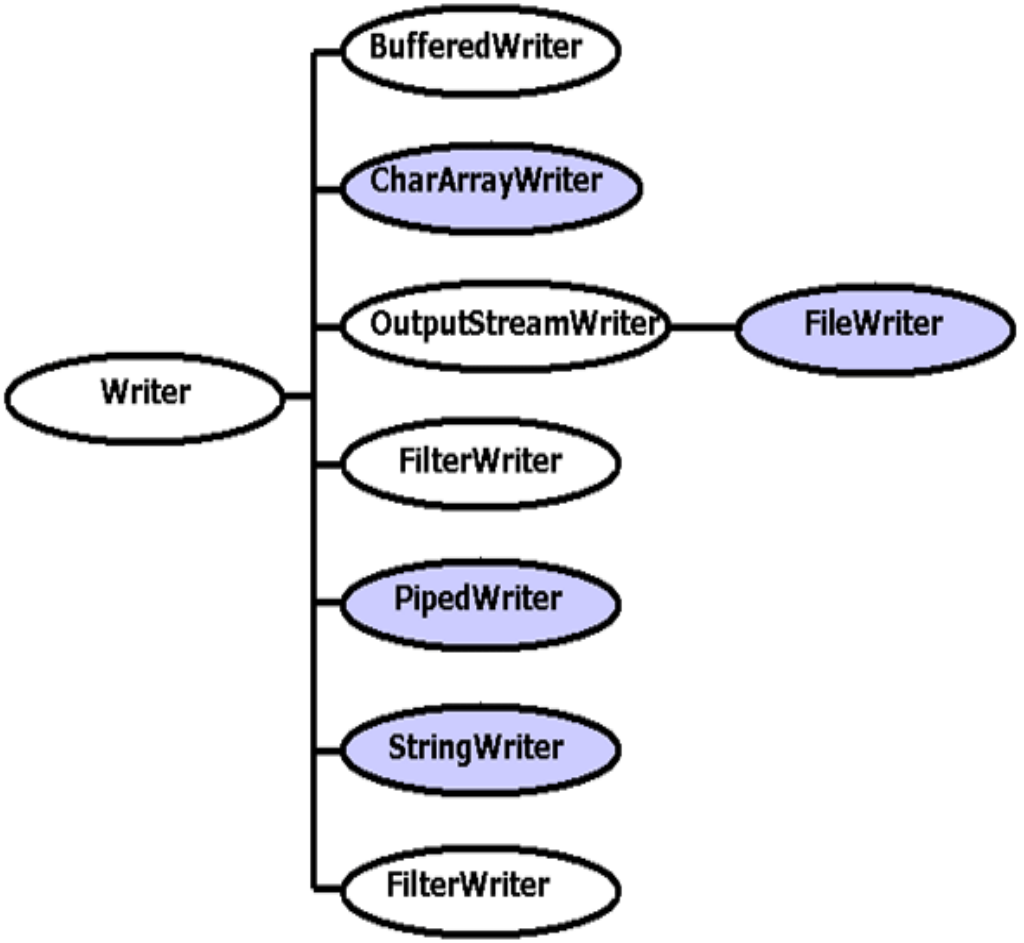
```
import java.io.*;

deschide canal comunicatie; //new ClasaFlux([argumente]);
while (mai sunt informatii de citit sau scris) {
  citeste/scrie informatie;
  //apelare read(), write() sau alte metode specifice
}
inchide canal comunicatie; //apelare close()
trateaza exceptii IOExeption;
```

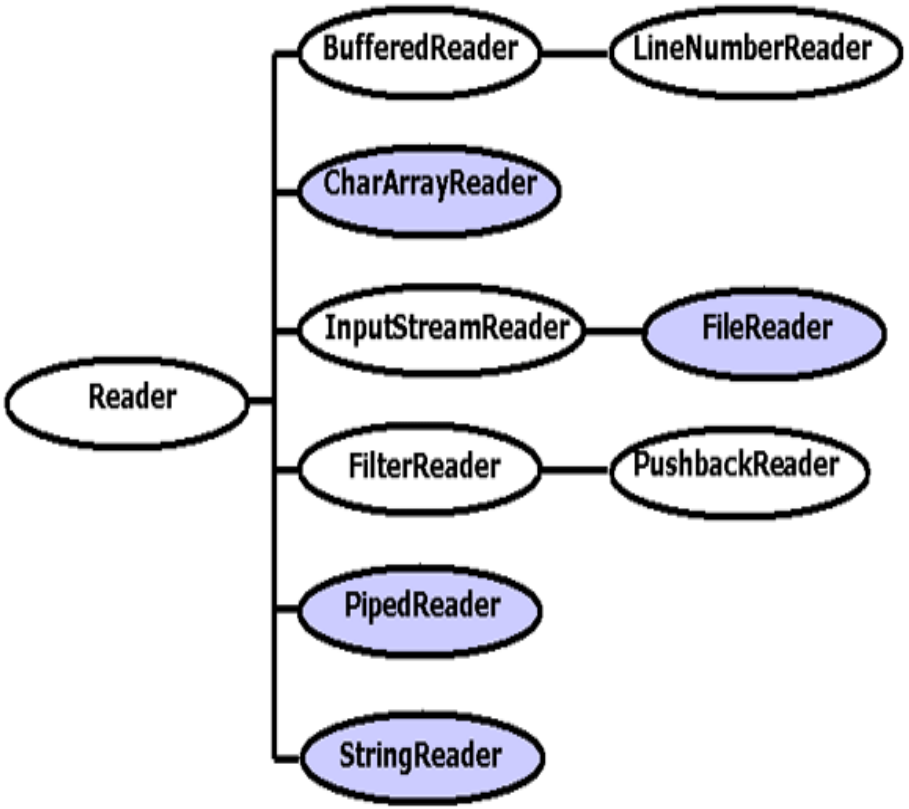
Fluxuri pe octeți





Fluxuri pe caractere





Fluxuri *primitive / filtrare*

- Fluxurile *primitive* “știu” să facă efectiv operațiie de citire/scriere de la/către un “partener” extern (fișier, memorie, fir de execuție, etc.) de exemplu:
  - `FileReader, FileWriter`
  - `ByteArrayInputStream,ByteArrayOutputStream`
  - `PipedReader, PipedWriter`
- Fluxurile de *filtrare (decorare)* “știu” să comunice cu un flux primitv (sau alt flux de filtrare) pentru a procesa și oferi datele într-un mod mai complex, de exemplu:
  - `BufferedReader, BufferedWriter, PrintWriter`
  - `DataInputStream, DataOutputStream`
  - `ObjectInputStream, ObjectOutputStream`

Utilizarea fluxurilor

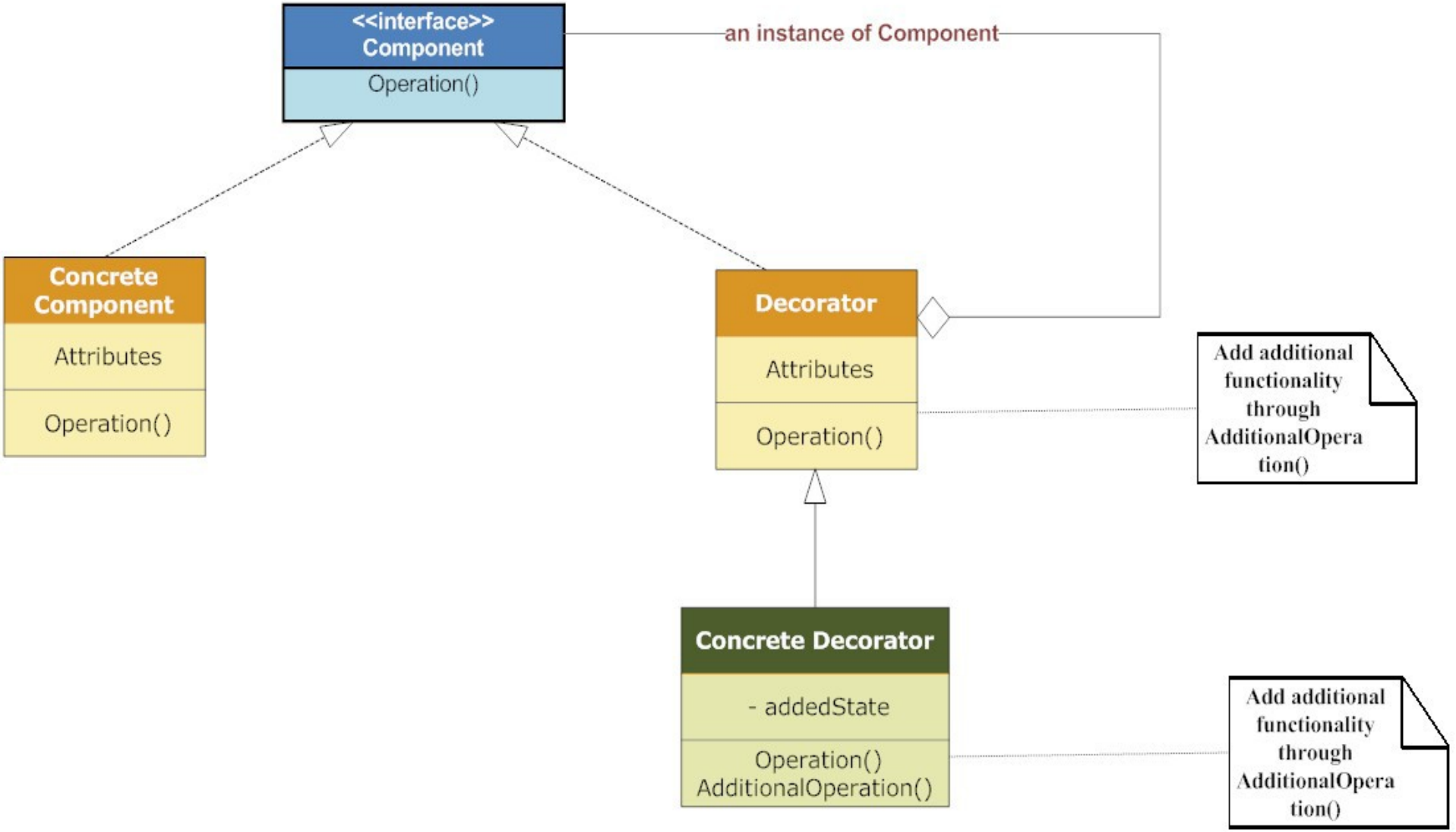
```
FluxPrimitiv fluxPrimitiv =
        new FluxPrimitiv(resursaExterna);
FluxDecorator fluxDecorator =
        new FluxDecorator(fluxPrimitiv);
```

Exemplu:

```
FileReader fileReader = new FileReader("fisier.txt");
BufferedReader bufferReader =
    new BufferedReader(fileReader);
String line = bufferedReader.readLine();
```

Decorator Design Pattern

Decorator Pattern Structure



Citirea și scrierea cu *buffer*

`BufferedReader, BufferedWriter, BufferedInputStream, BufferedOutputStream`

Introduc un buffer (zonă de memorie) în procesul de citire/scriere a informațiilor, cu scopul creșterii eficienței operației respective

```
BufferedOutputStream out = new BufferedOutputStream(
  new FileOutputStream("out.dat"), 1024)
  //1024 este dimensiunea bufferului

for(int i=0; i<100; 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
```

- Scade numărul de accesări ale dispozitivului extern
- Crește viteza de execuție

Serializarea datelor primitive

DataInputStream, DataOutputStream

- Citirea/scrierea datelor primitive în format binar, "*in a machine-independent way*"

```
//Scriere
FileOutputStream fos = new FileOutputStream("test.dat");
DataOutputStream out = new DataOutputStream(fos);
out.writeInt(12345);
out.writeDouble(12.345);
out.writeBoolean(true);
out.writeUTF("Sir de caractere");
out.flush();
fos.close();
...
//Citire
FileInputStream fis = new FileInputStream("test.dat");
DataInputStream in  = new DataInputStream(fis);
int i = in.readInt();
double d = in.readDouble();
boolean b = in.readBoolean();
String s = in.readUTF();
fis.close();
```

Serializarea obiectelor

ObjectInputStream, ObjectOutputStream

- Citirea/scrierea obiectelor în format binar, "*in a machine-independent way*"

```
//Scriere
FileOutputStream fos = new FileOutputStream("test.ser");
ObjectOutputStream out = new ObjectOutputStream(fos);
out.writeObject(new Date());
out.writeObject("Ora curenta:");
out.writeInt(12345);
out.flush();
fos.close();

//Citire
FileInputStream fis = new FileInputStream("test.ser");
ObjectInputStream in = new ObjectInputStream(fis);
Date data = (Date)in.readObject();
String mesaj = (String)in.readObject();
int i = in.readInt();
fis.close();
```

Fluxuri standard

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

Redirectarea fluxurilor standard:

```
setIn(InputStream)   - redirectare intrare
setOut(PrintStream)  - redirectare iesire
setErr(PrintStream)  - redirectare erori
```

Exemplu:

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

Citirea/scrierea formatată a datelor

java.util.Scanner,

java.util.Formatter, java.text.Format

```
Scanner s = Scanner.create(System.in);
String nume = s.next();
int varsta = s.nextInt();
double salariu = s.nextDouble();
s.close();
```

```
System.out.printf("%s %8.2f %n",nume, salariu, varsta);
SimpleDateFormat dateFormat =
  new SimpleDateFormat("dd-MM-yyyy");
String date = dateFormat.format(today);
System.out.println("Today in dd-MM-yyyy format : " + date);
```

Alte clase “utile”

- *java.io.File*

  lucrul cu fișiere și directoare
- *java.io.RandomAccesFile*

  acces nesecvențial (direct) la conținutul unui fișier
- *java.io.StreamTokenizer*

  analiză lexicală pe fluxuri (vezi și *StringTokenizer, String.split*)
- *java.nio.Files*
- *java.nio.Paths*

  acces la sistemul de fișiere (la nivel *low-level*)