Programare multiparadigmă - **JAVA**

Prof. univ. dr. **Claudiu Vințe**

claudiu.vinte@ie.ase.ro

Java Standard *I/O (java.io)*

- Clasele care implementează operațiile de intrare / ieșire sunt grupate în două pachete
  - ***java.io*** **–** Standard I/O: conține clasele de bază pentru intrare / ieșire bazate pe obiecte stream
  - ***java.nio*** **–** Non-blocking I/O: adaugă suport pentru facilități avansate de intrare / ieșire

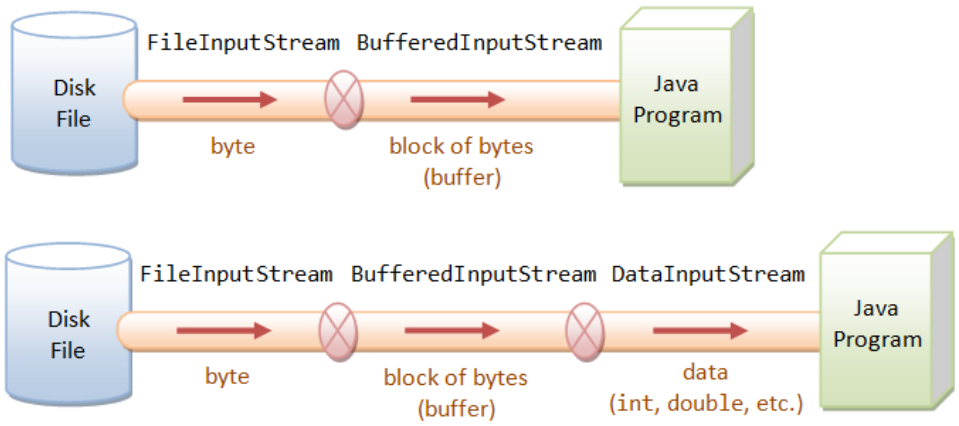
- Principalele funcționalități implementate în ***java.io***:
  - Accesul la sistemul de fișiere (creare, citire proprietăți, verificare existență, **...**) pentru fișiere și directoare
  - Accesul secvențial pentru citire și scriere la conținutul diverselor surse de date
  - Procesarea datelor (decodificare caractere, compresie, **...**)
  - Serializarea și deserializarea obiectelor
  - Acces aleator la datele din fișiere binare

Manipulare fișiere și directoare

- Se realizează prin intermediul obiectelor de tip ***File*** care:
  - Reprezintă o cale către un fișier sau director
  - Permite manipularea fișierelor și directoarelor, dar nu și a conținutului
- Principalele operații disponibile sunt:
  - ***isFile****() /* ***isDirectory****()*: determinarea tipului căii
  - ***exists****()*: determinare existență fișier sau director
  - ***length***(): determinare dimensiune în bytes pentru un fișier
  - *File[]* ***listFiles****()*: obține obiectele descendent (fișiere și directoare); *File* ***getParentFile****()*: obține părinte
  - ***mkDir****() /* ***mkDirs****()* sau ***createNewFile****()*: construire directoare sau fișier gol
  - ***renameTo****(File dest)*: redenumește fișierul sau directorul
  - ***delete****()*: șterge fișierul sau directorul
  - ***canRead****() /* ***canWrite****() /* ***canExecute****()*: verificare permisiuni

Manipularea conținutului

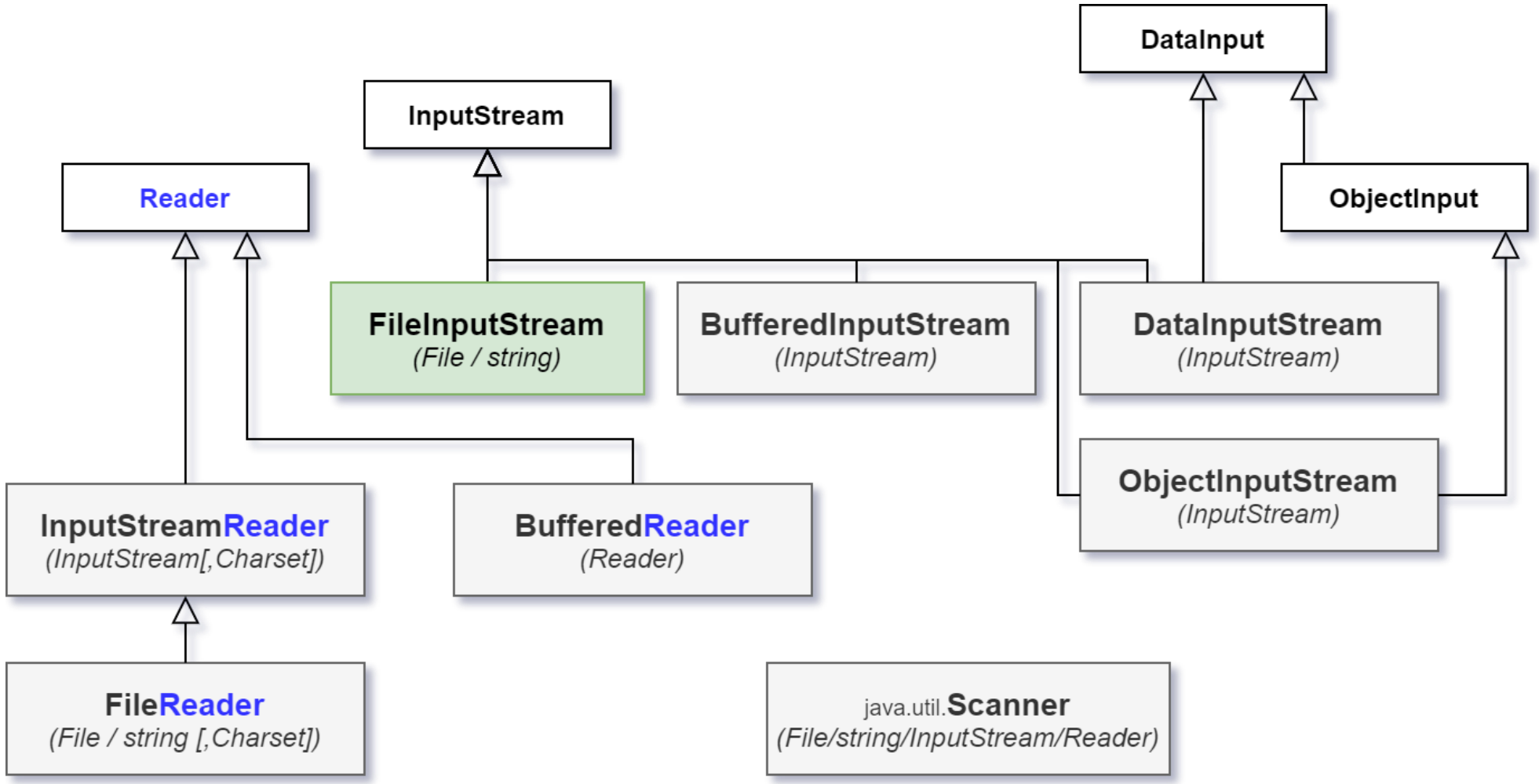
- Se bazează pe obiecte de tip ***stream*** care reprezintă o sursă (*input stream*) sau o destinație (*output stream*) pentru datele manipulate de către program.
- Permit accesul secvențial la date.
- Pot fi înlănțuite – *decorator design pattern.*
- În funcție de tipul de date manipulate se clasifică în *byte streams* și *character streams.*
- Există obiecte stream care permit accesul la o multitudine de surse de date (fișiere, rețea, *memory pipes*, ...).



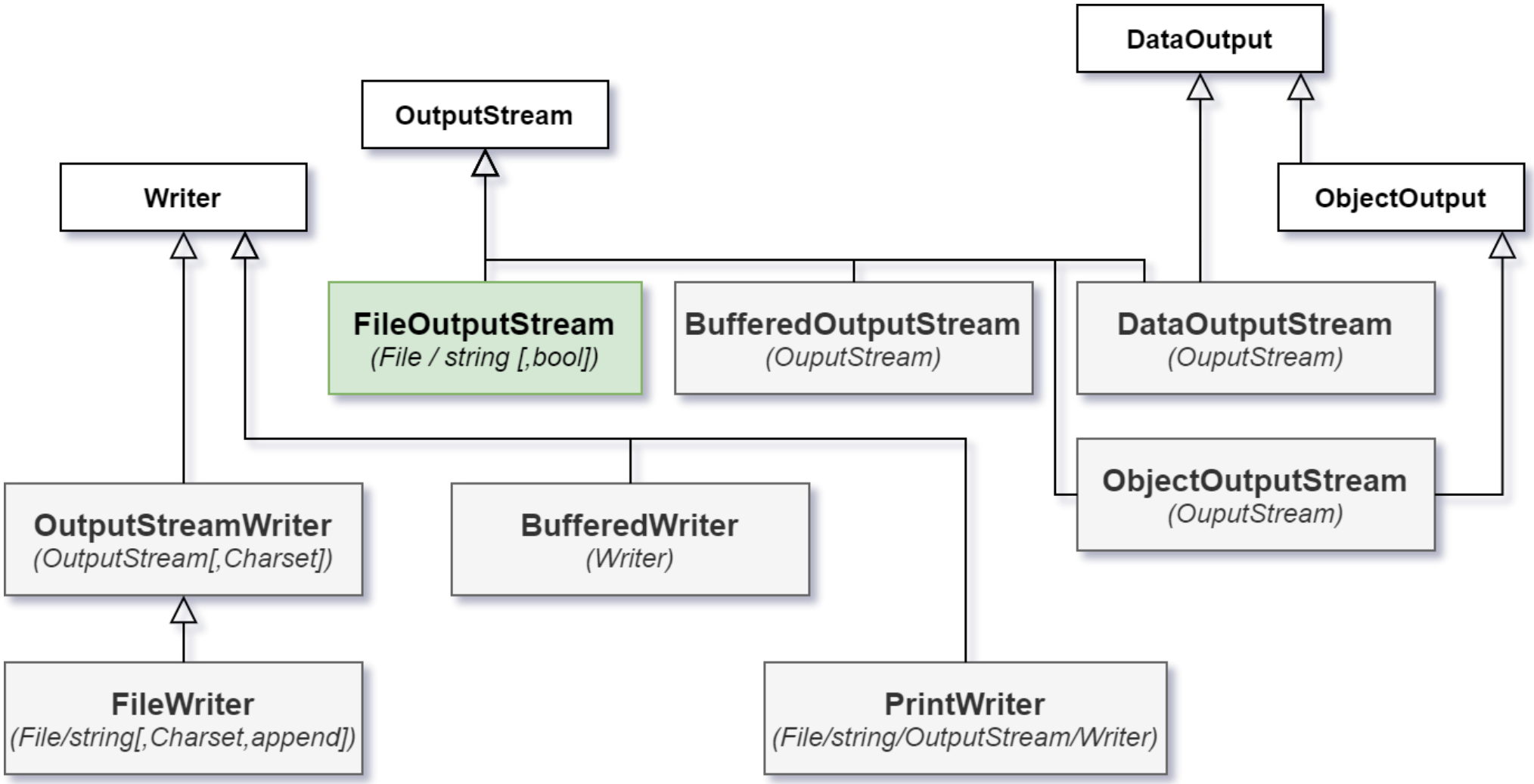
- Pentru că obiectele de tip **stream** gestionează resurse externe se recomandă folosirea construcțiilor de tip ***try-with-resources*** pentru gestionarea acestora.

```
try (FileInputStream stream =
        new FileInputStream("date.txt")) {
    // utilizare obiect stream
} // închidere automată la părăsirea blocului
```

Structura de clase - Citire



Structura de clase - Scriere



Citirea / scrierea la nivel de octet

- Clasa abstractă ***InputStream*** – permite citirea de octeți din sursa de date
- Metoda principală este *public abstract int* ***read()*** *throws IOException* care:
  - Întoarce octetul citit – o valoare de la **0** la **255** în caz de succes
  - Valoarea ***-1*** dacă sursa de date a fost epuizată (de exemplu sfârșit de fișier)
  - Aruncă o excepție în cazul unei erori de citire
  - **Blochează execuția** până la apariția unuia dintre cele trei cazuri de mai sus
- Clasa abstractă ***OutputStream*** – permite scrierea de octeți în destinație
- Implementări concrete pentru fișiere:
  - ***FileInputStream****(File fisier / string cale)* – citire din fișier
  - ***FileOutputStream****(File fisier / string cale[, boolean append])* – scriere în fișier
- ***BufferedInputStream*** și ***BufferedOutputStream*** – decoratori care implementează citirea (dintr-un *InputStream*) și scrierea (într-un *OutputStream*) folosind zone tampon

Citirea / scrierea la nivel de caracter

- Clasa abstractă ***Reader*** – permite citirea de caractere din sursa de date
- Metoda principală este *public abstract int* ***read()*** *throws IOException* care:
  - Funcționează similar cu metoda *read()* din InputStream
  - Întoarce caracterul citit – o valoare de la **0** la **65535** în caz de succes

- Clasa abstractă ***Writer*** – permite scrierea de caractere în destinație

- Legătura se realizează prin intermediul clasei ***InputStreamReader****(InputStream[, Charset])*
- Implementări concrete pentru fișiere:
  - ***FileReader****(File fisier / string cale[, Charset])* – citire din fișier
  - ***FileWriter****(File fisier / string cale[, Charset, boolean append])* – scriere în fișier
- ***BufferedReader*** și ***BufferedWriter*** – decoratori care implementează citirea (dintr-un *InputStream*) și scrierea (într-un *OutputStream*) folosind zone tampon
  - *BufferedReader* permite citirea de linii prin intermediul metodei ***readLine***

Fișiere text

- Clasa *java.util.**Scanner***
  - Constructor pe bază de *File* sau *InputStream* sau *Reader*
  - Permite citirea de tipuri de bază sau linii de text

- Clasa ***PrintWriter***
  - Constructor pe bază de *File* sau *OutputStream* sau *Writer*
  - Supraîncărcări pentru metoda *print* pentru scrierea tipurilor fundamentale
  - Metoda *printf* pentru scriere cu formatare

Fișiere binare

- Clasa ***DataInputStream***
  - Constructor pe bază de *InputStream*
  - Permite citirea de tipuri de bază dintr-un *stream* binar (exemplu *FileInputStream*)
  - Metode de forma *double readDouble(), int readInt(), String readUTF(), ...*
- Clasa ***DataOutputStream***
  - Constructor pe bază de *OutputStream*
  - Permite scrierea de tipuri de bază într-un *stream* binar (exemplu *FileOutputStream*)
  - Metode de forma *void writeDouble(double), void writeInt(int), void writeUTF(String), ...*

Serializarea obiectelor

- Procesul de transformare a unui graf de obiecte într-un șir de octeți
- Interfața de marcaj ***Serializable*** **–** serializare implicită
  - Poate fi ajustată prin intermediul metodelor *writeObject* și *readObject*
- Interfața ***Externalizable*** **–** controlul serializării prin metodele
  - *void* ***writeExternal****(ObjectOutput out) throws IOException*
  - *void* ***readExternal****(ObjectInput in) throws IOException, ClassNotFoundException*
- Pentru scrierea și citirea din fișier se utilizează clasele ***ObjectOutputStream*** și ***ObjectInputStream*** cu metodele ***writeObject*** și ***readObject***

Fișiere cu acces direct

- Clasa **RandomAccessFile**:
  - Permite acces bidirecțional (citire și scriere) și poziționare în cadrul fluxului
  - Constructori pe bază de obiect *File* sau *String cale* și *String mode* ("r", "rw", …)
  - Metode de poziționare
    - *void seek(long pos)* – *poziționarea în fișier la un anumit octet*
    - *int skipBytes(int numBytes)* – *ignorarea unui număr de octeți*
    - *long getFilePointer()* – *poziția curentă a pointerului de fișier*
    - *long length()* – *dimensiune fișierului in octeți*
  - Metode de citire și scriere la nivel de octet similare cu *InputStream* și *OutputStream*
  - Metode de citire și scriere pentru tipuri de bază conform interfețelor *DataInputStream* și *DataOutputStream*