Fișiere în Java

Un fișier reprezintă o structură organizată de date stocată pe un suport permanent (hard disk, SSD, mediu optic, memorie externă), care permite păstrarea informațiilor independent de execuția unui program. Spre deosebire de variabilele alocate în memorie RAM, datele din fișiere sunt persistente și pot fi reutilizate la rulări ulterioare ale aplicației.

Fișierele joacă un rol esențial în:

- stocarea datelor pe termen lung;
- transferul de informații între aplicații;
- realizarea bazelor simple de date;
- arhivarea și backup-ul informațiilor

1. Clasificarea fișierelor

Fișierele pot fi clasificate după mai multe criterii, însă unul dintre cele mai relevante în contextul programării este natura datelor stocate. Această clasificare influențează direct modul de procesare și clasele Java utilizate pentru citire și scriere.

După natura datelor

A. Fișiere text

Fișierele text conțin date reprezentate sub formă de șiruri de caractere, codificate folosind standarde precum ASCII sau Unicode. Conținutul lor este lizibil pentru utilizator și poate fi vizualizat direct cu ajutorul editoarelor de text (Notepad, VS Code etc.).

Caracteristici principale:

- Informațiile sunt stocate sub formă de caractere;
- Sunt ușor de interpretat de către utilizatori;
- Ocupă mai mult spațiu decât fișierele binare pentru aceeași cantitate de informație;
- Sunt independente de arhitectura hardware.

Exemple de fișiere text:

- .txt – documente simple
- .csv – tabele de date
- .json, .xml – fișiere de configurare

Clase Java utilizate:

1. FileReader – pentru citirea caracter cu caracter;
2. BufferedReader – pentru citire eficientă pe linii;
3. FileWriter – pentru scriere;
4. PrintWriter – pentru formatare avansată.

Fișiere în Java

B. Fișiere binare

Fișierele binare stochează date într-un format specific mașinii, reprezentând direct valorile numerice sau structurile interne ale programului sub formă de secvențe de biți (0 și 1). Aceste fișiere nu sunt lizibile fără un program care le interpretează corect.

Caracteristici principale:

- Conținutul nu este lizibil direct;
- Mai eficiente din punct de vedere al dimensiunii și vitezei;
- Depind de structura datelor;
- Pot stoca obiecte, imagini, sunet, video.

Exemple de fișiere binare:

- .jpg, .png – imagini
- .mp3 – fișiere audio
- .dat – date serializate
- .exe – programe executabile

Clase Java utilizate:

1. FileInputStream / FileOutputStream – citire/scriere octeți;
2. ObjectInputStream / ObjectOutputStream – pentru serializare;
3. DataInputStream / DataOutputStream – pentru tipuri primitive.

2. Conceptul de flux (Stream)

În Java, interacțiunea cu fișierele se realizează prin intermediul fluxurilor (streams), care reprezintă canale de transmitere a datelor între program și fișier.

Tipuri de fluxuri

Tip flux	Direcția datelor	Exemple clase
Input Stream	Fișier → Program	FileReader, Scanner
Output Stream	Program → Fișier	FileWriter, PrintWriter

1. Clasa File

Clasa java.io.File descrie metadate asociate fișierelor și directoarelor, fără a manipula direct conținutul acestora.

Exemple de metode fundamentale

Metodă	Descriere
exists()	verifică existența fișierului
createNewFile()	creează fișier nou
delete()	șterge fișierul
length()	dimensiunea în bytes

Fișiere în Java

getAbsolutePath()	cale completă
isFile()	verifică dacă este fișier
isDirectory()	verifică dacă este director

3. Citirea fișierelor text

A.Cu Scanner

Scanner este potrivit pentru date simple și structurale (numere, cuvinte).

```
Scanner sc = new Scanner(new File("date.txt"));
while(sc.hasNextLine()) {
System.out.println(sc.nextLine());
}
sc.close();
```

B. Cu BufferedReader

BufferedReader este recomandat pentru fișiere de dimensiuni mari, datorită mecanismului de bufferizare.

```
BufferedReader br = new BufferedReader(new FileReader("date.txt"));
String linie;
while((linie = br.readLine()) != null) {
System.out.println(linie);
}
br.close();
```

4. Scrierea fișierelor

Metodă	Descriere	Observații
FileWriter	scriere simplă	suprascrie conținutul
BufferedWriter	scriere optimizată	suportă append

Exemplu

```
BufferedWriter bw = new BufferedWriter(new FileWriter("output.txt", true));
bw.write("Text adăugat");
bw.newLine();
bw.close();
```

Fișiere în Java

1. Scrie un program Java care verifică dacă fișierul date.txt există folosind clasa File și metoda exists(), citește tot conținutul acestuia, numără numărul de linii, cuvinte și caractere (inclusiv spațiile), afișează rezultatele în consolă și tratează excepțiile, afișând un mesaj dacă fișierul nu există sau dacă apare o eroare de tip IOException.

Exemplu de text in fisierul date.txt

```
1 Ana are 3 mere
2 Ion are 5 pere
3 Total 8 fructe
```

```java
package fisiere;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileNotFoundException;
import java.io.IOException;

public class Problema1 {
    public static void main(String[] args) {
        File fisier = new File("date.txt");

        // Verificăm dacă fișierul există
        if (!fisier.exists()) {
            System.out.println("Fișierul date.txt nu există.");
            return;
        }

        int nrLinii = 0;
        int nrCuvinte = 0;
        int nrCaractere = 0;

        // try-with-resources: fișierul se închide automat
        try (BufferedReader br = new BufferedReader(new FileReader(fisier))) {
            String linie;
            while ((linie = br.readLine()) != null) {
                nrLinii++;

                // Numărăm caracterele (inclusiv spațiile din linie)
                nrCaractere += linie.length();

                // Împărțim linia în cuvinte (separare după spații)
                String[] cuvinte = linie.trim().split("\\s+");
```

2. Scrie un program Java care verifică dacă fișierul **note.txt** există folosind clasa File, citește conținutul acestuia linie cu linie prin BufferedReader, calculează media notelor și numărul de elevi promovați (nota ≥ 5), creează fișierul **raport.txt** în care scrie numărul total de elevi, media notelor și numărul celor promovați, și tratează excepțiile, afișând mesaje clare dacă fișierul nu există sau dacă apare o eroare la citire/scriere.
   Exemplu fisier note.txt

```
1 Ana Pop 9
2 Ion Ionescu 7
3 Maria Georgescu 10
4 Paul Andrei 5
5
```

Fișiere în Java

```java
package fisiere;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class NoteElevi {
    public static void main(String[] args) {

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

        if (!fisier.exists()) {
            System.out.println("Fișierul note.txt nu există.");
            return;
        }

        int numarElevi = 0;
        int promovati = 0;
        double sumaNote = 0;

        try (BufferedReader br = new BufferedReader(new FileReader(fisier))) {

            String linie;

            while ((linie = br.readLine()) != null) {
                String[] parti = linie.split(" ");
                double nota = Double.parseDouble(parti[2]);
                numarElevi++;
                sumaNote += nota;
                if (nota >= 5) {
                    promovati++;
                }
            }

        } catch (IOException e) {
            System.out.println("Eroare la citirea fișierului.");
            return;
        }

        double media = sumaNote / numarElevi;

        // Scriem rezultatul în raport.txt
        try (BufferedWriter bw = new BufferedWriter(new FileWriter("raport.txt"))) {

            bw.write("Număr elevi: " + numarElevi);
            bw.newLine();
            bw.write("Media notelor: " + media);
            bw.newLine();
            bw.write("Număr promovați: " + promovati);

            System.out.println("Raportul a fost generat în raport.txt");

        } catch (IOException e) {
            System.out.println("Eroare la scrierea raportului.");
        }
    }
}
```

Fișiere în Java

```
                // Atentie la linii goale
                if (!linie.trim().isEmpty()) {
                    nrCuvinte += cuvinte.length;
                }
            }

            System.out.println("Linii: " + nrLinii);
            System.out.println("Cuvinte: " + nrCuvinte);
            System.out.println("Caractere (fără \\n): " + nrCaractere);

        } catch (FileNotFoundException e) {
            System.out.println("Fișierul nu a fost găsit.");
        } catch (IOException e) {
            System.out.println("A apărut o eroare la citirea fișierului.");
            e.printStackTrace();
        }
    }
}
```

3. Scrie un program Java care verifică dacă fișierul **produse.txt** există, citește toate produsele folosind BufferedReader, calculează valoarea totală a stocului și produsul cu valoarea maximă, creează fișierul **raport_produse.txt** cu aceste rezultate, generează fișierul **produse_sub_stoc.txt** cu produsele care au cantitatea mai mică de 10, și tratează excepțiile, afișând mesaje clare pentru fișier inexistent sau pentru erori de citire/scriere.

Exemplu fișier produse.txt

```
P01;Lapte;5.5;20
P02;Paine;2.5;50
P03;Ulei;9.8;10
P04;Zahar;4.2;5
```

```
package fisiere;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class ProduseMagazin {
    public static void main(String[] args) {

        File fisier = new File("produse.txt");
        if (!fisier.exists()) {
            System.out.println("Fișierul produse.txt nu există.");
            return;
        }
        double valoareTotala = 0;
        double valoareMax = 0;
        String produsMax = "";
        try (
            BufferedReader br = new BufferedReader(new FileReader(fisier));
            BufferedWriter bwSubStoc = new BufferedWriter(new FileWriter("produse_sub_stoc.txt"))
        ) {
            String linie;
            while ((linie = br.readLine()) != null) {
                // Separăm datele
                String[] p = linie.split(";");
                String cod = p[0];
                String denumire = p[1];
                double pret = Double.parseDouble(p[2]);
                int cantitate = Integer.parseInt(p[3]);
                double valoareProdus = pret * cantitate;
                valoareTotala += valoareProdus;
                // produs cu valoarea maximă
```

Fișiere în Java

```
                // produs cu valoarea maximă
                if (valoareProdus > valoareMax) {
                    valoareMax = valoareProdus;
                    produsMax = cod + " - " + denumire;
                }
                // produse cu stoc mic
                if (cantitate < 10) {
                    bwSubStoc.write(linie);
                    bwSubStoc.newLine();
                }
            }

        } catch (IOException e) {
            System.out.println("Eroare la citirea/scrierea fișierelor.");
            return;
        }
        // Scriem raportul principal
        try (BufferedWriter bwRaport = new BufferedWriter(new FileWriter("raport_produse.txt"))) {
            bwRaport.write("Valoare totală stoc: " + valoareTotala + " lei");
            bwRaport.newLine();
            bwRaport.newLine();
            bwRaport.write("Produs cu valoarea maximă:");
            bwRaport.newLine();
            bwRaport.write(produsMax);
            bwRaport.newLine();
            bwRaport.write("Valoare: " + valoareMax + " lei");
            System.out.println("Rapoartele au fost generate cu succes.");
        } catch (IOException e) {
            System.out.println("Eroare la generarea raportului.");
        }
    }
}
```