Java – S04

Alin Zamfiroiu

alin.zamfiroiu@csie.ase.ro

Java - S04

Continut:

- FileWriter
- FileReader
- RandomAccessFile
- Clase abstracte
- Atribute statice

S04 – Scriere in fisiere text

- Se creaza un obiect de tip **File**. Constructorul primeste fisierul ce urmeaza a fi deschis.
- Pe baza obiectului de tip **File** creat, se creeaza un obiect de tip **FileWriter**.

- Pe baza obiectului de tip **FileWriter** se creaza un obiect de tip **BufferedWriter**.

- Obiectul de tip **BufferedWriter** are metoda **write()** pentru scrierea in fisier.

S04 – Scriere in fisiere text

```
try {
    File file=new File("fisier.txt");
    FileWriter writer = new FileWriter(file);
    BufferedWriter buffer=new BufferedWriter(writer);
    buffer.write("Se salveaza acest fisier");
    buffer.close();
    writer.close();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Dupa finalizarea scrierii in fisier se inchide Obiectul de tip **BufferedWriter** si obiectul de tip **FileWriter**.

S04 – Citire din fisiere text

- Pentru citire din fisier se creaza un obiect de tip **File**.
- Apoi se creeaza un obiect de tip **FileReader**.

- Pe baza obiectului de tip **FileReader** se creaza un obiect de tip **BufferedReader**.

- Obiectul de tip **BufferedReader** are metoda **readLine()** pentru citirea din fisier.

S04 – Citire din fisiere text

```
try {
    File file=new File("fisier.txt");
    FileReader reader = new FileReader(file);
    BufferedReader buff=new BufferedReader(reader);
    System.out.println(buff.readLine());
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

S04 – Citire din fisiere text

- Ce alte metode de citire putem sa folosim?

- Am folosit data trecuta la citirea de la tastatura.

S04 – Citire din fisiere text

Scanner

```
try {
    Scanner scaner=new Scanner(new File("fisier.txt"));
    System.out.println(scaner.nextLine());
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

S04 - RandomAccessFile

- In programul principal se creeaza un obiect de tip RandomAccessFile.
- Primeste ca parametri numele fisierului si modul de deschidere.

- Prin intermediul obiectului de tip RandomAccessFile se scrie in fisier un **integer**, un **boolean** si un **String**.

- Se inchide fluxul deschis.

S04 - RandomAccessFile

```
RandomAccessFile raf=new RandomAccessFile("fileJava4.txt", "rw");

raf.writeBoolean(true);
raf.writeUTF("Un text");
raf.writeInt(6);
raf.close();
```

Asemanator se face si citirea din fisier.

S04 – RandomAccessFile

```
RandomAccessFile raf2=new RandomAccessFile("fileJava4.txt", "r");
System.out.println(raf2.readBoolean());
System.out.println(raf2.readUTF());
System.out.println(raf2.readInt());
raf2.close();
```

S04 – Clase abstracte

- Ce sunt clasele abstracte?
- Clasa abstracta **Animal** cu atributele: **nr_picioare**, **inaltime**, **greutate**.
- Constructor in clasa abstracta. De ce?
- Metodele abstracte **vorbeste** si **deplaseaza**.
- Clasa **Caine** extinde clasa abstracta **Animal**.

S04 – Clase abstracte

```
public abstract class Animal {
    protected int nr_picioare;
    protected int inaltime;
    protected int greutate;

    public Animal(){
        nr_picioare=4;
        inaltime=30;
        greutate=2;
    }
```

```
public abstract void vorbeste();
public abstract void deplaseaza();
```

```
public class Caine extends Animal {

    @Override
    public void vorbeste() {
        System.out.println("Cainele spune Ham ham");
    }

    @Override
    public void deplaseaza() {
        System.out.println("Cainele merge in 4 picioare.");
    }
```

S04 – Clase abstracte

- Diferenta dintre **clase abstracte** si **interfete.**
- In programul principal se creaza un **animal** si se initializeaza cu un **caine.**
- Se apeleaza cele doua metode abstracte.

S04 – Atribute statice

- Ce sunt atributele statice?
- O clasa cu un atribut static de tip lista (**LinkedList**).
- In cadrul listei se adauga noi animale din programul principal (**void main**).
- Se face o noua clasa numita **Procesare** in care se implementeaza metoda statica **deplaseazaTurma**(), care apeleaza metoda **deplaseaza** pentru toate animalele din lista statica existenta in clasa creata mai sus.

S04 – Atribute statice

```
public class ClasaMea {
    public static LinkedList<Animal> animale=new LinkedList<Animal>();

}
```

```
public static void deplaseazaTurma(){
    for(Animal element:ClasaMea.animale){
        element.deplaseaza();
    }
}
```