Fitxers binaris

Fluxes a fitxers

Consorci d'Educació
de Barcelona
Generalitat de Catalunya
Ajuntament de Barcelona

Introducció

- Els **fitxers binaris** emmagatzemen seqüències de dígits binaris que no són llegibles directament per l'usuari com ocorria amb els fitxers de text. Un exemple clar serien les imatges.
- Com ja hem vist prèviament, a Java hi ha dues classes que permeten treballar amb fitxers: FileInputStream i FileOutputStream. Aquestes classes treballen amb fluxos de bytes i creen un enllaç entre el flux de bytes i el fitxer.
- Els mètodes que proporciona la classe FileInputStream per a lectura són similars als vistos per a la classe FileReader. Aquests mètodes retornen nombre de bytes llegits o **-1** si s'ha arribat al final del fitxer.

Introducció

- Mètodes de FileInputStream:
  - int read () → Llegeix un byte del flux i el retorna
  - int read (byte[] b) → Llegeix fins a *b.length* bytes del flux i els deixa en l'array b.
  - int read (byte [] b, int desplaçament, int n) → Llegeix fins a n bytes començant en b[desplaçament]
- Mètodes FileOutputStream:
  - void write (int b) → escriu un byte
  - void write (byte [] b) → escriu b.length bytes
  - void write (byte [] b, int desplaçament, int n) → escriu n bytes des de l'array b començant per b[desplaçament]

A continuació veurem un exemple en el qual primer s'escriuen els bytes en un fitxer i després es visualitzen.

Exemple

```
import java.io.* ;
public class EscriuFitxerBytes {
   public static void main (String [] args) throws IOException{
        File fitxer = new File ("FitxerDades.dat");
        FileOutputStream fileout = new FileOutputStream (fitxer);
        FileInputStream filein = new FileInputStream(fitxer);
        int i;
        for (i=1; i<100; i++)
            fileout.write(i);
        fileout.close();
        while ((i=filein.read()) != -1)
            System.out.println(i);
        filein.close();
    }
}
```

Bucle per l'escriptura de bytes

Bucle de lectura

DataInputStream / DataOutputStream

- DataInputStream i DataOutputStream s'utilitzen per a llegir i escriure dades de tipus primitius: int, float, long, etc.
- Aquestes classes defineixen diversos mètodes readXXX i writeXXX que són variacions dels mètodes read() i write() de la classe basi. Vegem alguns:

• boolean readBoolean(); • byte readByte(); • int readUnsignedByte(); • int readUnsignedShort(); • short readShort(); • char readChar(); • int readInt(); • float readFloat(); • String readUTF();	• void writeBoolean (boolean v); • void writeByte (int v); • void writeBytes (String s); • void writeShort (int v); • void writeChars (String s); • void writeChar (int v); • void writeInt (int v); • void writeFloat (float v); • void writeUTF (String str);

DataInputStream / DataOutputStream

Per a obrir un objecte DataInputStream s'utilitzen els mateixos mètodes que per a FileInputStream:

```
File fitxer = new File ("FitxerDades.dat");
FileInputStream filein= new FileInputStream (fitxer);
DataInputStream dataIS = new DataInputStream (filein);
```

Lògicament, el mateix succeeix per a DataOutputStream:

```
File fitxer = new File ("FitxerDades.dat");
DataOutputStream dataOS = new DataOutputStream ( new FileOutputStream (fitxer));
```

Activitats

A1- Escriu un programa que insereixi dades en "FitxerDades.dat". Les dades les prendrà de dos arrays fixes definits en el propi programa.

- Un array contindrà els noms d'una sèrie de persones.
- L'altre array contindrà les edats de els persones.

S'anirà recorrent els arrays i anirem escrivint en el fitxer el nom (mitjançant el mètode **writeUTF(String str)** i l'edat (**writeInt (int v)**).
NOTA: si volem afegir bytes al final del fitxer ("FitxerDades.dat") es pot usar el següent constructor: *FileOutputStream fileout = new FileOutputStream (fitxer, true)*.

A2- Ara escriu un programa que permeti visualitzar les dades gravades anteriorment en el fitxer "FitxerDades.dat". S'han d'obtenir en el mateix ordre en el qual es van escriure, és a dir, primer obtenim el nom i després l'edat.

Objectes en fitxers binaris

Hem vist com es guarden els tipus de dades primitives en un fitxer, però, què passa amb els objectes definits en el programa? Una opció seria anar guardant cada atribut per separat però seria molt farragós.

- Solució: Java permet **guardar objectes** que implementin la interfície Serializable.
- La interfície Serializable imposa una sèrie de mètodes per a guardar i llegir objectes. Els més importants són:
- **Object readObject ()** → per llegir objectes des d'arxiu.
  - Requereix un objecte del ObjectInputStream
  - Pot llançar **IOException** i **ClassNotFoundException**
- **void writeObject (Object obj)** → per escriure l'objecte ***obj*** en un arxiu.
  - Requereix un flux **ObjectOutputStream**
  - Pot llançar **IOException**

Exemple

Primer definirem la classe Persona. Contindrà dos atributs: nom i persona. També dels mètodes ***get*** i ***set*** corresponents.

```
import java.io.Serializable;
public class Persona implements Serializable {
    private String nom;
    private int edad;
    public Persona (String nom, int edad) {
        this.nom = nom;
        this.edad = edad;
    }
    public Persona () {
        this.nom=null;
    }
    public void setNom (String nom) {this.nom = nom;}
    public void setEdad (int edad) {this.edad = edad;}
    public String getNom () {return this.nom;}
    public int getEdad () {return this.edad;}
}
```

Exemple

A continuació, el següent exemple “escriu” un objecte Persona en un fitxer:

```
import.java.io.*;
public class EscriuFitxerObjecte {
   public static void main (String[] args) throws IOException {
        File fitxer = new File ("FitxerPersona.dat");
        FileOutputStream fileout = new FileOutputStream (fitxer);
        ObjectOutputStream dataOS = new ObjectOutputStream (fileout);
        Persona persona = new Persona ("pepito", 15);
        dataOS.writeObject (persona) ;
        dataOS.close();
   }
}
```

Ara veurem la lectura. El procés de lectura es realitza en un bucle while(true) i aquest es tanca en un bloc try-catch, ja que la lectura finalitzarà quan arribi a final de fitxer i es llanci l'excepció EOFException.

Exemple

```
import java.io.*;
public class LlegirFitxerObjecte {
   public static void main(String [] args) throws IOException, ClassNotFoundException {
   Persona persona;
   File fitxer = new File ("FitxerPersona.dat");
   FileInputStream filein = new FileInputStream (fitxer);
   ObjectInputStream dataIS = new ObjectInputStream (filein) ;
   try {
        while (true) {
             persona = (Persona) dataIS.readObject ();
             System.out.printf("Nom: %s, edad : %d %n",
             persona.getNombre(), persona.getEdad());
        }
   } catch (EOFException eo) {
   System.out.println ("Fin de la Lectura");
   } finally{
             dataIS.close();
        }
   }
}
```

Activitat

Basant-se (o no) en les activitats de M3 del curs passat:

A3- Dissenya una classe que anomenaràs ***EstatPartida*** per gestionar l'estat d'una partida de 3 en ratlla. Ha d'incloure: posició de les figures (serà una **matriu 3x3**) i a qui li toca tirar (**jugador1** o **jugador2**).

A4- Dissenya una classe que anomenaràs ***DesaiRecuperaEstat*** que disposi de dos mètodes: ***desarEstat*** que permetrà guardar en un arxiu l'estat de la partida i un altre, ***recuperarEstat***, que permetrà recuperar l'estat d'una partida des d'arxiu.

A5- Crea un programa en java que generi un objecte ***EstatPartida***, permeta modificar la posició d'algunes figures del tauler, guardi en un arxiu l'estat de la partida, recuperi l'estat des de l'arxiu i el mostri per pantalla.