Fluxes o Streams

Fluxes a fitxers de text

Introducció

- S'usa l'abstracció de flux (stream) en java per a tractar la comunicació entre una font i un destí.
- La font pot ser un fitxer situat en qualsevol dispositiu, un element perifèric o qualsevol altre programa.
- Hi han dos tipus de fluxos:
  - Fluxos de bytes (8 bits): el seu ús està orientat a la lectura/escriptura de dades binàries. Totes les classes de fluxos de bytes descendeixen de les classes InputStream i OutputStream. Cadascuna d'aquestes classes tenen diverses subclasses que controlen les diferències entre els diferents dispositius d'entrada/sortida que es poden utilitzar
  - Fluxos de caràcters (16 bits): realitzen operacions d'entrada/sortida de caràcters. Aquests fluxos venen governats per les classes Reader i Writer. La raó de ser d'aquestes classes és la internacionalització; l'antiga jerarquia de fluxos de E/S només suporta fluxos de 8 bits i, per tant, no controlova caràcters Unicode de 16bits.

Fluxos de byte (byte stream)

- InputStream representa les classes que produeixen entrades de diferents fonts (fitxer, connexió, canonada, etc).
- Tipus:
  - ByteArrayInputStream: permet usar un espai d'emmagatzematge intermedi de memòria (Buffer).
  - StringBufferInputStream: converteix un String en un InputStream.
  - FileInputStream: flux d'entrada cap a fitxer; ho usarem per a llegir informació d'un fitxer.
  - PipedInputStream: implementa el concepte de canonada (pipe).
  - FilterInputStream: proporciona funcionalitats a altres classes InputStream.
  - SequenceInputStream: converteix dos o més objectes InputStream en un InputStream únic.

Fluxos de byte (byte stream)

- OutputStream: inclou les classes que decideixen on anirà la sortida.
- Tipus:
  - ByteArrayOutputStream: crea un espai intermedi en memòria (buffer).
  - FileOutputStream: flux de sortida cap a fitxer; ho usarem per a enviar informació a fitxer.
  - PipedOutputStream: qualsevol informació que s'escrigui aquí acaba en un PipedInputStream associat. Implementen canonades.
  - FilterOutputStream: proporciona funcionalitats a altres OutputStream.

Fluxos de caràcter (character stream)

- Les classe Reader i Writer controlen fluxos de caràcters Unicode.
- Hi ha ocasions que convé usar les classes que controlen fluxos de byte amb les classes que treballen amb caràcters. Per a aconseguir-ho, existeixen classes pont:
  - InputStreamReader que converteix un InputStream en un Reader.
  - OutputStreamWriter que converteix un OutputStream en un Writer.
- Les classes de fluxos de caràcters més importants són:
  - FileReader i FileWriter: per a l'accés a fitxers de caràcters.
  - CharArrayReader i CharArrayWriter: llegeixen i escriuen un flux de caràcters en un array de caràcters.
  - BufferedReader i BufferedWriter: eviten l'accés directe al fitxer utilitzant un buffer intermig entre la memòria i el stream.

Formes d'accés a un fitxer

- **Accés seqüencial**: les dades es llegeixen i s'escriuen en ordre (seqüencialment). És semblant a una cinta de vídeo antiga: si volem accedir a una dada que està cap a la meitat de la cinta havem de llegir tota l'anterior informació. L'escriptura es farà a partir de l'última dada escrita i no és possible fer insercions entre dades ja escrites.
- **Accés directe o aleatori**: accedim directament a les dades sense necessitat de llegir totes les dades anteriors. Les dades estan emmagatzemades en registres de grandària coneguda, ens podem moure d'un registre a un altre de manera aleatòria per a llegir-los o modificar-los.
- A Java l'accés seqüencial s'implementa amb FileInputStream/FileOutputStream en el cas de dades binàries. I FileReader/FileWriter per a accedir a text. Per a l'accés aleatori s'utilitza RandomAccessFile.

Fluxos des de i cap a fitxers

- Els fitxers de text emmagatzemen caràcters alfanumèrics en un format estàndard (ASCII, UNICODE, UTF8...).
- Usarem les classes FileReader i FileWriter. Hem de controlar sempre les excepcions amb try-catch ja que:
  - En llegir un fitxer es pot generar FileNotFoundException si el nom del fitxer no és vàlid o no existeix.
  - En escriure es pot generar una IOException si no disposem de permisos o si el disc està ple.

Fluxos des de i cap a fitxers

FileReader

- Els mètodes que proporciona per a lectura són:
  - int read() → llegeix un caràcter i el retorna com a sencer.
  - int read (char[] buf) → llegeix fins a buf.length caràcters. Els caràcters llegits es van emmagatzemant en buf.
  - int read (char [] buf, int desplaçament, int n) → llegeix fins a **n caràcters** guardant-los en buf començant per buf[desplaçament] i retorna el nombre llegit de caràcters.
- Aquests mètodes retornen el nombre de caràcters llegits o -1 si s'ha arribat al final del fitxer.
- En un programa la seqüència de passos serà:
  - **Primer**: invocar a la classe File per a accedir al fitxer.
  - **Segon**: crear el flux d'entrada cap al fitxer amb la classe FileReader.
  - **Tercer**: operacions de lectura.
  - **Quart**: tancarem el flux mitjançant el mètode close().

Exemple

```
import java.io.*;
public class lectorFitxerText {
   public static void main (String [] args) throws IOException {
        File fitxer = new File ("lectorFitxerText.java"); // declaració del fitxer
        FileReader flux = new FileReader (fitxer); // creem flux d'entrada al fitxer
        int i;
        while ((i=flux.read())!=-1) //el llegim fins al final caràcter a caràcter
            System.out.println ((char) i); //Fem un cast a char del sencer llegit
        flux.close();
   }
}
```

Activitat

- **(A1)** Prova el codi anterior
- Modifica el codi anterior perquè el programa vagi llegint caràcters de 20 en 20.
- Modifica el codi anterior perquè se li pugui passar el nom del fitxer al programa.

Fluxos des de i cap a fitxers

FileWriter

- Els mètodes que proporciona per a escriptura són:
  - void write (int c) → Escriu un caràcter.
  - void write (char [] buf) → Escriu un array de caràcters.
  - void write (char [] buf, int desplaçament, int n) → Escriu n caràcters començant per buf[desplaçament].
  - void write (String str) → Escriu una cadena de caràcters.
  - void append (char c) → Afegeix un caràcter a un fitxer.
- Aquests mètodes poden llançar l'excepció IOException.
- Igual que abans, declarem el fitxer mitjançant la classe File i a continuació es crea el flux de sortida cap al fitxer amb la classe FileWriter.

Exemple

```
import java.io.*;
public class escriuFitxerText {
   public static void main(String[] args) throws IOException {
        File fitxer = new File("FitxerText.txt");
        FileWriter fitW = new FileWriter(fitxer);
        String cadena = "--- Prova usant FileWriter ---";
        char[] cad = cadena.toCharArray();
        for (int i = 0; i < cad.length; i++)
            fitW.write(cad[i]); // anem escrivint caràcter a caràcter
        fitW.append('*'); // afegim un asterisc al final
        fitW.close(); // tanquem el fitxer
   }
}
```

Activitat

- **(A2)** Còpia l'exemple anterior i prova'l.
- Modifica l'exemple anterior per a, en comptes d'escriure els caràcters un a un, s'escriga tot el array usant el mètode write (char [] buf).

NOTA: si volem afegir caràcters al final d'un fitxer de text podem utilitzar el següent constructor de FileWriter → FileWriter fit = new FileWriter (fitxer , true)

- Crea el següent array de String i insereix en el fitxer les cadenes una a una usant el mètode write (String str).

```
String prov[] = {"Albacete", "Avila", "Badajoz", "Caceres", "Huelva",
"Jaen",
            "Madrid", "Segovia", "Soria", "Toledo", "Valladolid", "Zamora"};
```

BufferedReader / BufferedWriter

BufferedReader

- FileReader no conté mètodes que ens permetin llegir línies completes.
- BufferedReader, en canvi, sí que disposa. En concret, el mètode readLine() que llegeix una línia del fitxer i la retorna, o retorna null si no hi ha res a llegir o arribem al final del fitxer. També disposa del mètode read() per llegir un caràcter.
- Per construir un BufferedReader necessitem un objecte FileReader.

Exemple

```
import java.io.*;
public class LlegirFitxerTextBuf{
   public static void main (String [] args) {
       try {
           BufferedReader fitxer = new BufferedReader ( new FileReader
           ("LlegirFitxerTextBuf.java")); // Atenció: utilitzem un altre constructor de FileReader.
           //Ara no usem un objete File, sino passem el nom de l'arxiu
           String linia;
           while ((linia = fitxer.readLine()) != null) // llegim de línia en línia
               System.out.println (linia);
           fitxer.close();
       }
       catch (FileNotFoundException fn) {
           System.out.println ("No es troba el fitxer enlloc"); }
       catch (IOException io) {
         System.out.println ("Error d'E/S"); }
   }
}
```

BufferedReader / BufferedWriter

BufferedWriter

- Igual que BufferedReader, BufferedWriter aporta funcionalitats d'escriptura a FileWriter. Disposa, per exemple, del mètode newLine() que insereix un salt de línia.
- Segueix el mateix model de construcció que BufferedReader; és a dir, requereix un objecte FileWriter per a instanciar-se.

PrintWriter

- Una altra classe que deriva de Writer. Posseeix els mètodes print (String) i println (String) (idèntics als de System.out) per a escriure en un fitxer.
- Per a construir un PrintWriter necessitem un FileWriter

```
PrintWriter fitxer = new PrintWriter ( new FileWriter (Nom_Fitxer));
```

Activitat

- **(A3)** Escriu un programa en java que mostri per pantalla un fitxer de text que li passem com a argument (o utilitzant scanner) utilitzant la classe BufferedReader.
- **(A4)** Escriu un programa que, utilitzant la classe BufferedWriter, escrigui 10 files de caràcters en un fitxer de text i després d'escriure cada fila salte una línia amb el mètode newLine().
- **(A5)** Repeteix l'exercici anterior però ara utilitzant la classe PrintWriter. Podeu fer-ho a un nou fitxer o al mateix que l'anterior, comentant les línies innecessàries.