Tema 12
Fitxers

1. Introducció

Tots els llenguatges de programació tenen alguna forma d'interatuar amb els fitxers.

Operacions que podem fer en un fitxer:

Consultar característiques del fitxer	Llegir del fitxer	Escriure en el fitxer
- Comprovar si existeix, si tenim permisos, si és un directori... - Consultar data del fitxer, grandària...	1r. Obrir el fitxer 2n. Mentre resten dades per llegir Llegir dades del fitxer 3r. Tancar el fitxer	1r. Obrir el fitxer 2n. Mentre tenim dades per escriure Escriure dades al fitxer 3r. Tancar el fitxer

Moltes operacions sobre fitxers generen excepcions (per intentar llegir d'un fitxer inexistent, no tindre permisos, etc). Per tant, abans de vore què podem fer amb els fitxers, veiem primer com tractar, en general, les excepcions.

Tractament de les excepcions

Veiem 2 formes de tractar les excepcions en general, amb un exemple de fitxers:

Ja veiérem que podem tractar les excepcions amb *try-catch*:

```
public static void llegirFitxer() {
    ...
    try {
        // Ús de mètodes de lectura de fitxers que poden generar excepcions
    }
    catch(FileNotFoundException e){
        System.out.println("Fitxer no trobat");
    catch(IOException e){
        System.out.println( "Error accedint al fitxer " + e.getMessage() );
    }
    ...
}
```

Però altra solució és "passar el marró" a qui haja cridat a la funció on està el codi que pot provocar l'error:

```
public static void llegirFitxer() throws FileNotFoundException, IOException {
    ...
    // Ús de mètodes de lectura de fitxers
    ...
}

public static void gestioFitxers() throws FileNotFoundException IOException() {
    ...
    llegirFitxer();
    ...
}

public static void main(String [] args) {
    ...
    try {
        gestioFitxers();
    }
    catch(IOException e){
        System.out.println( e.getMessage() );
    }
    ...
}
```

Si hi ha error, esta part no s'executa i la funció acaba.

La funció *llegirFitxer()* hauria de tractar l'error de les operacions de lectura de fitxers amb *try-catch* però en compte d'això li "passa el marró" de tractar l'error a qui ha invocat a *llegirFitxer()*.

Si *llegirFitxer()* provoca error, la funció acaba.

La funció *gestioFitxers()* invoca a *llegirFitxer()* i, com esta llança una excepció, l'hauria de tractar amb *try-catch* però en compte d'això li "passa el marró" a qui haja invocat a gestioFitxers().

El *main()* invoca a *gestioFitxers()* i, com esta llança una excepció, la tracta amb un *try-catch*.
També s'haguera pogut posar el *throws* en el *main()* però realment no estaria passant el marró a ningú (ja que ningú crida a *main()*).

Si *gestioFitxers()* genera *IOException*, esta part sí que s'executa.

2. Consultar característiques d'un fitxer

En un programa en Java podrem saber coses la grandària d'un fitxer, qui és el seu directori pare, si té permís de lectura, etc.

Per a fer això existeix la classe ***File***. Per a saber les característiques d'un fitxer crearem un objecte d'eixa classe instanciant-lo amb el nom del fitxer. Després, accedirem a les propietats del fitxer amb els mètodes d'eixa classe.

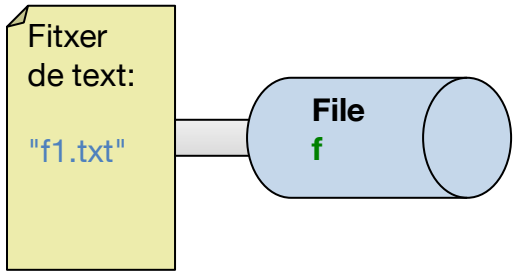
1r pas: associar al fitxer un objecte de la classe *File* (cal importar *java.io.File*);

Es pot fer amb 3 constructors diferents:

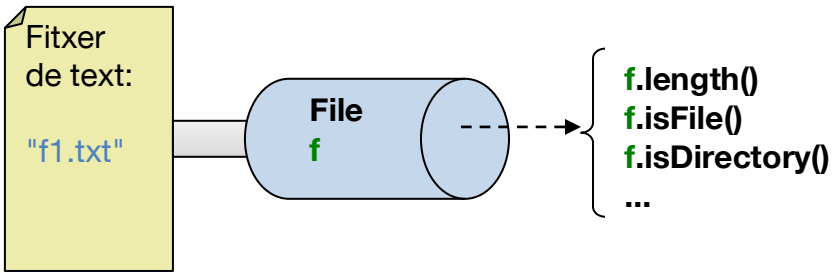
```
File f = new File( "/Users/abdo/f1.txt");
```

```
File f = new File( "/Users/abdo", "f1.txt");
```

```
File directori = new File("c:\kk");
File f = new File(directori, "f1.txt");
```



2n pas: accedir a les propietats del fitxer usant els mètodes de la classe.

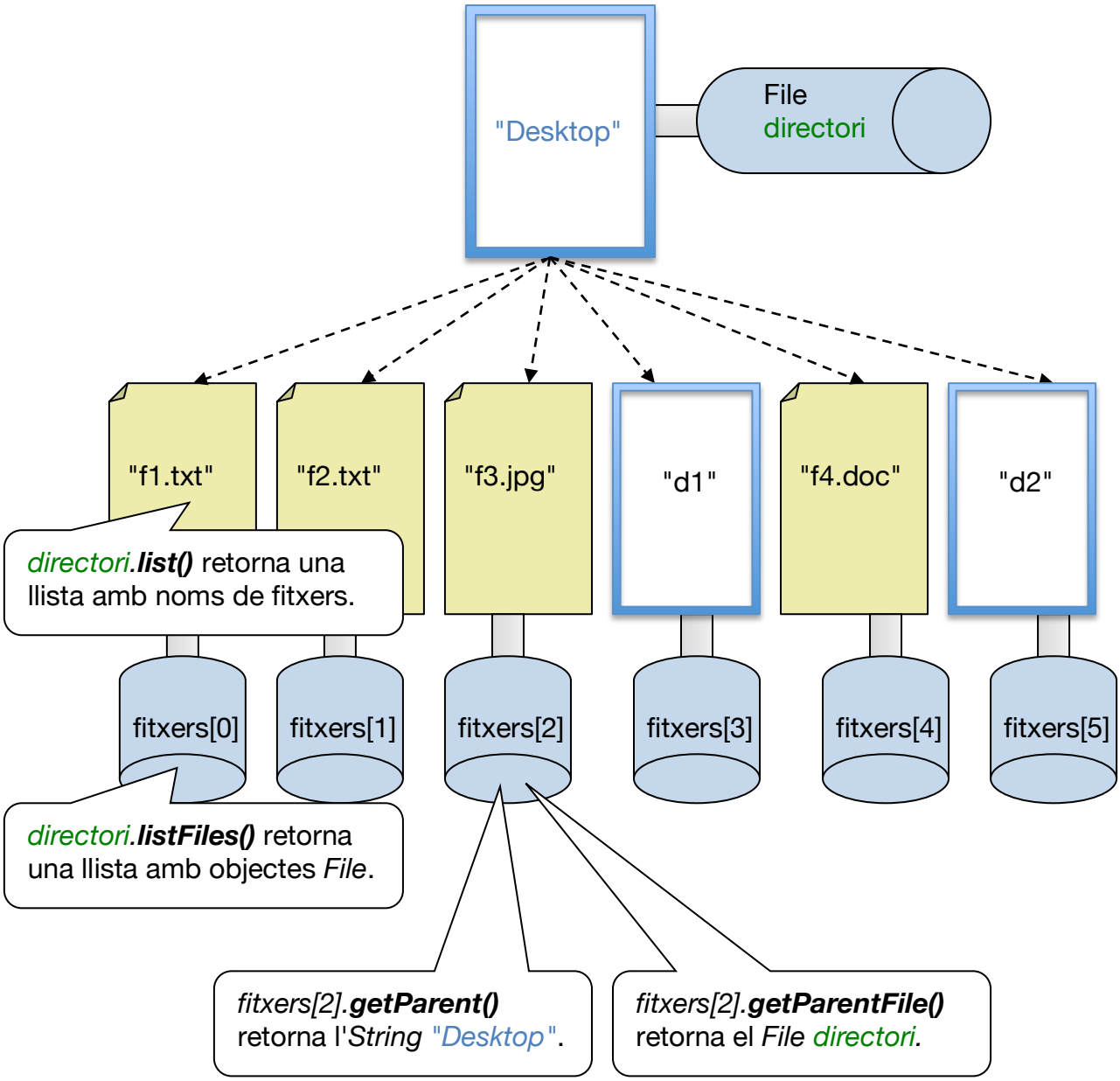



```
System.out.println("Existeix: " + f.exists());
System.out.println("És directori: " + f.isDirectory());
System.out.println("És fitxer: " + f.isFile());
System.out.println("Bytes: " + f.length());
System.out.println("Nom: " + f.getName());
System.out.println("Ruta: " + f.getPath());
System.out.println("Pare: " + f.getParent());
System.out.println("Permís execució: " + f.canExecute());
System.out.println("Permís lectura: " + f.canRead());
System.out.println("Permís escriptura: " + f.canWrite());
System.out.println("toString: " + f.toString());
...
```

També hi ha mètodes per a obtindre la llista dels fitxers que hi ha dins d'un directori. Si el fitxer que tenim en el File correspon a un directori, podem fer:

```
File directori = new File("/Users/abdo/Desktop");

String nomsFitxers [] = directori.list();

File fitxers [] = directori.listFiles();
```

Amb ***list()*** obtenim un array dels noms de fitxers que estan dins del directori. Però si volem accedir a les propietats de cadascun d'eixos fitxers, seria millor ***listFiles()***, ja que ens retorna un array d'objectes *File* que estaran apuntant a cadascun d'eixos fitxers. Veiem-ho gràficament:

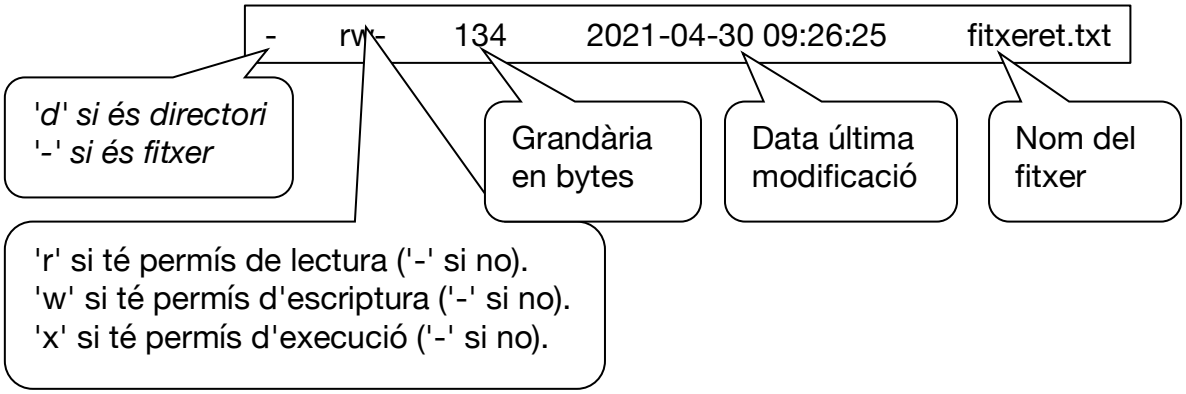


Exercicis sobre consultar les característiques de fitxers

1. Fes la funció *lligFitxer()* que demane per teclat un nom de fitxer (o directori). Si existeix, retornarà el File que apunta al fitxer. Si no existeix retornarà null.

2. Fes la funció *mostraAtributsFitxer()*, a la qual se li passa com a paràmetre un fitxer (no nom de fitxer sinó objecte *File*). La funció mostrarà informació del fitxer. Si el fitxer no existeix retornarà false (true en cas contrari).

   Exemple:



La data la tindrem com un número *long*. Per a obtindre la data en format *String* podeu usar esta funció:

```
public static String dataString(long data) {
   return new SimpleDateFormat("yyyy-MM-dd hh:mm:ss").format(new Date(data));
}
```

3. Fes la funció *llistaFitxers()*, a la qual se li passa com a paràmetre un directori (no el nom del directori sinó l'objecte *File*). Mostrarà les dades de cada fitxer (o subdirectori) del directori (usa la funció *atributsFitxer()*).

4. Fes la funtció llistaFitxersRecursivament(), qui rep com a paràmetre un directori (File). Per a cada fitxer o subdirectori seu, mostrarà les seues dades. A més, si és un subdirectori, mostrarà el seu contingut, cridant recursivament a la funció.

```
-       rw-     2020-12-03 06:34:21      658  f6.rtf
-       rw-     2019-03-22 08:13:29    21300  f3.docx
d       rwx     2021-05-01 09:08:32      192  dir2
        -       rw-     2020-12-03 06:34:21      658   f23.rtf
d       rwx     2021-05-01 09:07:03      224  dir1
        d       rwx     2021-05-01 09:07:43      128   d11
                -       rw-     2019-03-22 08:14:25     7809   f112.odt
-       rw-     2019-03-22 08:14:25     7809  f8.odt
```

Afig un sagnat en cada nivell de l'arbre de directoris.
Per a aconseguir-ho, posa altre paràmetre a la funció recursiva.

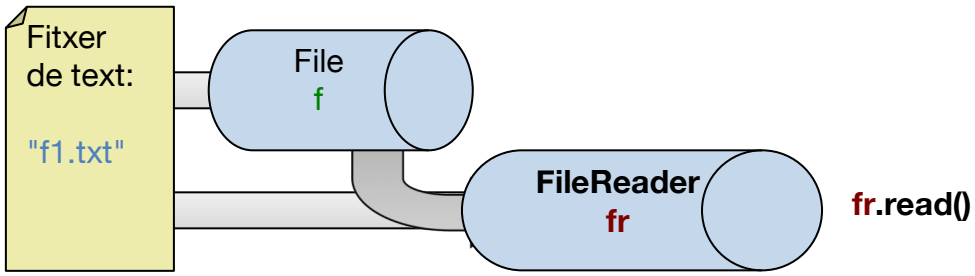
3. Fitxers de text. Streams d'entrada

En un programa en Java potser ens interessarà llegir el contingut d'un fitxer de text on puga haver un escrit qualsevol, un document html, un fitxer de configuració de Linux...

A vegades ens interessarà llegir del fitxer caràcter a caràcter. En eixe cas usarem la classe *FileReader*. Però si volem llegir línia a línia usarem la classe *BufferedReader*.

3.1. Lectura caràcter a caràcter

1r pas: Associar al fitxer un objecte de la classe ***FileReader*** (i importar *java.io.FileReader*).



El podem crear amb diversos constructors:

```
FileReader fr = new FileReader( f );
FileReader fr = new FileReader( "f1.txt" );
```

La crida als constructors obliga a tractar l'excepció *FileNotFoundException* (amb *try-catch* o *throws*).

2n pas: Llegir del fitxer

Executarem un ***read()*** per cada caràcter que volem llegir. Este mètode retorna un enter, que és el codi del caràcter llegit (o -1 si no hem pogut llegir del fitxer). Després cal fer casting a ***char*** per a treballar amb eixe caràcter.

```
int num = fr.read();

char lletra = (char) num;
```

El *read()* obliga a tractar l'excepció *IOException.*

Nota: com *FileNotFoundException* és filla d'*IOException*, només estarem obligats a tractar *IOException*. Però si volguérem un tracte diferent per a cada tipus d'error, caldria capturar les 2 excepcions.

Exemple: llegim d'un fitxer de text i ho mostrem per pantalla:

```
public static void mostraCaractersDeFitxer(String nomFitxer) throws IOException {
        FileReader fr = new FileReader(nomFitxer);
        int c;
        while ( (c = fr.read()) != -1 ) {
                System.out.print( (char) c );
        }
}
```

Exercicis sobre lectura de fitxers de text amb *FileReader* (caràcter a caràcter)

5. Donat un fitxer de text fes un programa que indique:

   Quantes vocals hi ha al fitxer, quants espais en blanc, quantes majúscules i quantes minúscules.

   Ampliació: compta també les paraules. Es consideraran separadors de paraules (a més de l'espai en blanc): . , ; : ! ?

6. Corregir exàmens tipus test.

   En un examen de tipus test, cada alumne deixa en un fitxer amb el seu nom (Pep.txt) les seues 20 respostes (que poden ser A, B, C, D o bé un guionet si no vol contestar-se una pregunta) en la primera línia del fitxer. Per exemple un fitxer podria tindre esta línia: ABCD-BBA-CCDBACBC-DA.

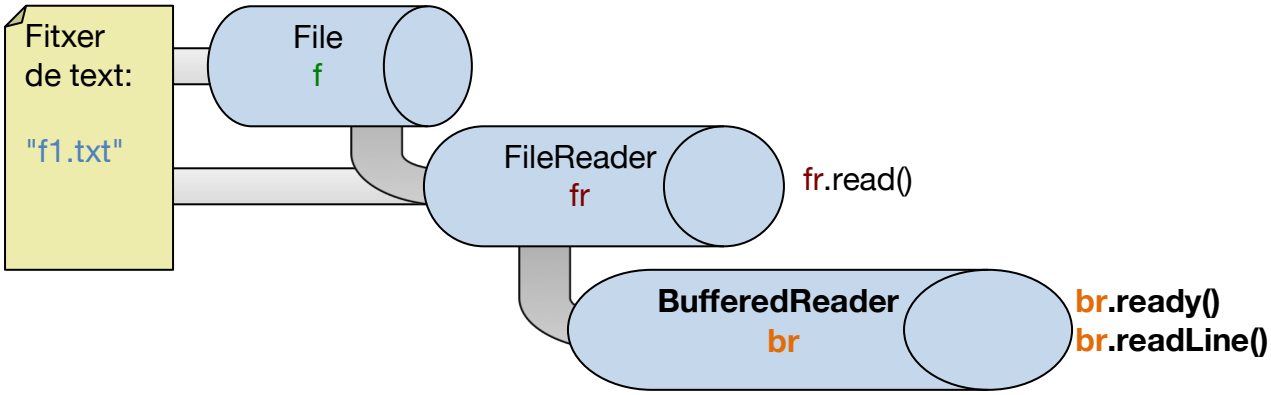
   En altre fitxer de text (solucio.txt) tindrem la solució, que serà: ABCDDCBAACCDBAABCDDA.

   Fes un programa tal que demane el nom del fitxer de l'alumne. El programa haurà de mostrar-nos quantes respostes ha encertat i quantes ha fallat. També la nota que ha tret l'alumne, tenint en compte que cada pregunta correcta suma 0.5 punts i cada pregunta incorrecta resta 0.125 punts.

   Modifica el programa per a que demane en bucle els noms dels fitxers de tots els alumnes fins que li posem de nom de fitxer "fi".

3.2 Lectura línia a línia

Si en compte de llegir d'un fitxer caràcter a caràcter volem fer-ho línia a línia, necessitarem unobjecte de la classe ***BufferedReader***, associat a un objecte de la classe *FileReader*. Caldrà importar: *java.io.BufferedReader*.



Constructor i mètodes:

```
BufferedReader br = new BufferedReader(fr);

br.ready();         // retorna true/false si queden línies per llegir

br.readLine();      // retorna un String amb la següent línia del fitxer
```

Exemple: llegim d'un fitxer de text, línia a línia, i ho mostrem per pantalla:

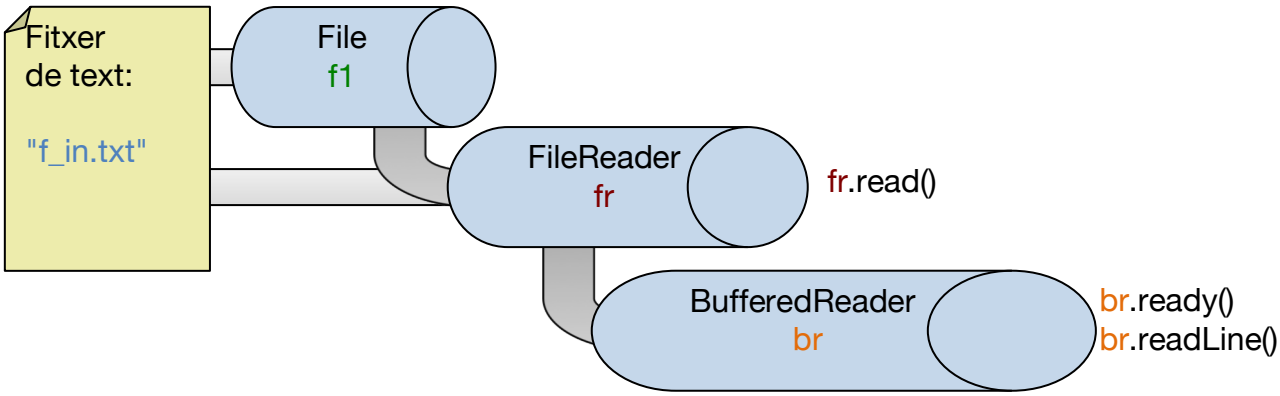
```
public static void mostraLiniesDeFitxer(String nomFitxer) throws IOException  {
    FileReader fr = new FileReader(nomFitxer);
    BufferedReader br = new BufferedReader(fr);
    String s;
    while (br.ready()) {
        s = br.readLine();
        System.out.println(s);
    }
}
```

Exercici sobre lectura de fitxers de text amb *BufferedReader* (línia a línia)

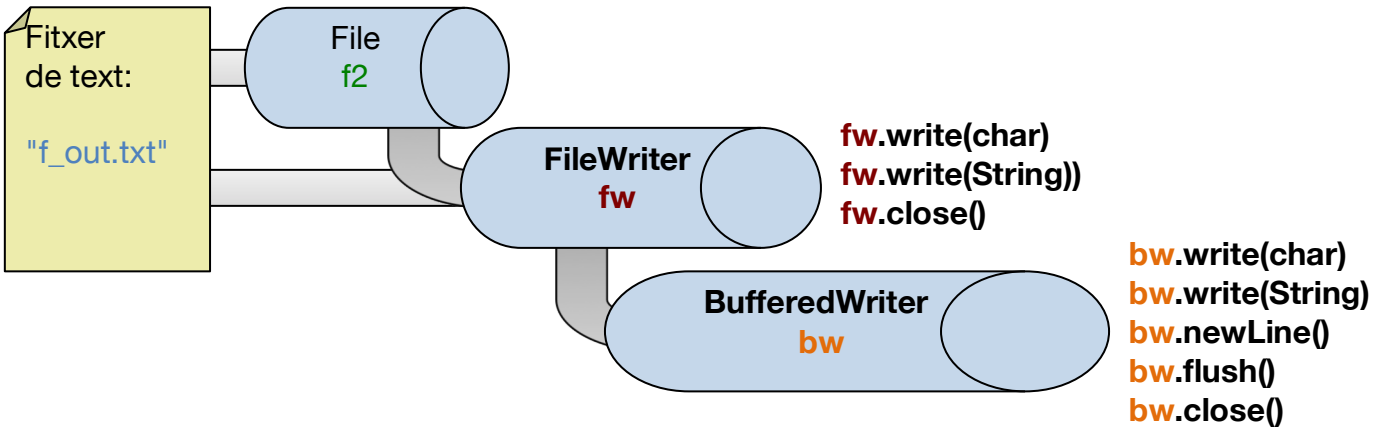
7. Escriu un programa que diga quants paràgrafs té un fitxer de text. Que mostre també una llista amb el número de paràgraf i la quantitat de lletres que té cadascun.

4. Fitxers de text. *Streams* d'eixida

Recordem que tenim estes 2 classes per a la **lectura** de fitxers:



Per a l'**escriptura** de fitxers tindrem estes classes anàlogues:

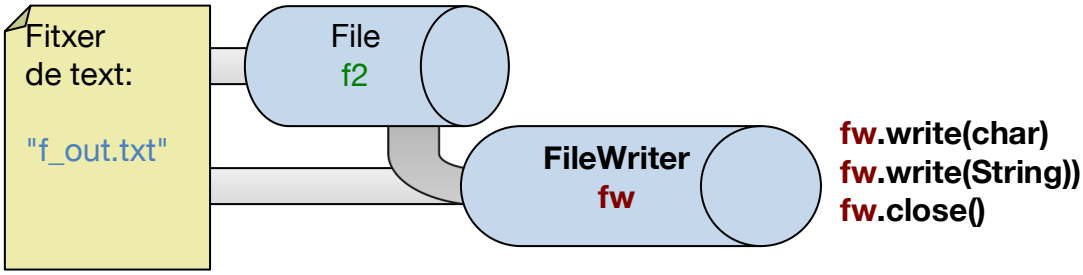


La classe *BufferedWriter* fa pràcticament el mateix que *FileWriter* (escriure caràcters o cadenes) però pot ser més eficient si es fan moltes escriptures ja que:

- ***FileWriter***: cada vegada que s'invoca *write(...)*, es fa un accés a memòria secundària.
- ***BufferedWriter***: write no escriu directament en memòria secundària, sinó en un *buffer* temporal de la RAM (més ràpid). Quan volem bolcar el *buffer* al fitxer, invocarem el mètode *flush()*.

BufferedWriter també té el mètode *newLine()*, que és com un *write('\n')*.

4.1 Escriptura amb *FileWriter*



Exemple: escriu en un fitxer les frases introduïdes per teclat (fins escriure "fi"):

```
FileWriter fw = null;
System.out.println("Escriu frases ('fi' per a acabar):");
try {
   fw = new FileWriter(nomFitxer, true);

   String frase = teclat.nextLine();

   while (!frase.equalsIgnoreCase("fi")) {
      fw.write(frase + "\n");
      frase = teclat.nextLine();
   }
} catch (IOException ex) {
   System.out.println("Error escrivint a fitxer");

} finally {
   try {
      fw.close();

   } catch (IOException ex) {
      System.out.println("Error tancant el fitxer");
   }
}
```

Creem un *FileWriter* amb el constructor.
true --> s'afegiran dades al fitxer
false --> se sobreescriurà el fitxer
Per defecte és *false*.

Escriu al fitxer la frase (i un intro).

- Si no posem el *close()* no es guarda res al fitxer.
- Convé posar-lo al *finally* per a que sempre tanque el fitxer encara que haja passat algun error no capturat.
- El *close()* també genera una excepció, que cal capturar.

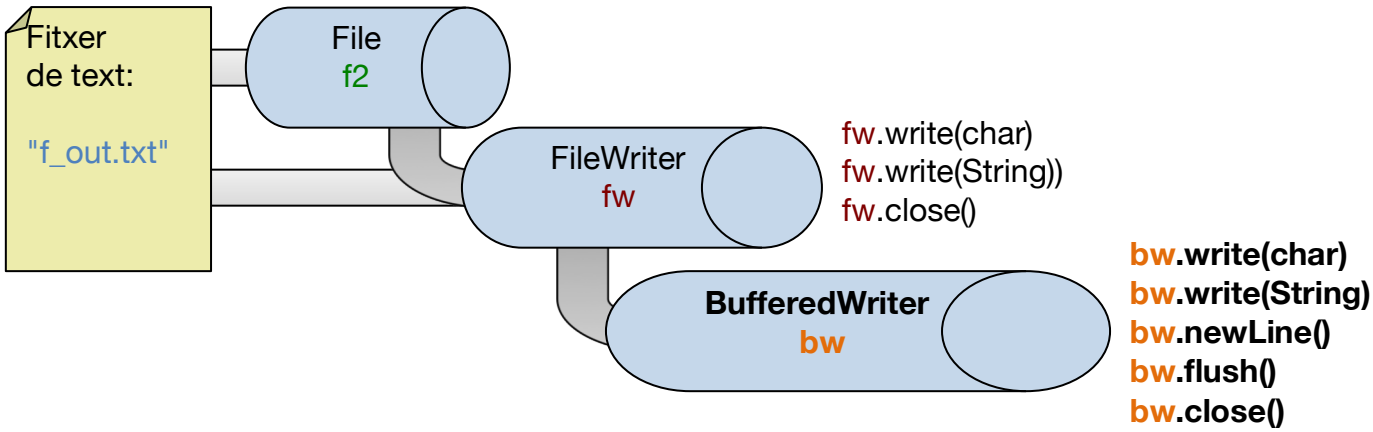
Abans d'escriure al fitxer convindria fer un control:

```
File f = new File(nomFitxer);
if ( f.exists() ) {
      System.out.println("El fitxer ja existeix. Vols sobreescriure'l?");
      ...
}
```

Exercici sobre escriptura de fitxers de text amb *FileWriter*

8. Funció *copiaFitxer* que rep 2 noms de fitxers de text i copia un damunt l'altre.

4.2 Escriptura amb *BufferedWriter*



Exemple: Demanar dades d'alumnes i anar escrivint-les a un fitxer de text.

```
FileWriter fw = null;
BufferedWriter bw = null;
System.out.println("Dades alumnes('fi' per a acabar):");

try {
   fw = new FileWriter(nomFitxer, true);
   bw = new BufferedWriter(fw);
   do {
      String nom = lligString("Nom:");
      String tel = lligString("Tel:");
      bw.write(nom);
      bw.newLine();
      bw.write(tel);
      bw.newLine();
      bw.newLine();
      bw.flush();

   } while (lligBoolean("Altre alumne?"));
} catch (IOException ex) {
   System.out.println("Error escrivint a fitxer");
} finally {
   try {
      bw.close();
      fw.close();
   } catch (IOException ex) {
      System.out.println("Error tancant el fitxer");
   }
}
```

Creem un *BufferedWriter* amb el constructor, a partir d'un *FileWriter*.

write(...) i *newLine()* escriuen al *buffer* (RAM).

flush() copia el *buffer* al fitxer (i es buida el *buffer*).
Si no l'usem es copiarà tot el *buffer* quan fem *bw.close()*.

Tanquem els 2 objectes d'escriptura.

El mètode close()

Les classes de lectura de fitxers (*FileReader* i *BufferedReader*) i les d'escriptura (*FileWriter* i *BufferedWriter*) usen una série de recursos que cal alliberar quan hem acabat tota la lectura i/o escriptura. Això es farà invocant el mètode *close()* dels respectius objectes de lectura/escriptura.

Ara bé, en l'escriptura té encara més sentit:

- L'escriptura amb la classe *FileWriter* no escriurà res al fitxer fins que no fem *close()*.
- L'escriptura amb la classe *BufferedWriter* no escriurà res al fitxser fins que no fem *flush()* o *close()*. Per tant, si no fem cap *flush()*, estem obligats a usar el *close()*. Usarem el flush() quan volem fer "bolcats parcials" al fitxer.

Ordre de tancament: primer es tanca l'últim en ser obert. Per tant:

```
bw.close();
fw.close();
```

Si després d'haver tancat un objecte de lectura/escriptura tornàrem a accedir a ell, donaria l'excepció *IOException* (*Stream closed*). Si volguérem tornar a llegir o escriure, caldria tornar a obrir l'objecte tornant a invocar el constructor.

Exercici sobre escriptura de fitxers de text amb *BufferedWriter*

9. Fes la funció *juntaFitxers*, que juntarà molts fitxers de text en un únic fitxer destí. Rep com a paràmetres un ArrayList de noms de fitxers i, a més, el nom del fitxer destí. Usa la classe *BufferedWriter* on, per cada fitxer copiat al *buffer* es passarà al fitxer, amb *flush()*.

5. Fitxers binaris. Accés directe

5.1. Format dels fitxers binaris

Els fitxers, a més de textos, poden tindre guardats **números** enters, números amb decimals, etc. Esta informació es guarda als fitxers en **format binari**. És a dir, abans de guardar-se una dada, es transforma a la seua representació binària.

5.2. Accés als fitxers binaris

En les formes vistes fins ara de llegir/escriure fitxers de text, la manera d'accedir a estos és **seqüencial**. És a dir: quan obrim un fitxer, el cursor (punt de lectura o escriptura) se situa al principi del fitxer i, conforme llegim o escrivim, el cursor va avançant.

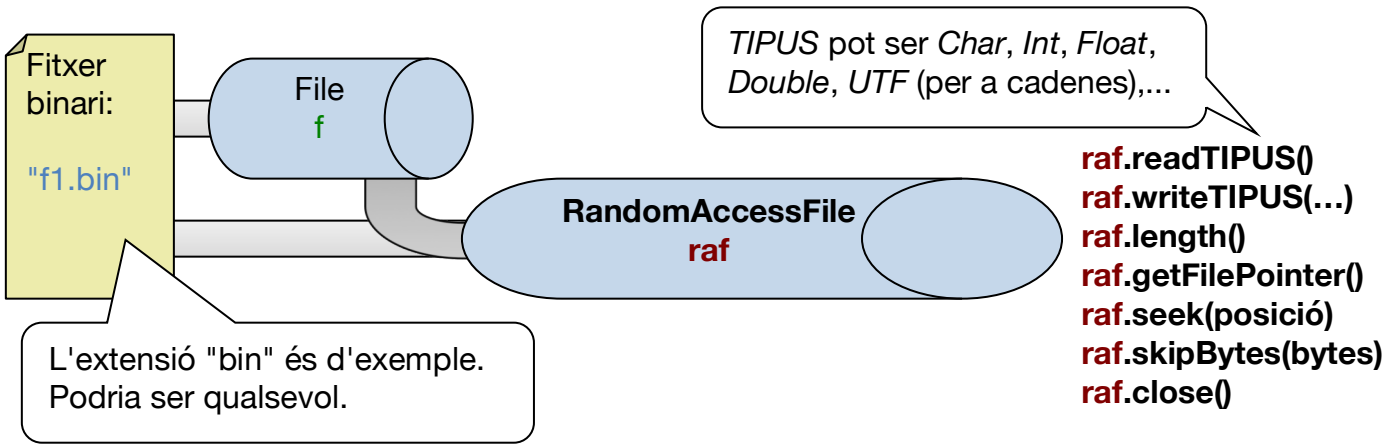
En canvi, els fitxers binaris tenen accés **directe**. Consistix en poder accedir a una posició determinada del fitxer sense haver de passar per les anteriors (els vectors també tenen accés directe). També s'anomena "accés aleatori", ja que el temps que es tarda en accedir a una determinada posició no depén de la quantitat d'elements que tinga davant, sinó que tarda un temps "aleatori".

Per posar un símil en la vida real, una cinta VHS té accés seqüencial, mentre que un CD té accés aleatori.

Cal tindre en compte que les dades que guardem en un fitxer d'esta forma, si intentem vore el contingut del fitxer en un editor de textos, possiblement vorem caràctes estranys, ja que la informació està guardada en binari i l'editor no sap interpretar-ho.

5.3. Lectura i escriptura en fitxers binaris

La classe *RandomAccessFile* ens permetrà l'accés a fitxers binaris (accés aleatori), tant per a llegir del fitxer com per a escriure. Cal importar *java.io.RandomAccessFile*.



Constructors:

Com veiem a l'esquema anterior, podem crear-lo a partir del nom d'un fitxer o a partir d'un objecte *File* ja creat:

```
File f = new File("f1.bin");
RandomAccessFile raf = new RandomAccessFile(f, mode);
```

```
RandomAccessFile raf = newRandomAccessFile("f1.bin", mode);
```

El "mode" és un *String* on indiquem si obrim el fitxer només per a lectura o també per a escriptura:

"r" → Només lectura (el fitxer ha d'existir)
"rw" → Lectura i escriptura (si no existeix el fitxer, el crea)

Mètodes:

MÈTODE	DESCRIPCIÓ
TIPUS readTIPUS()	Llig del fitxer el tipus *TIPUS*.
void writeTIPUS(...)	Escriu al fitxer el tipus *TIPUS*.
long length()	Torna la grandària del fitxer.
long getFilePointer()	Torna la posició del cursor del fitxer.
void seek(long pos)	Posiciona el cursor a la posició *pos* (des de l'inici del fitxer).
int skipBytes(int n)	Posiciona el cursor *n* bytes més a partir de la posició actual.
void close()	Tanca el fitxer.

Exemple:

```
RandomAccessFile raf = null;
try {
   raf = new RandomAccessFile(nomFitxer, "rw");

   raf.writeUTF(lligString("Nom:"));
   raf.writeInt(lligInt("Edat:"));
   raf.writeChar(lligChar("Lletra grup:"));

   raf.seek(0);

   System.out.println(raf.readUTF());
   System.out.println(raf.readInt());
   System.out.println(raf.readChar());

} catch (FileNotFoundException ex) {
   System.out.println("Fitxer no trobat");
} catch (IOException ex) {
   System.out.println("Error accedint al fitxer");
} finally {
   try {
      if (raf != null) {
         raf.close();
      }
   } catch (IOException ex) {
      System.out.println("Error tancant el fitxer");
   }
}
```

Amb només "r" haurien donat error els mètodes *write(...)*.

Podem escriure distints tipus de dades.

Com hem fet crides a *write(...)* el cursor estarà al final del fitxer. L'hem de posar a la posició 0 si volem llegir des del principi.

Hem de llegir les dades segons l'ordre en que hem escrit al fitxer, clar.

Convé tancar el fitxer per a alliberar recursos.

Exercicis sobre lectura i escriptura en fitxers binaris

10. Fes un programa que demane per teclat el teu nom, any de naixement, el número del NIF i la lletra del NIF. Caldrà vore si la lletra correspon amb el número (busca l'algorisme corresponent en Internet i implementa-ho). A continuació, guarda-ho en un fitxer.

11. Fes un programa que llija les dades del fitxer de l'exercici anterior i les mostre per pantalla.

12. Fes un programa que mostre el contingut d'un fitxer d'enters, que li afegisca més enters i que torne a mostrar el fitxer. Per a això:

13. Fitxer d'enters

a) Fes el mètode *creaFitxerEnters(...)*. Rep un ArrayList d'enters i un nom de fitxer. Ha de guardar els enters al fitxer usant un RandomAccessFile. Si el fitxer ja existia, afegirà els enters als que ja tenia el fitxer.

b) Fes el mètode *mostraFitxerEnters(...).* Rep un nom de fitxer (corresponent a un fitxer d'enters existent) El mètode ha de mostrar tots els números del fitxer per pantalla, separats per un tabulador.

c) Fes el mètode *modificarNumeroEnFitxer(...).* Rep un nom del fitxer de números. Es mostraran els enters del fitxer i es demanarà per teclat la posició que ocupa l'enter a modificar, es mostrarà el valor a modificar, es demanarà el nou valor i s'escriurà el nou valor en la posició indicada. Pistes:

   - ✓ La posició a demanar haurà de ser entre 1 i la quantitat d'enters del fitxer, que, com 1 enter ocupa 4 bytes, es calcula així: *fitxer.length() / 4*.
   - ✓ Caldrà posar el punter en la posició on està el número, llegir el número, tornar a posar el punter en la posició on està el número i escriure el nou número.

d) En el *main* invoca els mètodes anteriors per comprovar que funcionen.

14. Programa per a convertir una paraula a majúscules en un fitxer de text. Es demana una paraula, la busca en un fitxer de text i la modifica escrivint-la en majúsucules cada vegada que apareix en el fitxer. Pistes:

   - ✓ Com volem llegir i escriure en el fitxer, millor accedir al fitxer de text amb un *RandomAccesFile*, on llegirem el fitxer per línies.
   - ✓ Per a cada línia llegida, la modificarem usant els mètodes *replaceAll(...)* i *toUpperCase()* de la classe *String*.
   - ✓ Se sobreescriu al fitxer la línia modificada.

6. Fitxers d'objectes. Lectura i escriptura

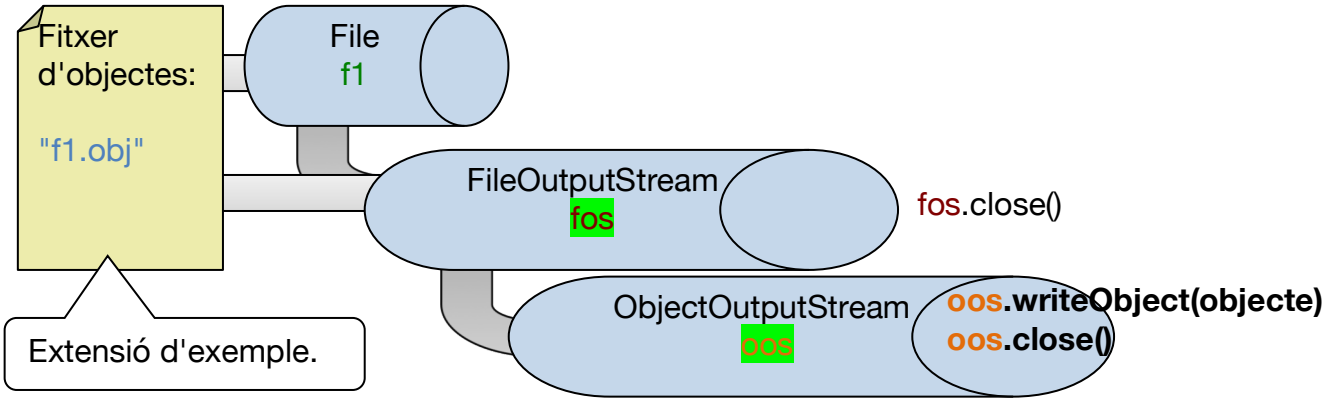
L'inconvenient de guardar la informació amb RandomAccessFile és que cal llegir en el mateix ordre en què hem escrit les dades.

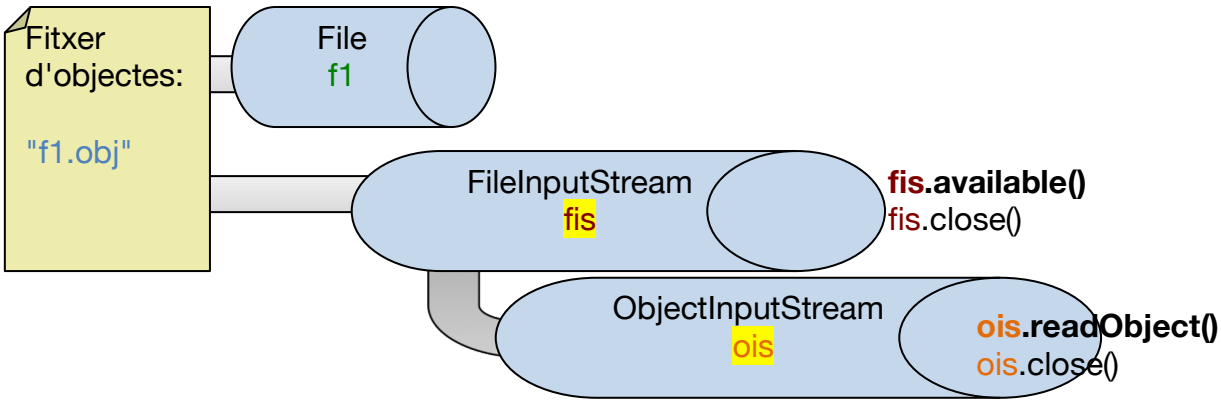
La solució passa per tindre una classe que ens permeta escriure objectes. Així escriurem per exemple, objectes de la classe Alumne, i llegirem objectes de la classe Alumne.

Per a fer això, **els objectes a escriure necessiten ser *serialitzats*** (que l'objecte es puga convertir en una seqüència de bytes). Simplement, hem d'indicar-ho en la definició de la classe. Així:

```
class Alumne implements java.io.Serializable {
        ...
}
```

En l'esquema següent tenim les classes que cal utilitzar per a utilitzar els mètodes de lectura i escriptura sobre fitxers d'objectes:





Caldrà importar les classes corresponents del paquet *java.io*.

Per a escriure objectes al fitxer

Cridarem al mètode ***writeObject()***, passant-li com a paràmetre l'objecte que volem escriure.

L'escriptura és seqüencial i destructiva (cada volta es crearà un nou fitxer).

Per a llegir del fitxer d'objectes

Cridarem al mètode ***readObject()***, que ens retorna un objecte però de la classe *Object*. Per tant, haurem de fer-li un casting per a convertir-lo al tipus d'objecte que estem llegint.

Ara bé, abans de llegir objectes, caldrà assegurar-se que en queden per llegir. Per a això, cridarem al mètode ***available()*** (compte! de la classe *FileInputStream*), el qual retorna la quantitat de bytes que falten per llegir al fitxer.

Exemple: escriure objectes de la classe *Alumne*, i llegir-los:

Primer que res haurem de fer que la classe *Alumne* implemente *Serializable*:

```
class Alumne implements java.io.Serializable {
        String nom;
        int edat;
        ...
}
```

I ara, en el main, per exemple, escriurem objectes de la classe Alumne a un fitxer.

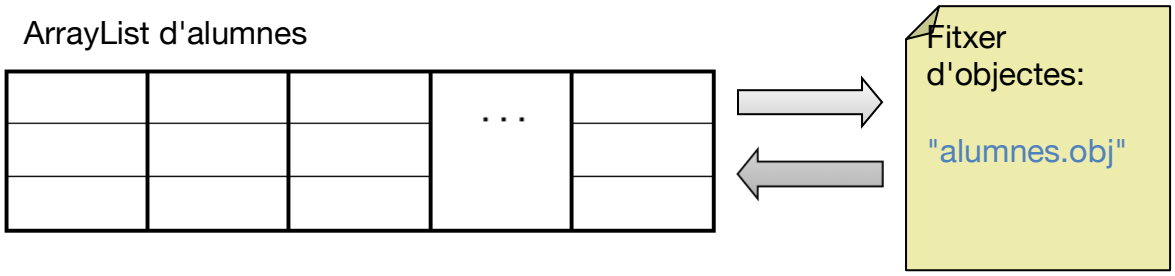
Després, els llegirem per a mostrar-los per pantalla.

```
public static void main(String arg[]) {

        Alumne alu1 = new Alumne("Pep", 10);
        Alumne alu2 = new Alumne("Pepa", 11);

        FileOutputStream fos = null;
        ObjectOutputStream oos = null;

        FileInputStream fis = null;
        ObjectInputStream ois = null;

        try {
            // ----------- ESCRIVIM AL FITXER --------

            fos = new FileOutputStream("alumnes.obj");
            oos = new ObjectOutputStream(fos);

            oos.writeObject(alu1);
            oos.writeObject(alu2);
            oos.writeObject(new Alumne("Pepet", 10));

            oos.close();

            // ----------- LLEGIM DEL FITXER --------

            fis = new FileInputStream("alumnes.obj");
            ois = new ObjectInputStream(fis);

            Alumne alu;
            while (fis.available()>0){
                alu = (Alumne) ois.readObject();
                System.out.println(alu);
            }

            ois.close();

        } catch (FileNotFoundException ex) {
            System.out.println("Fitxer no trobat");
        } catch (IOException ex) {
            System.out.println("Error accedint al fitxer");
        } catch (ClassNotFoundException ex) {
            System.out.println("Classe no trobada");
        }
}
```

Com treballem amb els fitxers d'objectes?

La idea és tindre en un fitxer tota la informació dels objectes (alumnes, per exemple) i poder inserir nous alumnes, modificar-los o esborrar-los, de forma que sempre estiga en memòria secundària per a continuar en una altra ocasió amb eixes dades.

La solució passa per fer un bolcat de memòria secundària a principal (RAM), fer les actualitzacions allí i després fer el bolcat de la RAM a memòria secundària.



Per tant:

- Abans de començar el programa, importar les dades del fitxer a un ArrayList.
- El programa farà insercions/modificacions o esborrats dels objectes en l'ArrayList.
- En acabar el programa, s'exportaran les dades de l'ArrayList al fitxer.

Exercici sobre lectura i escritura amb fitxers d'objectes

15. Volem tindre una agenda on pugam guardar el nom, telèfon i domicili de cada amic. Fes un programa amb un bucle i un menú amb les següents opcions:

    1. Donar d'alta un amic
    2. Mostrar l'agenda per pantalla
    3. Consultar un amic pel seu nom
    4. Esborrar un amic
    5. Modificar les dades d'un amic
    6. **Importar dades**
    7. **Exportar dades**

Part opcional

Demanarà nom de fitxer on estan els contactes i ho passarà a l'ArrayList.

Demanarà un nom de fitxer i guardarà allí les dades de l'ArrayList