

Programació

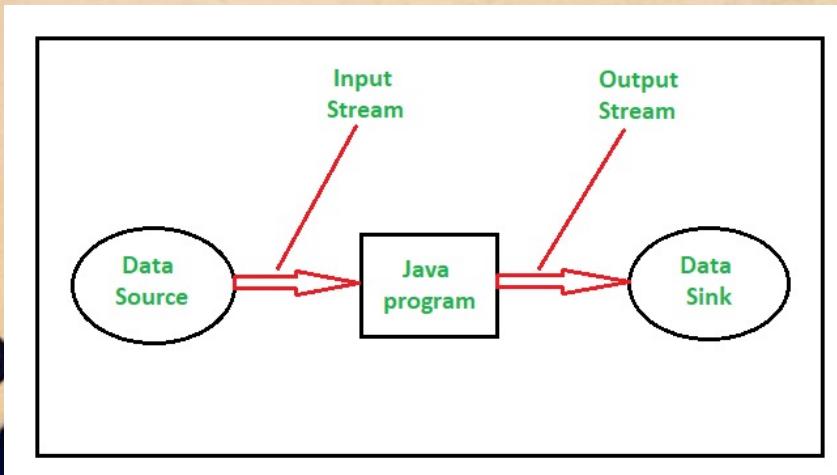


UT13.4 Java NIO

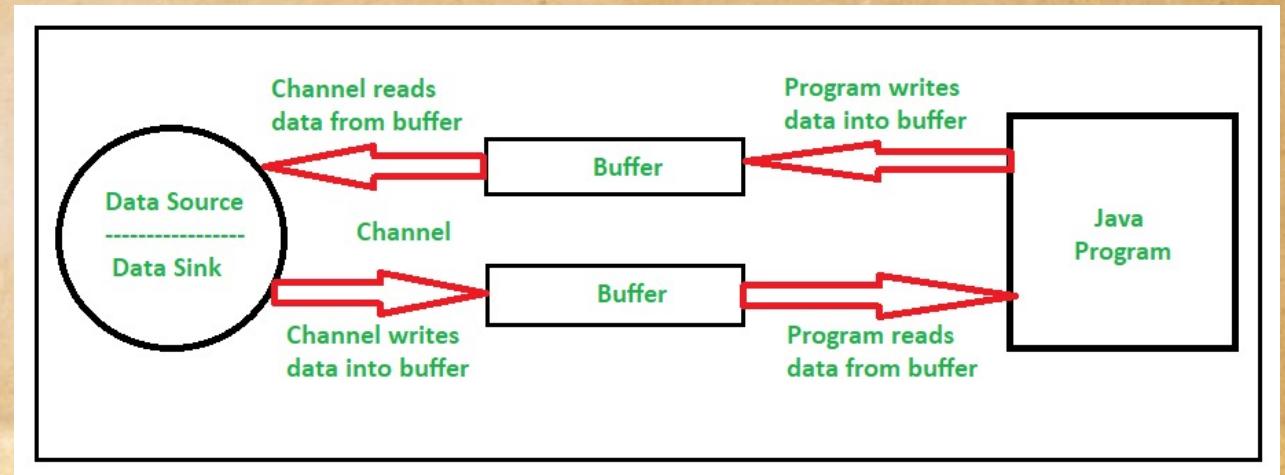
Introducció

Java IO (entrada/eixida) s'utilitza per realitzar operacions de lectura i escriptura. El paquet `java.io` conté totes les classes necessàries per al funcionament d'entrada i eixida. Mentre que, Java NIO (New IO) es va introduir per implementar **operacions d'IO d'alta velocitat**. És una alternativa a les API d'IO estàndard.

Java IO



Java NIO



Introducció

Utilitzem Java NIO per les dues raons principals següents:

Operació d'IO sense bloqueig: Java NIO realitza operacions d'IO sense bloqueig. Això vol dir que llegeix les dades que estiguuen preparades. Per exemple, un fil pot demanar a un canal que llegisca les dades d'un buffer i el fil pot realitzar altres treballs durant eixe període i continuar de nou des del punt anterior on ho ha deixat. Mentrestant, s'ha completat l'operació de lectura, la qual cosa augmenta l'eficiència global.

Enfocament orientat al buffer: l'enfocament orientat al buffer (la memòria intermèdia) de Java NIO ens permet avançar i retrocedir en la memòria intermèdia segons ho necessitem. Les dades es llegeixen en un buffer i s'emmagatzemen en forma de memòria cau (cache). Sempre que es requereixen les dades, es processen des del buffer.

Interfície “Path”

- Representa una ruta en el sistema de fitxers.
- Conté el nom de fitxer i la llista de directoris usada per a construir la ruta.
- Permet manejar diferents sistemes de fitxers (Windows, Linux, MacOS, ...)

Path fa referència a un directori, fitxer o link que tinguem dins del nostre sistema de fitxers.

Operacions amb “Path”

- Crear un path
 - Obtindre informació d'un path
 - Eliminar redundàncies
 - Unir dos paths
 - Comparar dos paths
- ...

Fes una ullada a l'API de “Path” per a comprendre millor els seus mètodes:

[API Interfície “Path”](#)

La creació d'un “Path” farà quasi sempre ús dels diferents mètodes estàtics de “Paths”

Exemple creació de “Path”

```
16  
17     public static void main(String[] args) {  
18         Path p1 = Paths.get("java", "dades.txt");  
19         Path p2 = Paths.get("java/dades.txt");  
20         Path p3 = FileSystems.getDefault().getPath("java", "dades.txt");  
21         Path p4 = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");  
22  
23         System.out.println(p1.toAbsolutePath().toString());  
24         System.out.println(p2.toAbsolutePath().toString());  
25         System.out.println(p3.toAbsolutePath().toString());  
26         System.out.println(p4.toAbsolutePath().toString());  
27     }  
28  
»
```

Output - NewIO (run) ×

```
run:  
/home/ciclost/NetBeansProjects/NewIO/java/dades.txt  
/home/ciclost/NetBeansProjects/NewIO/java/dades.txt  
/home/ciclost/NetBeansProjects/NewIO/java/dades.txt  
/home/ciclost/Documents/java/dades.txt  
BUILD SUCCESSFUL (total time: 1 second)
```

Exemple informació de “Path”

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");

System.out.println("toString:\t" + path.toString());
System.out.println("getFileName:\t" + path.getFileName());
System.out.println("getName(0):\t" + path.getName(0));
System.out.println("getNameCount:\t" + path.getNameCount());
System.out.println("subpath(0,2):\t" + path.subpath(0, 2));
System.out.println("getParent:\t" + path.getParent());
System.out.println("getParent x2:\t" + path.getParent().getParent());
System.out.println("getRoot:\t" + path.getRoot());
```

Path.Ruta > main >

t - NewIO (run) ×

```
run:
toString:      /home/ciclost/Documents/java/dades.txt
getFileName:   dades.txt
getName(0):   home
getNameCount: 5
subpath(0,2):  home/ciclost
getParent:    /home/ciclost/Documents/java
getParent x2: /home/ciclost/Documents
getRoot:      /
```

Exemple redundàncies en “Path”

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "...", "...", "dades.txt");
System.out.println(path.toString());

Path normalitzat = path.normalize();
System.out.println(normalitzat.toString());
```

- NewIO (run) ×

```
run:
/home/ciclost/Documents/java/.../.../dades.txt
/home/ciclost/dades.txt
```

Exemple unió de dos “Path”



The screenshot shows a Java code editor and a terminal window. The code editor displays a file named 'Ruta.java' with the following content:

```
17 public static void main(String[] args) {  
18  
19     Path base = Paths.get(System.getProperty("user.home"), "Documents", "java");  
20     Path fitxer = Paths.get("dades.txt");  
21  
22     Path complet = base.resolve(fitxer);  
23  
24     System.out.println(complet.toString());  
25  
26  
27 }
```

The terminal window below shows the output of running the program:

```
path.Ruta >  
Output - NewIO (run) ×  
run:  
/home/ciclost/Documents/java/dades.txt  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Exemple comparació de “Path”

```
public static void main(String[] args) {  
  
    Path p1 = Paths.get(System.getProperty("user.home"), "Documents", "dades.txt");  
    Path p2 = Paths.get(System.getProperty("user.home"), "Documents", "java", "..", "dades.txt");  
  
    System.out.print("Sense normalitzar: ");  
    if(p1.equals(p2)) System.out.println("Són iguals");  
    else System.out.println("No són iguals");  
  
    System.out.print("Normalitzats: ");  
    if(p1.normalize().equals(p2.normalize())) System.out.println("Són iguals");  
    else System.out.println("No són iguals");  
}
```

Ruta >

- NewIO (run) ×

run:

Sense normalitzar: No són iguals
Normalitzats: Són iguals
BUILD SUCCESSFUL (total time: 1 second)

Pregunta 1

Si tenim el següent path:

```
Path p = Paths.get("/home/export/tom/documents/coursefiles/JDK7");
```

I la instrucció:

```
Path sub = p.subPath(x, y);
```

Quins valors tindràn “x” i “y” per a retornar un Path que continga documents/coursefiles?

Pregunta 2

Quin serà el resultat d'executar el següent codi?

```
Path p1 = Paths.get("D:/temp/foo/");  
Path p2 = Paths.get("../bar/documents");  
Path p3 = p1.resolve(p2).normalize();  
System.out.println(p3);
```

Classe “Files”

Té desenes de mètodes estàtics per a fer múltiples operacions amb fitxers i directoris.

Provarem alguns dels seus mètodes.

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/nio/file/Files.html>

Exemple comprovacions amb “Files”

```
try {
    Path p1 = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");
    Path p2 = Paths.get(System.getProperty("user.home"), "Documents");

    System.out.println("Existeix p1: " + Files.exists(p1));
    System.out.println("Existeix p2: " + Files.exists(p2));

    System.out.println("Permís de lectura (p1): " + Files.isReadable(p1));
    System.out.println("Permís d'escriptura (p1): " + Files.isWritable(p1));
    System.out.println("Permís d'execució (p1): " + Files.isExecutable(p1));
    System.out.println("Fitxer ocult (p1): " + Files.isHidden(p1));
    System.out.println("És un directori (p1): " + Files.isDirectory(p1));
    System.out.println("És un directori (p1.getParent): " + Files.isDirectory(p1.getParent()));
    System.out.println("És un fitxer (p1): " + Files.isRegularFile(p1));
    System.out.println("És un fitxer (p1.getParent): " + Files.isRegularFile(p1.getParent()));

} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

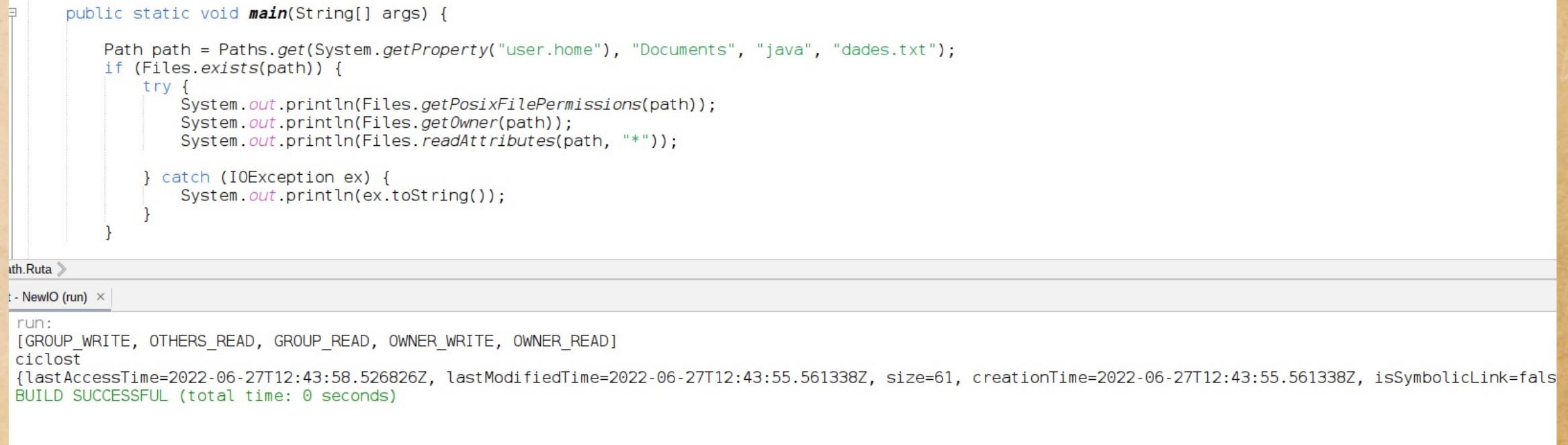
Path.Ruta > main >

ut - NewIO (run) >

```
run:
Existeix p1: true
Existeix p2: true
Permís de lectura (p1): true
Permís d'escriptura (p1): true
Permís d'execució (p1): false
Fitxer ocult (p1): false
És un directori (p1): false
És un directori (p1.getParent): true
És un fitxer (p1): true
És un fitxer (p1.getParent): false
```

Exemple comprovacions amb “Files”

També podem comprovar els permisos dels fitxers, el seu propietari i altres metadades:



The screenshot shows a Java code editor and a terminal window. The code in the editor is as follows:

```
public static void main(String[] args) {
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");
    if (Files.exists(path)) {
        try {
            System.out.println(Files.getPosixFilePermissions(path));
            System.out.println(Files.getOwner(path));
            System.out.println(Files.readAttributes(path, "*"));
        } catch (IOException ex) {
            System.out.println(ex.toString());
        }
    }
}
```

The terminal window below shows the output of running the code:

```
Ruta >
- NewIO (run) x
run:
[GROUP_WRITE, OTHERS_READ, GROUP_READ, OWNER_WRITE, OWNER_READ]
ciclost
{lastAccessTime=2022-06-27T12:43:58.526826Z, lastModifiedTime=2022-06-27T12:43:55.561338Z, size=61, creationTime=2022-06-27T12:43:55.561338Z, isSymbolicLink=false}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Crear fitxers i directoris

```
public static void main(String[] args) {  
  
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "proval.txt");  
    if(Files.notExists(path)){  
        try {  
            System.out.println("No existeix el path. Es crea...");  
            Files.createFile(path);  
            System.out.println("Fitxer creat");  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

- NewIO (run) ×
run:
No existeix el path. Es crea...
Fitxer creat
BUILD SUCCESSFUL (total time: 0 seconds)

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "practiques");  
if(Files.notExists(path)){  
    try {  
        System.out.println("No existeix el path. Es crea...");  
        Files.createDirectory(path);  
        System.out.println("Directori creat");  
    } catch (IOException ex) {  
        System.out.println(ex.toString());  
    }  
}
```

th.Ruta > main >

- NewIO (run) ×
run:
No existeix el path. Es crea...
Directori creat
BUILD SUCCESSFUL (total time: 0 seconds)

ERROR!

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "exemples", "text", "dades.txt");
if(Files.notExists(path)){
    try {
        System.out.println("No existeix el path. Es crea...");
        Files.createFile(path);
    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
}
```

t - NewIO (run) ×

```
run:
No existeix el path. Es crea...
java.nio.file.NoSuchFileException: /home/ciclost/Documents/exemples/text/dades.txt
```

El mètode Files.createFile ens dona un error, ja que no existeix la ruta on es vol crear “dades.txt”

Crear fitxers i directoris



Podem crear la ruta de directoris directament amb el mètode Files.createDirectories:

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "exemples", "text", "dades.txt");
if(Files.notExists(path)){
    try {
        System.out.println("No existeix el path. Es crea...");
        // Prime es crea la ruta de directoris on volem guardar "dades.txt"
        Files.createDirectories(path.getParent());
        System.out.println("Ruta creada");
        // Posteriorment creem el fitxer
        Files.createFile(path);
        System.out.println("Fitxer creat");
    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
}
```

Ruta.Ruta > main >

ut - NewIO (run) x

```
run:
No existeix el path. Es crea...
Ruta creada
Fitxer creat
BUILD SUCCESSFUL (total time: 0 seconds)
```

Eliminar fitxers i directoris

```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "proval.txt");
if(Files.exists(path)){
    // També disposem del mètode "deleteIfExists(path)"
    try {
        System.out.println("Existeix el path. A esborrar-lo...");
        Files.delete(path);
        System.out.println("Fitxer esborrat");

    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
}
```

Path.Ruta > main

ut - NewIO (run) ×

```
run:
Existeix el path. A esborrar-lo...
Fitxer esborrat
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "practiques");
if(Files.exists(path)){
    // També disposem del mètode "deleteIfExists(path)"
    try {
        System.out.println("Existeix el path. A esborrar-lo...");
        Files.delete(path);
        System.out.println("Directori esborrat");

    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
}
```

Path.Ruta > main

ut - NewIO (run) ×

```
run:
Existeix el path. A esborrar-lo...
Directori esborrat
BUILD SUCCESSFUL (total time: 0 seconds)
```



ERROR!



```
Path path = Paths.get(System.getProperty("user.home"), "Documents", "exemples");
if(Files.exists(path)){
    // També disposem del mètode "deleteIfExists(path)"
    try {
        System.out.println("Existeix el path. A esborrar-lo...");
        Files.delete(path);
        System.out.println("Directori esborrat");
    } catch (IOException ex) {
        System.out.println(ex.toString());
    }
}

ath.Ruta > main >
it - NewIO (run) x
run:
Existeix el path. A esborrar-lo...
java.nio.file.DirectoryNotEmptyException: /home/ciclost/Documents/exemples
BUILD SUCCESSFUL (total time: 0 seconds)
```

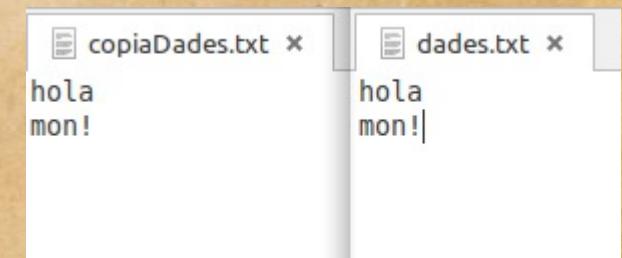
*El mètode `Files.delete` ens dona un error, ja que el directori a eliminar no està buit.
Abans de borrar un directori hem d'eliminar el seu contingut.*

Copiar fitxers i directoris

- Podem copiar fitxers o directoris amb el mètode `copy(Path, Path, CopyOption...)`
- Quan copiem un directori, els fitxers i carpetes que conté al seu interior no seran copiats.

```
public static void main(String[] args) {  
  
    Path orige = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");  
    Path desti = Paths.get(System.getProperty("user.home"), "Documents", "java", "copiaDades.txt");  
    if(Files.exists(orige)){  
        // També disposem del mètode "deleteIfExists(path)"  
        try {  
            System.out.println("Existeix el path. A copiar-lo...");  
            Files.copy(orige, desti, StandardCopyOption.REPLACE_EXISTING, StandardCopyOption.COPY_ATTRIBUTES);  
            System.out.println("Fitxer copiat");  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

Path.Ruta >
ut - NewIO (run) x
run:
Existeix el path. A copiar-lo...
Fitxer copiat
BUILD SUCCESSFUL (total time: 1 second)



Moure fitxers i directoris

- Podem moure i reanomenar fitxers o directoris amb el mètode `move(Path, Path, CopyOption...)`

```
public static void main(String[] args) {  
  
    Path orige = Paths.get(System.getProperty("user.home"), "Documents", "java", "dades.txt");  
    Path desti = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes.txt");  
    if(Files.exists(orige)){  
        try {  
            System.out.println("Existeix el path. A moure'l...");  
            Files.move(orige, desti, StandardCopyOption.REPLACE_EXISTING);  
            System.out.println("Fitxer mogut");  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
  
}
```

h.Ruta >
- NewIO (run) ×
run:
Existeix el path. A moure'l...
Fitxer mogut
BUILD SUCCESSFUL (total time: 0 seconds)

Llegir el contingut d'un fitxer

- Podem llegir tots els bytes (readAllBytes) o bé totes les línies d'un fitxer (readAllLines)

The screenshot shows a Java application running in an IDE. The code in the editor reads a file named 'notes.txt' located at the user's home directory. It prints a message if the file exists, reads all lines from it, and prints each line. If an IOException occurs, it prints the exception's string representation.

```
public static void main(String[] args) {  
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes.txt");  
    if (Files.exists(path)) {  
        try {  
            System.out.println("Existeix el path. A llegir-lo...");  
            List<String> llinies = Files.readAllLines(path);  
            System.out.println("Fitxer llegit:");  
            for (String llinia : llinies) {  
                System.out.println(llinia);  
            }  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

The terminal window shows the output of the program:

```
run:  
Existeix el path. A llegir-lo...  
Fitxer llegit:  
hola  
mon!  
Això és un fitxer  
BUILD SUCCESSFUL (total time: 0 seconds)
```

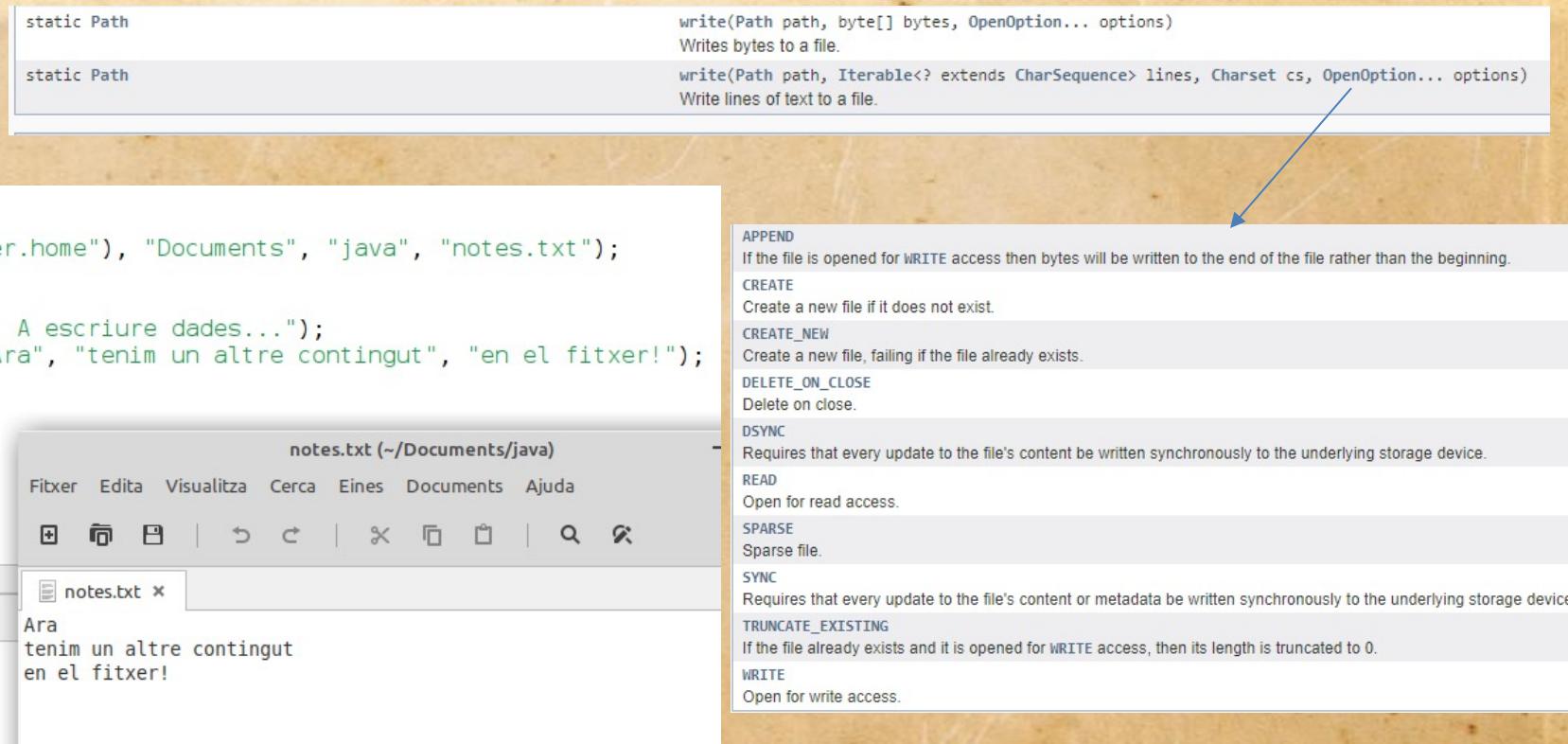
A separate window titled 'notes.txt (~/Documents/java)' shows the contents of the file 'notes.txt':

```
notes.txt (~/Documents/java)  
Fitxer Edita Visualitza Cerca Eines Documents Ajuda  
+ - NewIO (run) x  
notes.txt x  
hola  
mon!  
Això és un fitxer|
```

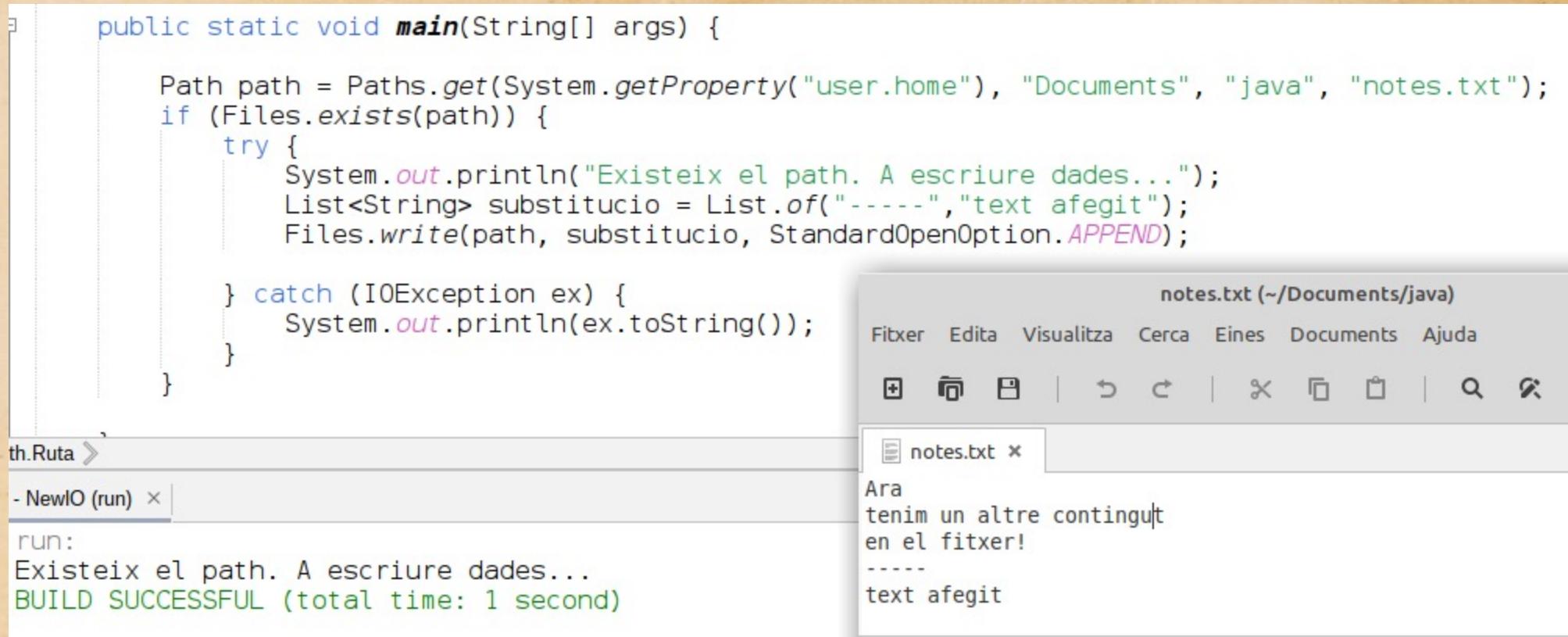
Escriure contingut en un fitxer

- Utilitzarem el mètode `write`

```
public static void main(String[] args) {  
  
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes.txt");  
    if (Files.exists(path)) {  
        try {  
            System.out.println("Existeix el path. A escriure dades...");  
            List<String> substitucio = List.of("Ara", "tenim un altre contingut", "en el fitxer!");  
            Files.write(path, substitucio);  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}  
  
th.Ruta >  
- NewIO (run) x  
run:  
Existeix el path. A escriure dades...  
BUILD SUCCESSFUL (total time: 1 second)
```



Escriure contingut en un fitxer



Escriure contingut en un fitxer

ERROR!

```
public static void main(String[] args) {  
  
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes3.txt");  
    if (Files.notExists(path)) {  
        try {  
            System.out.println("Existeix el path. A escriure dades...");  
            List<String> substitucio = List.of("-----", "text afegit");  
            Files.write(path, substitucio, StandardOpenOption.APPEND);  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

th.Ruta >
- NewIO (run) ×
run:
Existeix el path. A escriure dades...
java.nio.file.NoSuchFileException: /home/ciclost/Documents/java/notes3.txt

SOLUCIÓ!

```
public static void main(String[] args) {  
  
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes3.txt");  
    if (Files.notExists(path)) {  
        try {  
            System.out.println("Existeix el path. A escriure dades...");  
            List<String> substitucio = List.of("-----", "text afegit");  
            Files.write(path, substitucio, StandardOpenOption.APPEND, StandardOpenOption.CREATE);  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

th.Ruta >
- NewIO (run) ×
run:
Existeix el path. A escriure dades...
BUILD SUCCESSFUL (total time: 0 seconds)

Modificar metadades

Amb el mètode `setLastModifiedTime` podem modificar la data de l'última modificació

```
public static void main(String[] args) {
    Path path = Paths.get(System.getProperty("user.home"), "Documents", "java", "notes.txt");
    if (Files.exists(path)) {
        try {
            System.out.println("Ultima modificació inicial: " + Files.getLastModifiedTime(path));
            LocalDateTime dataLocal = LocalDateTime.of(1998, 5, 30, 22, 30, 25);
            FileTime dataFitxer = FileTime.from(dataLocal.toInstant(ZoneOffset.UTC));
            Files.setLastModifiedTime(path, dataFitxer);
            System.out.println("Ultima modificació final: " + Files.getLastModifiedTime(path));

        } catch (IOException ex) {
            System.out.println(ex.toString());
        }
    }
}

th.Ruta >
- NewIO (run) x
run:
Ultima modificació inicial: 2022-06-27T12:43:55.561338Z
Ultima modificació final: 1998-05-30T22:30:25Z
BUILD SUCCESSFUL (total time: 1 second)
```

Modificar metadades

Amb el mètode `setOwner` podem modificar el propietari del fitxer

```
public static void main(String[] args) {
    Path path = Paths.get("/home/ciclost/Documents/java/notes.txt");
    Path pathDeRoot = Paths.get("/");
    if (Files.exists(path)) {
        try {
            System.out.println("Propietari inicial: " + Files.getOwner(path));
            Files.setOwner(path, Files.getOwner(pathDeRoot));
            System.out.println("Propietari final: " + Files.getOwner(path));
        } catch (IOException ex) {
            System.out.println(ex.toString());
        }
    }
}

h.Ruta >
- NewIO (run) ×
run:
Propietari inicial: ciclost
Propietari final: root
BUILD SUCCESSFUL (total time: 1 second)
```

Modificar metadades

Amb el mètode `setPosixFilePermissions` podem modificar els permisos del fitxer

```
public static void main(String[] args) {
    Path path = Paths.get("/home/ciclost/Documents/java/notes.txt");
    if (Files.exists(path)) {
        try {
            System.out.println("Permisos iniciais: " + Files.getPosixFilePermissions(path).toString());
            Files.setPosixFilePermissions(path, PosixFilePermissions.fromString("rw-rw-rw-"));
            System.out.println("Permisos finals: " + Files.getPosixFilePermissions(path).toString());
        } catch (IOException ex) {
            System.out.println(ex.toString());
        }
    }
}

Ruta >
- NewIO (run) ×
run:
Permisos iniciais: [GROUP_WRITE, OTHERS_READ, GROUP_READ, OWNER_WRITE, OWNER_READ]
Permisos finals: [OTHERS_WRITE, GROUP_WRITE, OTHERS_READ, GROUP_READ, OWNER_WRITE, OWNER_READ]
BUILD SUCCESSFUL (total time: 1 second)
```

Llistar el contingut d'un directori

- La classe `DirectoryStream` proporciona un mecanisme per iterar sobre totles entrades d'un directori.
- Per a comprendre millor el funcionament has de conèixer la programació funcional

```
public static void main(String[] args) {  
    Path path = Paths.get("/home/ciclost/Documents/java/");  
    if (Files.exists(path)) {  
        try {  
            // Mostra tot el contingut de la carpeta de forma recursiva  
            Files.walk(path).forEach(System.out::println);  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

Path.Ruta >
- NewIO (run) ×
run:
/home/ciclost/Documents/java
/home/ciclost/Documents/java/notes2.txt
/home/ciclost/Documents/java/copiaDades.txt
/home/ciclost/Documents/java/notes3.txt
/home/ciclost/Documents/java/notes.txt
/home/ciclost/Documents/java/examples
/home/ciclost/Documents/java/examples/text
/home/ciclost/Documents/java/examples/text/dades.txt
/home/ciclost/Documents/java/examples/aa
BUILD SUCCESSFUL (total time: 0 seconds)

```
public static void main(String[] args) {  
    Path path = Paths.get("/home/ciclost/Documents/java/");  
    if (Files.exists(path)) {  
        try {  
            // Recorre tot el contingut de la carpeta de forma recursiva  
            // I mostra només els fitxers regulars que no estiguin buits  
            Files.walk(path)  
                .filter(Files::isRegularFile)  
                .filter(p -> p.toFile().length() > 0L)  
                .forEach(System.out::println);  
  
        } catch (IOException ex) {  
            System.out.println(ex.toString());  
        }  
    }  
}
```

- NewIO (run) ×
run:
/home/ciclost/Documents/java/notes2.txt
/home/ciclost/Documents/java/copiaDades.txt
/home/ciclost/Documents/java/notes3.txt
/home/ciclost/Documents/java/notes.txt
BUILD SUCCESSFUL (total time: 0 seconds)

Conversió File <->Path

- Finalment, podrem convertir objectes de tipus File a tipus Path, i a l'inrevés.

```
File file = onePath.toFile();
Path path = oneFile.toPath();
```