Module
II.102 Algorithme et Programmation

Manipulation des fichiers et répertoires

isep
École d'ingénieurs du numérique

Sommaire

- Lister le contenu dans un repértoire
- Lecture du contenu d'un fichier
- Ecriture dans un fichier

Lister le contenu d'un repértoire

Un répertoire contient une liste de nom de fichiers ou de répertoires (fils ou sous-répertoires)

- possède un nom,
- appartient à un utilisateur propriétaire,
- est accessible ou non, en lecture ou en écriture, relativement à son propriétaire ou aux autres utilisateurs,
- accessible en lecture, on peut lire la liste des noms qu'il contient.

Lister le contenu d'un repértoire

```
import java.io.*;

public class ListerRepertoire {

    public static void main(String args[]) {
        File repertoire = new File(args[0]);
        String liste[] = repertoire.list();

        if (liste != null) {
            for (int i = 0; i < liste.length; i++) {
                System.out.println(liste[i]);
            }
        } else {
            System.err.println("Nom de repertoire invalide");
        }
    }
}
```

Lecture d'un contenu dans un fichier

- Java propose plusieurs méthodes pour lire des fichiers.
- Chacune de ces méthodes convient à la lecture de différents types de fichiers dans différentes situations.
- Certains sont meilleurs pour lire des fichiers plus longs, d'autres pour lire des fichiers plus courts, etc.
- La lecture peut se faire par :
  - BufferedReader,
  - Files,
  - Scanner.

Lecture d'un contenu dans un fichier BufferedReader

- La classe BufferedReader lit un flux de saisie de caractères. Idéale pour la lecture d'un fichier ligne par ligne.

```
import java.io.*;
public class FileReaderWithBufferedReader {
    public static void main(String[] args) throws IOException{
        String file = "src/file.txt";
        BufferedReader bufferedReader = new BufferedReader(new FileReader(file));

        String curLine;
        while ((curLine = bufferedReader.readLine()) != null){
            //process the line as required
            System.out.println(curLine);
        }
        bufferedReader.close();
    }
}
```

Lecture d'un contenu dans un fichier

Class Files

- La classe Java Files se compose entièrement de méthodes statiques qui opèrent sur les fichiers.
- En utilisant la classe Files, nous pouvons lire le contenu complet d'un fichier dans un tableau. Cela en fait un bon choix pour lire des fichiers plus petits.

```
import java.io.IOException;
import java.nio.file.*;
import java.util.*;

public class SmallFileReaderWithFiles {

    public static void main(String[] args) throws IOException {
        String file = "src/file.txt";
        Path path = Paths.get(file);
        List<String> lines = Files.readAllLines(path);
    }
}
```

Lecture d'un contenu dans un fichier

Class Files : BufferedReader

- Si nous souhaitons lire un fichier volumineux avec la classe Files, on peut utiliser la méthode newBufferedReader() pour obtenir une instance de la classe BufferedReader et lire le fichier ligne par ligne à l'aide d'un BufferedReader.

```
import java.io.*;
import java.nio.file.*;
public class LargeFileReaderWithFiles {
    public static void main(String[] args) throws IOException {
        String file = "src/file.txt";
        Path path = Paths.get(file);
        BufferedReader bufferedReader =
Files.newBufferedReader(path);
        String curLine;
        while ((curLine = bufferedReader.readLine()) != null){
            System.out.println(curLine);
        }
        bufferedReader.close();
    }
}
```

Lecture d'un contenu dans un fichier

Class Files : Files.lines()

- Une méthode dans la classe Files pour lire l'intégralité du fichier dans un Stream de chaînes.

```
import java.io.IOException;
import java.nio.file.*;
import java.util.stream.Stream;

public class FileReaderWithFilesLines {

    public static void main(String[] args) throws IOException {
        String file = "src/file.txt";
        Path path = Paths.get(file);
        Stream<String> lines = Files.lines(path);

        lines.forEach(s -> System.out.println(s));
        lines.close();
    }
}
```

Lecture d'un contenu dans un fichier

Lecture avec Scanner

La classe Scanner divise le contenu d'un fichier en parties à l'aide d'un délimiteur donné et le lit partie par partie. Cette approche est la mieux adaptée à la lecture de contenu séparé par un délimiteur.

```
import java.io.IOException;
import java.util.Scanner;
import java.io.File;

public class FileReaderWithScanner {
    public static void main(String[] args) throws IOException{
        String file = "src/file.txt";
        Scanner scanner = new Scanner(new File(file));
        scanner.useDelimiter(" ");

        while(scanner.hasNext()){
            String next = scanner.next();
            System.out.println(next);
        }
        scanner.close();
    }
}
```

Ecriture d'un contenu dans un fichier

Quatre options possibles pour l'opération de l'écriture d'un contenu dans un fichier avec Java.

- FileWriter : c'est le moyen le plus simple d'écrire un fichier en Java. Il fournit une méthode d'écriture surchargée pour écrire un entier, un tableau d'octets et une chaîne dans le fichier.
- BufferedWriter : BufferedWriter est presque similaire à FileWriter mais il utilise un tampon interne pour écrire des données dans un fichier.
- FileOutputStream : FileWriter et BufferedWriter sont destinés à écrire des données de flux brutes dans un fichier
- Class Files : Nous pouvons écrire un fichier en utilisant sa fonction 'Write'. En interne, il utilise OutputStream pour écrire un tableau d'octets dans un fichier

Ecriture d'un contenu dans un fichier

BufferedWriter

```
public static void WriteToFile{
        String content = "Content to write to file";
        //Name and path of the file
        File file = new File("writefile.txt");

        if(!file.exists()){
            file.createNewFile();
        }

        FileWriter fw = new FileWriter(file);

        BufferedWriter bw = new BufferedWriter(fw);
        bw.write(content);
        bw.close();

    }
```

Ecriture d'un contenu dans un fichier

Files Class

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
public class WriteToFile {
    public static void main( String[] args ) {
        Path path = Paths.get("writefile.txt");
        String content = "Content to write to file";
        try {
            byte[] bytes = content.getBytes();
            Files.write(path, bytes);
        } catch(IOException ex) {
            System.out.println("Exception occurred:");
            ex.printStackTrace();
        }
    }
}
```