Université Mohamed V Rabat Année 2019/2020

Ecole Supérieure de Technologie de Salé

Département d'informatique

**TD3: java**

Fichiers & BD

**GL 2**

**Partie 1 : Les fichiers**

**Exercice 1 : Les fluts de caractères**

1. **Lecture/écriture des caractères :** FileWriter et FileReader
2. En utilisant la classe FileWriter, écrire un programme java qui lit une chaine de caractères et la copie dans un fichier texte.
3. Afficher le contenu du fichier.

class CreateFile {

    public static void main(String[] args) throws IOException     {

        // Accept a string

        String str = "FileWriter and FileReader";

        FileWriter fw=new FileWriter("output.txt");

        // read character from string and write  into FileWriter

        for (int i = 0; i < str.length(); i++)

            fw.write(str.charAt(i));

        System.out.println("Writing successful");

        fw.close();

    }

}

class ReadFile {

    public static void main(String[] args) throws IOException {

        int ch;

        // check if File exists or not

        FileReader fr=null;

        try   {

            fr = new FileReader("text");

        }

        catch (FileNotFoundException fe) {

            System.out.println("File not found");

        }

        while ((ch=fr.read())!=-1)

            System.out.print((char)ch);

        fr.close();

    } }

1. Afficher 5 caractères à partir du 3ème.
2. Afficher les deux premiers caractères, faites un saut de 3 caractères puis afficher les autres caractères.

Tenir compte des exception IOException et FileNotFoundException.

1. **Lecture / écriture ligne par ligne : BuferedReader** et **BuferedWriter, printWriter**
2. Copier des chaines de caractères dans un fichier texte.
3. Afficher le contenu de ce fichier.

**public** **class** BufferedWriterExample {

**public** **static** **void** main(String[] args) **throws** Exception {

    String s;

FileWriter writer = **new** FileWriter("D:\\testout.txt");

    BufferedWriter buffer = **new** BufferedWriter(writer);

    s=texte1.getText();

buffer.write(s);

    buffer.close();

    System.out.println("Success");

    }

}

FileReader reader = **new** FileReader ("D:\\testout.txt");

    BufferedReader buffer = **new** BufferedReader (reader);

while ((String s = in.readLine()) != null)

System.out.println(s);

    buffer.close();

    System.out.println("Success");

}

**Exercice 2 : Les flux d'octets : InputStream ou OutputStream.**

1. **Octets simples**

Les classes **FileInputStream** et **FileOutputStream** offrent les primitives de lecture/écriture de byte.

* Copier un tableau d'octets dans un fichier.
* Afficher le contenu de ce fichier.

**class** IOTest{

String str = "Hello www.codesjava.com";

**public** **void** writeFile(){

**try** {

FileOutputStream fos = **new** FileOutputStream("F:**\\**New folder**\\**data2.txt");

**byte** b[]=str.getBytes();

fos.write(b);

fos.flush();

fos.close();

System.out.println("Contents written successfully.");

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**class** IOTest{

**public** **void** readFile(){

**try** {

FileInputStream fis = **new** FileInputStream("F:**\\**New folder**\\**data1.txt");

**int** i;

**while**((i=fis.read())!=-1){

System.out.print((**char**)i);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

1. **Flux d'octets bufférisés : DataInputStream** et **DataOutputStream**
2. Remplir un fichier d'octets par des nombres réels puis afficher le contenu de ce fichier.

Ces classes permettent la lecture/écriture des types primitifs (int, double, boolean, char, ...) dans leur représentation en byte.

FileInputStream ne vous donne qu'une interface très basique. Lorsque vous voulez lire des chiffres, des chaînes (ou même des objets complexes) plutôt que des octets, c'est très pénible.

1. Etant donné les informations : numéro, nom, prénom, âge et salaire.

* Copier ces données dans un fichier.
* afficher le contenu de ce fichier

class DataInputStreamDemo{

    public static void main(String args[]) throws IOException  {

try {

DataOutputStream dout=new DataOutputStream(new FileOutputStream("fil.dat"))

            dout.writeDouble(1.1);

            dout.writeInt(55);

            dout.writeBoolean(true);

            dout.writeChar('4');

        }

        catch(FileNotFoundException ex) {

            System.out.println("Cannot Open the Output File");

            return;

        }

        // reading the data back.

        try {

DataInputStream din = new DataInputStream(new FileInputStream("fil.dat")) )

            double a = din.readDouble();

           int b = din.readInt();

            boolean c = din.readBoolean();

            char d=din.readChar();

            System.out.println("Values: " + a + " " + b + " " + c+" " + d);

        }

        catch(FileNotFoundException e)  {

            System.out.println("Cannot Open the Input File");

            return;

        }

    }

**Exercice 3 : Fichiers à accès direct : classe RamdonAccessFile**

* Copier le contenu du fichier précédent dans un fichier à accès direct.
* Afficher ce contenu
* Aller à la position 5 puis afficher l'élément.

**Exercice 4 : La classe File**

* Créer un dossier
* Créer des fichiers dans ce dossier.
* Afficher le contenu d'un dossier.
* Supprimer un fichier de ce dossier

**Exercice 5 : Sérialisation**

* Copier une collection de personnes dans un fichier sachant que le salaire est secret et ne doit pas figurer dans le fichier.

**Partie 2 : JDBC**

**Exercice 6 : inscription des étudiants**

1. Créer un formulaire d'inscription des étudiants (attention aux étudiants déjà inscrits).
2. Rechercher un étudiant
3. Supprimer un étudiant
4. Mise à jour d'un étudiant
5. Afficher la liste des étudiants