

Youssef SAADE

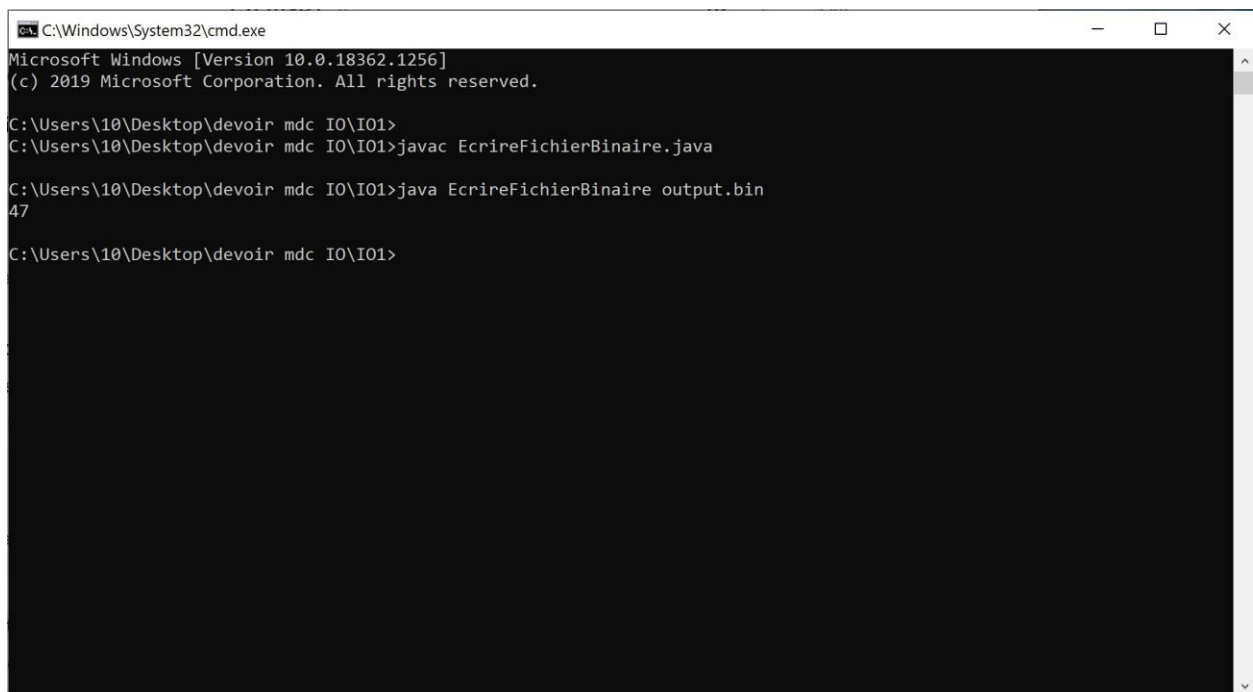
220958

Le code de ce devoir est present dans le Github repository dont le lien est le suivant :

[https://github.com/Yousseffsaade/USJ\\_devoirs\\_mdc\\_Youssef\\_SAADE\\_2024.git](https://github.com/Yousseffsaade/USJ_devoirs_mdc_Youssef_SAADE_2024.git)

## Partie 1:

### 1-1



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

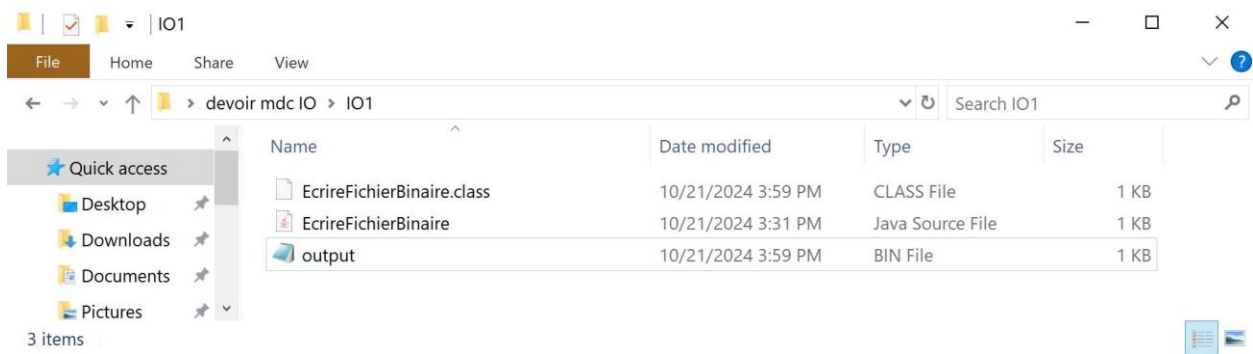
C:\Users\10\Desktop\devoir mdc IO\IO1>
C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierBinaire.java

C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierBinaire output.bin
47

C:\Users\10\Desktop\devoir mdc IO\IO1>
```

**Résultat de l'exécution :** Le programme écrit des données binaires dans un fichier, incluant des types de données comme des chaînes de caractères, des entiers, des nombres à virgule flottante, et des booléens. La taille du fichier est imprimée à la fin.

## 1-2



### Lecture du fichier binaire :

Voici ce que contient le fichier binaire :



### Commentaire:

Le fichier créé contient des données binaires, et non des données textuelles lisibles. Lorsqu'on ouvre ce fichier avec un éditeur de texte comme Notepad, on voit des caractères spéciaux ou illisibles, car les types de données (comme int, float, double, etc.) sont stockés en binaire. Chaque type de données a une représentation spécifique dans le fichier, qui n'est pas directement interprétable sous forme de texte.


Dans la screenshot, on peut voir que certains segments comme "bonjour" et "au revoir" sont visibles car ils ont été écrits avec la méthode `writeUTF()`, qui encode des chaînes de caractères en binaire, mais certains autres caractères ne sont pas lisibles, comme les entiers ou les nombres à virgule flottante.

### 1-3

**Classe et méthodes à utiliser pour lire ce fichier :** Pour lire ce fichier, on peut utiliser la classe `DataInputStream` avec des méthodes comme `readUTF()`, `readInt()`, `readLong()`, etc., qui permettent de lire des types de données spécifiques à partir du fichier binaire.

### 1-4

**Programme qui permet de lire le contenu de ce fichier binaire :**



```
File Edit Format View Help
LireFichierBinaire - Notepad
import java.io.*;

class LireFichierBinaire {
    public static void main(String[] argv) throws IOException {
        DataInputStream lecteur = new DataInputStream(new BufferedInputStream(new FileInputStream(argv[0])));

        System.out.println(lecteur.readUTF());
        System.out.println(lecteur.readInt());
        System.out.println(lecteur.readLong());
        System.out.println(lecteur.readFloat());
        System.out.println(lecteur.readDouble());
        System.out.println(lecteur.readChar());
        System.out.println(lecteur.readBoolean());
        System.out.println(lecteur.readUTF());

        lecteur.close();
    }
}
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

**Execution :**

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierBinaire.java

C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierBinaire output.bin
47

C:\Users\10\Desktop\devoir mdc IO\IO1>javac LireFichierBinaire.java

C:\Users\10\Desktop\devoir mdc IO\IO1>java LireFichierBinaire output.bin
bonjour
3
100000
2.0
3.5
a
false
au revoir

C:\Users\10\Desktop\devoir mdc IO\IO1>
```

## Partie 2 :

### a-1 Rôle des classes :

InputStreamReader : Elle convertit les octets en caractères. Elle est utilisée pour lire des données d'une source d'entrée comme le clavier.

StringTokenizer : Cette classe permet de découper une chaîne de caractères en plusieurs morceaux (ou "tokens"), souvent à partir d'un séparateur comme l'espace.

### b- execution (avec `ecrivain.close()` )

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierTexte.java
EcrireFichierTexte.java:22: warning: [removal] Integer(int) in Integer has been deprecated and marked for removal
    ecrivain.println(new Integer(36));
                        ^
1 warning

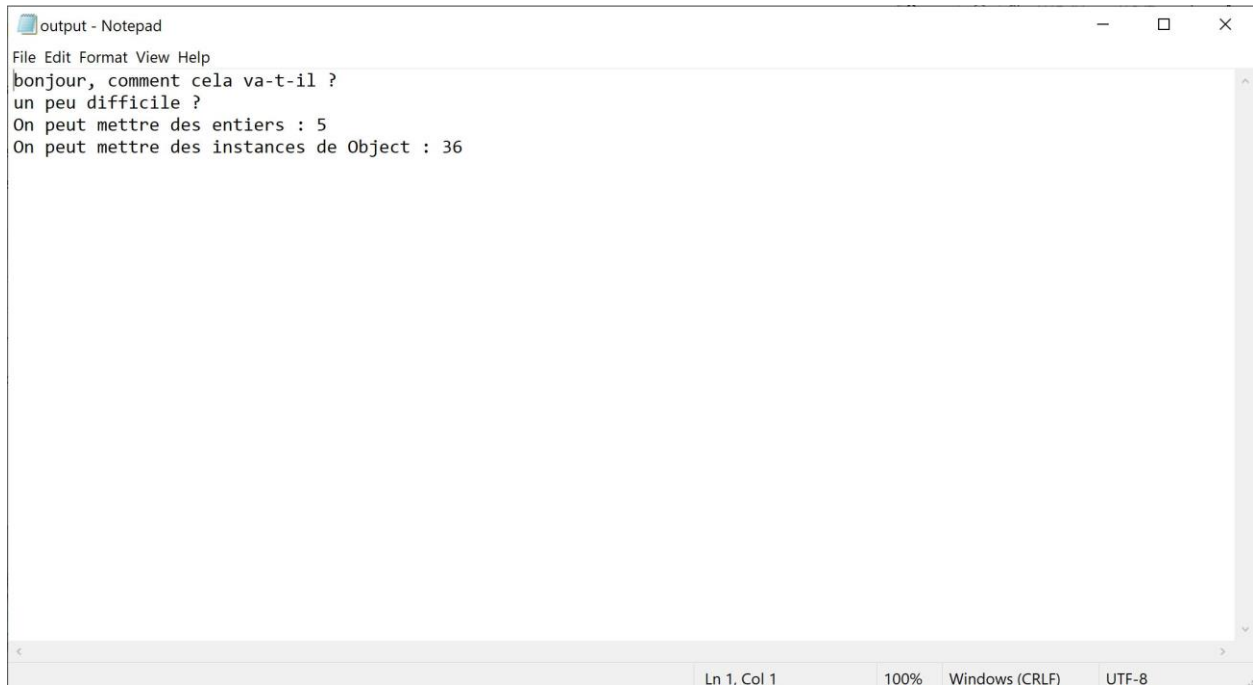
C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierTexte output.txt

C:\Users\10\Desktop\devoir mdc IO\IO1>
```

b-1

Avec `ecrivain.close()` :

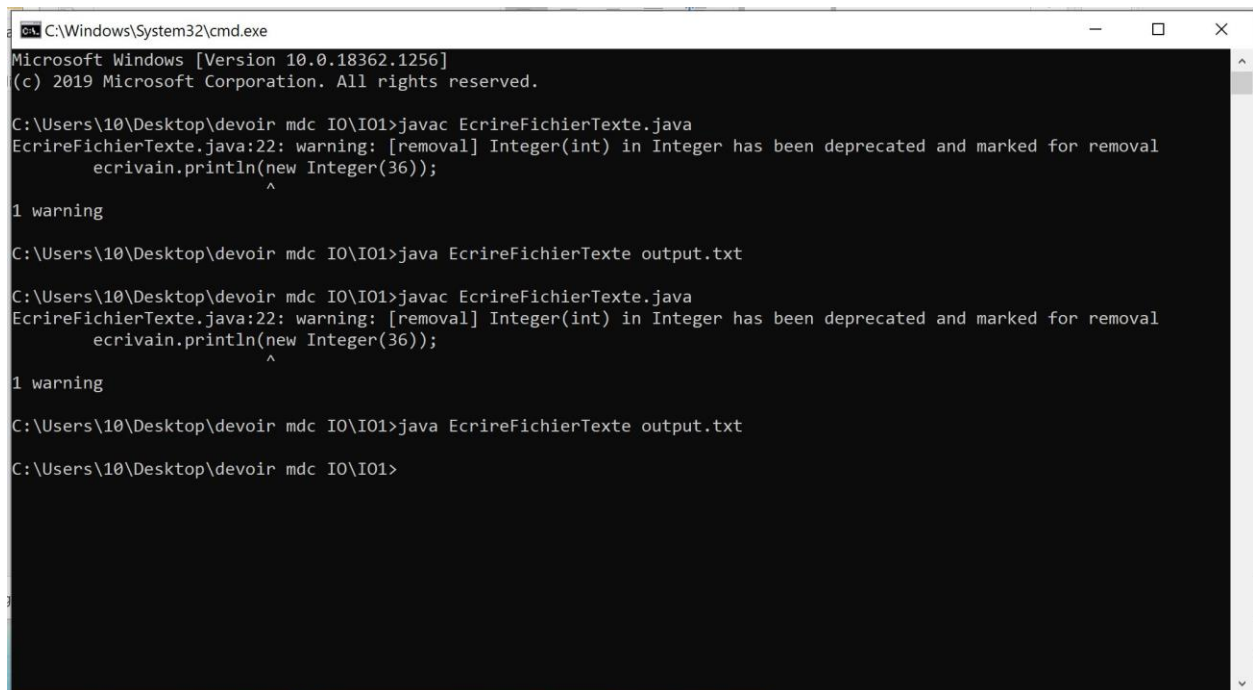
## Contenu du fichier texte



```
output - Notepad
File Edit Format View Help
bonjour, comment cela va-t-il ?
un peu difficile ?
On peut mettre des entiers : 5
On peut mettre des instances de Object : 36
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Sans `ecrivain.close()` :

Execution :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierTexte.java
EcrireFichierTexte.java:22: warning: [removal] Integer(int) in Integer has been deprecated and marked for removal
    ecrivain.println(new Integer(36));
                        ^
1 warning

C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierTexte output.txt

C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierTexte.java
EcrireFichierTexte.java:22: warning: [removal] Integer(int) in Integer has been deprecated and marked for removal
    ecrivain.println(new Integer(36));
                        ^
1 warning

C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierTexte output.txt
C:\Users\10\Desktop\devoir mdc IO\IO1>
```

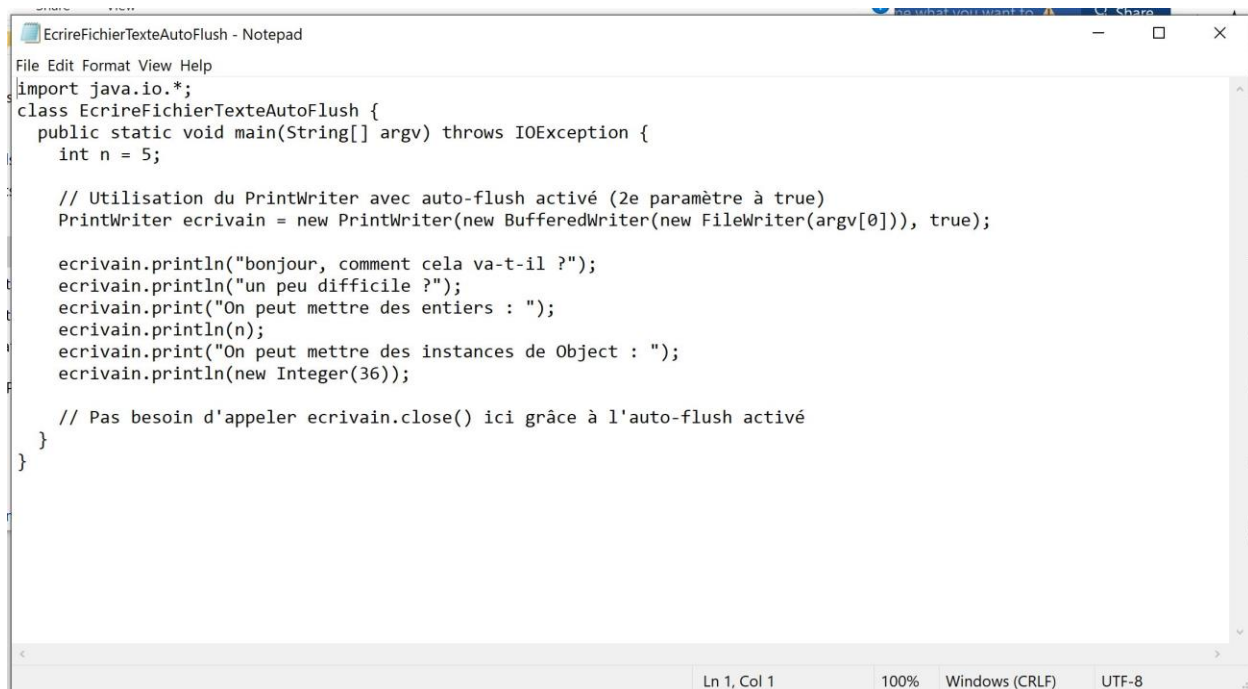
Contenu du fichier texte :



**Différence avec ou sans `ecrivain.close()`** : Si on n'appelle pas `ecrivain.close()`, les données peuvent ne pas être écrites complètement dans le fichier car elles peuvent être stockées dans un tampon. L'appel de `close()` assure que tout est écrit et que les ressources sont libérées.

**b-2**

**code modifié :**



```
File Edit Format View Help
import java.io.*;
class EcrireFichierTexteAutoFlush {
    public static void main(String[] argv) throws IOException {
        int n = 5;

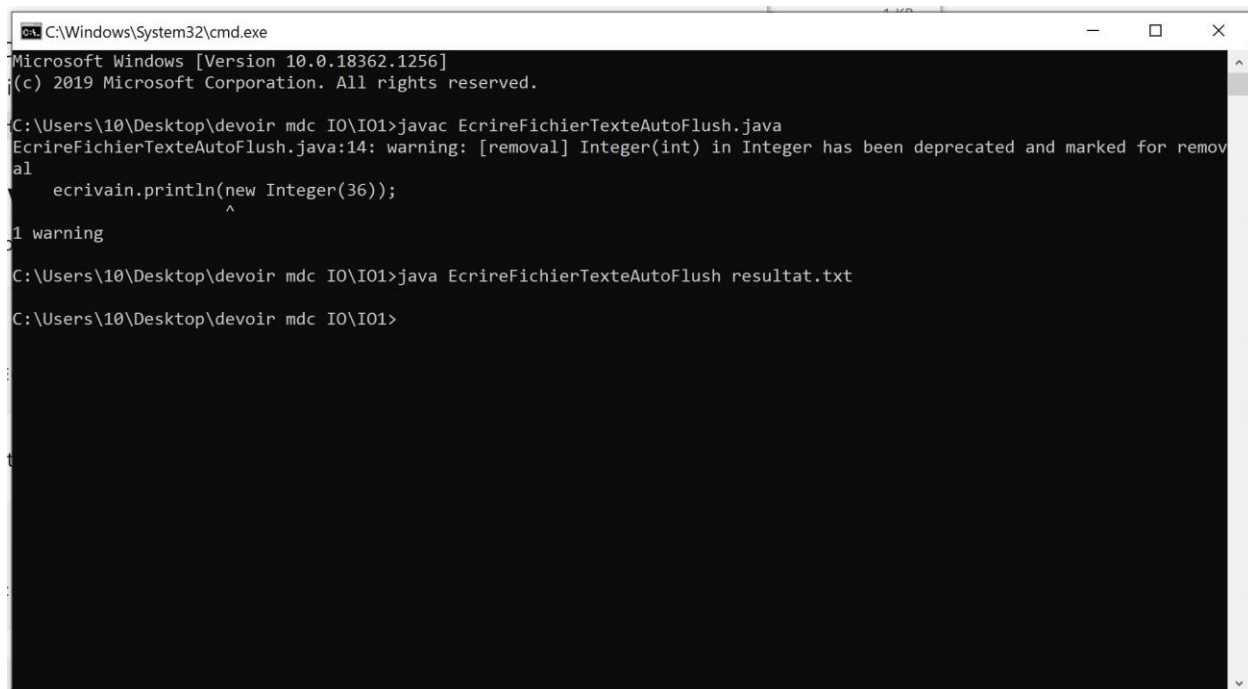
        // Utilisation du PrintWriter avec auto-flush activé (2e paramètre à true)
        PrintWriter ecrivain = new PrintWriter(new BufferedWriter(new FileWriter(argv[0])), true);

        ecrivain.println("bonjour, comment cela va-t-il ?");
        ecrivain.println("un peu difficile ?");
        ecrivain.print("On peut mettre des entiers : ");
        ecrivain.println(n);
        ecrivain.print("On peut mettre des instances de Object : ");
        ecrivain.println(new Integer(36));

        // Pas besoin d'appeler ecrivain.close() ici grâce à l'auto-flush activé
    }
}
```

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8

**execution :**



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\10\Desktop\devoir mdc IO\IO1>javac EcrireFichierTexteAutoFlush.java
EcrireFichierTexteAutoFlush.java:14: warning: [removal] Integer(int) in Integer has been deprecated and marked for removal
    ecrivain.println(new Integer(36));
                        ^
1 warning

C:\Users\10\Desktop\devoir mdc IO\IO1>java EcrireFichierTexteAutoFlush resultat.txt

C:\Users\10\Desktop\devoir mdc IO\IO1>
```

Si on active l'auto-flush, cela forcera l'écriture immédiate dans le fichier à chaque appel de `println()` sans avoir besoin d'appeler `close()`. Cela permet de s'assurer que les données sont écrites en temps réel.

### Partie 3:

Code de la classe Segment :



```
Segment - Notepad
File Edit Format View Help
import java.io.*;
class Segment implements Serializable {
    private static final long serialVersionUID = 1L;
    int abscisse_debut, ordonnee_debut, abscisse_fin, ordonnee_fin;

    public Segment(int x1, int y1, int x2, int y2) {
        abscisse_debut = x1;
        ordonnee_debut = y1;
        abscisse_fin = x2;
        ordonnee_fin = y2;
    }

    @Override
    public String toString() {
        return "Segment [de (" + abscisse_debut + ", " + ordonnee_debut + ") à (" + abscisse_fin + ", " + ordonnee_fin + ")]";
    }
}

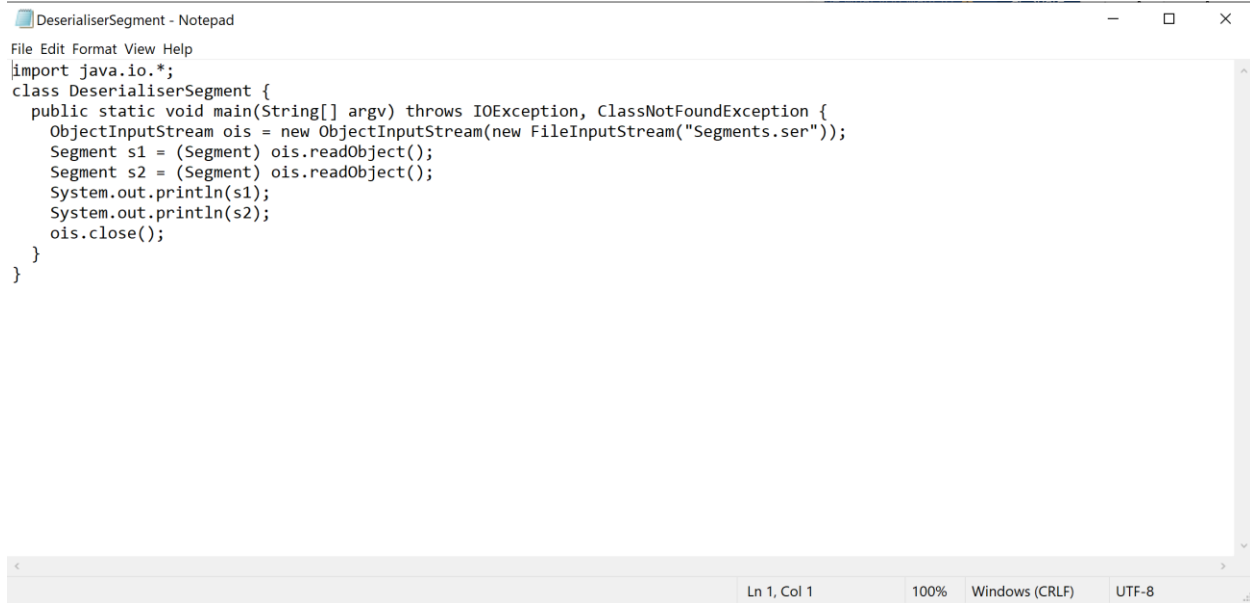
Ln 18, Col 1    100%    Windows (CRLF)    UTF-8
```

Code de la classe SerialiserSegment:

```
SerialiserSegment - Notepad
File Edit Format View Help
import java.io.*;
class SerialiserSegment {
    public static void main(String[] argv) throws IOException {
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("Segments.ser"));
        Segment s1 = new Segment(0, 0, 3, 4);
        Segment s2 = new Segment(5, 1, 7, 8);
        oos.writeObject(s1);
        oos.writeObject(s2);
        oos.close();
    }
}

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8
```

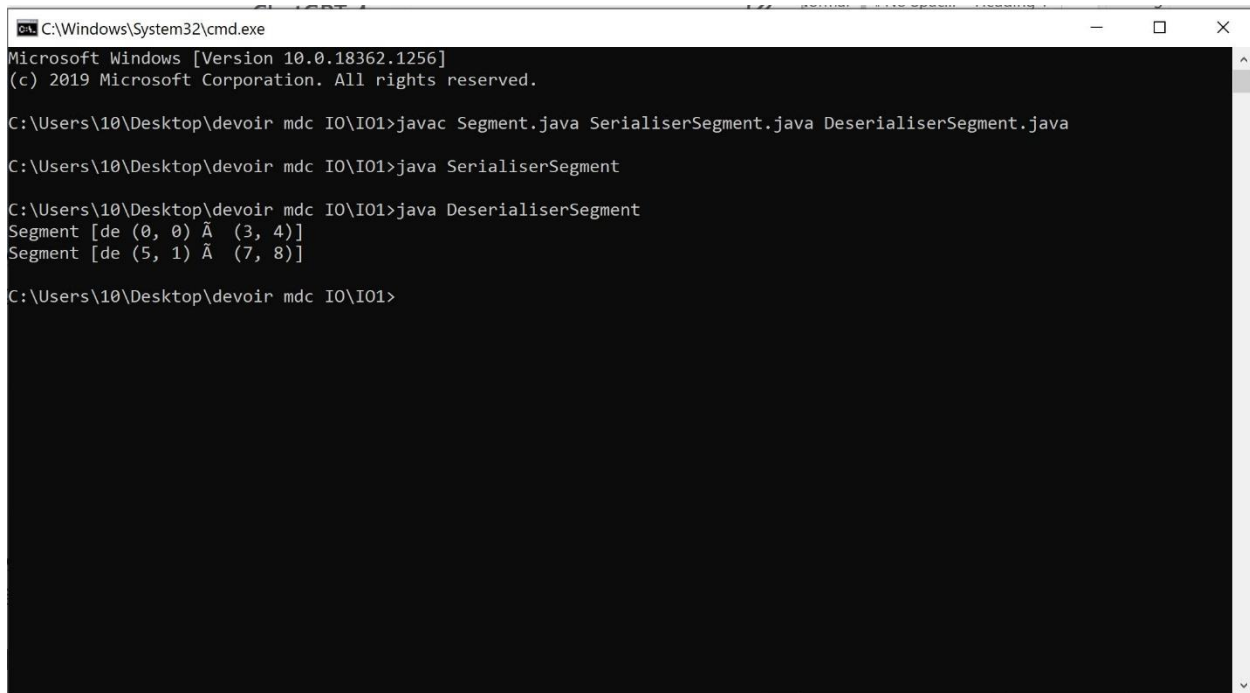
## Code de la classe DeserialiserSegment :



```
DeserialiserSegment - Notepad
File Edit Format View Help
import java.io.*;
class DeserialiserSegment {
    public static void main(String[] argv) throws IOException, ClassNotFoundException {
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("Segments.ser"));
        Segment s1 = (Segment) ois.readObject();
        Segment s2 = (Segment) ois.readObject();
        System.out.println(s1);
        System.out.println(s2);
        ois.close();
    }
}
```

Ln 1, Col 1    100%    Windows (CRLF)    UTF-8

## Execution :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

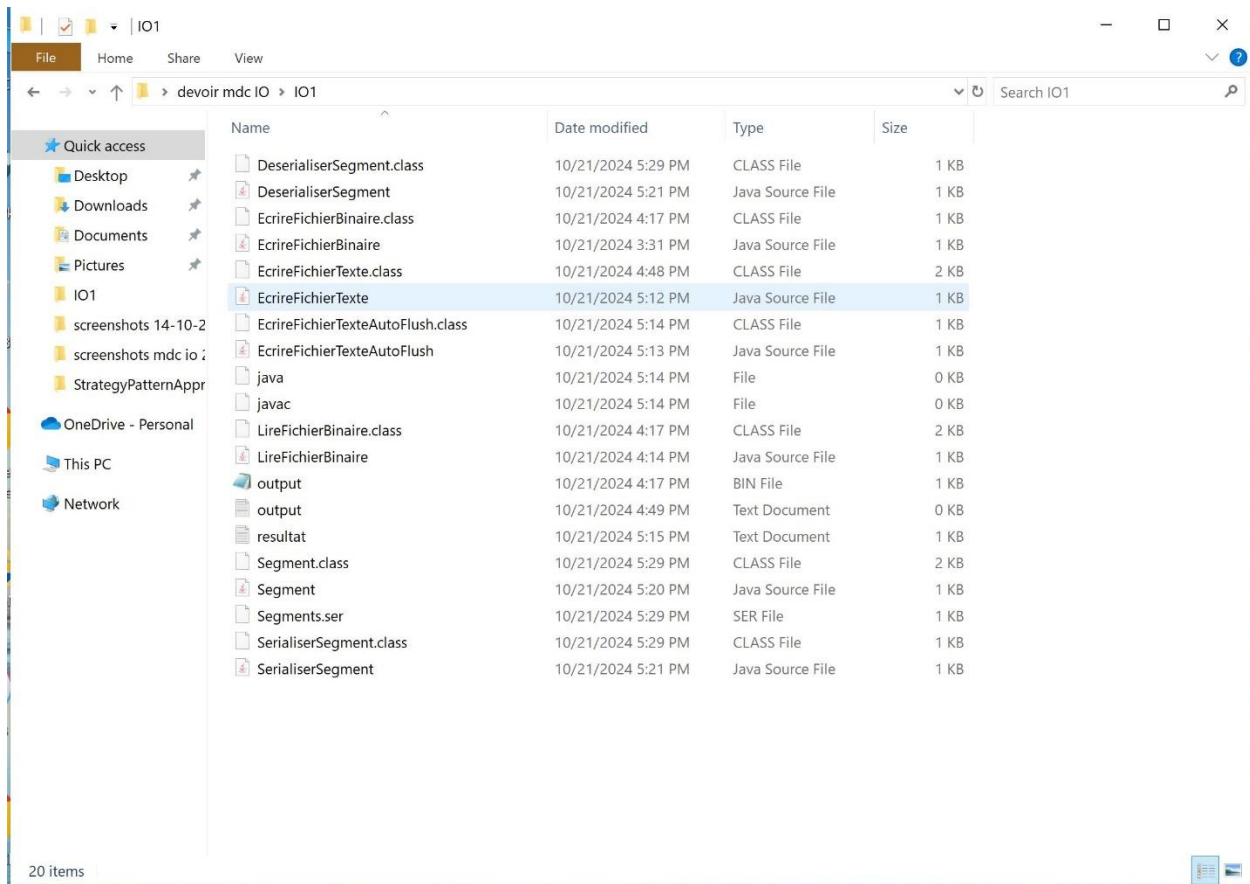
C:\Users\10\Desktop\devoir mdc IO\IO1>javac Segment.java SerialiserSegment.java DeserialiserSegment.java

C:\Users\10\Desktop\devoir mdc IO\IO1>java SerialiserSegment

C:\Users\10\Desktop\devoir mdc IO\IO1>java DeserialiserSegment
Segment [de (0, 0) Å (3, 4)]
Segment [de (5, 1) Å (7, 8)]

C:\Users\10\Desktop\devoir mdc IO\IO1>
```

Fichiers obtenus :



Ce travail a ete realise avec l'aide de ChatGPT 4o.