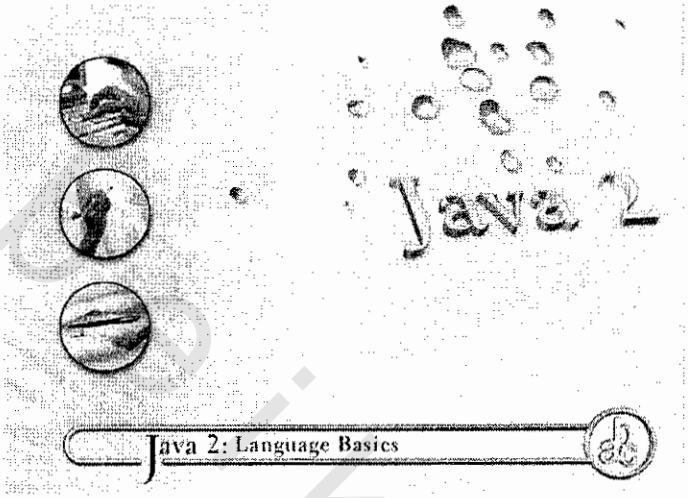
الفصل الثالث عشر



Streams ملفات

في هذا الفصل سوف نتناول عملية إدخال واخراج البيانات سواء من خلال وسائل الإدخال والإخراج التقليدية أو من خلال الملفات وذلك من خلال النقاط التالية:

- فصائل classes تدفق البيانات Streams.
- الفصيلة InputStream والفصيلة OutputStream والفصيلة Reader والفصيلة Writer.
- التعامل مع الملفات Files.
- التعامل مع المجلدات (الفهارس) Directories.
- قراءة بايت Byte من الفصيلة System.in.
- الفصائل المغلفة للتدفق Wrapper Streams.
- القراءة من ملف Reading from files.
- الكتابة إلي ملف Writing to files.
- نسخ الملفات File Copy ونقل ملف File Move والغاء ملف File Delete وإعادة تسمية ملف File Rename.
- عملية النشر التسلسلي Serialization.
- المتغيرات العابرة Transient Variables.

مقدمة:

- جميع البرامج تتعامل مع المستخدم وتطلب منه بعض البيانات والمعلومات ثم تتعامل مع هذه البيانات ثم يتم عرض نتائج معالجة هذه البيانات ، إذن لابد أن تتعامل لغات البرمجة مع عمليتي الإدخال والإخراج.
- سوف نتناول فى هذا الفصل كيفية قراءة البيانات كإدخال من المستخدم وكيفية عرض النتائج كإخراج وكذلك كيفية القراءة من بيانات من ملفات Files وكذلك كتابة بيانات فى ملفات.

تدفق البيانات Streams:

- هو تدفق البيانات من مصدر معين Source إلى جهة وصول Destination والعكس.
- فمثلاً ، إذا أدخلت بيانات إلى البرنامج فإن النتائج تظهر على الشاشة ، إذن هنا مصدر البيانات هو لوحة المفاتيح Keyboard وجهة الوصول هى الشاشة.
- إذا طبعت ملفاً فإن النتائج تظهر علي آلة الطباعة ، إذن هنا مصدر البيانات هو الملف الذى طبعته وجهة الوصول هى آلة الطباعة.
- إذا قمت بكتابة بيانات ثم قمت بحفظ هذه البيانات فى ملف فإنك تحفظه على القرص الصلب (Hard Disk) ، إذن هنا مصدر المعلومات هو الملف وجهة الوصول هى القرص الصلب.
- إذن هناك أكثر من مثال لعملية تدفق البيانات Streams فهو ببساطة مجموعة بيانات تنتقل من مصدر إلي جهة وصول بصرف النظر عن نوع البيانات المنتقلة.
- قد تظن أنه لكي تستطيع التعامل مع أنواع التدفق Streams المختلفة ، فإنك لا بد أن تعرف عدة طرق فى البرمجة بحيث تستطيع التعامل مع كل نوع من أنواع التدفق Streams المختلفة ، ولكن هذا ليس صحيحاً ، فكما سنرى ، توفر لغة Java عدة فصائل classes للتعامل مع أنواع التدفق Streams المختلفة بمنتهى السهولة.
- لاحظ أنك إذا كنت تكتب بيانات فإنك فى هذه الحالة تكون المصدر Source ، أما إذا كنت تقرأ البيانات فإنك تمثل جهة الوصول Destination لهذه البيانات.
- وتوجد في لغة Java أربعة فصائل classes رئيسية للتعامل مع عمليات تدفق البيانات Streams هى:

InputStream OutputStream Reader Writer

الفصيلة InputStream:

- توفر هذه الفصيلة class وظائف قراءة بايت byte من المعلومات من تدفق البيانات Stream.
- تحتوى هذه الفصيلة class على دالة method تسمى ()read هى المسئولة عن وظيفة القراءة من التدفق Stream.
- كل أنواع التدفق Streams التي تتعامل مع عملية الإدخال Input Streams ترث من هذه الفصيلة class.

الفصيلة OutputStream:

- توفر هذه الفصيلة class وظائف كتابة بايت byte من المعلومات إلى التدفق Stream.
- تحتوى هذه الفصيلة class على دالة method تسمى ()write هى المسئولة عن وظيفة الكتابة إلى التدفق Stream.
- كل أنواع التدفق Streams التي تتعامل مع عملية الإخراج Output Streams ترث من هذه الفصيلة class.

الفصيلة Reader:

- وهى تماثل فى عملها الفصيلة InputStream.

الفصيلة Writer :

- وهى تماثل فى عملها الفصيلة OutputStream.
- **إذن هنا نتساءل : ما الفرق بين هذه الفصائل classes ؟**
- الفرق هو أن الفصائل classes المسماة Reader وWriter تدعم عملية تسمى دعم اللغات المتعددة Internationalization.

دعم اللغات المتعددة Internationalization:

- هى عملية تصميم البرنامج بحيث يمكنه أن يعمل مع أكثر من لغة بدون أى تغيير في البرمجة.
- فمثلاً انظر لنظام تشغيل النوافذ Windows ، فتجد أنه يدعم بالطبع اللغة الإنجليزية ، ولكن إذا أردت أن تحول لغة الواجهة لتصبح العربية ، فإنك لا تستطيع والحل أنك ستشترى نسخة أخرى لها واجهة باللغة العربية.
- إذن لكى تستطيع تغيير اللغة فإنك ستبتاع نسخة أخرى ، ولكن عملية دعم اللغات المتعددة Internationalization تعنى أنه بتصميم البرنامج بطريقة معينة ، فإنك ستستطيع تغيير اللغة بلا مشاكل وبدون شراء نسخة أخرى.
- هذه العملية في غاية الأهمية خصوصاً في شبكة الإنترنت Internet ، فمثلاً إذا أنشأت موقعك علي شبكة الإنترنت Internet باللغة العربية وحاول أحد مستخدمي الشبكة الدخول علي موقعك ، وكان نظام التشغيل Operating System علي جهازه لا يدعم اللغة العربية ، فلن يستطيع قراءة اللغة العربية وبذلك فلن يمكنه معرفة أي معلومات عن موقعك ، وبالتالي لا بد أن يكون موقعك مجهزاً بحيث يستطيع تقديم المعلومات بأكثر من لغة.
- تعرف عملية دعم اللغات المتعددة Internationalization بعملية i18n لأنه يوجد 18 حرف يقعون بين حرفي I و n.
- والفصيلة InputStream تعبر عن تدفق Stream مصدره هو لوحة المفاتيح Keyboard ، والفصيلة OutputStream تعبر عن تدفق Stream جهة وصوله Destination هى الشاشة.
- تذكر أننا استخدمنا الدالة println() الموجودة فى الفصيلة System.out لطباعة رسالة على الشاشة ، إذن فالفصيلة System.out هي المسئولة عن الطباعة على الشاشة.
- سوف نرى العديد من الاستخدامات لهاتين الفصيلتين classes للقراءة وأخذ البيانات من المستخدم فى الأمثلة القادمة.

الملفات Files:

- بالطبع لابد أن تتعامل مع الملفات لأنك تستطيع القراءة من ملف أو الكتابة إلى ملف ،

والكتابة والقراءة من الملفات من العمليات القديمة والتقليدية التى توفرها لغة البرمجة ، وسنتناول في النقاط القادمة العديد من عمليات الملفات.

عمليات الملفات:

توجد مجموعة من العمليات التى يمكن إجراؤها على الملفات ، والجدول التالى يعرض الدوال methods المسئولة عن تحقيق هذه العمليات وعمل كل دالة method.

اسم الدالة method	نوع القيمة المرتجعة Return Type	وظيفة الدالة method
Exists ()	boolean	تحدد ما إذا كان الملف موجوداً أم لا
canRead ()	boolean	تحدد ما إذا كان باستطاعتك القراءة من الملف أم لا
canWrite()	boolean	تحدد ما إذا كان باستطاعتك الكتابة إلى الملف أم لا
getAbsolutePath ()	String	تحدد المسار الكامل للملف
mkdir ()	boolean	تقوم بإنشاء مجلد (directory)
getName()	String	تقوم بتحديد اسم الملف
isHidden ()	Boolean	تقوم بتحديد ما إذا كان الملف مختفياً (Hidden) أم لا
isDirectory ()	boolean	تقوم بتحديد ما إذا كان الملف مجلداً Folder أم لا
isFile ()	boolean	تقوم بتحديد ما إذا كان الملف عبارة عن ملف File عادي أم لا
length ()	long	تقوم بإرجاع حجم الملف بالبايت Byte
setReadOOnly ()	boolean	تقوم بجعل الملف للقراءة فقط (Read –Only)

مثال (1) : التعامل مع الملفات:

أولاً: هدف المثال:

المطلوب فى هذا المثال إنشاء برنامج يستطيع الحصول على بعض البيانات من ملف نصي اسمه Data.txt ، فلا بد أن تتأكد أولاً من وجود هذا الملف ، وإذا كان موجوداً فنريد معرفة بعض المعلومات عن هذا الملف باستخدام الدوال method السابقة الشرح في الجدول السابق.

ثانياً: كود البرمجة:

```
1:   import java.io.*;
2:
3:   class MyFile
4:   {
5:     public static void main (String[]args)
6:   {
7:   File myFile = new File ("data.txt");
8:
9:   //Now the file does not exist on your hard
10:  System.out.println("The file does not exist on your hard");
11:  System.out.println("--------------------------------");
12:  System.out.println("Exists: " + myFile.exists());
13:  System.out.println("Can Read: " + myFile.canRead());
14:  System.out.println("Can Write: " + myFile.canWrite());
15:  System.out.println("Absolute Path: "+
myFile.getAbsolutePath());
16:  System.out.println("File Name: " + myFile.getName());
17:
18:  //create the file
19:  try
20:  {
21:  System.out.println("---------------------------");
22:  System.out.println("Creating File: " +
myFile.createNewFile());
23:  }
24:  catch (IOException e)
```

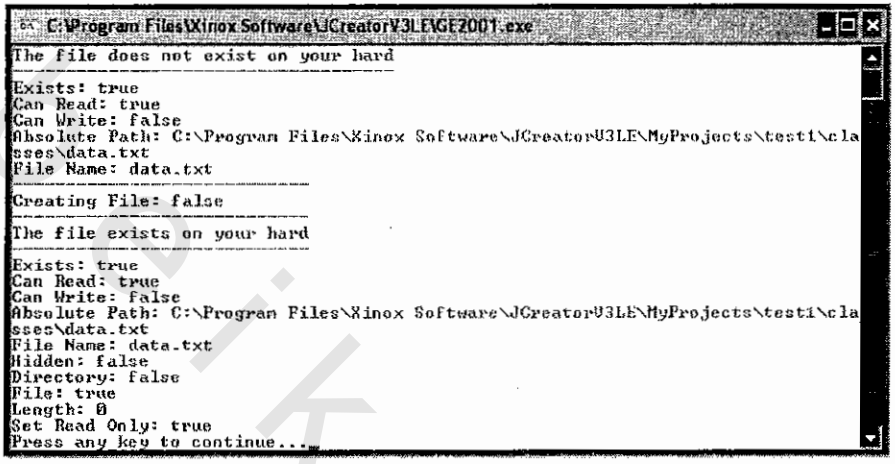
```
25:  {}
26:  //Now the file exists on your hard
27:  System.out.println("---------------------------");
28:  System.out.println("The file exists on your hard");
29:  System.out.println("---------------------------");
30:  System.out.println("Exists: " + myFile.exists());
31:  System.out.println("Can Read: " + myFile.canRead());
32:  System.out.println("Can Write: " + myFile.canWrite());
33:  System.out.println("Absolute Path: " +
myFile.getAbsolutePath());
34:  System.out.println("File Name: " + myFile.getName());
35:  System.out.println("Hidden: " + myFile.isHidden());
36:  System.out.println("Directory: " + myFile.isDirectory());
37:  System.out.println("File: " + myFile.isFile());
38:  System.out.println("Length: " + myFile.length());
39:  System.out.println("Set Read Only: " +
myFile.setReadOnly());
40:  }
41:  }
```

ثالثاً: شرح الكود:

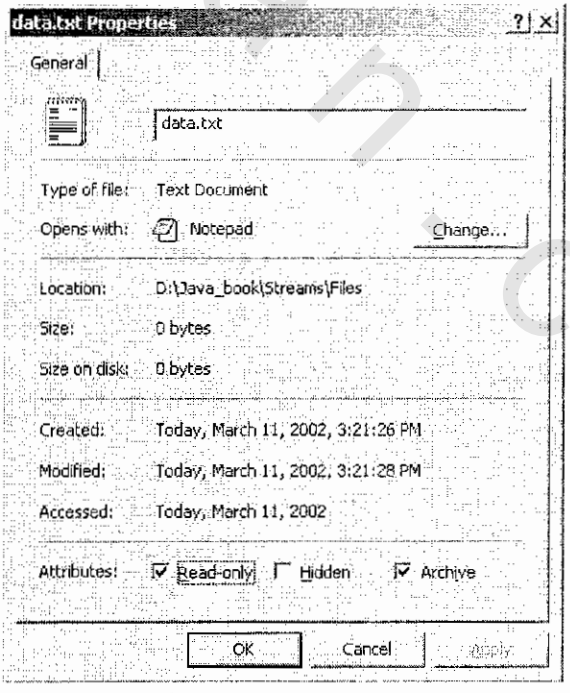
- في السطر رقم 1 تضمين الحزم packages اللازمة لعمل البرنامج.
- في السطر رقم 7 يتم إنشاء ملف اسمه data.txt.
- لاحظ أننا أنشأنا هدفاً object من نوع الفصيلة File وتم تمرير القيمة القيمة data.txt لدالة البناء constrcutor وهي تمثل اسم الملف الذي نريد التعامل معه.
- لاحظ أيضاً أنه كونك أنشأت هذا الهدف object ، فإن ذلك لا يعني أن الملف أصبح موجوداً على القرص الصلب Hard Disk وستعرف حالاً كيف تستطيع إنشاؤه على القرص الصلب Hard Disk.
- في السطور من 12 إلى 16 يتم استخدام الدوال methods الموجودة فى الجدول السابق.
- لاحظ أن الملف data.txt غير موجود فى نفس مسار البرنامج لأننا لم ننشئه بعد ، إذن فالملف غير موجود وبالطبع لا يمكنك القراءة منه أو الكتابة إليه لأنه غير موجود أصلاً.

- في السطر رقم 22 يتم إنشاء الملف فعلياً على القرص الصلب Hard Disk عن طريق الدالة createNewFile().
- لاحظ أنك فى هذه الحالة أنشأت الملف فى نفس مسار البرنامج.
- فمثلاً نفترض أن الملف Files.java موجود فى المسار E:\ علي القرص الصلب Hard Disk ، إذن بعد تنفيذ السطر رقم 22 فإنك ستجد الملف data.txt موجوداً فى نفس المسار وهو E:\.
- تم تنفيذ السطر رقم 22 داخل جملة try بسبب إمكانية حدوث استثناء Exception ، فقد لا تستطيع إنشاء ملف بسبب عدم وجود مساحة علي القرص الصلب Hard Disk مثلاً ، وهنا نوع الاستثناء Exception هو IOException.
- في السطور من 30 إلى 39 يتم بيان استخدام باقى الدوال methods الموجودة فى الجدول السابق.
- لاحظ أنك أنشأت الآن الملف data.txt فعلياً على القرص الصلب Hard Disk ، إذن فهو موجود الآن وتستطيع الكتابة إليه والقراءة منه وتستطيع معرفة ما إذا كان مخفياً Hidden أم لا.
- في السطر رقم 36 يتم اختبار الهدف object الذى أنشأناه باسم myFile ، هل هو مجلد Directory أم لا ، بالطبع نحن أنشأنا ملفاً عادياً وسنرى فيما بعد كيف ننشئ مجلداً Directory.
- في السطر رقم 37 يتم اختبار myFile ، هل هو ملف عادى أم لا ، وبالطبع الإجابة هى نعم أو true.
- في السطر رقم 38 يتم معرفة حجم الملف عن طريق الدالة length().
- في السطر رقم 39 يتم جعل الملف Read–only أى للقراءة فقط ، أى أنك إذا ذهبت للمسار E:\ الموجود فيه الملف النصي data.txt وقمت بالضغط بالزر الأيمن للفارة Mouse على الملف data.txt ثم اخترت Properties ، فإنك ستجد أن الملف للقراءة فقط Read-only فعلاً. انظر شكل (13-2).
- لاحظ أن Read–only تعنى أنه يمكنك القراءة من الملف فقط ولا يمكنك الكتابة إليه أو تعديله.

- لاحظ أيضاً أنه إذا كان الملف موجوداً ، فإن محاولة إنشاء الملف فى السطر رقم 22 ستعيد القيمة false لأنك لن تستطيع إنشاء ملف موجود أصلاً.
- بعد تنفيذ البرنامج تظهر النتيجة علي الشاشة كما في شكل (13-1).



(الشكل 13-1) المثال الأول



(الشكل 13-2) المثال الأول

المجلدات Directories (Folders):

كما تعاملنا مع الملفات ، فإننا نريد أيضاً أن نتعامل مع المجلدات Directories (Folders) ، فنريد أن نعرف كيف ننشئ مجلداً Folder وكيف نعرض أسماء المجلدات Folders والملفات الموجودة فى مسار معين (أى مثل أمر dir فى نظام تشغيل الدوس Dos).

مثال (2): إنشاء مجلدات Directories:

أولاً: هدف المثال:

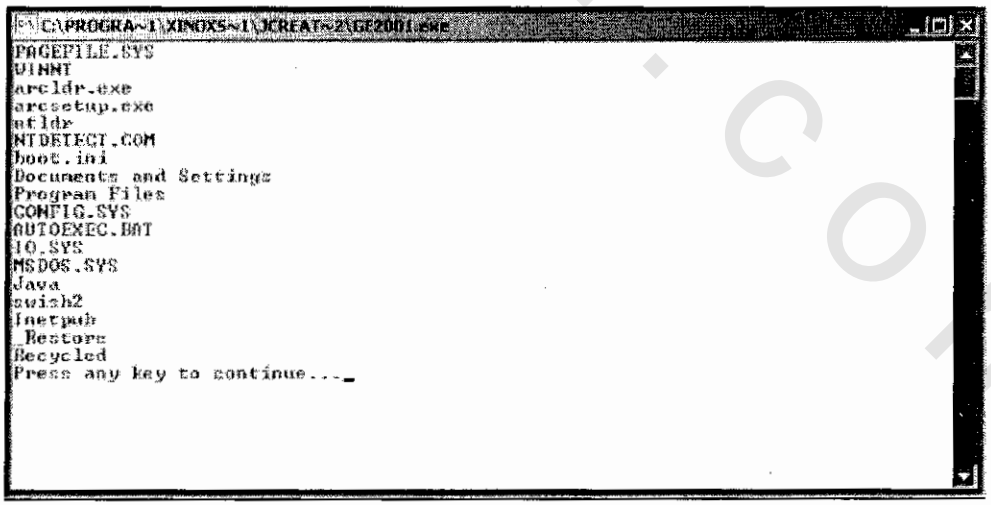
نتعلم في هذا المثال كيفية إنشاء مجلد Directory (Folder) وكيفية عرض أسماء المجلدات Folders والملفات الموجودة فى مسار معين.

ثانياً: كود البرمجة:

```
1:   import java.io.*;
2:
3:   class MyDirectory
4:   {
5:    public static void main ( String[] args )
6:    {
7:        //creating a directory
8:        File myDirectory = new File ("C:\\Java");
9:        myDirectory.mkdir();
10:
11:       //listing directory files
12:       File myDir = new File ("C:\\");
13:
14:       String[] dirList = myDir.list();
15:
16:       for ( int i = 0 ; i < dirList.length ; i++ )
17:       {
18:           System.out.println(dirList[i]);
19:       }
20:   }
21:  }
```

ثالثاً: شرح الكود:

- في السطر رقم 8 يتم إنشاء ملف ويتم وضعه في المسار C:\\Java. لاحظ استخدام \\ وليس \ واحدة فى كتابة المسار.
- في السطر رقم 9 يتم إنشاء المجلد Folder فعلياً على القرص الصلب Hard Disk عن طريق الدالة ()mkdir.
- في السطر رقم 12 يتم إنشاء ملف موجود فى المسار C:\\ مباشرة.
- في السطر رقم 14 يتم إنشاء مصفوفة array من نوع String بحيث أن كل عنصر من عناصر المصفوفة array يحمل اسم ملف ، واستخدمنا الدالة ()list لعرض جميع الملفات الموجودة في المسار C:\\.
- في السطور من 16 إلى 19 يتم طباعة محتويات المصفوفة array التى ستحمل أسماء جميع الملفات فى المسار C:\\.
- لاحظ استخدام dirList.length فى السطر رقم 16 وهى مهمة جداً لأننا لا نعرف عدد الملفات فى المسار المحدد فلا تستطيع كتابة رقم معين هنا.
- بعد تنفيذ البرنامج فستظهر النتيجة علي الشاشة كما في شكل (13-3).



(الشكل 13-3) المثال الثاني

- إذا أردنا عرض الملفات الموجودة فى نفس المسار الحالى للبرنامج ، فإننا نقوم بتغيير السطر رقم 12 إلى

```
File myDir = new File (".");
```

حيث النقطة (.) ترمز إلى المسار الحالى

قراءة بايت Byte من الفصيلة System.in:

- من الإمكانيات المتاحة فى لغة Java إمكانية قراءة حرف بحرف من المستخدم بحيث يستطيع إدخال بعض البيانات للبرنامج.
- يتضح لنا استخدام هذه الفصيلة class من المثال التالي.

مثال (3): قراءة بايت Byte من الفصيلة System.in:

أولاً: هدف المثال:

المطلوب أن نطلب من المستخدم إدخال حرف ، فإذا أدخل الحرف x ، فيتم إنهاء البرنامج والخروج منه (Exit) حيث نتعلم كيفية قراءة بايت (حرف) Byte باستعمال الفصيلة System.in.

ثانياً: كود البرمجة:

```
1:    import java.io.*;
2:
3:    class Read
4:    {
5:      public static void main ( String[]args )
6:      {
7:          int i = 0;
8:          InputStream is = System.in;
9:
10:         System.out.println("Enter a character");
11:
12:         while ( (char)i != 'x' )
13:         {
14:             try
```

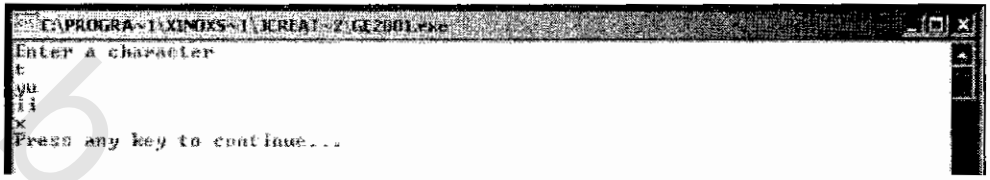
```
15:             {
16:                 i = is.read();
17:             }
18:
19:             catch ( IOException e )
20:             {
21:                 System.out.println(e);
22:             }
23:         }
24:         }
25:     }
```

ثالثاً: شرح الكود:

- في السطر رقم 8 يتم إنشاء هدف object من نوع الفصيلة InputStream واسمه is ثم جعلنا is هو System.in. تذكر أن System.in هو عبارة عن InputStream ومصدره هو لوحة المفاتيح Keyboard.
- في السطر رقم 10 يتم طباعة رسالة للمستخدم تطلب منه إدخال حرف.
- في السطر رقم 16 يتم استخدام الدالة ()read لقراءة البايت Byte من لوحة المفاتيح Keyboard.
- لاحظ أن الدالة ()read لها قيمة مرتجعة Return Value من نوع int ولذلك قمنا بتعريف متغير من نوع int اسمه i فى السطر رقم 7 وجعلنا له قيمة ابتدائية Initial Value تساوى صفراً ، وكما تعلم فإنك لابد أن تعطى قيمة ابتدائية Initial Value لأى متغير فى لغة Java.
- لاحظ أيضاً أن الدالة ()read من الممكن أن تسبب استثناءاً Exception من نوع IOException ولذلك استخدمنا جملة try وcatch لمعالجة هذا الاستثناء Exception.
- في السطر رقم 12 يتم استخدام الدوارة while loop لأننا نريد أن نقرأ بايت Byte طالما أن الحرف المدخل ليس الحرف x.
- لاحظ أن البايت Byte الذى سيدخله المستخدم تم حفظه فى المتغير i ، ولكن المتغير i من نوع int ونريد أن نقارن المتغير i بالحرف x ، وذلك مستحيل لاختلاف نوعى

المتغيرين ولذلك استخدمنا كلمة char لعمل تحويل Casting للمتغير i ليعامل كحرف وبذلك تصبح المقارنة ممكنة.

- بعد تنفيذ البرنامج تظهر النتيجة علي الشاشة كما في شكل (13-4).



(الشكل 13-4) المثال الثالث

الفصائل المغلفة للتدفق Wrapper Streams:

- ويمكن تعريف الفصيلة المغلفة Wrapper class هي الفصيلة class التى تضيف بعض الوظائف إلى الفصائل classes الخاصة بالتدفق Streams ، فقد لاحظنا فى المثال السابق أننا كنا نقرأ بايت Byte ، وهذا نادراً ما يحدث فالطبيعى أنك تطلب إدخال عدة سطور من الأحرف ، ولذلك فإننا نستخدم فصيلة class تسمى BufferedReader التي تحتوى على الدالة readLine() التى تسمح لنا بقراءة عدة سطور ، كما أنها تتميز بالقدرة على تخزين الحروف المدخلة بحيث يمكن القراءة منها مباشرة بدلاً من القراءة من نظام التشغيل Operating System وهذا يعطى سرعة أكبر في التنفيذ.
- سوف نرى استخدام الفصيلة BufferedReader لاحقاً.

عمليات الملفات File Operations:

- عمليات القراءة والكتابة فى الملفات تعتبر من العمليات المشهورة وكثيرة الاستعمال في البرمجة ، ولذلك توفر لغة Java اثنتين من الفصائل classes لعملية القراءة والكتابة للملفات هما:

1. FileReader: للقراءة من ملف.
2. FileWriter: للكتابة إلى ملف.

- ولكن المشكلة أنك تريد أن تقرأ سطراً من ملف ، ولذلك لا تستطيع استخدام الفصيلة FileReader مباشرة ، إذن نريد فصيلة class لقراءة سطر من الملف وهذه الفصيلة class هي BufferedReader.
- أيضاً توجد نفس المشكلة فى الكتابة إلى ملف ، فلا يمكنك استخدام الفصيلة FileWriter مباشرة لأنك ستكتب سطراً في الملف ، ولذلك نستخدم فصيلة class لكتابة سطر إلى ملف وهذه الفصيلة class هي PrintWriter.
- قد تبدو هذه السطور غامضة ولكننا سنعطى مثالاً صغيراً لبيان كيفية القراءة والكتابة بالنسبة للملفات.

أولاً: القراءة من ملف:

- لتحقيق ذلك يتم استعمال السطور الثلاثة التالية:

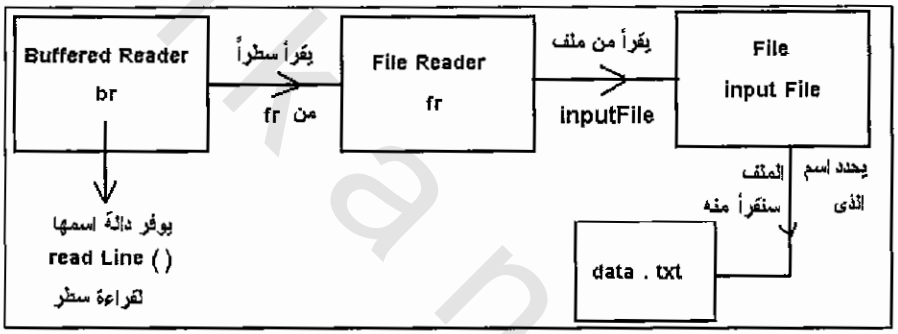
```
File inputFile = new File ("data.txt");
FileReader fr = new FileReader (inputFile);
BufferedReader br = new BufferedReader (fr);
```

- فى السطر الأول أنشأنا هدفاً object من نوع الفصيلة class المسماه File ، إذن هنا أنشأنا ملفاً جديداً اسمه data.txt.
- حيث أنك تريد أن تقرأ من هذا الملف ، فإنك تستخدم هدفاً object من نوع الفصيلة class المسماة FileReader.
- دالة البناء constructor لهذه الفصيلة class تستقبل هدفاً object من نوع الفصيلة File ، إذن هنا حددنا أننا سنقرأ من inputFile الذى بدوره هو عبارة عن الملف data.txt.
- كما ذكرنا لا يمكن استخدام الفصيلة FileReader مباشرة للقراءة من الملف وذلك لأنك تقرأ سطراً من الملف وليس الملف كاملاً ، ولذلك استخدمنا هدفاً object من نوع الفصيلة class المسماة BufferedReader ، ودالة البناء constructor لهذه الفصيلة class تستقبل هدفاً object من نوع FileReader لتحديد الملف الذى ستقرأ منه.
- كملخص ، فإن السطر الثالث ينشئ الهدف br الذى يقرأ سطراً من الهدف fr.

- والهدف fr هو الذي يستخدم للقراءة من الملف وهذا الملف هو inputFile.
- والملف inputFile هو عبارة عن الملف الفعلى الذى سنقرأ منه وهو الذى اسمه data.txt.
- بعد هذه الثلاثة سطور فيمكننا القراءة مباشرة من الهدف br ، وكما ذكرنا فإن الفصيلة BufferedReader توفر لنا دالة method اسمها readLine() لقراءة سطر كامل من الملف.
- أى أنه للقراءة من br فإننا نكتب.

```
String s = br.readLine();
```

- الرسم التالى يوضح هذه العملية.



- سوف نتناول فى الأمثلة القادمة كيفية تحقيق ذلك.

ثانياً: الكتابة إلى ملف:

- ولتحقيق ذلك يتم استعمال السطور الثلاثة التالية:

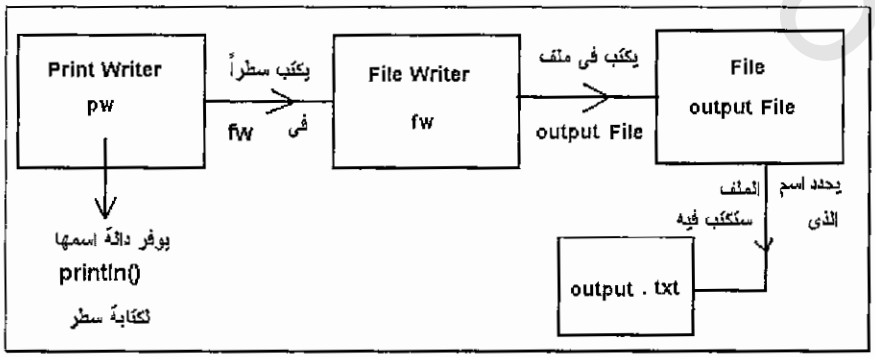
```
File outputFile = new File ("output.txt");
FileWriter fw = new FileWriter (outputFile);
PrintWriter pw = new PrintWriter (fw);
```

- بالطبع يوجد تشابه شديد بين عمليتي الكتابة والقراءة ولكننا سنعيد شرحهم لمزيد من الفهم.
- فى السطر الأول أنشأنا هدفاً object من نوع الفصيلة class المسماة File ، إذن هنا أنشأنا ملفاً للكتابة فيه اسمه output.txt.

- حيث أنك تريد أن تكتب فى هذا الملف ، فإنك تستخدم هدفاً object من نوع الفصيلة class المسماة FileWriter.
- دالة البناء constructor لهذه الفصيلة class تستقبل هدفاً object من نوع الفصيلة File ، إذن هنا حددنا أننا سنكتب فى outputFile الذى بدوره هو عبارة عن الملف output.txt.
- كما ذكرنا لا يمكن استخدام الفصيلة FileWriter مباشرة للكتابة فى الملف وذلك لأنك ستكتب فيه سطر بسطر وليس السطور كاملة ، ولذلك استخدمنا هدفاً object من نوع الفصيلة class المسماة PrintWriter ، ودالة البناء constructor لهذه الفصيلة class تستقبل هدفاً object من نوع الفصيلة FileWriter لتحديد الملف الذى ستكتب فيه.
- كملخص ، فإن السطر الثالث ينشىء الهدف pw الذى يكتب سطراً فى الهدف fw.
- الهدف fw هو الذى يستخدم للكتابة فى الملف ، وهذا الملف هو outputFile.
- الملف outputFile هو عبارة عن الملف الفعلى الذى سنكتب فيه وهو الذى اسمه output.txt.
- بعد هذه الثلاثة سطور فيمكننا الكتابة مباشرة فى الهدف pw ، وكما نعلم أنه توجد الدالة method المسماة println() التى تستخدم للكتابة.
- أى أنه للكتابة فى الهدف pw فإننا نكتب.

```
String s = br.readLine ();
pw.println (s);
```

- أى أننا سنقرأ من الهدف br ونكتب فى الهدف pw العبارة s.
- الرسم التالى يوضح هذه العملية.



عمليات الملفات File Operations:

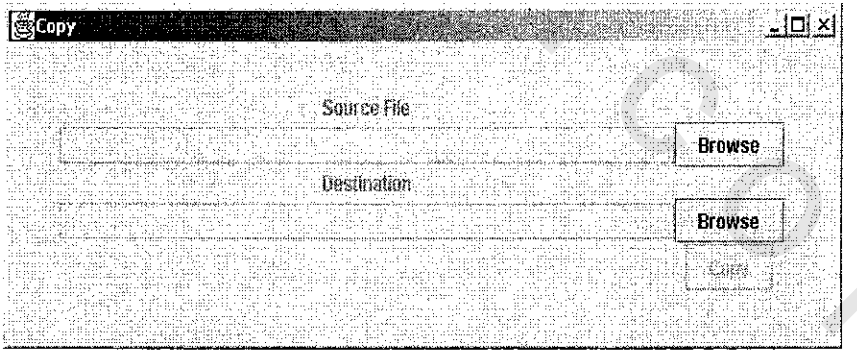
لتوضيح ما سبق شرحه من العمليات الممكن إجراؤها على الملفات ، فسوف نتناول أربعاً من أشهر العمليات التى تتم على الملفات وهى:

1- عمل نسخة من ملف Copy.
2- نقل ملف Move.
3- إلغاء ملف Delete.
4- إعادة تسمية الملف Rename.

مثال (4): عمل نسخة من ملف Copy:

أولاً: هدف المثال:

المطلوب إنشاء برنامج يحقق عملية نسخ الملفات الشهيرة فى نظام التشغيل Operating System ، وبالتالى يطلب البرنامج من المستخدم إدخال اسم ومسار الملف الأول (المراد النسخ منه) ثم اسم ومسار الملف الثانى (المراد النسخ إليه) ، وبالتالى نحتاج إلى واجهة تطبيق للمستخدم User Interface تحتوى على عدد JLabel 2 و JTextField 2 و JButton 3 و JFileChooser 1 كما بالشكل (5-13).



(الشكل 13-5) نسخ ملف

نريد أن نضغط على الزر Browse الأول فيفتح لنا مربع حوار فتح ملف Dialog Box لنختار الملف الذى نريد أن ننسخه.

- وإذا ضغطنا على الزر Browse الثانى فيفتح لنا أيضاً مربع فتح ملف Dialog Box لنختار المسار الذى نريد أن نضع نسخة الملف فيه.
- وبالطبع فإن مسار الملفات المختارة سوف يظهر فى أداة النص JTextField.
- ثم بالضغط على الزر Copy فإن عملية النسخ تتم فعلياً.

ثانياً: كود البرمجة:

```
1:    import javax.swing.*;
2:    import java.io.*;
3:    import java.awt.*;
4:    import java.awt.event.*;
5:
6:    class CopyFile extends JFrame
7:    {
8:     JLabel labelSourceFile = new JLabel ("Source File");
9:     JLabel labelDestination = new JLabel ("Destination");
10:   JTextField textSourceFile = new JTextField (40);
11:   JTextField textDestination = new JTextField (40);
12:   JButton buttonSourceBrowse = new JButton ("Browse");
13:   JButton buttonDestinationBrowse = new JButton
("Browse");
14:   JButton buttonCopy = new JButton ("Copy");
15:   JFileChooser fileChooser = new JFileChooser ("C:\\");
16:   Container c;
17:   String sourceFile;
18:   String destinationFile;
19:   File inputFile;
20:   File outputFile;
21:   FileReader fr;
22:   BufferedReader br;
23:   FileWriter fw;
24:   PrintWriter pw;
25:
26:   CopyFile ()
```

```
27:     {
28:         c = getContentPane ();
29:         GridBagLayout gbl = new GridBagLayout ();
30:         GridBagConstraints gbc = new GridBagConstraints ();
31:         c.setLayout(gbl);
32:
33:         textSourceFile.setEditable(false);
34:         textDestination.setEditable(false);
35:
36:         buttonCopy.setEnabled(false);
37:
38:         gbc.gridx = 1;
39:         gbc.gridy = 1;
40:         gbl.setConstraints(labelSourceFile , gbc);
41:         c.add(labelSourceFile);
42:
43:         gbc.gridx = 1;
44:         gbc.gridy = 2;
45:         gbl.setConstraints(textSourceFile , gbc);
46:         c.add(textSourceFile);
47:
48:         gbc.gridx = 2;
49:         gbc.gridy = 2;
50:         gbl.setConstraints(buttonSourceBrowse , gbc);
51:         c.add(buttonSourceBrowse);
52:
53:         gbc.gridx = 1;
54:         gbc.gridy = 3;
55:         gbl.setConstraints(labelDestination , gbc);
56:         c.add(labelDestination);
57:
58:         gbc.gridx = 1;
59:         gbc.gridy = 4;
60:         gbl.setConstraints(textDestination , gbc);
```

```
61:         c.add(textDestination);
62:
63:         gbc.gridx = 2;
64:         gbc.gridy = 4;
65:         gbl.setConstraints(buttonDestinationBrowse , gbc);
66:         c.add(buttonDestinationBrowse);
67:
68:         gbc.gridx = 2;
69:         gbc.gridy = 5;
70:         gbl.setConstraints(buttonCopy , gbc);
71:         c.add(buttonCopy);
72:
73:         setTitle("Copy");
74:         setSize(600,200);
75:         setVisible(true);
76:
77:         //window listener
78:         addWindowListener(new WindowAdapter()
79:         {
80:             public void windowClosing (WindowEvent w)
81:             {
82:         Object options[] = {"Yes, please" , "No, thanks"};
83:
84:                 int i = JOptionPane.showOptionDialog(c,
85:                 "Are you sure you want to exit?",
86:                 "Exit?",
87:                 JOptionPane.YES_NO_OPTION,
88:                 JOptionPane.QUESTION_MESSAGE,
89:                 null,
90:                 options,
91:                 options[1]);
92:
93:                 if ( i == JOptionPane.YES_OPTION )
94:                 {
```

```
95:                         System.exit(0);
96:                     }
97:
98:         else if ( i == JOptionPane.NO_OPTION )
99:                     {
100:
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CL
OSE);
101:                    }
102:                }
103:        });
104:
105:        //button listeners
106:        ButtonListener blisten = new ButtonListener();
107:        buttonSourceBrowse.addActionListener(blisten);
108:        buttonDestinationBrowse.addActionListener(blisten);
109:        buttonCopy.addActionListener(blisten);
110:    }
111:
112:    class ButtonListener implements ActionListener
113:    {
114:        public void actionPerformed ( ActionEvent e )
115:        {
116:            Object obj = e.getSource();
117:
118:            if ( obj == buttonSourceBrowse )
119:            {
120:        int i = fileChooser.showOpenDialog(c);
121:
122:        if ( i == JFileChooser.APPROVE_OPTION )
123:                    {
124:    sourceFile =
fileChooser.getSelectedFile().getAbsolutePath();
125:                        textSourceFile.setText(sourceFile);
```

```
126:
127:        if ( destinationFile != null && sourceFile != null )
128:                        {
129:                            buttonCopy.setEnabled(true);
130:                        }
131:                    }
132:                }
133:
134:                if ( obj == buttonDestinationBrowse )
135:                {
136:            int i = fileChooser.showOpenDialog(c);
137:
138:            if ( i == JFileChooser.APPROVE_OPTION )
139:                    {
140:  destinationFile=
fileChooser.getSelectedFile().getAbsolutePath();
141:                        textDestination.setText(destinationFile);
142:
143:  if ( destinationFile != null && sourceFile != null )
144:                        {
145:                            buttonCopy.setEnabled(true);
146:                        }
147:                    }
148:                }
149:
150:                if ( obj == buttonCopy )
151:                {
152:                    try
153:                    {
154:            inputFile = new File (sourceFile);
155:            fr = new FileReader (inputFile);
156:            br = new BufferedReader (fr);
157:
158:            outputFile = new File (destinationFile + ".txt");
```

```
159:        fw = new FileWriter (outputFile);
160:        pw = new PrintWriter (fw);
161:
162:        String s;
163:
164:        while ( ( s = br.readLine() ) != null )
165:                    {
166:            pw.write(s + "\n");
167:                    }
168:
169:                    br.close();
170:                    pw.close();
171:
172:                    JOptionPane.showMessageDialog(c,"Copy
Successful");
173:                    System.out.println("Copy Successful");
174:                }
175:
176:                catch (IOException e1)
177:                {
178:                    System.out.println(e1);
179:                }
180:            }
181:        }
182:    }
183:
184:    public static void main ( String[]args )
185:    {
186:        new CopyFile();
187:    }
188: }
```

ثالثاً: شرح الكود:

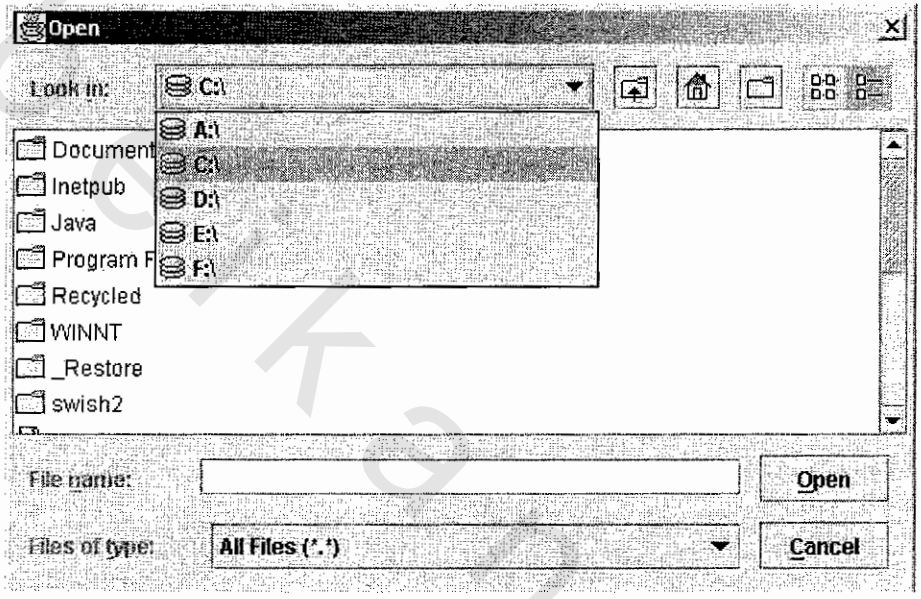
في السطور من 8 إلى 24 يتم تعريف المتغيرات variables التى نحتاجها.

- في السطر رقم 26 يتم تعريف دالة البناء constructor وفيها نبدأ إنشاء الواجهة الرسومية User Interface ، ولذلك فإننا سنختار مدير تخطيط Layout Manager مناسب.
- ومدير تخطيط Layout Manager المناسب هنا هو GridBagLayout وجعلناه هو مدير التخطيط Layout Manager فى السطر رقم 31.
- في السطرين 33 و34 يتم جعل أداة النص JTextField غير قابلة للكتابة والتعديل فيها من المستخدم. فعلنا ذلك لمنع كتابة مسار الملف الذى قد يحدث فيه خطأ فى الكتابة ، فإذا أردت كتابة مسار الملف فلا بد أن تستخدم الزر Browse.
- طالما أنك لم تحدد بعد الملف الذى ستنسخه ، فلا يمكن أن تتم عملية النسخ ولذلك جعلنا الزر Copy غير فعالاً disabled كما في السطر رقم 36.
- في السطور من 38 إلى 71 يتم إنشاء الواجهة الرسومية User Interface كما فعلنا فى الفصول السابقة وليس بها جديد.
- في السطور من 73 إلى 75 يتم تحديد عنوان نافذة البرنامج ليصبح Copy وحجم النافذة 200*600 pixel وتم جعلها ظاهرة.
- في السطور من 77 إلى 103 يتم إضافة دالة method معالجة حدث إغلاق نافذة البرنامج Window Event ، فإذا حاولت إغلاق النافذة فسيظهر لك صندوق به رسالة تحدد هل أنت متأكد أنك تريد إغلاق النافذة أم لا ، فإذا ضغطت "نعم" فإنك ستغلق النافذة ، أما إذا ضغطت "لا" فلا شئ يحدث. هذه السطور شرحناها من قبل فى فصل معالجة الحدث Event Handling ولا جديد بها.
- في السطر رقم 106 يتم تعريف هدف object من نوع الفصيلة ButtonListener وهي الفصيلة class التى سننشئها لمعالجة حدث الضغط على الأزرار.
- في السطور 107 و 108 و 109 يتم إضافة الهدف blisten كمستمع لحدث الضغط علي الأزرار الثلاثة.
- فى السطر رقم 112 يتم تعريف الفصيلة ButtonListener.
- في السطر رقم 116 يتم تحديد مصدر الحدث Event Source عن طريق الدالة getSource().

- في السطر رقم 118 يتم اختبار مصدر الحدث Event Source ، فإذا كنت ضغطت على الزر Browse لتحدد الملف الذي تريد نسخه فإنك تفعل الآتى.
- في السطر رقم 120 يتم إظهار المربع الحوارى الخاص بفتح ملف Dialog Box. لاحظ أن المربع الحوارى Dialog Box لا بد أن يظهر فى محتوى أب Parent Container ، فهنا الهدف object المسمي c -الذي من نوع الفصيلة Container- هو المحتوي الأب Parent Container.
- في السطر رقم 122 يتم اختبار الزر الذى ضغطت عليه ، فإذا كنت ضغطت على الزر Open أو المعروف باسم JFileChooser.APPROVE_OPITON ، فإنك تريد أن تعرف مسار الملف المختار. إذن فلا بد أن نعرف الملف الذى اخترته ، ويتم ذلك عن طريق الدالة getSelectedPath() ، ولمعرفة مسار الملف فإننا نستخدم الدالة getAbsolutePath(). إذن هنا حددنا مسار الملف وسوف نحفظ هذا المسار فى String اسمها sourceFile التى عرفناها فى السطر رقم 17. أيضاً نريد أن نضع مسار الملف الذى نريد نسخه فى أداة النص JTextField الخاصة بالملف المصدر ، ولذلك استخدمنا الدالة setText() الخاصة بأداة النص JTextField المسماة textSourceFile ، ووضعنا فيها الـ String المسماة sourceFile والتى تحدد مسار الملف المصدر.
- في السطر رقم 127 يتم اختبار الـ String المسماة destinationFile و sourceFile ، فإذا كان لهما قيمة -أى لا يساويان null- ، فإننا نجعل الزر Copy فعالاً enabled عن طريق الدالة setEnabled() وتمرير القيمة true لها كما فى السطر رقم 129.
- السطور من 134 إلى 148 تشبه بالضبط السطور من 118 إلى 132 مع الاختلاف أننا هنا سنحدد الملف الذى نريد النسخ إليه.
- في السطر رقم 150 يتم اختبار مصدر الحدث Event Source ، فإذا كان هو الزر buttonCopy فإننا نفعل الآتى.
- في السطور 154 و 155 و 156 يتم تجهيز الملف المختار للقراءة منه ، وقد سبق لنا شرح هذه السطور الثلاثة.

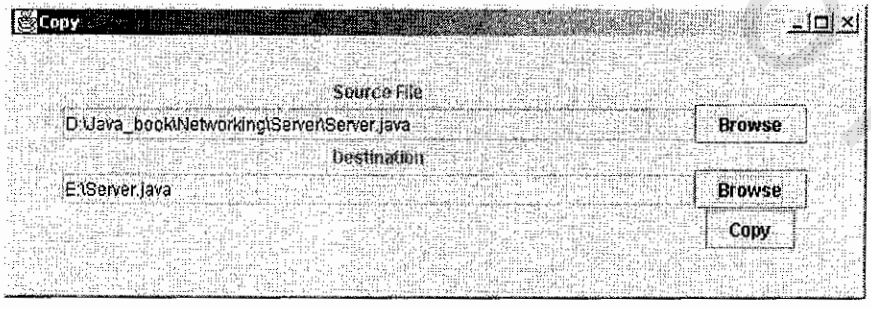
- في السطور 158 و 159 و 160 يتم تجهيز الملف الذى سننسخ إليه ، وقد سبق لنا أيضاً شرح هذه السطور الثلاثة.
- حيث أنك ستقرأ سطراً ثم سطراً من الملف ، إذن فلا بد أن تحفظ هذا السطر فى String ولذلك قمنا بتعريف String اسمها s كما فى السطر رقم 162.
- في السطر رقم 164 يتم استخدام الدالة readLine() التى تستخدم لقراءة سطر كما ذكرنا. الشرط الذى نختبره هو: هل وصلنا إلى نهاية الملف؟ ولذلك استخدمنا الدوارة while loop ، فطالما أنك لم تصل إلى نهاية الملف -التى تعرف بالقيمة null- ، فإننا نستخدم الدالة write() (ارجع للسطر رقم 166) لكتابة هذا السطر فى الملف الجديد ، ثم نضع سطر جديد حتى لا تظهر الجمل وراء بعضها عن طريق القيمة "\n" التى تعنى سطر جديد.
- بعد أن انتهيت من النسخ ، فلا بد أن تغلق التدفق Streams المفتوح حتى لا يعتقد نظام التشغيل Operating System أن الملفات لا زالت مفتوحة وحتى نخلى الذاكرة من هذه الملفات لأننا فعلاً انتهينا منها ، ولذلك نستخدم الدالة close() لإغلاق الـ InputStream والـ OutputStream كما هو واضح في السطر رقم 169 و 170.
- بعد انتهاء عملية النسخ ، فإننا نريد أن نظهر رسالة لبيان أن عملية النسخ قد تمت بنجاح ، ولذلك استخدمنا الأداة JOptionPane ، ولأننا نريد أن نظهر رسالة فقط ، فلقد استخدمنا الدالة showMessageDialog() ، وكما ذكرنا ، لا بد أن تحدد المحتوى الأب Parent Container للأداة JOptionPane وهو الـ Container c ، والقيمة الثانية هى الرسالة التى نريد إظهارها وهى Copy Successful لبيان نجاح عملية النسخ كما هو واضح في السطر رقم 172. انظر شكل (13-8).
- في السطر رقم 173 يتم طباعة نفس الرسالة على شاشة الدوس Dos.
- لاحظ أن التعامل مع التدفق Streams يمكن أن يسبب استثناءاً Exception ، ولذلك استخدمنا جملة try و catch، وهنا الاستثناء Exception الذى يمكن حدوثه هو IOException.

- قم بكتابة سطور البرنامج وتنفيذه لتحصل على نتيجة التنفيذ كما هو واضح فى الشكل (13-5) السابق.
- اضغط الزر Browse الأول ليظهر مربع حوار Open لفتح الملف كما هو واضح فى الشكل (13-6).



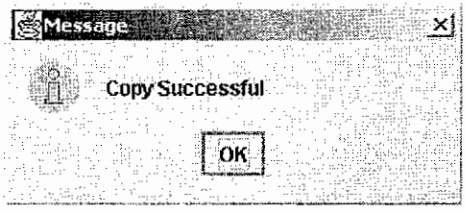
(الشكل 13-6) نسخ ملف

- حدد مسار الملف المطلوب لتحصل على الاسم والمسار كما فى الشكل (13-7) مع تكرار ذلك مع الزر Browse الثانى لتحديد المسار الجديد للملف المنسوخ.



(الشكل 13-7) نسخ ملف

اضغط الزر Copy لتحصل على رسالة تفيد نجاح العملية كما فى الشكل (13-8).



(الشكل 13-8) نسخ ملف

ثانياً: نقل ملف من مكان لأخر File Move:

نقل الملف هى عملية تشبه بالضبط عملية النسخ مع فارق أنك ستلغى الملف المصدر بعد عملية النسخ ، وهذه العملية معروفة باسم القص Cut المشهورة فى نظام التشغيل Operating System كما يتضح لنا من المثال التالي.

مثال (5): نقل ملف من مكان لأخر File Move:

أولاً: هدف المثال:

في هذا المثال نريد عمل نقل لملف من مكان لأخر بلاً من نسخه.

المثال التالى يحتوى على نفس سطور المثال السابق مع زيادة واحدة فقط فى السطور من 178 إلى 183 وهى الخاصة بإلغاء الملف المصدر. استخدمنا الدالة delete() لإلغاء الملف ، وإذا تمت العملية الإلغاء إذن فقد نجحت عملية النقل ولذلك تظهر رسالة تفيد ذلك باستخدام السطر رقم 180.

ثانياً: كود البرمجة:

```
1: import javax.swing.*;
2: import java.io.*;
3: import java.awt.*;
4: import java.awt.event.*;
5:
6: class MoveFile extends JFrame
7: {
8:  JLabel labelSourceFile = new JLabel ("Source File");
```

```
9:  JLabel labelDestination = new JLabel ("Destination");
10:
11:     JTextField textSourceFile = new JTextField (40);
12:     JTextField textDestination = new JTextField (40);
13:
14:     JButton buttonSourceBrowse = new JButton ("Browse");
15:     JButton buttonDestinationBrowse = new JButton
    ("Browse");
16:     JButton buttonMove = new JButton ("Move");
17:
18:     JFileChooser fileChooser = new JFileChooser ("C:\\");
19:
20:     Container c;
21:
22:     String sourceFile;
23:     String destinationFile;
24:
25:     File inputFile;
26:     File outputFile;
27:     FileReader fr;
28:     BufferedReader br;
29:     FileWriter fw;
30:     PrintWriter pw;
31:
32:     MoveFile()
33:     {
34:         c = getContentPane ();
35:         GridBagLayout gbl = new GridBagLayout ();
36:         GridBagConstraints gbc = new GridBagConstraints ();
37:         c.setLayout(gbl);
38:
39:         textSourceFile.setEditable(false);
40:         textDestination.setEditable(false);
41:
42:         buttonMove.setEnabled(false);
```

```
43:
44:         gbc.gridx = 1;
45:         gbc.gridy = 1;
46:         gbl.setConstraints(labelSourceFile , gbc);
47:         c.add(labelSourceFile);
48:
49:         gbc.gridx = 1;
50:         gbc.gridy = 2;
51:         gbl.setConstraints(textSourceFile , gbc);
52:         c.add(textSourceFile);
53:
54:         gbc.gridx = 2;
55:         gbc.gridy = 2;
56:         gbl.setConstraints(buttonSourceBrowse , gbc);
57:         c.add(buttonSourceBrowse);
58:
59:         gbc.gridx = 1;
60:         gbc.gridy = 3;
61:         gbl.setConstraints(labelDestination , gbc);
62:         c.add(labelDestination);
63:
64:         gbc.gridx = 1;
65:         gbc.gridy = 4;
66:         gbl.setConstraints(textDestination , gbc);
67:         c.add(textDestination);
68:
69:         gbc.gridx = 2;
70:         gbc.gridy = 4;
71:         gbl.setConstraints(buttonDestinationBrowse , gbc);
72:         c.add(buttonDestinationBrowse);
73:
74:         gbc.gridx = 2;
75:         gbc.gridy = 5;
76:         gbl.setConstraints(buttonMove , gbc);
77:         c.add(buttonMove);
```

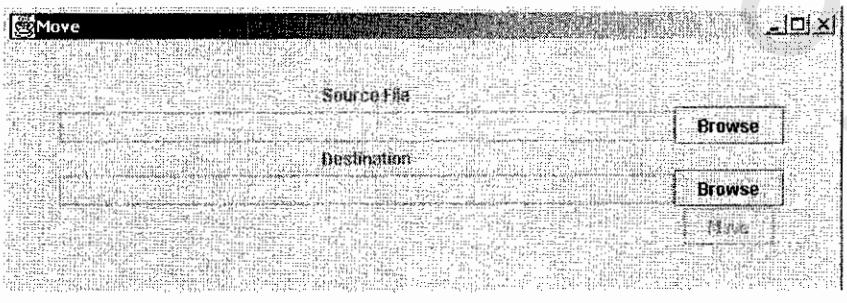
```
78:
79:         setTitle("Move");
80:         setSize(600,200);
81:         setVisible(true);
82:
83:         //window listener
84:         addWindowListener(new WindowAdapter()
85:         {
86:     public void windowClosing (WindowEvent w)
87:             {
88:     Object options[] = {"Yes, please" , "No, thanks"};
89:
90:         int i = JOptionPane.showOptionDialog(c,
91:                 "Are you sure you want to exit?",
92:                 "Exit?",
93:                 JOptionPane.YES_NO_OPTION,
94:                 JOptionPane.QUESTION_MESSAGE,
95:                 null,
96:                 options,
97:                 options[1]);
98:
99:                 if ( i == JOptionPane.YES_OPTION )
100:                 {
101:                     System.exit(0);
102:                 }
103:
104:        else if ( i == JOptionPane.NO_OPTION )
105:                 {
106:
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_C
    LOSE);
107:                 }
108:             }
109:         });
110:
```

```
111:        //button listeners
112:        ButtonListener blisten = new ButtonListener();
113:            buttonSourceBrowse.addActionListener(blisten);
114:
        buttonDestinationBrowse.addActionListener(blisten);
115:            buttonMove.addActionListener(blisten);
116:        }
117:
118:        class ButtonListener implements ActionListener
119:        {
120:            public void actionPerformed ( ActionEvent e )
121:            {
122:                Object obj = e.getSource();
123:
124:                if ( obj == buttonSourceBrowse )
125:                {
126:    int i = fileChooser.showOpenDialog(c);
127:
128:    if ( i == JFileChooser.APPROVE_OPTION )
129:                        {
130:    sourceFile                                                    =
      fileChooser.getSelectedFile().getAbsolutePath();
131:                            textSourceFile.setText(sourceFile);
132:
133:        if ( destinationFile != null && sourceFile != null )
134:                            {
135:                                buttonMove.setEnabled(true);
136:                            }
137:                        }
138:                }
139:
140:    if ( obj == buttonDestinationBrowse )
141:                {
142:    int i = fileChooser.showOpenDialog(c);
143:
```

```
144:   if ( i == JFileChooser.APPROVE_OPTION )
145:                       {
146:         destinationFile=fileChooser.getSelectedFile().getAbsolut
     ePath();
147:
         textDestination.setText(destinationFile);
148:
149:                           if ( destinationFile != null &&
   sourceFile != null )
150:                           {
151:                               buttonMove.setEnabled(true);
152:                           }
153:                       }
154:                   }
155:
156:                   if ( obj == buttonMove )
157:                   {
158:                   try
159:                       {
160:                   inputFile = new File (sourceFile);
161:                   fr = new FileReader (inputFile);
162:                   br = new BufferedReader (fr);
163:
164:         outputFile = new File (destinationFile + ".txt");
165:         fw = new FileWriter (outputFile);
166:         pw = new PrintWriter (fw);
167:
168:                   String s;
169:
170:                   while ( ( s = br.readLine() ) != null )
171:                   {
172:                   pw.write(s + "\n");
173:                   }
174:
175:                   br.close();
```

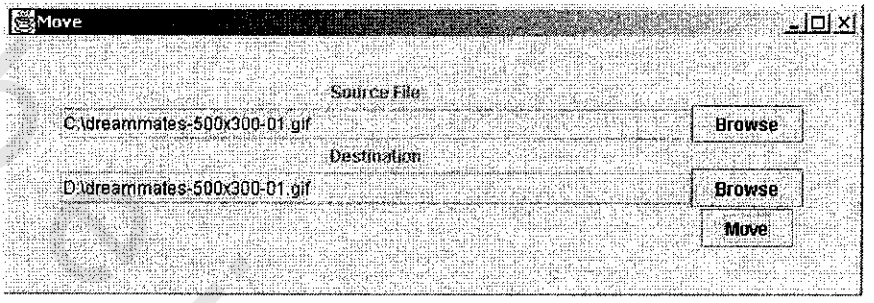
```
176:                pw.close();
177:
178:                if ( inputFile.delete() == true )
179:                {
180:
      JOptionPane.showMessageDialog(c,"Move Successful");
181:                                System.out.println("Move
  Successful");
182:                            }
183:                        }
184:
185:                        catch (IOException e1)
186:                        {
187:                            System.out.println(e1);
188:                        }
189:                }
190:            }
191:        }
192:
193:        public static void main ( String[]args )
194:        {
195:            new MoveFile();
196:    }
197:    }
```

قم بتنفيذ البرنامج لتظهر لك الشاشة كما هو واضح في شكل (13-9).



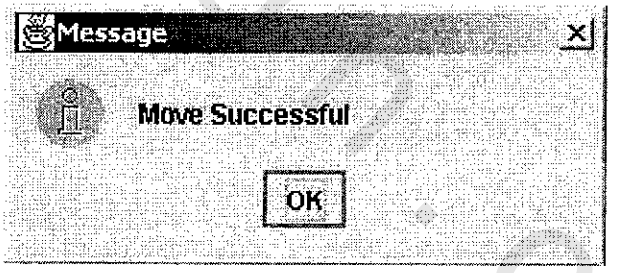
(الشكل 13-9) نقل ملف

- اضغط الزر Browse الأول ليظهر مربع حوار Open لفتح الملف.
- حدد مسار الملف المطلوب لتحصل على الاسم والمسار كما فى الشكل (13-10) مع تكرار ذلك مع الزر Browse الثانى لتحديد المسار الجديد.



(الشكل 13-10) نقل ملف

- اضغط الزر Move لتحصل على رسالة تفيد نجاح العملية كما هو واضح فى الشكل (13-11).



(الشكل 13-11) نقل ملف

مثال (6): إلغاء ملف File Delete:

- يتشابه هذا البرنامج أيضاً مع البرامج السابقة ولكن بتعديلات بسيطة لتتوافق مع الواجهة الرسومية الجديدةUser Interface .
- الجديد فى هذا البرنامج هو السطور من 112 إلى 118.
- السطر رقم 112 يختبر مصدر الحدث Event Source فإذا كان مصدر الحدث Event Source هو الزر Delete ، فإننا نستخدم الدالة ()delete لإلغاء الملف

الذى نريده كما في السطر رقم 115 ، ثم نظهر رسالة لبيان نجاح عملية الإلغاء كما في السطر رقم 116.

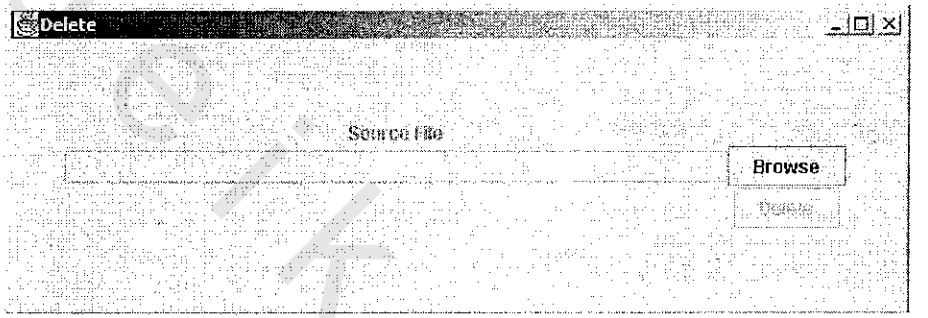
```
1:      import javax.swing.*;
2:      import java.io.*;
3:      import java.awt.*;
4:      import java.awt.event.*;
5:
6:      class DeleteFile extends JFrame
7:      {
8:          JLabel labelSourceFile = new JLabel ("Source File");
9:
10:         JTextField textSourceFile = new JTextField (40);
11:
12:         JButton buttonSourceBrowse = new JButton
("Browse");
13:         JButton buttonDelete = new JButton ("Delete");
14:
15:         JFileChooser fileChooser = new JFileChooser ("C:\\");
16:
17:         Container c;
18:
19:         String sourceFile;
20:
21:         DeleteFile ()
22:         {
23:             c = getContentPane ();
24:             GridBagLayout gbl = new GridBagLayout ();
25:             GridBagConstraints gbc = new
GridBagConstraints ();
26:             c.setLayout(gbl);
27:
```

```
28:             textSourceFile.setEditable(false);
29:
30:             buttonDelete.setEnabled(false);
31:
32:             gbc.gridx = 1;
33:             gbc.gridy = 1;
34:             gbl.setConstraints(labelSourceFile , gbc);
35:             c.add(labelSourceFile);
36:
37:             gbc.gridx = 1;
38:             gbc.gridy = 2;
39:             gbl.setConstraints(textSourceFile , gbc);
40:             c.add(textSourceFile);
41:
42:             gbc.gridx = 2;
43:             gbc.gridy = 2;
44:             gbl.setConstraints(buttonSourceBrowse , gbc);
45:             c.add(buttonSourceBrowse);
46:
47:             gbc.gridx = 2;
48:             gbc.gridy = 5;
49:             gbl.setConstraints(buttonDelete , gbc);
50:             c.add(buttonDelete);
51:
52:             setTitle("Delete");
53:             setSize(600,200);
54:             setVisible(true);
55:
56:             //window listener
57:             addWindowListener(new WindowAdapter()
58:             {
59:                 public void windowClosing (WindowEvent w)
60:                 {
```

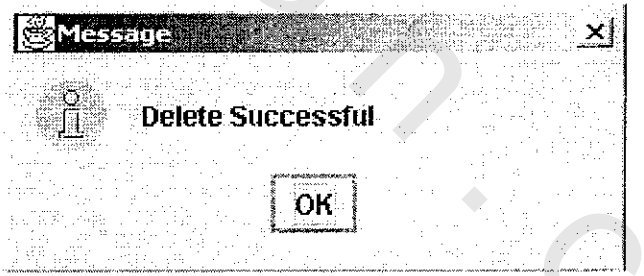
```
61:                     Object options[] = {"Yes, please" , "No,
thanks"};
62:
63:                     int i = JOptionPane.showOptionDialog(c,
64:                     "Are you sure you want to exit?",
65:                     "Exit?",
66:                     JOptionPane.YES_NO_OPTION,
67:                     JOptionPane.QUESTION_MESSAGE,
68:                     null,
69:                     options,
70:                     options[1]);
71:
72:                     if ( i == JOptionPane.YES_OPTION )
73:                     {
74:                         System.exit(0);
75:                     }
76:
77:                     else if ( i == JOptionPane.NO_OPTION )
78:                     {
79:
      setDefaultCloseOperation(JFrame.DO_NOTHING_ON_C
LOSE);
80:                     }
81:                 }
82:             });
83:
84:             //button listeners
85:             ButtonListener blisten = new ButtonListener();
86:             buttonSourceBrowse.addActionListener(blisten);
87:             buttonDelete.addActionListener(blisten);
88:         }
89:
90:         class ButtonListener implements ActionListener
```

```
91:         {
92:             public void actionPerformed ( ActionEvent e )
93:             {
94:                 Object obj = e.getSource();
95:
96:                 if ( obj == buttonSourceBrowse )
97:                 {
98:                     int i = fileChooser.showOpenDialog(c);
99:
100:                    if ( i ==
JFileChooser.APPROVE_OPTION )
101:                    {
102:                        sourceFile =
fileChooser.getSelectedFile().getAbsolutePath();
103:                        textSourceFile.setText(sourceFile);
104:
105:                        if ( sourceFile != null )
106:                        {
107:                            buttonDelete.setEnabled(true);
108:                        }
109:                    }
110:                }
111:
112:                if ( obj == buttonDelete )
113:                {
114:                    File DeletedFile = new File (sourceFile);
115:                    DeletedFile.delete();
116:
        JOptionPane.showMessageDialog(c,"Delete Successful");
117:                    System.out.println("Delete successful");
118:                }
119:            }
120:        }
```

```
121:
122:        public static void main ( String[]args )
123:        {
124:            new DeleteFile();
125:        }
126:    }
```



(الشكل 13-12) إلغاء ملف



(الشكل 13-13) إلغاء ملف

مثال (7): إعادة تسمية ملف File Rename:

- أيضاً يتشابه هذا البرنامج مع البرامج السابقة والجديد هنا هو السطور من 127 إلى 134.
- السطر رقم 127 يقوم بعملية هامة جداً ، فهو يقوم بإنشاء String تحمل مسار الملف القديم ثم اسمه الجديد ، فمثلاً إذا أردت أن تعيد تسمية الملف Rename الموجود في المسار C:\\MyDocuments\\File1.txt إلى الملف C:\\MyDocuments\\File2.txt ، فإنك تريد معرفة مسار المجلد الأب Parent

Directory للملف الأول وهو C:\\MyDocuments ، ولذلك استخدمنا الدالة getParent() التى تتولى هذه الوظيفة ، وكما ذكرنا فإن الملف الجديد يصبح مساره هو C:\\MyDocuments\\File2.txt.

- الآن حددنا المسار C:\\MyDocuments ، ولذلك أضفنا \\ ثم أضفنا اسم الملف الجديد الذى كتبته فى أداة النص JTextField عن طريق الدالة getText().
- السطر رقم 129 ينشىء هدفاً object من نوع File للملف القديم الذى مساره يعرف من الـ String المسماة sourceFile كما هو واضح في السطر رقم 115.
- السطر رقم 130 ينشىء هدفاً object من نوع File للملف الجديد الذى مساره يعرف من الـ String المسماة newFileName كما هو واضح في السطر رقم 127.
- السطر رقم 131 يعيد تسمية الملف القديم Rename ليصبح هو الملف الجديد عن طريق الدالة renameTo() ، وبالطبع لا بد أن تمرر لهذه الدالة method الاسم الجديد وهو newFileName.
- السطر رقم 133 يظهر رسالة تفيد نجاح عملية إعادة التسمية Rename. انظر شكل (13-14) و (13-15) و (13-16).

```
1: import javax.swing.*;
2: import java.io.*;
3: import java.awt.*;
4: import java.awt.event.*;
5:
6: class RenameFile extends JFrame
7: {
8:       JLabel labelSourceFile = new JLabel ("Source File");
9:       JLabel labelNewName = new JLabel ("Enter File New
         Name");
10:
11:          JTextField textSourceFile = new JTextField (40);
12:          JTextField textNewName = new JTextField (40);
13:
```

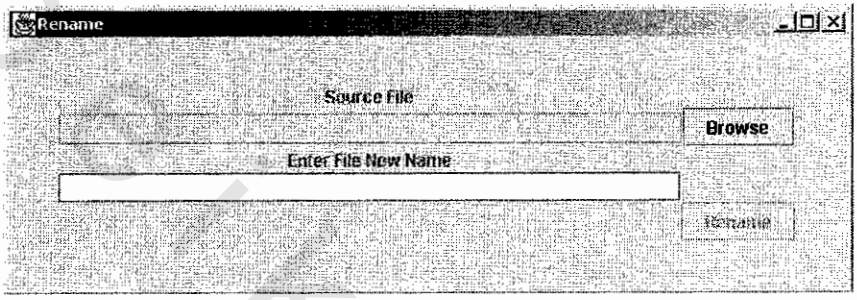
```
14:         JButton buttonSourceBrowse = new JButton
      ("Browse");
15:         JButton buttonRename = new JButton ("Rename");
16:
17:         JFileChooser fileChooser = new JFileChooser ("C:\\");
18:
19:         Container c;
20:
21:         String sourceFile;
22:         String newFileName;
23:
24:         RenameFile ()
25:         {
26:             c = getContentPane ();
27:             GridBagLayout gbl = new GridBagLayout ();
28:             GridBagConstraints gbc = new
      GridBagConstraints ();
29:             c.setLayout(gbl);
30:
31:             textSourceFile.setEditable(false);
32:
33:             buttonRename.setEnabled(false);
34:
35:             gbc.gridx = 1;
36:             gbc.gridy = 1;
37:             gbl.setConstraints(labelSourceFile , gbc);
38:             c.add(labelSourceFile);
39:
40:             gbc.gridx = 1;
41:             gbc.gridy = 2;
42:             gbl.setConstraints(textSourceFile , gbc);
43:             c.add(textSourceFile);
44:
45:             gbc.gridx = 2;
```

```
46:             gbc.gridy = 2;
47:             gbl.setConstraints(buttonSourceBrowse , gbc);
48:             c.add(buttonSourceBrowse);
49:
50:             gbc.gridx = 1;
51:             gbc.gridy = 3;
52:             gbl.setConstraints(labelNewName , gbc);
53:             c.add(labelNewName);
54:
55:             gbc.gridx = 1;
56:             gbc.gridy = 4;
57:             gbl.setConstraints(textNewName , gbc);
58:             c.add(textNewName);
59:
60:             gbc.gridx = 2;
61:             gbc.gridy = 5;
62:             gbl.setConstraints(buttonRename , gbc);
63:             c.add(buttonRename);
64:
65:             setTitle("Rename");
66:             setSize(600,200);
67:             setVisible(true);
68:
69:             //window listener
70:             addWindowListener(new WindowAdapter()
71:             {
72:                 public void windowClosing (WindowEvent w)
73:                 {
74:                     Object options[] = {"Yes, please" , "No,
    thanks"};
75:
76:                     int i = JOptionPane.showOptionDialog(c,
77:                     "Are you sure you want to exit?",
78:                     "Exit?",
```

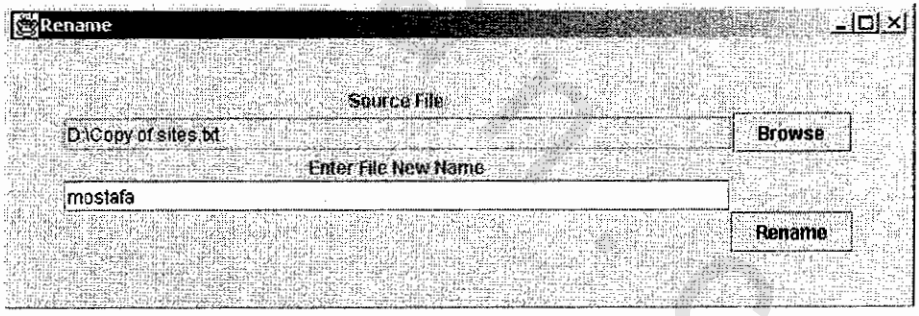
```
79:                     JOptionPane.YES_NO_OPTION,
80:                     JOptionPane.QUESTION_MESSAGE,
81:                     null,
82:                     options,
83:                     options[1]);
84:
85:                     if ( i == JOptionPane.YES_OPTION )
86:                     {
87:                         System.exit(0);
88:                     }
89:
90:                     else if ( i == JOptionPane.NO_OPTION )
91:                     {
92:
        setDefaultCloseOperation(JFrame.DO_NOTHING_O
    N_CLOSE);
93:                     }
94:                 }
95:             });
96:
97:             //button listeners
98:             ButtonListener blisten = new ButtonListener();
99:             buttonSourceBrowse.addActionListener(blisten);
100:            buttonRename.addActionListener(blisten);
101:        }
102:
103:        class ButtonListener implements ActionListener
104:        {
105:            public void actionPerformed ( ActionEvent e )
106:            {
107:                Object obj = e.getSource();
108:
109:                if ( obj == buttonSourceBrowse )
110:                {
```

```
111:                    int i = fileChooser.showOpenDialog(c);
112:
113:                    if ( i ==
    JFileChooser.APPROVE_OPTION )
114:                    {
115:                        sourceFile =
    fileChooser.getSelectedFile().getAbsolutePath();
116:                    textSourceFile.setText(sourceFile);
117:
118:                    if ( newFileName != "" && sourceFile
    != null )
119:                    {
120:                        buttonRename.setEnabled(true);
121:                    }
122:                }
123:                }
124:
125:                if ( obj == buttonRename )
126:                {
127:                    newFileName =
    fileChooser.getSelectedFile().getParent()+"\\"+textNewNa
    me.getText();
128:
129:                    File oldFile = new File (sourceFile);
130:                    File newFile = new File (newFileName);
131:                    oldFile.renameTo(newFile);
132:                    System.out.println(newFileName);
133:
        JOptionPane.showMessageDialog(c,"Rename
    Successful");
134:                    System.out.println("Rename successful");
135:                }
136:            }
137:        }
```

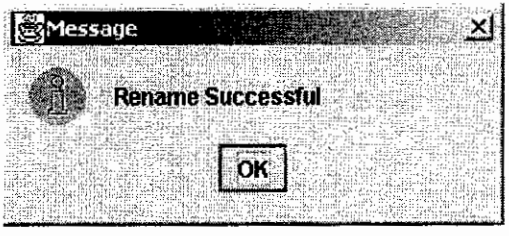
```
138:
139:        public static void main ( String[]args )
140:        {
141:            new RenameFile();
142:        }
143:    }
```



(الشكل 13-14) إعادة تسمية ملف Rename



(الشكل 13-15) إعادة تسمية ملف Rename



(الشكل 13-16) إعادة تسمية ملف Rename

كيفية استخدام الأربع برامج السابقة:

- شرحنا البرامج السابقة من وجهة نظرك كمبرمج يكتب الكود الخاص بالبرنامج ، وهنا نريد أن نتأكد بأنفسنا من صحة ما فعلناه.

أولاً: برنامج النسخ Copy:

- اضغط على الزر Browse الأول ليظهر لك مربع فتح ملف Dialog Box. اختر أي مجلد Folder ثم اضغط Open.
- لاحظ كتابة مسار الملف فى أداة النص JTextField الأولي.
- اضغط على الزر Browse الثانى ليظهر لك مربع فتح ملف Dialog Box. اختر D:\ ثم فى الجزء الخاص باسم الملف File Name اكتب الاسم الجديد للملف وليكن New ثم اضغط على Open.
- لاحظ كتابة مسار الملف فى أداة النص JTextField الثانية.
- لاحظ أن الملف الأول موجود فعلياً على القرص الصلب Hard Disk ولكن الثاني غير موجود ولكننا نريد الآن إنشاءه.
- لاحظ أيضاً أنه الآن أصبح الزر Copy فعالاً enabled ويمكننا الضغط عليه.
- اضغط الآن على الزر Copy ولاحظ ظهور رسالة Copy successful.
- اضغط على Ok.
- الآن حاول الضغط على زر إغلاق النافذة. سوف يظهر لك مربع حوار Dialog Box للتأكد من أنك تريد إغلاق البرنامج. إذا ضغطت على No,Thanks فلا شئ يحدث ، وأما إذا ضغطت على Yes , Please فذلك يغلق البرنامج.
- الآن إذا فتحت المسار :D من My Computer من على شاشة سطح المكتب Desktop ، فإنك ستجد الملف New.txt موجوداً وتستطيع فتحه ببرنامج NotePad ، وبذلك نكون قد تأكدنا من نجاح البرنامج وعمله بالشكل المطلوب.

ثانياً: باقى البرامج:

- باقى البرامج تتشابه مع البرنامج السابق ولذلك سنترك لك فرصة تجربة البرامج بنفسك والتأكد من صحتها وعملها بالشكل الذى أردته.

حفظ واسترجاع بيانات الأهداف (النشر التسلسلي) Serialization:

- تعلمنا فيما سبق كيفية إنشاء فصائل classes وكيفية إنشاء هدف object منها. أيضاً تعلمنا أن كل هدف object له متغيرات خاصة به ، وعند تشغيلك للبرنامج فإن قيم هذه المتغيرات تتغير طبقاً للبرنامج ، ولكن للأسف لا يمكنك حفظ قيم هذه المتغيرات عندما تغلق البرنامج.
- فمثلاً افترض أنك صممت برنامجاً يطلب من المستخدم اسمه الأول ولقب العائلة ومدينته ، إذا أدخل المستخدم هذه البيانات ثم أغلق البرنامج فإن هذه البيانات تضيع.
- ما نريده الآن هو: هل توجد وسيلة لحفظ بيانات الهدف object الذى أنشأناه فى ملف بحيث أننا إذا أردنا حفظ واسترجاع البيانات فذلك يصبح فى منتهى السهولة.
- وسوف تجد أن البيانات تكون محفوظة فى ملف حتى إذا أغلقت البرنامج فإنه يمكن استرجاع البيانات من هذا الملف.
- إذن النشر التسلسلي Serialization هى عملية حفظ متغيرات الهدف object بتحويلها إلى بايت Byte وتخزينها فى ملف مع إمكانية استرجاعها فيما بعد.

حفظ بيانات الأهداف objects:

- إذا كان عندك فصيلة class وفيها بيانات وتريد أن تحفظ هذه البيانات ، فلا بد أن تنفذ هذه الفصيلة class الـ interface المسمى Serializable ، وفى هذه الحالة يمكن حفظ متغيرات الهدف object كما في السطور التالية.

```
class Data implements Serializable
{
    String firstName;
    String lastName;
    String country;
}
```

- فهنا قمنا بتعريف فصيلة class اسمها Data تحتوى على ثلاث متغيرات.
- كوننا جعلنا الفصيلة Data تنفذ الـ interface المسمي Serializable ، فذلك معناه أننا بإمكاننا حفظ هذه البيانات فى ملف.

- لاحظ أن عملية الحفظ لم تتم بعد مع العلم أن عملية الحفظ تتم على بيانات الهدف object وليس الفصيلة class.
- إذن هنا قمنا بتمكين عملية الحفظ دون القيام بعملية الحفظ نفسها.

حفظ بيانات الهدف object فى ملف:

- لكى نحفظ بيانات الهدف object فلا بد أن ننشئه أولاً كما فى السطر التالي:

```
Data Employee = new Data ();
```

- ثم نملأ بياناته كما فى السطور التالية:

```
Employee.firstName = "Mostafa";
Employee.lastName = "Maged";
Employee.country = "Egypt";
```

- الآن نريد حفظ هذه البيانات Egypt, Maged, Mostafa ، ولذلك نستخدم الفصيلة FileOutputStream والفصيلة ObjectOutputStream.
- لاحظ أنه فى هذه الحالة فإنك تتعامل مع مجموعة بيانات أى تتعامل مع تدفق Stream بحيث أن مصدره هو بيانات الهدف object ثم يتم حفظه فى ملف أي أن الملف هو جهة الوصول ، ولذلك لا تصلح الفصيلة FileWriter لأن مصدر البيانات هنا ليس ملفاً ولكنه بيانات الهدف object.
- أيضاً لا يمكننا استخدام الفصيلة FileOutputStream مباشرة لأننا سنكتب هنا بيانات هدف object ، ولذلك نستخدم الفصيلة ObjectOutputStream كالآتى.

```
FileOutputStream fos = new FileOutputStream ("data .txt");
ObjectOutputStream oos = new ObjectOutputStream (fos);
```

- في السطر الأول يتم تحديد الملف الذى سنقوم بحفظ البيانات فيه.
- في السطر الثانى يتم تحديد أننا سنحفظ بيانات الهدف object فى fos الذى بدوره يحفظ البيانات فى الملف data.txt.
- توفر الفصيلة ObjectOutputStream دالة method تسمى writeObject() لحفظ البيانات ، وهي تستقبل معاملاً Parameter يمثل الهدف object الذى تريد

حفظه ، فمثلاً إذا أردت حفظ بيانات الهدف object المسمي Employee الذى من نوع الفصيلة Data وهى Egypt, Maged, Mostafa فإننا نكتب.

```
oos.writeObject (Employee);
```

- إذا أنشأت هدفاً object أخر مثلاً اسمه Engineer كالآتى:

```
Data Engineer = new Data ();
Engineer.firstName = "Ahmed";
Engineer.lastName = "Ali";
Engineer.country = "Egypt";
```

- لحفظ هذه البيانات فإنك تكتب

```
oos.writeObject (Engineer);
```

- وهكذا كلما أردت حفظ بيانات هدف object فإنك تمرر اسمه إلى الدالة writeObject().
- بعد انتهائك من حفظ جميع بيانات الهدف object ، فلا بد أن تغلق ملفاتك كما فعلنا من قبل مع الملفات وأيضاً سنستخدم الدالة close() من أجل ذلك كالآتى:

```
oos.close ();
```

قراءة بيانات الهدف object من ملف:

- بعد حفظ البيانات وإغلاق البرنامج فإنك تريد أن تعيد استرجاع البيانات التي قمت بحفظها في الملف.
- لاسترجاع البيانات فإننا نستخدم الفصيلة FileInputStream لأن مصدر البيانات هو ملف وسيتم تحميل هذه البيانات فى هدف object ، أي أن جهة الوصول هي هدف object.
- أيضاً لا يمكنك القراءة من الفصيلة FileInputStream مباشرة لأننا هنا سنقرأ هدفاً object ولذلك نستخدم الفصيلة ObjectInputStream كالآتى.

```
FileInputStream fis = new FileInputStream ("data.txt");
ObjectInputStream ois = new ObjectInputStream (fis);
```

- السطر الأول يحدد اسم الملف الذى سنقرأ منه ، لاحظ أن هذا الملف لا بد أن يكون موجوداً ولا بد أن يحوى البيانات التى حفظتها من قبل.
- السطر الثانى يحدد أننا سنقرأ بيانات الهدف object من fis الذى بدوره يقرأ من الملف المسمى data.txt.
- توفر الفصيلة ObjectInputStream دالة method تسمى readObject() لقراءة البيانات ، فمثلاً إذا أردت استرجاع بيانات الهدف object المسمى Engineer ، فلا بد أن تنشئ هدفاً object جديداً وتساوى بياناته ببيانات Engineer كالآتى:

```
Data eng = (Data) ois.readObject ();
```

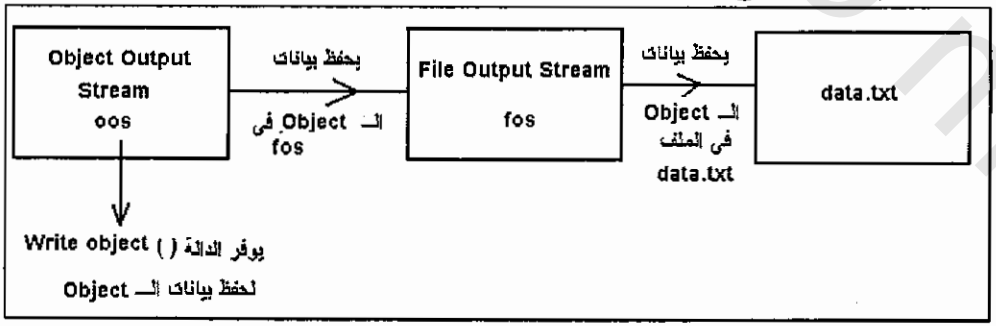
- لاحظ عملية التحويل Casting وذلك لأن القيمة المرتجعة Return Value من الدالة readObject() تكون من نوع الفصيلة Object ولذلك عملنا بعمل تحويل Casting لكى نستطيع مساواة الهدف object الذى قرأناه من الملف بالهدف object الذى أنشأناه باسم eng.
- أيضاً كلما أردت قراءة بيانات أي هدف object قمت بتخزينه ، فإنك تستخدم الدالة readObject() مع القيام بعملية التحويل Casting ، فمثلاً لقراءة بيانات object أخر.

```
Data eng2 = (Data) ois.readObject ();
```

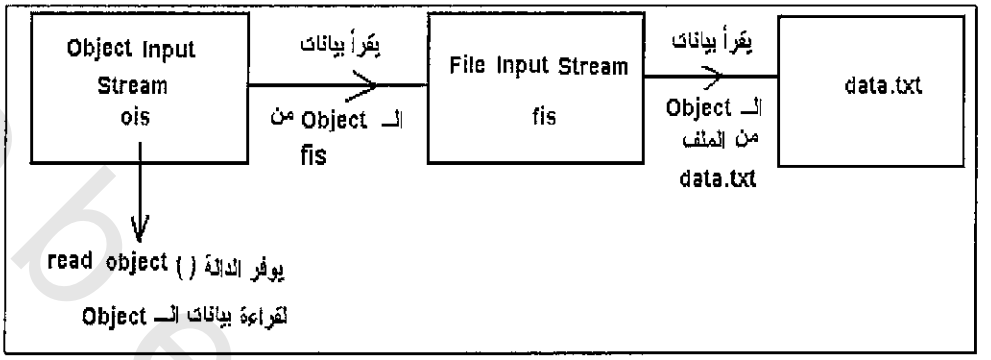
- وكما تعودنا لا بد أن نغلق التدفق Stream عن طريق الدالة close () كالآتى:

```
ois.close ();
```

- الرسم التالى يوضح عملية حفظ الهدف object فى ملف.



الرسم التالى يوضح عملية قراءة بيانات الهدف object من ملف.



المثال التالى يوضح عملية النشر التسلسلي Serialization.

مثال (8): النشر التسلسلي Serialization:

أولاً: هدف المثال:

نريد أن ننشىء واجهة بسيطة نطلب فيها اسم الشخص ولقب العائلة ومدينته ، ثم يكون عندنا 3 أزرار حيث يقوم الزر Save بحفظ البيانات فى ملف نختار اسمه ، ويقوم الزر Load باسترجاع البيانات من الملف الذى نقوم بتحديده ، ويقوم الزر Clear بإلغاء البيانات التي أدخلتها.

ثانياً: كود البرمجة:

```
1:  import javax.swing.*;
2:  import java.awt.*;
3:  import java.awt.event.*;
4:  import java.io.*;
5:
6:  class Input extends JFrame
7:  {
8:          JLabel labelFirstName = new JLabel ("First Name");
9:          JLabel labelLastName = new JLabel ("Last Name");
10:         JLabel labelCountry = new JLabel ("Country");
11:
```

```
12:             JTextField textFirstName = new JTextField(15);
13:             JTextField textLastName = new JTextField(15);
14:             JTextField textCountry = new JTextField(15);
15:
16:             JButton buttonSave = new JButton ("Save");
17:             JButton buttonLoad = new JButton ("Load");
18:             JButton buttonClear = new JButton ("Clear");
19:
20:             JFileChooser choose = new JFileChooser("C:\\");
21:
22:             Container c;
23:
24:             Input()
25:             {
26:                GridBagLayout gbl = new GridBagLayout ();
27:                GridBagConstraints gbc = new
    GridBagConstraints ();
28:
29:                c = getContentPane();
30:                c.setLayout(gbl);
31:
32:                gbc.gridx = 1;
33:                gbc.gridy = 1;
34:                gbl.setConstraints(labelFirstName , gbc);
35:                c.add(labelFirstName);
36:
37:                gbc.gridx = 2;
38:                gbc.gridy = 1;
39:                gbl.setConstraints(textFirstName , gbc);
40:                c.add(textFirstName);
41:
42:                gbc.gridx = 1;
43:                gbc.gridy = 2;
44:                gbl.setConstraints(labelLastName , gbc);
45:                c.add(labelLastName);
```

```
46:
47:             gbc.gridx = 2;
48:             gbc.gridy = 2;
49:             gbl.setConstraints(textLastName , gbc);
50:             c.add(textLastName);
51:
52:             gbc.gridx = 1;
53:             gbc.gridy = 3;
54:             gbl.setConstraints(labelCountry , gbc);
55:             c.add(labelCountry);
56:
57:             gbc.gridx = 2;
58:             gbc.gridy = 3;
59:             gbl.setConstraints(textCountry , gbc);
60:             c.add(textCountry);
61:
62:             gbc.gridx = 1;
63:             gbc.gridy = 4;
64:             gbl.setConstraints(buttonSave , gbc);
65:             c.add(buttonSave);
66:
67:             gbc.gridx = 2;
68:             gbc.gridy = 4;
69:             gbl.setConstraints(buttonLoad , gbc);
70:             c.add(buttonLoad);
71:
72:             gbc.gridx = 3;
73:             gbc.gridy = 4;
74:             gbl.setConstraints(buttonClear , gbc);
75:             c.add(buttonClear);
76:
77:             //button listeners
78:             ButtonListener blisten = new ButtonListener ();
79:             buttonSave.addActionListener (blisten);
80:             buttonLoad.addActionListener (blisten);
```

```
81:             buttonClear.addActionListener (blisten);
82:
83:             //window listener
84:             addWindowListener(new WindowAdapter()
85:             {
86:                 public void windowClosing (WindowEvent w)
87:                 {
88: Object options[] = {"Yes, please" , "No, thanks"};
89:
90:             int i = JOptionPane.showOptionDialog(c,
91:             "Are you sure you want to exit?",
92:                     "Exit?",
93:                     JOptionPane.YES_NO_OPTION,
94:                     JOptionPane.QUESTION_MESSAGE,
95:                     null,
96:                     options,
97:                     options[1]);
98:
99:             if ( i == JOptionPane.YES_OPTION )
100:                     {
101:                         System.exit(0);
102:                     }
103:
104:            else if ( i == JOptionPane.NO_OPTION )
105:                    {
106:
            setDefaultCloseOperation(JFrame.DO_NOTHING_
    ON_CLOSE);
107:                    }
108:                }
109:            });
110:
111:            setTitle("Serialization");
112:            setSize(400,200);
113:            setVisible(true);
```

```
114:            }
115:
116:          class ButtonListener implements ActionListener
117:            {
118:          public void actionPerformed ( ActionEvent e )
119:                {
120:            Object obj = e.getSource();
121:
122:                      if ( obj == buttonSave )
123:                      {
124:          Data Employee = new Data ();
125:          Employee.firstName = textFirstName.getText();
126:          Employee.lastName = textLastName.getText();
127:          Employee.country = textCountry.getText();
128:
129:          try
130:          {
131:          int i = choose.showSaveDialog(c);
132:
133:          if ( i == JFileChooser.APPROVE_OPTION )
134:          {
135:          String                    fileName                    =
    choose.getSelectedFile().getAbsolutePath();
136:                                FileOutputStream    fos    =    new
    FileOutputStream (fileName);
137:                                ObjectOutputStream  oos  =  new
    ObjectOutputStream (fos);
138:                                      oos.writeObject(Employee);
139:                                      oos.close();
140:
141:                      JOptionPane.showMessageDialog(c,"Save
    Successful");
142:            }
143:            }
144:
```

```
145:              catch (Exception e1 )
146:              {
147:                                   System.out.println("Input/Output
    exception");
148:              }
149:              }
150:
151:              else if ( obj == buttonLoad )
152:              {
153:              try
154:                                   {
155:              int i = choose.showOpenDialog(c);
156:
157:              if ( i == JFileChooser.APPROVE_OPTION )
158:                                   {
159:              String                    fileName                    =
    choose.getSelectedFile().getAbsolutePath();
160:                                   FileInputStream fis = new
    FileInputStream (fileName);
161:                                   ObjectInputStream ois = new
    ObjectInputStream (fis);
162:                                   Data SavedEmployee =
    (Data)ois.readObject();
163:                                   ois.close();
164:
165:
                  textFirstName.setText(SavedEmployee.firstName);
166:
                  textLastName.setText(SavedEmployee.lastName);
167:
                  textCountry.setText(SavedEmployee.country);
168:                                   }
169:                              }
170:
171:                              catch (Exception e1 )
```

```
172:                        {
173:
    JOptionPane.showMessageDialog(c,"Wrong File Format");
174:                            System.out.println("Wrong      File
    Format");
175:                        }
176:                    }
177:
178:                    else if ( obj == buttonClear )
179:                    {
180:                        textFirstName.setText("");
181:                        textLastName.setText("");
182:                        textCountry.setText("");
183:                    }
184:                }
185:            }
186:
187:            public static void main ( String[]args )
188:            {
189:                new Input();
190:            }
191:        }
192:
193:        class Data implements Serializable
194:        {
195:            String firstName;
196:            String lastName;
197:                String country;
198:        }
```

ثالثاً: شرح الكود:

- في السطور من 1 إلى 4 يتم تضمين الحزم Packages اللازمة لعمل البرنامج.
- في السطور من 8 إلى 22 يتم تعريف المتغيرات variables التى سنستخدمها لإنشاء الواجهة الرسومية GUI.

- في السطور من 26 إلى 75 يتم إنشاء الواجهة الرسومية GUI كما فعلنا من قبل ولا جديد هنا.
- في السطر رقم 78 يتم تعريف مستمع حدث الضغط على الزر Event Listener.
- في السطور من 79 إلى 81 يتم إضافة blisten كمستمع لحدث الضغط على الثلاثة أزرار.
- في السطور من 84 إلى 109 يتم إضافة مستمع لحدث إغلاق النافذة.
- في السطر رقم 111 يتم تحديد عنوان النافذة باسم Serialization.
- في السطر رقم 112 يتم تحديد حجم النافذة 200*400.
- في السطر رقم 113 يتم إظهار النافذة.
- في السطر رقم 116 يتم تعريف مستمع حدث الضغط على الزر والمسمى ButtonListener.
- في السطر رقم 122 يتم اختبار مصدر الحدث Event Source ، فإذا كان مصدر الحدث Event Source هو الزر Save ، فإننا ننشيء هدفاً object من نوع Data واسمه Employee كما هو واضح في السطر رقم 123.
- الفصيلة Data موجود تعريفها في السطر رقم 193 وهي تحتوى على 3 متغيرات يمثلون الاسم الأول ولقب العائلة والدولة.
- حيث أننا نريد أن نجعل حفظ هذه البيانات ممكناً ، فإن الفصيلة Data لا بد أن تنفذ الـ interface المسمي Serializable لتمكين عملية الحفظ.
- في السطور من 125 إلى 127 يتم الحصول على البيانات من المستخدم التى سيدخلها فى أداة النص JTextField الأولي والثانية والثالثة عن طريق الدالة getText() وجعلنا كل قيمة منهم هي أحد المتغيرات فى الهدف object المسمى Employee.
- في السطر رقم 131 يتم إظهار مربع حفظ ملف.
- في السطر رقم 133 يتم اختبار الزر الذى ضغطت عليه ، فإذا كان Save فإننا لا بد أن نعرف مسار الملف وهذا ما يفعله السطر رقم 135.
- في السطور من 136 إلى 139 يتم تخزين البيانات فى الملف الذى حددته فى الخطوة السابقة وقد شرحنا هذه السطور من قبل.

- بعد عملية الحفظ نريد أن نظهر رسالة تفيد نجاح عملية الحفظ وهذه هى وظيفة السطر رقم 141.
- السطور من 131 إلى 141 من الممكن أن تسبب استثناءاً Exception ولذلك وضعناها فى جملة try و catch ، وهنا الاستثناء Exception المكن حدوثه يسمى Exception.
- في السطر رقم 151 يتم اختبار مصدر الحدث Event Source ، فإذا كان هو الزر Load ، فإننا نظهر مربع فتح ملف وهذا ما يفعله السطر رقم 155.
- في السطر رقم 157 يتم اختبار الزر الذى ضغطت عليه ، فإذا كان Open فإننا لا بد أن نعرف مسار الملف المختار وهذا ما يفعله السطر رقم 159.
- في السطور من 160 إلى 163 تتم عملية قراءة بيانات الهدف object من الملف المختار. لاحظ أننا فى السطر رقم 162 أنشأنا هدفاً object جديداً من نوع الفصيلة Data بحيث أن بياناته يتم قراءتها من البيانات المحفوظة فى الملف. لاحظ عملية التحويل Casting أيضاً كما ذكرنا من قبل.
- في السطور من 165 إلى 167 يتم وضع بيانات الهدف object الجديد المسمى SavedEmployee فى الثلاثة أدوات النص JTextField الموجودين فى الواجهة الرسومية عن طريق الدالة setText().
- هنا قد يخطىء المستخدم ويطلب تحميل البيانات من ملف أخر لا يحمل بيانات المستخدم كما نريدها ، ولذلك قد يحدث استثناء Exception نتيجة لاختلاف تنسيق الملف File Format واختلاف بياناته عن التى نريدها ، ولذلك نظهر رسالة تفيد أن الملف غير صالح للقراءة منه كما فى السطر رقم 173.
- في السطور من 180 إلى 182 يتم إلغاء البيانات فى حالة الضغط على الزر Clear.

تشغيل البرنامج:

- يظهر لنا البرنامج بواجهته الرسومية التى أردناها.
- الآن قم بإدخال بيانات الاسم الأول واسم العائلة والدولة مثل Maged, Mostafa, Egypt.

- الآن اضغط على الزر Save ليظهر مربع حفظ ملف. فى الجزء الخاص باسم الملف اكتب مثلاً Employee1 ثم اضغط على الزر Save لتظهر لك رسالة تفيد إتمام عملية الحفظ. الآن اضغط على Ok للرجوع إلى نافذة البرنامج الأصلية.
- من على شاشة سطح المكتب افتح My Computer ثم اضغط على :C وهو المسار الذى حددناه لحفظ الملف ، وستجد ملفاً اسمه employee1 يمثل البيانات التى حفظناها.
- إذا حاولت فتح الملف employee1 باستخدام برنامج المفكرة Notepad فلن تجد البيانات بالضبط وستجد بعض الزيادات نتيجة لأننا حولنا البيانات إلى بايتByte للتخزين وهذا قام بالتغيير في شكل البيانات قليلاً.
- إذا ضغطت على الزر Clear فذلك يمسح البيانات التى أدخلتها كما أردنا.
- قم بإغلاق البرنامج لتظهر لك رسالة للتأكيد على إغلاق البرنامج ، اضغط على yes, please.
- الآن سيتضح لنا أنه مع أننا أغلقنا البرنامج فإن البيانات التى حفظناها ستستطيع استرجاعها كالآتى:
- نفذ البرنامج مرة أخرى لتظهر لك الواجهة الرسومية.
- الآن اضغط على الزر Load.
- اذهب إلى الملف C:\employee1 ثم اضغط على الزر Open.
- ستجد البيانات التى حفظناها موجودة الآن فى أدوات النص JTextField الأولي والثانية والثالثة أى أنها لم تفقد.
- الآن اضغط على الزر Load وحاول فتح أى ملف أخر. حيث أن هذا الملف الأخر لا يحتوى على البيانات الخاصة ببرنامجنا ، فذلك يسبب استثناءاً Exception ، ولذلك سوف تظهر رسالة تبين حدوث خطأ فى تنسيق الملف الذى اخترته.

المتغيرات العابرة Transient Variables:

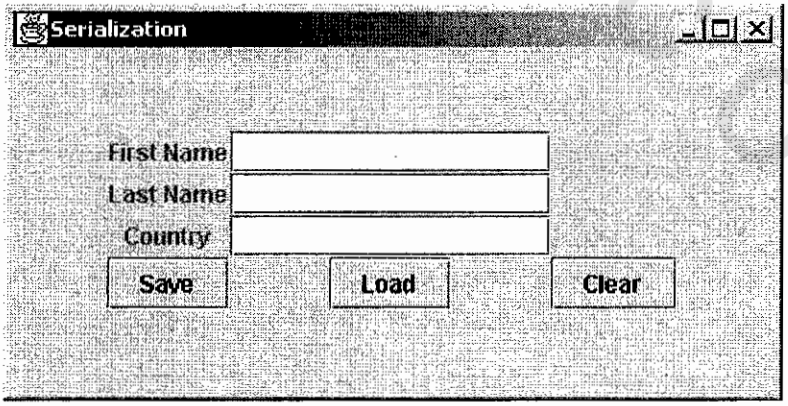
- لنفرض أننا نريد حفظ الاسم الأول واسم العائلة فقط ولا نهتم بحفظ الدولة ، المشكلة أن الدولة لا بد أن تكون من ضمن بيانات ومتغيرات الفصيلة class المسماة Data.

- ولكن حيث أن الفصيلة Data تنفذ الـ interface المسمي Serializable ، فذلك يجعل كل بيانات الفصيلة class المسماة Data من الممكن تخزينها.
- فإذا أردت أن تجعل أحد المتغيرات غير قابل للحفظ فقم بوضع كلمة transient قبل تعريفه.
- أى قم بتعديل السطر رقم 197 ليصبح

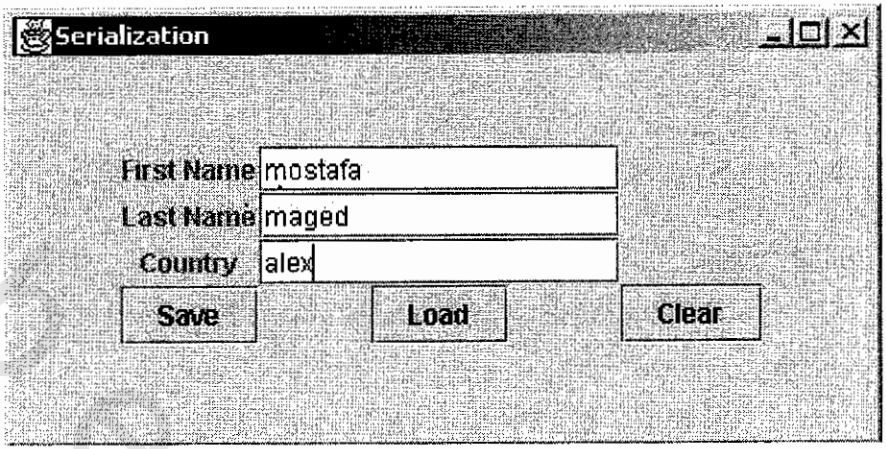
```
transient String country;
```

- الآن نفذ البرنامج واكتب البيانات Mostafa و Maged و Egypt مرة أخرى.
- اضغط على الزر Save ثم احفظ الملف باسم t لتظهر الرسالة التى تفيد عملية الحفظ وعندها اضغط على الزر Ok.
- اضغط على الزر Clear لمسح البيانات.
- اضغط الآن على الزر Load واختر الملف t ، الآن ستجد أن ما تم حفظه هو Mostafa وMaged فقط أما اسم الدولة فلم يتم حفظه لأنه أصبح transient.

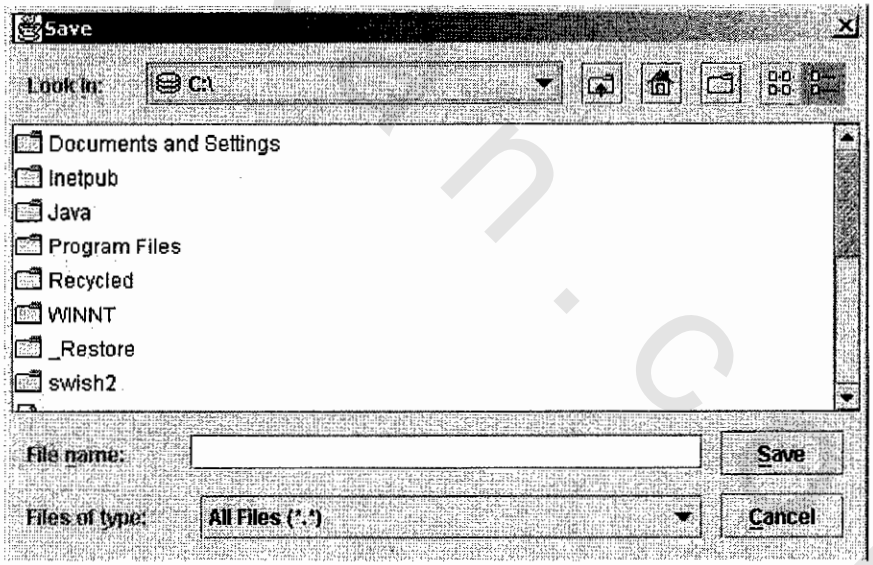
إذا كان أحد المتغيرات فى الفصيلة class له المعدل static فلن يتم حفظه حتى لو لم يتم تعريفه كأنه transient ، ويمكنك التأكد من ذلك بنفسك وسنترك هذا التمرين لك لكى تتمرن على النشر التسلسلي Serialization.



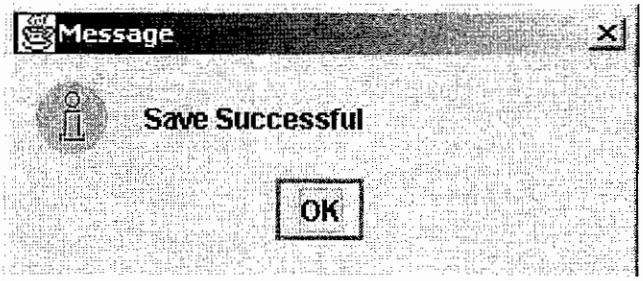
(الشكل 13-17) النشر التسلسلي Serialization



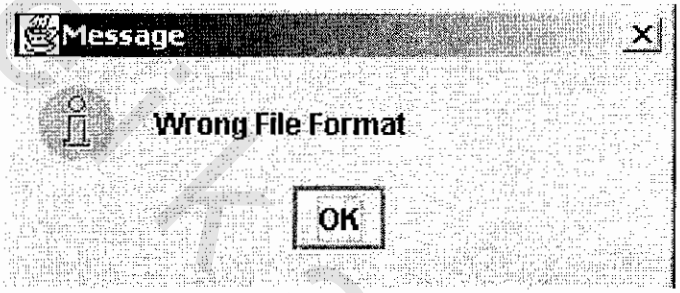
(الشكل 13-18) النشر التسلسلي Serialization



(الشكل 13-19) النشر التسلسلي Serialization



(الشكل 13-20) النشر التسلسلي Serialization



(الشكل 13-21) النشر التسلسلي Serialization

ملخص الفصل:

- تعلمنا في هذا الفصل العديد من عمليات الملفات وتعرفنا علي أهم موضوعات التدفق Streams.
- في الفصل القادم نتعلم - بإذن الله- كيفية استخدام العديد من فصائل المجموعات Collections حيث نتعرف علي القائمة List والمجموعة Set والخريطة Map ، فتابع معنا الفصل القادم.