الفصل السابع عشر

التعامل مع النصوص والأعداد والملفات
String,Number,Date,files

Programming Concepts فى هذا الفصل

17

فى هذا الفصل نشرح كيفية التعامل مع حافظه برنامج النوافذ Clipboard ذلك بالإضافه إليها والنسخ فيها من خلال النقاط التالية :

- التعامل مع النصوص String
- التعامل مع الأعداد Numbers
- التعامل مع التواريخ Dates
- التعامل مع الملفات النصية

فى VB.NET,C#

التعامل مع النصوص String

الـ String هو أى عبارة حرفية توضع بين علامتى تنصيص " " ويتم الإعلان عنها والتعامل معها كما فى السطور التالية:

```
Dim s As  String ="Hi  From VB.Net"
```

أو الطريقة:

```
Dim s As String
S="Hi From VB.Net"
```

وتحتوى الفصيلة string على مجموعة من الأعضاء التى تسهل التعامل مع العبارات الحرفية كما فى الجدول التالى:

Chars	تعيد حرف من داخل العبارة الحرفية المحدد بالموضع داخل العبارة الحرفية.
Length	تعيد عدد حروف العبارة الحرفية.
Compare	تقارن متغيرين من نوع String.
CompareTo	تقارن المتغير الذى تستدعى معه مع متغير أخر.
Concat	لاضافة عبارة حرفية او محتوى متغير Stringالى عبارة حرفية أو متغير String.
Copy	تنسخ محتوى متغير String الى أخر.
EndsWith	تختبر نهاية متغير String هل ينتهى بكلمة محددة أم لا.
Equals	تحدد نتيجة مقارنة متغيرين String.
Format	تقوم بتحويل العبارة المكتوبة الى الصيغة التى ترغبها.
IndexOf	تحدد أول ظهور لحرف معين داخل عبارة حرفية.
Insert	تضيف حرف فى مكان محدد داخل عبارة حرفية.
LastIndexOf	تحدد أخر ظهور لحرف معين داخل عبارة حرفية

Remove	تحذف عدد محدد من الحروف حسب التحديد.
Replace	تستبدل حروف بحروف أخرى.
StartsWith	تختبر بداية عبارة حرفية.
Substring	تعيد عدد محدد من الحروف من عبارة حرفية حسب العدد وحسب البداية.
ToLower	تحول الحروف الكبيرة الى الحروف الصغيرة.
ToString	تحول قيمة معينة الى عبارة حرفية String حتى يمكن التعامل معها بالدوال الحرفية.
ToUpper	تحول الحروف الصغيرة الى الحروف الكبيرة.
Trim	تقوم بقص (حذف)حرف محدد من بداية ونهاية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.
TrimEnd	تقوم بقص (حذف)حرف محدد من نهاية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.
TrimStart	تقوم بقص (حذف)حرف محدد من بداية عبارة حرفية ويستخدم بكثرة لحذف المسافة Space.

التعامل مع الأعداد Numbers

الإعداد Numbers سواء الصحيحة Integers أو الحقيقية Float تختلف عن العبارة الحرفية String فى إمكانية إجراء العمليات الحسابية عليها مثل الجمع والطرح.

التعامل مع التواريخ Dates

هى القيم التى تأخذ شكل وقيمة تاريخ Date أو وقت Time وتوجد بعض الدوال التى توفر معظم العمليات المطلوبة مع قيم التاريخ والوقت مثل الدوال التالية:

Date, Day, DayOfWeek, DayOfYear, Hour, Millisecond, Minute, Month, Now, Second, Ticks, TimeOfDay, Today, UtcNow, Year, Add,AddDays, AddHours, AddMilliseconds, AddMinutes, AddMonths, AddSeconds, AddTicks, AddYears, Compare, CompareTo, DaysInMonth, Equals, FromFileTime, FromOADate, GetDateTimeFormats

فى لغة Java

الملفات النصية

توجد أكثر من طريقة للتعامل مع الملفات ، وتستخدم كل طريقة مع الحالة التى تناسبها وهذه الطرق هى :

1- التعامل مع ملفات قواعد البيانات : وتم شرحها فى الفصول السابقة.

2- فتح الملفات بطرق التسلسلية : ويتم استعمال عند الرغبة فى التعامل مع الملفات النصية سواء بالكتابة أو بالقراءة مثل الملف AUTOEXEC.BAT.

3- التعامل مع الملفات العشوائية : وهو حفظ البيانات فى ملف بالصورة الثنائية (Binary) وذلك حسب التطبيقات التى تتطلب ذلك.

وتوجد مجموعة من الفصائل classes التى توفر عمليات التعامل مع الملفات منها.

BinaryReader, BinaryWriter, BufferedStream, Directory, DirectoryInfo DirectoryNotFoundException, EndOfStreamException, ErrorEventArgs, File FileInfo, FileLoadException, FileNotFoundException, FileStream FileSystemEventArgs ,FileSystemInfo, FileSystemWatcher

وتأخذ عملية القراءة من الملفات مجموعة الخطوات التالية:

1- التحقق من وجود الملف المطلوب التعامل معه.

2- فتح الملف.

3- تعريف متغير هدف Object من الفصيلة BinaryReader أو الفصيلة StreamReader حسب الطريقة التى ترغب فى التعامل مع الملف.

4- استخدام دوال Methods لقراءة البيانات.

5- إغلاق الملف.

وتأخذ عملية الكتابة فى الملفات مجموعة الخطوات التالية:

افتح الملف المطلوب التعامل معه وإذا كان موجود تتأكد من المستخدم هل تريد استبداله والكتابة عليه أم لا تعريف متغير هدف Object من الفصيلة BinaryWriter أو الفصيلة StreamWriter حسب الطريقة التى ترغب فى التعامل مع الملف.

4-استخدام دوال Methods لكتابة البيانات فى الملف.

5- إغلاق الملف.

مع ملاحظة استعمال بلوك التعامل مع الأخطاء try..catch للتعامل مع أى أخطاء تظهر مثل عدم وجود الملف المطلوب القراءة منه أو عدم إمكانية إنشاء ملف جديد.

الفصيلة File

توفر هذه الفصيلة إمكانية تحقيق العمليات الأساسية مع الملفات وذلك من خلال مجموعة من الدوال Methods وأشهر هذه الدوال الدالة ()Open المسئولة عن فتح ملف.

AppendText,Copy,Create,CreateText,Delete,Exists,GetAttributes,GetCreationTime,GetLastAccessTime, GetLastWriteTime, Move, Open, OpenRead, OpenText, OpenWrite, SetAttributes, SetCreationTime, SetLastAccessTime, SetLastWriteTime

وتوفر الفصيلة File عمليات التعامل مع الملفات والفهارس مثل اختبار وجود الملف أو الفهرس وإيجاد معلومات عنه مثل الحجم وتاريخ الإنشاء.

التدفق Stream

فى لغة Java

جميع البرامج تتعامل مع المستخدم وتطلب منه بعض البيانات والمعلومات ثم تتعامل مع هذه البيانات ثم يتم عرض النتائج، 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.

هو فيض من المعلومات والبيانات من مصدر معين Source إلى جهة وصول Destination.

فمثلاً:

- إذا أدخلت بيانات إلى البرنامج فإن النتائج تظهر على الشاشة إذن هنا مصدر المعلومات هو لوحة المفاتيح وجهة الوصول هى الشاشة.
- إذا طبعت ملف فإن النتائج تظهر على آلة الطباعة إذن هنا مصدر المعلومات هو الملف الذى طبعته وجهة الوصول هى آلة الطباعة.
- إذا قمت بكتابة ملف قمت بحفظة فإنك تحفظه على القرص الصلب (Hard Disk) إذن هنا مصدر المعلومات هو الملف وجهة الوصول هى القرص الصلب.

- إذن هناك أكثر من مثال للتدفق Streams فهو ببساطة مجموعة بيانات تنتقل من مصدر إلى جهة وصول بصرف النظر عن نوع البيانات المنتقلة.
- قد تظن أنه لكي تستطيع التعامل مع أنواع التدفق Streams المختلفة، فإنك لابد أن تعرف عدة طرق في البرمجة بحيث تستطيع التعامل مع كل نوع من أنواع التدفق Streams المختلفة. ولكن هذا ليس صحيحاً، فكما سنرى توفر لغة الجافا Java عدة فصائل classes للتعامل مع أنواع التدفق Streams المختلفة بمنتهى السهولة.
- لاحظ أنك إذا كنت تكتب بيانات فإنك في هذه الحالة تكون المصدر Source اما إذا كنت تقرأ البيانات أو قبض المعلومات فإنك تمثل جهة الوصول Destination لهذه البيانات.
- إذا نظرنا نظرة عميقة للتدفق Streams فإننا سنجد أنها في الواقع أربعة أنواع رئيسية: InputStream, OutputStream, Reader, and Writer.

1. InputStream

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

2. OutputStream

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

3. Reader

وهو يماثل الـ InputStream

4. Writer

وهو يماثل الـ OutputStream

إذن هنا نتساءل : ما الفرق بين هذه الفصائل classes؟

الفرق أن الفصائل classes المسماة Reader و Writer تدعم عملية تسمى Internationalization.

Internationalization

هى عملية تصميم البرنامج بحيث يمكنه أن يعمل مع أكثر من لغة بدون أى تغيير في البرمجة بمعنى:

انظر مثلاً لنظام تشغيل النوافذ Windows فإنه 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 الـ Internationalizatio 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 فمثلاً إذا أنشأت موقعك على شبكة الإنترنت باللغة العربية مثلاً وحاول أحد مستخدمي الشبكة الدخول على موقعك ونظام التشغيل على جهازه لا يدعم اللغة العربية ، فلن يستطيع قراءة اللغة العربية وبذلك فلن يمكنه معرفة أي معلومات عن موقعك.

تعرف عملية الـ Internationalization بعملية I18n لأنه يوجد 18 حرف يقعون بين حرفي I و n.

أشهر فصائل classes التدفق Streams

System.in

هو InputStream يكون مصدره لوحة المفاتيح

System.out

هو OutputStream تكون جهة وصوله هى الشاشة

تذكر أننا استخدمنا دالة println() الموجودة فى الفصيلة System.out لطباعة رسالة على الشاشة. إذن فالفصيلة class المسماة System.out هي المسئوله عن الطباعة على الشاشة.

سوف نرى بعض 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.
سوف ترى فى المثال القادم كيفية التعامل مع الملفات.

مثال 1: التعامل مع الملفات

أولاً: هدف المثال

تريد أن تنشئ تطبيقاً يستطيع الحصول على بعض البيانات من ملف نصي اسمه **Data.txt** فلابد أن تتأكد أولاً من وجود هذا الملف وإذا كان موجوداً فإنك تريد أن تعرف بعض المعلومات عن هذا الملف باستخدام الدوال الآتية:

وظيفة الدالة	نوع القيمة المرتجعة	اسم الدالة
تحدد ما إذا كان الملف موجوداً أم لا	boolean	exists ()
تحدد ما إذا كان باستطاعتك القراءة من الملف أم لا	boolean	canRead ()
تحدد ما إذا كان باستطاعتك الكتابة إلى الملف أم لا	boolean	canWrite ()
تحدد المسار الكامل للملف	String	getAbsolutePath ()
تقوم بإنشاء مجلد (directory)	boolean	mkdir ()
تقوم بتحديد اسم الملف	String	getName ()
تقوم بتحديد ما إذا كان الملف مختفياً (Hidden) أم لا.	Boolean	isHidden ()
تقوم بتحديد ما إذا الـ object المنشأ من نوع File، مجلداً أم لا.	boolean	isDirectory ()
تقوم بتحديد ما إذا كان الـ object المنشأ من نوع File ملفاً أم لا	boolean	isFile()
تقوم بإرجاع حجم الملف بالبايت	long	length ()

اسم الدالة	نوع القيمة المرتجعة	وظيفة الدالة
setReadOnly ()	boolean	تقوم بجعل الملف للقراءة فقط (Read –Only)

يقوم هذا البرنامج ببيان استخدام هذه الدوال.

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:
25:       catch (IOException e)
26:       {}
27:
28:       //Now the file exists on your hard
29:       System.out.println("---------------------------");
30:       System.out.println("The file exists on your hard");
31:       System.out.println("---------------------------");
32:       System.out.println("Exists: " + myFile.exists());
33:       System.out.println("Can Read: " + myFile.canRead());
34:       System.out.println("Can Write: " + myFile.canWrite());
35:       System.out.println("Absolute Path: " + myFile.getAbsolutePath());
36:       System.out.println("File Name: " + myFile.getName());
```

```
37:     System.out.println("Hidden: " + myFile.isHidden());
38:     System.out.println("Directory: " + myFile.isDirectory());
39:     System.out.println("File: " + myFile.isFile());
40:     System.out.println("Length: " + myFile.length());
41:     System.out.println("Set Read Only: " + myFile.setReadOnly());
42: }
43:}
```

ثالثاً: شرح الكود:

- في السطر 1 يتم تضمين الحزمة package المسماة (java.io.*) لانها تتضمن فصائل التدفق STREAMS اللازمة لعمل البرنامج.
- في السطر 7 يتم إنشاء ملف اسمه data.txt
- لاحظ أنك أنشأت هدف object من نوع الفصيلة class المسماة File ومررت لدالة البناء constrcutor القيمة data.txt التي تمثل اسم الملف الذي نريد التعامل معه.
- لاحظ أيضاً أنه كونك أنشأت هذا الهدف object فإن ذلك لا يعني أن الملف اصبح موجوداً على القرص الصلب (Hard Disk) وستعرف حالاً كيف تستطيع إنشاءه على القرص الصلب.
- في السطور من 12 إلى 16 يتم استخدام الدوال الموجودة في الجدول السابق.
- لاحظ أن الملف data.txt غير موجود في نفس مسار البرنامج لأننا لم ننشئه بعد إذن فالملف غير موجود وبالطبع لا يمكنك القراءة منه أو الكتابة إليه لأنه غير موجود.
- في السطر 22 يتم إنشاء الملف فعلياً على القرص الصلب عن طريق الدالة createNewFile() .
- لاحظ أنك في هذه الحالة أنشأت الملف في نفس مسار البرنامج.
- فمثلاً نفترض أن الملف Files.java موجود في المسار E:\ على القرص الصلب (Hard Disk) إذن بعد تنفيذ السطر 22 فإنك ستجد الملف Data.txt موجوداً في نفس المسار وهو E:\.
- تم تنفيذ السطر 22 داخل try بسبب إمكانية حدوث استثناء Exception فقد لا تستطيع عمل ملف بسبب مثلاً عدم وجود مساحة وهنا نوع الاستثناء Exception هو IOException.
- في السطور من 29 إلى 41 يتم بيان استخدام باقي الدوال الموجودة في الجدول السابق.

- لاحظ أنك أنشأت الآن الملف data.txt فعلياً على القرص الصلب إذن فهو موجود وتستطيع الكتابة إليه والقراءة منه وتستطيع معرفة إذا كان مخفياً (Hidden) أم لا.
- في السطر 38 يتم اختبار الهدف object الذى انشأناة باسم myFile. هل هو مجلد (Directory) أم لا. بالطبع فنحن أنشأنا ملفاً عادياً وسنرى فيما بعد كيف ننشئ مجلداً.
- في السطر 39 يتم اختبار myFile هل هو ملف عادى أم لا وبالطبع الإجابة هى نعم أو true.
- في السطر 40 يتم معرفة حجم الملف عن طريق الدالة () length. انظر شكل (15-1).
- في السطر 41 يتم جعل الملف Read – only اى للقراءة فقط أى أنك إذا ذهبت للمسار E:\ الموجود فيه الملف النصي data.txt وقمت بالضغط بالزر الأيمن للفأرة على الملف Data.txt ثم اخترت Properties فإنك ستجد أن الملف Read only
- 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 تخليق الملف فى سطر 22 ستعيد القيمة false لأنك لن تستطيع تخليق ملف موجود أصلاً.

رابعاً: نتيجة التنفيذ:

بعد تنفيذ البرنامج فستظهر النتيجة على الشاشة كما في الشكل (1-17)

الشكل (1-17)

الشكل (17-1)

Directories : (المجلدات)

كما تعاملنا مع الملفات فإننا نريد أن نتعامل مع المجلدات Directories فنريد أن نعرف كيف ننشئ مجلداً وكيف نعرض أسماء المجلدات والملفات الموجودة فى مسار معين أى مثل أمر dir فى الـ Dos.

قراءة بايت Byte من الفصيلة System.in

ما نريد أن نفعله الآن هو أن نطلب من المستخدم إدخال حرف فإذا أدخل حرف X فذلك يعنى أنه يريد إنهاء البرنامج والخروج منه (Exit).

مثال 3: قراءة بايت Byte من الفصيلة System.in

أولاً: هدف المثال:

نريد توضيح كيفية قراءة بايت Byte من الفصيلة System.in أي أننا سنقرأ البايت المدخلة من لوحة المفاتيح.

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 8 يتم إنشاء هدف object من نوع الفصيلة InputStream واسمه is ثم جعلنا is هو System.in.
- لاحظ أننا نريد أن نقرأ بايت من لوحة المفاتيح وكما ذكرنا من قبل فإن System.in هو عبارة عن InputStream ومصدرة هو لوحة المفاتيح .
- في السطر 10 يتم طباعة رسالة للمستخدم تطالب منه إدخال حرف .
- في السطر 16 يتم استخدام الدالة ()read لقراءة البايت Byte من لوحة المفاتيح.
- لاحظ أن الدالة ()read لها قيمة مرتجعه من نوع int ولذلك عرفنا متغير من نوع int اسمه i في سطر 7 وجعلنا له قيمة ابتدائية تساوى صفر وكما تعلم فإنك لابد أن تعطى قيمة ابتدائية لأى متغير في لغة الجافا Java.

- لاحظ أيضاً أن الدالة ()read من الممكن أن تسبب استثناء Exception من نوع IOException ولذلك استخدمنا try و catch لمعالجة هذا الاستثناء.
- في السطر 12 يتم استخدام الدوارة while لأننا نريد أن نقرأ بايت طالما أن الحرف المدخل ليس x.
- لاحظ أن البايت الذي سيدخلة المستخدم تم حفظة في المتغير i ولكن i من نوع int كما ذكرنا. إذن فإننا نريد أن نقارن i بالحرف x وذلك مستحيل لاختلاف نوعي المتغيرين ولذلك استخدمنا كلمة char لعمل تحويل Casting للمتغير i ليعامل كحرف وبذلك تصبح المقارنة ممكنة. انظر الشكل (2-17).

رابعاً: نتيجة التنفيذ:

بعد تنفيذ البرنامج فستظهر النتيجة على الشاشة كما في الشكل (2-17).

الشكل (2-17)

Wrapper Streams

- لاحظنا في المثال السابق أننا كنا نقرأ بايت Byte. وهذا نادراً ما يحدث فالطبيعي أنك تطلب إدخال عدة سطور من الأحرف.
- إذن الـ Wrapper class هي الفصيلة class التي تضيف بعض الوظائف إلى الفصائل classes الخاصة بالتدفق Streams.
- إذن فإننا سنستخدم فصيلة class تسمى BufferedReader والتي تحتوى على الدالة

readLine() التي تسمح لنا بقراءة عدة سطور. كما أنها تتميز بالقدرة على تخزين الحروف المدخلة بحيث يمكن القراءة منها مباشرة بدلاً من القراءة من نظام التشغيل وهذا يعطي سرعة أكبر.

- سوف نرى استخدام الفصيلة BufferedReader لاحقاً.

عمليات الملفات File Operations

عمليات القراءة والكتابة في الملفات تعتبر من أهم العمليات في البرمجة ولذلك توفر لغة الجافا Java اثنين من الفصائل classes لعملية القراءة والكتابة للملفات وهما:

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 تريد فصيلة class لقراءة سطر من الملف وهذه الفصيلة class هي BufferedReader.
- ايضاً توجد نفس المشكلة في الكتابة إلى ملف فلا يمكنك استخدام الفصيلة FileWriter مباشرة لانك ستكتب سطر في الملف ولذلك نستخدم فصيلة class لكتابة سطر إلى ملف وهذه الفصيلة class هي PrintWriter.
- قد تبدو هذه السطور غامضة ولكننا سنعطي مثال صغير لبيان كيفية القراءة والكتابة بالنسبة للملفات.

أولاً: القراءة من ملف

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

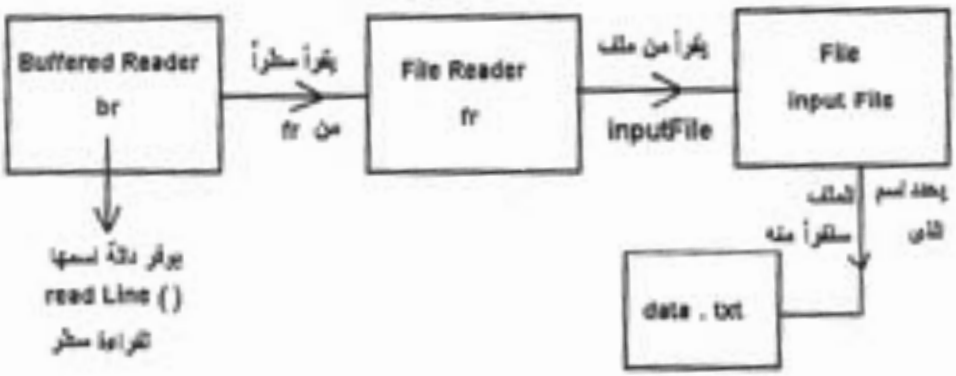
- في السطر الأول أنشأنا هدف object من نوع الفصيلة class المسماة File. إذن هنا أنشأنا ملفاً جديداً اسمه data.txt.
- حيث أنك تريد أن تقرأ من هذا الملف، فإنك تستخدم هدف object من نوع الفصيلة class المسماة FileReader. دالة البناء constructor لهذه الفصيلة class تستقبل هدف

object من نوع الفصيلة File. إذن هنا حددنا أننا سنقرأ من inputFile الذى بدورة هو عبارة عن الملف data.txt.

- كما ذكرنا لا يمكن استخدام FileReader مباشرة للقراءة من الملف وذلك لأنك تقرأ سطر من الملف وليس الملف كاملاً ولذلك استخدمنا هدف object من نوع الفصيلة class المسماة BufferedReader. ودالة البناء constructor فهذه الفصيلة class تستقبل هدف object من نوع FileReader لتحديد الملف الذى ستقرأ منه.
- كملخص: فإن السطر الثالث ينشئ br الذى يقرأ سطراً وسوف يقرأ هنا السطر من fr.
- أيضاً fr هو الذي يستخدم للقراءة من الملف وهذا الملف هو inputFile.
- أيضاً inputFile هو عبارة عن الملف الفعلى الذى ستقرأ منه وهو الذى اسمه data.txt.
- بعد هذه الثلاثة سطور فيمكننا القراءة مباشرة من br وكما ذكرنا فإن BufferedReader يوفر لنا دالة اسمها readLine() للقراءة منه. أى انه للقراءة من br فإننا نكتب. String

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

- الرسم التالى يوضح هذه العملية.



سوف ترى فى المثال القادم كيف يتم ذلك.

ثانياً: الكتابة إلى ملف:

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

- بالطبع يوجد تشابة شديد بين الكتابة والقراءة ولكننا سنعيد شرحهم لمزيد من الفهم.
- فى السطر الأول أنشأنا هدف 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 هدف object من نوع الفصيلة class المسماة FileWriter. دالة البناء constructor لهذه الفصيلة class تستقبل هدف object من نوع الفصيلة File. إذن هنا حددنا أننا ستكتب فى outputFile الذى بدورة هو عبارة عن الملف output.txt.
- كما ذكرنا لا يمكن استخدام FileWriter مباشرة للكتابة فى الملف وذلك لأنك ستكتب فيه سطر بسطر وليس السطور كاملة ولذلك استخدمنا هدف object من نوع الفصيلة class المسماة PrintWriter. ودالة البناء constructor هذه الفصيلة class تستقبل هدف object من نوع الفصيلة FileWriter لتحديد الملف الذى ستكتب فيه.
- كملخص: فإن السطر الثالث ينشىء pw الذى يكتب سطراً وسوف يكتب هذا السطر فى fw. أيضاً fw هو الذى يستخدم للكتابة فى الملف وهذا الملف هو outputFile أيضاً outputFile هو عبارة عن الملف الفعلى الذى سنكتب فيه وهو الذى اسمة output.txt .
- بعد هذه الثلاثة سطور فيمكننا الكتابة مباشرة فى pw وكما نعرف أنه يوجد الدالة المسماة println() التى تستخدم للكتابة. أى أنه للكتابة فى pw فإننا نكتب

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

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