آموزش نوشتن در یک فایل

نوشتن در یک فایل کمی آسانتر از خواندن یک فایل می باشد. برای نوشتن یک فایل از چند گروه داخلی استفاده خواهیم کرد: گروه FileWriter و .PrintWriter

با کلیک کردن بر روی File > New File از منوی NetBeans یک گروه جدید در پروژه ی خود ایجاد کنید. در بخش Categoriesاز دیالوگ باکس Java و Class را از لیست File Types انتخاب کنید. روی دکمه ی Next در پایین کلیک کنید .برای نام گروه WriteFile را تایپ کرده و سپس روی Finish کلیک کنید. سه عبارت زیر را به کد خود وارد کنید:

```
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;
```

گروه جدید شما باید مشابه زیر باشد:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {


}
```

مجددا عبارت هایی که زیر آنها خط کشیده شده وجود دارند، زیرا هنوز از گروه وارد شده استفاده نکرده ایم.

وقتی در یک فایل می نویسید، یا می توانید از ابتدا آغاز کرده و هر موردی را بنویسید. یا می توانید از انتها آغاز کرده و به فایل ضمیمه کنید. گروه FileWriterبه شما اجازه می دهد تا روش آن را تعیین کنید. یک فیلد اضافه خواهیم کرد که مقدار ضمیمه را برای گروه FileWriter تنظیم می کند. ما یک فیلد هم برای تنظیم نام فایل اضافه خواهیم کرد.

بنابراین دو فیلد زیر را به اضافه ی یکconstructor ، به کد خود اضافه کنید:

```
public class WriteFile {

    private String path;
    private boolean append_to_file = false;

    public WriteFile(String file_path) {
        path = file_path;
    }
}
```

فیلد Boolean در واقع append_to_file نامیده می شود و روی مقدار falseتنظیم کرده است. این یک مقدار پیش فرض برای گروه FileWriter می باشد و به این معناست که ضمیمه نمی خواهید، اما هر چیزی را در فایل حذف می کند.

سازنده (constructor) مقداری را برای فیلد path تنظیم می کند که نام و موقعیت فایل می باشد. این برنامه یک آبجکت جدید از گروه WriteFile را توزیع خواهد کرد.

به هرحال همانطور که در بخش قبل ذکر شد، می توانید بیشتر از یک سازنده در کد خود تنظیم کنید. می توانیم سازنده ی دوم را در تنظیم کرده و آن را در یک مقدار ضمیمه انتقال دهیم. به این روش یک یوزر یا می تواند از اولین سازنده استفاده کند و یا اینکه نام یک فایل و یا نام یک فایل و یک مقدار ضمیمه را توزیع کند. بنابراین مورد زیر را در زیر اولین سازنده اضافه کنید:

مثال :

```
public WriteFile( String file_path , boolean append_value ) {

path = file_path;

append_to_file = append_value;

}
```

اکنون سازنده ی دوم دارای دو مقدار در بین پرانتزها می باشد: یک مسیر فایل و یک مقدار ضمیمه. اگر می خواهید به این فایل ضمیمه کنید، می توانید در هنگام ایجاد یک آبجکت جدید از این سازنده استفاده کنید. اگر فقط می خواهید فایل متن را بنویسید، می توانید از اولین سازنده (constructor) استفاده کنید.

پنجره ی کد شما باید شبیه به تصویر زیر باشد:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {

 private String path;
 private boolean append_to_file = false;

 public WriteFile(String file_path) {
     path = file_path;
 }

 public WriteFile( String file_path, boolean append_value ){
     path = file_path;
     append_to_file = append_value;
 }

}
```

برای نوشتن روی فایل، متود زیر را در زیر سازنده های خود اضافه کنید:

```
public void writeToFile( String textLine ) throws IOException {

}
```

این متود نیازی به بازگشت یک مقدار ندارد، بنابراین آن را void می سازیم. در بین پرانتزهای مربوط به نام متود، یک متغیر String به نام textLineرا داریم. مشخص است که این متنی است که می خواهیم در یک فایل بنویسیم. مجددا واضح است که باید "throws IOException" را اضافه کنیم چرا که برای کنترل خطاهای مربوط به file-writing لازم است.

اولین چیزی که در متود لازم داریم یک آبجکت FileWriter می باشد FileWriter .مراقب باز کردن فایل درست و مرتب کردن متن به عنوان بایت می باشد. خط زیر را به متود writeToFile خود اضافه کنید:

```
FileWriter write = new FileWriter( path , append_to_file);
```

بنابراین ما یک آبجکت FileWriter جدید با نام write ایجاد می کنیم. بین پرانتزهای FileWriter نام و موقعیت فایل به علاوه ی مقدار ضمیمه را انتقال می دهیم. این مقدارtrue (ضمیمه شده به یک فایل) یا) false ضمیمه نشده (خواهد بود. اگر فایلی با نامی که انتقال داده اید، وجود نداشته باشد، FileWriterبرای شما ایجاد می کند.

به هرحال FileWriter بایت ها را می نویسد. اما می توانیم به کمک گروه PrintWriterیک متن ساده به FileWriter ارائه دهیم PrintWriter .چند متود ساده برای این کار در دست دارد. اما در هنگام ایجاد آبجکت از گروه نیاز به نام یک FileWriterدارد. بنابراین خط زیر را به متود خود اضافه کنید:

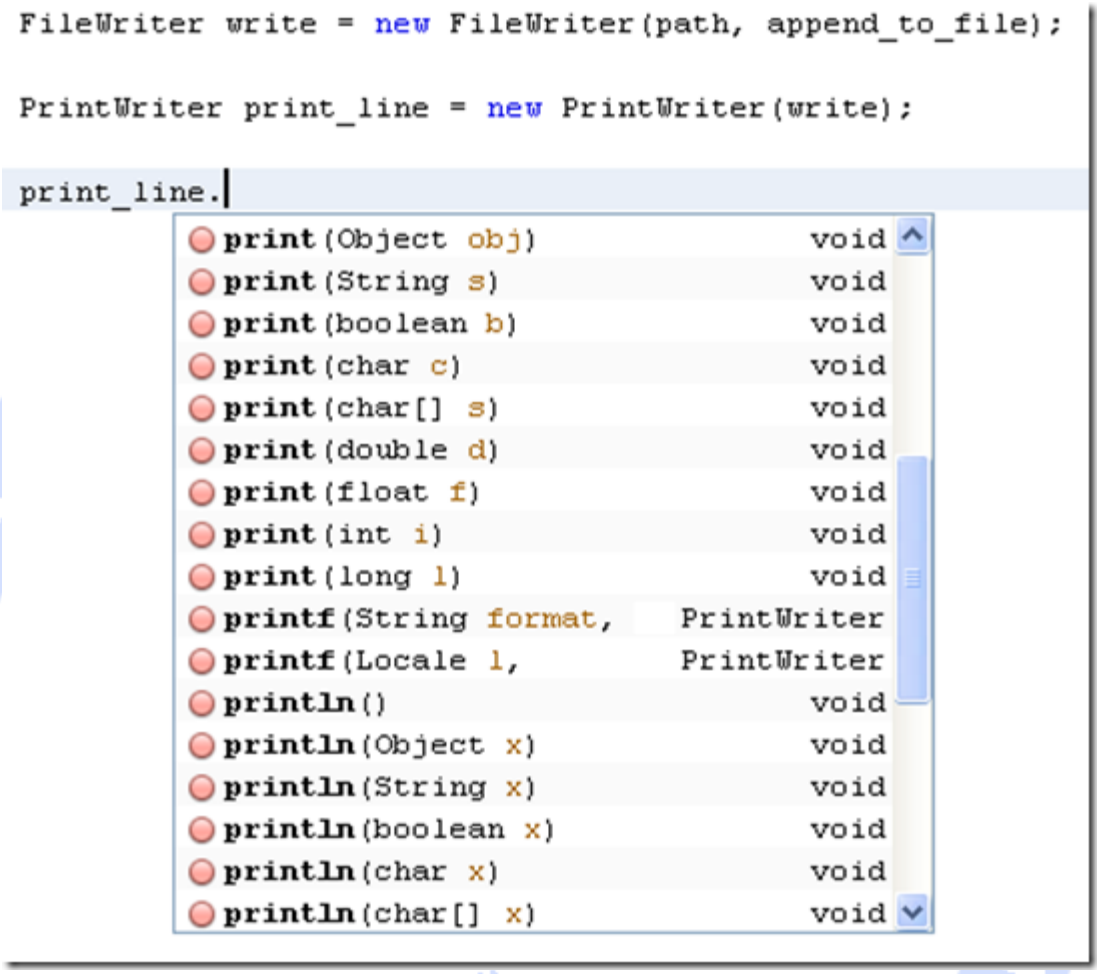
```
PrintWriter print_line = new PrintWriter( write );
```

آبجکت PrintWriter ما در واقع print_line نامیده می شود. بین پرانتزهایPrintWriter ، نام آبجکت FileWriter خود را اضافه می کنیم.

برای افزودن متن به یک فایل، نام آبجکت PrintWriter را تایپ کرده که با یک نقطه (dot) دنبال می شود:

```
print_line.
```

به محض اینکه این نقطه را تایپ می کنید، NetBeansلیستس از گزینه های متغیر را نمایش خواهد داد:



گزینه های خیلی زیادی در لیست وجود دارند.

یکی از گزینه هایی که استفاده خواهیم کرد، یکی از متودهای printf می باشد. این متود به شما اجازه می دهد تا یک رشته ی قالب بندی شده از متن را به PrintWriter انتقال دهید. یک دلیل خوب برای استفاده ازprintf ، بررسی کاراکترهای خط جدید می باشد. کاراکترهای خط جدید بسته به سیستم عاملی که استفاده می کنید، متفاوت می باشند. ویندوز کاراکترهای \r\nرا برای یک خط جدید اضافه خواهد کرد. اما سیستم Unix فقط از \n استفاده می کند. استفاده از عملکرد printf کد گذاری صحیح را، بدون توجه به سکو، تضمین خواهد داد.

خط زیر را به کد خود اضافه کنید:

مثال :

```
print_line.printf( "%s" + "%n" , textLine);
```

به متود printf دو مورد را تحویل داده ایم: فرمت مربوط به متن، رشته ای که روی فایل بنویسیم. هر دوی این موارد با استفاده از یک ویرگول (comma) از یکدیگر جدا می شوند. به این دو دقت کنید:

```
"%s" + "%n"
```

%sبین علامت های نقل قول به معنای رشته ای از کاراکترها با هر طولی می باشد .%n نیز به معنای خط جدید (newline)می باشد. بنابراین ما به متود printfمی گوییم که یک رشته از کاراکترها را فرمت کرده و یک خط جدید در انتها اضافه کند. متن حقیقی که نیاز به فرمت شدن دارد بعد از ویرگول قرار می گیرد. متود printf بسیار مفید است و در بخش های بعدی به جزئیات بیشتری در مورد گزینه ها خواهیم پرداخت. اکنون اجازه بدهید ادامه دهیم.

تنها یک خط دیگر به متود خود اضافه کنید:

```
print_line.close();
```

این خط فایل متن را بسته و منابعی را که استفاده می کرد، آزاد می کند.

اکنون گروه WriteFile مانند زیر به نظر خواهد رسید:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {

  private String path;
  private boolean append_to_file = false;

  public WriteFile(String file_path) {
      path = file_path;
  }

  public WriteFile( String file_path, boolean append_value ) {
      path = file_path;
      append_to_file = append_value;
  }

public void writeToFile(String textLine) throws IOException {

    FileWriter write = new FileWriter(path, append_to_file);
    PrintWriter print_line = new PrintWriter(write);

    print_line.printf("%s" + "%n", textLine);

    print_line.close();

  }
}
```

برای اینکه گروه جدید خود را امتحان کنید) گروهی با متود اصلی (main به گروه FileData بازگردید. برای ایجاد یک آبجکت جدید از گروهWriteFile ، خط زیر را اضافه کنید:

```
WriteFile data = new WriteFile( file_name , true );
```

بنابراین یک آبجکت جدید WriteFile به نام data تنظیم کرده ایم. بین پرانتزهای WriteFile دو مورد اضافه کرده ایم: نام فایل و یک مقدار .true این امر فراخوانی سازنده های دوم (constructor) را تضمین خواهد کرد. اگر بخواهیم فقط فایل را بنویسیم، کد آن مانند زیر خواهد بود:

```
WriteFile data = new WriteFile( file_name );
```

از آنجایی که مقدار ضمیمه ی پیش فرض را با عنوان false تنظیم کرده ایم، اگر بخواهیم کل محتوا را بنویسیم، تنها به نام فایل احتیاج داریم.

برای فراخوانی متود writeToFile از آبجکتWriteFile ، خط زیر را اضافه کنید:

```
data.writeToFile( "This is another line of text" );
```

برای تغییر متن بین پرانتزهای مربوط به متود آزاد هستید.

برای اینکه به یوزر اجازه بدهید متوجه اتفاقی که افتاده شود، باید چیزی از پنجره ی Output چاپ کنید:

```
System.out.println( "Text File Written To" );
```

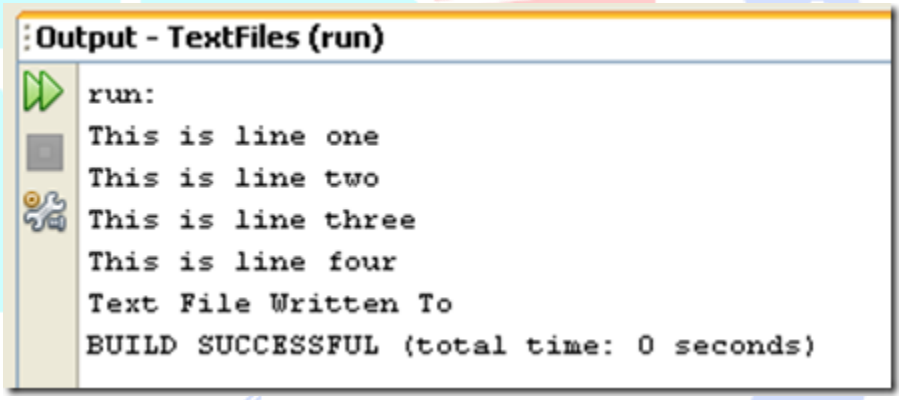
اکنون کد FileData باید مانند زیر باشد (چند کامنت اضافه کرده ایم.):

```
package textfiles;
import java.io.IOException;

public class FileData {

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

        String file_name = "C:/test.txt";
        try {
            ReadFile file = new ReadFile(file_name);
            String[] aryLines = file.OpenFile();
            int i;
            for (i=0; i < aryLines.length; i++) {
                System.out.println(aryLines[i]);
            }
        }
        catch (IOException e) {
              System.out.println( "Sorry, dude - no can do!" );
        }

        //============================
        //  WRITE TO A FILE
        //============================
        WriteFile data = new WriteFile(file_name, true);
        data.writeToFile("This is another line of text");

        System.out.println("Text File Written To");
    }
}
```

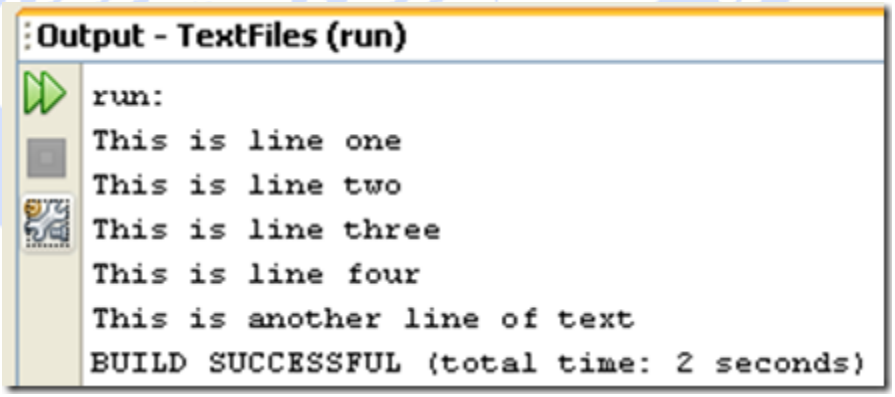
اگر تمایل داشته باشید، می توانید برای نوشتن متن یک بخش catch … try دیگر نیز اضافه کنید. به جای مورد بالا، آن را مانند زیر تغییر دهید:

```
//============================
//   WRITE TO A FILE
//============================
try {
   WriteFile data = new WriteFile(file_name, true);
   data.writeToFile("This is another line of text");
}
catch (IOException e) {
      System.out.println( e.getMessage() );
}
```

اکنون کد خود را اجرا کرده تا آن را امتحان کنید. شما باید به وسیله ی پیغامی که فایل متن نوشته، محتوای فایل متن خود را پنجره ی Output مشاهده کنید:



مجددا برنامه را اجرا کرده و یک خط جدید مشاهده خواهید کرد. (می توانید به کدی که در فایل متن نوشته می شود، کامنت بگذارید.)



و این تمام آن مورد می باشد- اکنون می توانید یک فایل متن نوشته و محتوای آن را بخوانید. در بخش بعد ادامه خواهیم داد و به برنامه نویسی با فرما های جاوا خواهیم پرداخت.

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد7

88146323 - 88446780 - 88146330

http://www.tahlildadeh.com/