Bereich: Input/Output*

JGrep

Package: `de.dhbwka.java.exercise.io`	**Klasse:** `Jgrep`

Aufgabenstellung:

Das UNIX-Shellkommando *grep* sucht Zeilen in (Text-)Dateien, die einem bestimmten Textmuster („Pattern“) entsprechen und gibt diese Zeilen auf der Konsole aus.

Entwickeln Sie ein Java-Programm *Jgrep*, welches die folgenden Eigenschaften erfüllt:

Das Programm *Jgrep* soll folgende Argumente akzeptieren:

-R	(optional)
-F	(optional)
<pattern>	(Pflicht)
< pfadname >	(Pflicht)
<ausgabedateiname>	(optional, aber nur zusammen mit „*-F*“)

<pattern> bezeichnet ein Textmuster, das aus beliebigen Buchstaben und Ziffern bestehen darf (Sonderzeichen, insb. das Leerzeichen, darf dieses Muster also nicht enthalten).
<pfadname> soll ein einfacher Datei- oder Verzeichnisname oder ein absoluter Pfad (zu einer Datei oder einem Verzeichnis) sein.

Handelt es sich bei *<pfadname>* um eine Datei, sollen alle Zeilen dieser Datei ausgegeben werden, in denen das Muster *<pattern>* enthalten ist. Dabei sollen vor dem Zeileninhalt jeweils der Dateiname und die Zeilennummer in dieser Datei (gezählt ab 1) ausgegeben werden.
Handelt es sich bei dem Parameter *<pfadname>* um ein Verzeichnis, sollen alle Dateien in diesem Verzeichnis durchsucht und die Trefferzeilen wie oben ausgegeben werden.

Der Parameter *-R* darf, muss aber nicht angegeben werden.
Ist er angegeben und bei *<pfadname>* handelt es sich um ein Verzeichnis, sollen auch die Dateien in allen Unterverzeichnissen (und in deren Unterverzeichnisse etc.) rekursiv behandelt werden.

Ist der ebenfalls optionale Parameter *-F* angegeben, muss als weiterer (letzter) Parameter *<ausgabedateiname>* der relative oder absolute Pfad einer Datei angegeben werden. In diesem Fall sollen alle Ausgaben statt auf die Konsole in diese Datei gemacht werden.

Fangen Sie alle Exceptions ab, die bei den Dateizugriffen auftreten können und ersetzen Sie diese durch eine eigene „sprechende“ `JgrepIOException`.

Im Falle von fehlerhaften Parametern (z.B. Anzahl der Parameter < 2, fehlender Parameter *<pfadname>* oder fehlender Paratemer *<ausgabedateiname>* bei „-F“) soll eine `JgrepIllegalParameterException` geworfen werden, welche mindestens den Standard-Konstruktor sowie einen Konstruktur zur Übergabe eine Strings – dieser soll zum Erzeugen einer „sprechenden“ Fehlermeldung verwendet werden – haben soll.

Hinweise:
Die Reihenfolge der Parameter ist wie oben angegeben festgelegt („-R“ ist optional, wenn vorhanden, steht es aber an erster Stelle. „-F“ steht vor <pattern>, <pattern> steht vor <pfadname> und <ausgabedateiname> steht ggf. nach <pfadname>).
Wir gehen davon aus, dass es sich bei allen Dateien um Textdateien (z.B. Java-Quellcode) handelt.
Die Reihenfolge der Dateien bei der Ausgabe spielt keine Rolle.
Beim Vergleich der Zeilen mit <pattern> soll Groß- bzw. Kleinschreibung berücksichtigt werden.
Exceptions, die bei der Ausgabe in eine Datei entstehen können, müssen nicht abgefangen werden.

Beispiel

Beispiel-Dateistruktur:

```
c:\files [DIR]
   |
   +--myDir [DIR]
   |  |
   |  +--myDir2 [DIR]
   |  |   |
   |  |   +ArrayTest.java [FILE]
   |  |
   |  +--Fibonacci.java [FILE]
   |
   +--Babylon.java [FILE]
   |
   +--BubbleSort1.java [FILE]
```

Aufruf 1:

`java Jgrep print c:\files\Babylon.java` *(Es wird also nach „print“ gesucht.)*

Hinweis: Der Aufruf muss nicht unbedingt in der Command-Shell getestet werden. Es reicht, wenn Sie Ihre Entwicklungsumgebung (Parameter) entsprechend konfigurieren!

Ausgabe:

```
Babylon.java, 10:    System.out.print("Bitte geben Sie eine ganze Zahl ein: ");
Babylon.java, 12:    System.out.println("Ermittle Quadratwurzel aus " + a);
Babylon.java, 19:         System.out.printf("Schritt %2d: %10f \n",i++,x);
```

Aufruf 2:

```
java Jgrep -R –F print c:\files\ c:\output.txt
```

Ausgabe in der Datei c:\output.txt:

```
Babylon.java, 10:    System.out.print("Bitte geben Sie eine ganze Zahl ein: ");
Babylon.java, 12:    System.out.println("Ermittle Quadratwurzel aus " + a);
Babylon.java, 19:         System.out.printf("Schritt %2d: %10f \n",i++,x);
BubbleSort1.java, 25:         System.out.print(nums[i] + " ");
BubbleSort1.java, 27:     System.out.println();
BubbleSort1.java, 50:              System.out.print(nums[i] + " ");
BubbleSort1.java, 52:         System.out.println();
Fibonacci.java, 16:     for(int i=0;i<n;i++) System.out.println(fib[i]);
ArrayTest.java, 27:         System.out.println("p1 gleich p2");
ArrayTest.java, 29:         System.out.println("p1 ungleich p2");
ArrayTest.java, 31:         System.out.println(a.p1[i]);
ArrayTest.java, 35:         System.out.println(a.p2[i]);
```