File-Regex

In windows files & Folders, In Linux files & Directories but java files or directories instance are same.

--- if occupy memory in system that is called file. For segregation & data storing
-- Every file has extensions.

File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.

File class

File class in Java is part of the java.io package and is used to represent file and directory pathnames in an abstract manner. It allows you to create, delete, inspect, and manipulate files and directories.

```
import java.io.File;
File myFile = new File("path/to/file.txt");
```

• boolean exists()	: Checks if the file or directory exists
• boolean isFile()	: Returns true if it's a file
• boolean isDirectory()	: Returns true if it's a directory
• String getName()	: Returns the name of the file/directory
• String getAbsolutePath()	: Gets the full path
• boolean createNewFile()	: Creates a new file (throws IOException)
• boolean mkdir()	: Creates a directory
• boolean delete()	: Deletes the file or directory
• long length()	: Returns the file size in bytes
• String[] list()	: Lists names of files in a directory

Streams

A sequence of data is known as a stream. This concept is used to perform I/O operations on a file.

1. Input Stream
   InputStream class is the superclass of all input streams. The input stream is used to read data from numerous input devices like the keyboard, network …

@ AudioInputStream
@ ByteArrayInputStream
@ FileInputStream
@ FilterInputStream
@ StringBufferInputStream
@ ObjectInputStream

InputStream obj = new FileInputStream();

<u>Methods of InputStream</u>

• obj.read()	: Reads one byte of data
• obj.read(byte[] array)()	: Reads byte from the stream and stores that byte
• obj.mark()	: Marks the position in the input stream
• obj.available()	: Returns the number of bytes available
• obj.markSupported()	: Check if the mark() method & the reset() method
• obj.reset()	: Returns the control to the point where the mark
• obj.skips()	: Skips and removes a particular number of bytes
• obj.close()	: Closes the input stream.

2. Output Stream
output stream is used to write data to numerous output devices like the monitor, file

@ ByteArrayOutputStream
@ FileOutputStream
@ StringBufferOutputStream
@ ObjectOutputStream
@ DataOutputStream
@ PrintStream

OutputStream obj = new FileOutputStream();

<u>Methods of OutputStream</u>

• obj.write()	: Writes the specified byte to the output stream.
• obj.write(byte[] array)	: Writes the bytes which are inside a specific array
• obj.close()	: Closes the output stream.
• Obj.flush()	: Forces to write all the data present in an output

Create a New File
Two standard methods to create a new file:

1. Using the File Class	: File.createNewFile()
2. Using the FileOutputStream Class	: new FileOutputStream(String filePath);

@ayekiran

Read Files
Multiple ways of writing and reading a text file in Java. this is required while dealing with many applications.

1. Using BufferedReader class
2. Using Scanner class
3. Using File Reader class
4. Reading the whole file in a List
5. Read a text file as String

***Note*:** We can also use both BufferReader and Scanner to read a text file line by line in Java. Then Java SE 8 introduces another Stream class **java.util.stream.Stream** which provides a lazy and more efficient way to read a file.

Write on Files
FileWriter class is used to write character-oriented data to a file as this class is character-oriented because it is used in file handling.

1. Using writeString() method
2. Using FileWriter Class
3. Using BufferedWriter Class
4. Using FileOutputStream Class

Delete a File
To delete files programmatically. In contrast to normal delete operations in any operating system, files being deleted.

1. java.io.File.delete() Method
2. java.nio.file.files.deleteifexists(Path p) Method

🗎 TEXT Files
Text file contains only Palin text information like .txt, .csv, .log … These are files that contain plain text, and Java provides several classes for reading and writing text files.

Ways of reading the data from text files
1. FileInputStream class
2. Scanner class
3. FileReader class
4. BufferedReader class

Ways of writing the data from text files
1. FileOutputStream class
2. FileWriter class
3. BufferedWriter class

🖼 Image Files (e.g., .jpg, .png) & Binary Files

Image files include formats like .jpg, .png, .gif, .bmp, etc. These files store binary data representing visual information such as photographs, graphics, or icons. In Java, working with image files typically involves reading and writing binary data or using specialized libraries for image processing.

Ways of reading image files
1. FileInputStream class – Reads image data as raw bytes from a file.
2. BufferedInputStream class – Improves performance when reading large image files.
3. ImageIO.read(File) – Part of javax.imageio.ImageIO; reads an image
4. Files.readAllBytes(Path) – Reads the entire file into a byte array, useful for binary

Ways of writing image files in Java
1. FileOutputStream class – Writes image data as raw bytes to a file.
2. BufferedOutputStream class – Buffers output for better performance with large img
3. ImageIO.write(BufferedImage, formatName, File) – Saves a BufferedImage to disk
4. Files.write(Path, byte[]) – Writes byte array data (such as image bytes) directly file.

🎵 Audio Files (e.g., .wav)

Audio files contain sound data, typically in formats such as .wav, .mp3, .aac, etc. Java provides various classes and APIs for reading, writing, and processing audio files.

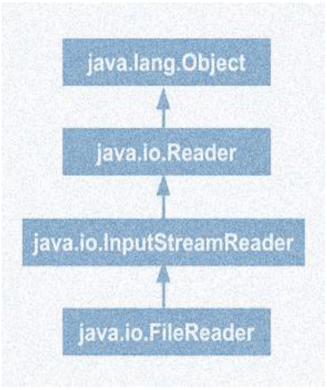
Ways of reading data from audio files
1. AudioInputStream class (from javax.sound.sampled)
2. Clip class
3. TargetDataLine class
4. SourceDataLine class

Ways of writing data to audio files
1. AudioSystem.write() method
2. TargetDataLine (with AudioSystem.write)
3. ByteArrayOutputStream

FileReader Class

FileReader is a class in the java.io package which can be used to read a stream of characters from the files. Java IO FileReader class uses either specified charset or the platform's default charset for decoding from bytes to characters.

- Charset class is used to define methods for producing encoders and decoders
- Default charset is defined during implicit computer start-up and it depends

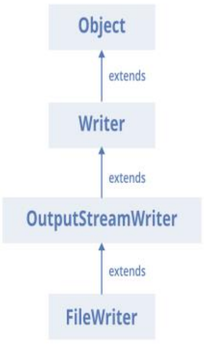


FileWriterClass

FileWriter class of the java.io package is used to write data in character form to a file. The FileWriter class in Java is used to write character-oriented data to a file.

FileWriter extends OutputStreamWriter and Writer classes.

@ Closeable
@ Flushable
@ Appendable
@ AutoCloseable



FilePermission Class

java.io.FilePermission class represents access to a file or directory. These accesses are in the form of a path name and a set of actions associated with the path name.

- read: Grants permission to read the file.
- write: Grants permission to write to the file.
- delete: Grants permission to delete the file by calling File.delete
- readlink: Grants permission to read symbolic links.
- execute: Grants permission to execute the file.

FileDescriptor Class

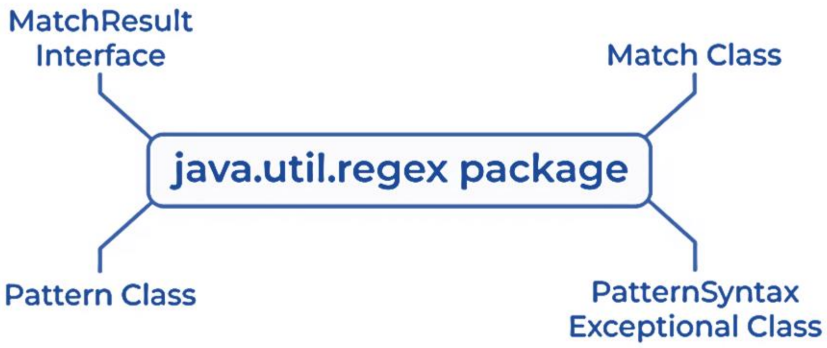
java.io.FileDescriptor class works for opening a file having a specific name.

--- java.io.FileDescriptor class in Java works for opening a file having a specific name.
--- Applications should not create their own file descriptors.

@ayekiran

Regular Expressions:
Regular Expressions or Regex (java.util.regex) is an API for defining String patterns that can be used for searching, manipulating, and editing a string.

--- Email validation and passwords are a few areas of strings.
--- Regex is widely used to define the constraints.



Pattern Class:
Pattern class is part of the java.util.regex package and is used for defining and working with regular expressions—text patterns used for searching, matching, or manipulating strings.

--- It compiles a regular expression into a pattern that can be used for pattern matching.
--- It is immutable and thread-safe.
--- You typically use it with the Matcher class to perform matching operations.

Key Methods of Pattern

• compile(String regex)	--- Compiles the given regular expression
• matcher(CharSequence input)	--- Creates a matcher that can be used to match
• split(CharSequence input)	--- Splits the input string around matches
• pattern()	--- Returns the regular expression from which this pattern was compiled.

Character Classes

• .	--- Any character except newline (\n)
• [abc]	--- a, b, or c
• [^abc]	--- Any character except a, b, or c
• [a-z]	--- Any lowercase letter
• [A-Z]	--- Any uppercase letter

[0-9]	--- Any digit
\\d	--- Digit (same as [0-9])
\\D	--- Non-digit
\\w	--- Word character (letters, digits, underscore)
\\W	--- Non-word character
\\s	--- Whitespace (spaces, tabs, etc.)
\\S	--- Non-whitespace

Quantifiers

*	--- 0 or more times
+	--- 1 or more times
?	--- 0 or 1 time
{n}	--- Exactly n times
{n,}	--- n or more times
{n,m}	--- Between n and m times

Common Patterns

\\d{4}-\\d{2}-\\d{2}	--- Date format: YYYY-MM-DD
\\w+@\\w+\\.\\w+	--- Basic email pattern
https?://\\S+	--- Matches http or https URLs
[A-Za-z0-9_]{8,}	--- At least 8 alphanumeric characters or underscores.

Matcher class

Matcher class is typically associated with regular expressions, where it's part of the java.util.regex package. It's used to perform match operations on a character sequence using a pattern defined by a Pattern object.

--- Search for matches of a regular expression in a given text.
--- Extract parts of a string based on regex groups.
--- Replace matching parts of a string.
--- Iterate over multiple matches.

Key Methods of Matcher

find()	: Attempts to find the next subsequence of the input sequence
matches()	: Attempts to match the entire region against the pattern.
lookingAt()	: Attempts to match the beginning of the input against the pattern.
group()	: Returns the matched subsequence.
start(), end()	: Return the start and end indices of the current match.
replaceAll()	: Replace matching substrings, Also, replaceFirst ()

Character Class

Character class is a wrapper class for the primitive data type char. It is part of the java.lang package and provides a number of useful static methods and constants for working with characters.

```
public final class Character extends Object
            implements Serializable, Comparable
```

1. Wrapper for char

   Allows a char primitive to be used as an object, which is useful for collections like ArrayList that work with objects, not primitives.

   ```
   char c = 'a';
   Character ch = Character.valueOf(c);
   ```

2. Character Testing Methods

   ```
   Character.isLetter('a');              // true
   Character.isDigit('1');               // true
   Character.isWhitespace(' ');          // true
   Character.isUpperCase('A');           // true
   Character.isLowerCase('a');                     // true
   ```

3. Character Conversion Methods:

   ```
   Character.toUpperCase('a'); // 'A'
   Character.toLowerCase('A'); // 'a'
   ```

Quantifiers

Quantifiers allow users to specify the number of occurrences to match against. It used with regular expressions to specify the number of times a particular pattern or character can appear in the Input.

X*	--- Zero or more occurrences of X
X?	--- Zero or One occurrence of X
X+	--- One or More occurrences of X
X{n}	--- Exactly n occurrences of X
X{n, }	--- At least n occurrences of X
X{n, m}	--- Count of occurrences of X is from n to m

Types of Quantifiers

1. Greedy Quantifier (Default)
2. Reluctant Quantifier (Appending ? after a quantifier)