Files

Published Jul 29, 2022

Contribute to Docs

Java provides a number of different classes and methods for utilizing files and a computer's file system. They include the `File`, `FileReader`, and `FileWriter` classes (all from the `java.io` package).

Accessing Files

Files and directories can be accessed with the `File` class.

Syntax

```
import java.io.File;

File myFile = new File(pathString);
```

The constructor of the `File` class accepts a `pathString` that specifies path/filename. Once declared, the new `myFile` object can be used to manipulate and gather information about the file or directory specified by `pathString`.

Methods

The `File` object includes the following methods to manipulate the specified file or directory:

- `canRead()`: Returns `true` if the file is readable.
- `canWrite()`: Returns `true` if the file is writable.
- `createNewFile()`: Creates an empty file. Returns `true` if successful.
- `delete()`: Deletes a file. Can delete a directory if it is empty.

- exists(): Returns true if the file/directory exists.
- getName(): Returns the name of the file/directory.
- getAbsolutePath(): Returns the full pathname of the file/directory.
- isDirectory(): Returns true if instance points to a directory.
- isFile(): Returns true if instance points to a file.
- length(): Returns the size of the file in bytes.
- list(): Returns a String[] array of the files in the directory.
- mkdir(): Creates a directory.

Example

The following example creates a File object, checks if its corresponding file exists, and if not, creates it (file access is placed within a try ... catch block in case the file system throws any errors):

```
import java.io.File;
import java.io.IOException;

public class FileExample {
 public static void main(String[] args) {
  try {
   File myFile = new File("test.txt");
   if (myFile.exists()) {
    System.out.println("File exists: " + myFile.getName());
   } else {
    if (myFile.createNewFile()) {
     System.out.println("File created: " + myFile.getName());
    } else {
     System.out.println("File was not created.");
    }
   }
  } catch (IOException e) {
   System.out.println("An error was thrown.");
   e.printStackTrace();
  }
 }
}
```

Writing to a File

Data can be written to a file with the FileWriter class.

Syntax

```
import java.io.FileWriter;

FileWriter myWriter = new FileWriter(pathString, append);
```

The constructor of the `FileWriter` class takes a `pathString` that specifies a path/filename. Once declared, the new `myWriter` object can be used to write to the file specified by `pathString`. The optional `append` boolean specifies if writing will append to the file.

Methods

The following methods are provided by the `FileWriter`:

- `close()`: Closes the `FileWriter`. Should be done after all writes are complete.
- `write()`: Writes a string or char sequence to the file.

Example

The following example writes a string out to a file (again, within a `try ... catch` block):

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

public class FileWriterExample {
 public static void main(String[] args) {
  try {
   FileWriter myWriter = new FileWriter("test.txt");
   myWriter.write("Hello World!");
   myWriter.close();
  } catch (IOException e) {
   System.out.println("An error was thrown.");
   e.printStackTrace();
  }
 }
}
```

Reading Files

Data can be read from a file with the `FileReader` class.

Syntax

```
import java.io.FileReader;
```

```
FileReader myReader = new FileReader(pathString);
```

The constructor of the `FileReader` class takes a pathString that specifies a path/filename. Once declared, the new `myReader` object can be used to read characters from the file specified by `pathString`.

Methods

The `FileReader` class offers the following methods to read from a file:

- `close()`: Closes the `FileReader`. Should be done after all reads are complete.
- `read()`: Reads a character from the file, or reads characters into a buffer.

Example

The following example reads all the characters from a file (again, within a `try ... catch` block):

```
import java.io.FileReader;
import java.io.IOException;

public class FileReaderExample {
  public static void main(String[] args) {
    try {
      FileReader myReader = new FileReader("test.txt");
      int ;
      while (( = myReader.read()) != -1)
        System.out.print((char) );
      myReader.close();
    } catch (IOException e) {
      System.out.println("An error was thrown.");
      e.printStackTrace();
    }
  }
}
```

All contributors

1.