File Handling in Java

Dr. Rab Nawaz Jadoon

Assistant Professor
COMSATS University, Abbottabad
Pakistan

Department of Computer Science
DCS
COMSATS University, Islamabad
(Abbottabad Campus)

Object Oriented Programming (OOP)

File handling in Java

- File handling is an important part of any application.
- The File class from the java.io package, allows us to work with files.
- To use the File class, create an object of the class, and specify the filename or directory name.
- Example
  - On the next slide

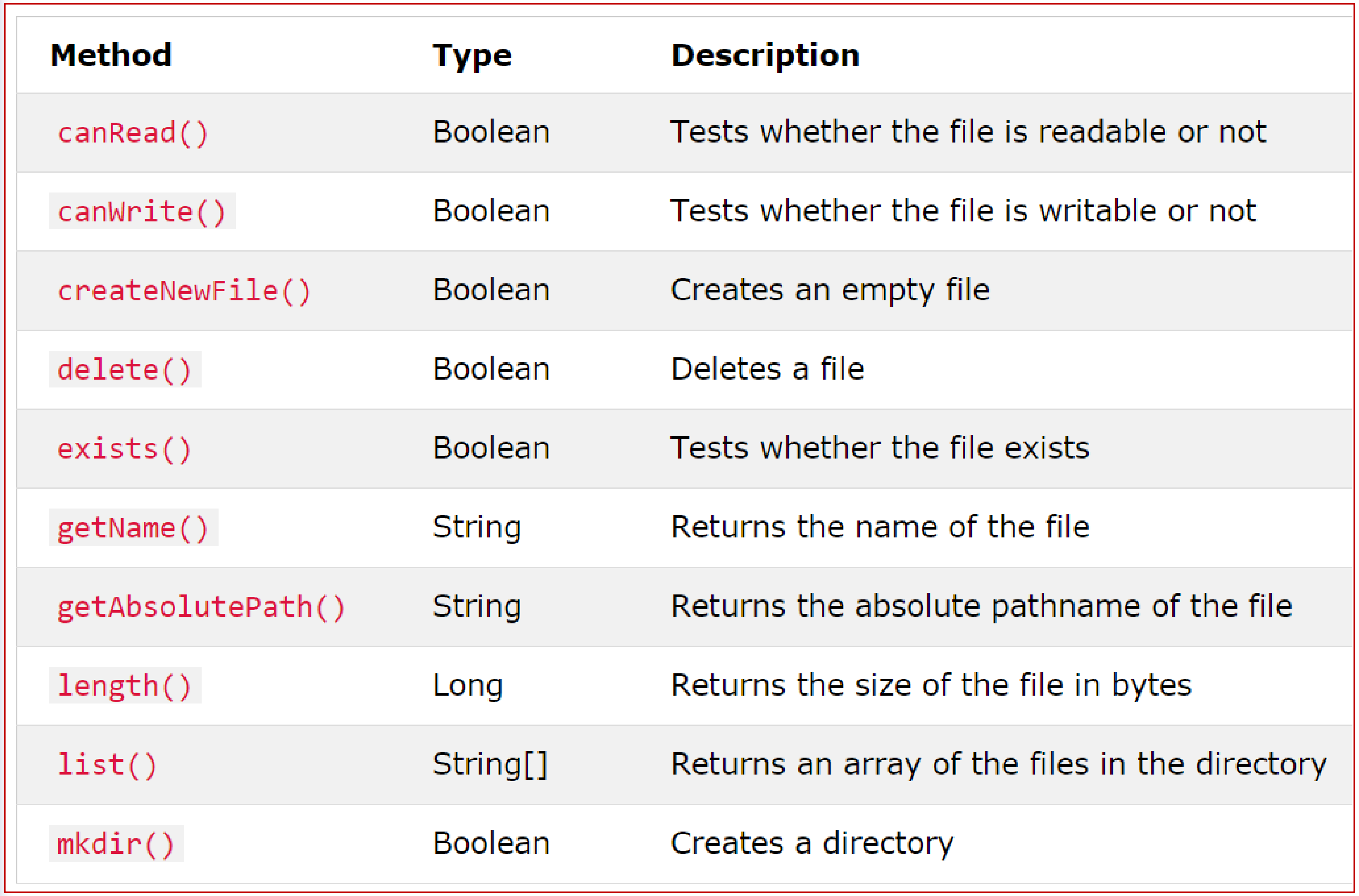
Cont…

```
import java.io.File;  // Import the File class

File myObj = new File("filename.txt"); // Specify the filename
```

1. Java File class represents the files and directory pathnames in an abstract manner.
2. This class is used for creation of files and directories, file searching, file deletion, etc.
3. The File object represents the actual file/directory on the disk.

File Class Methods

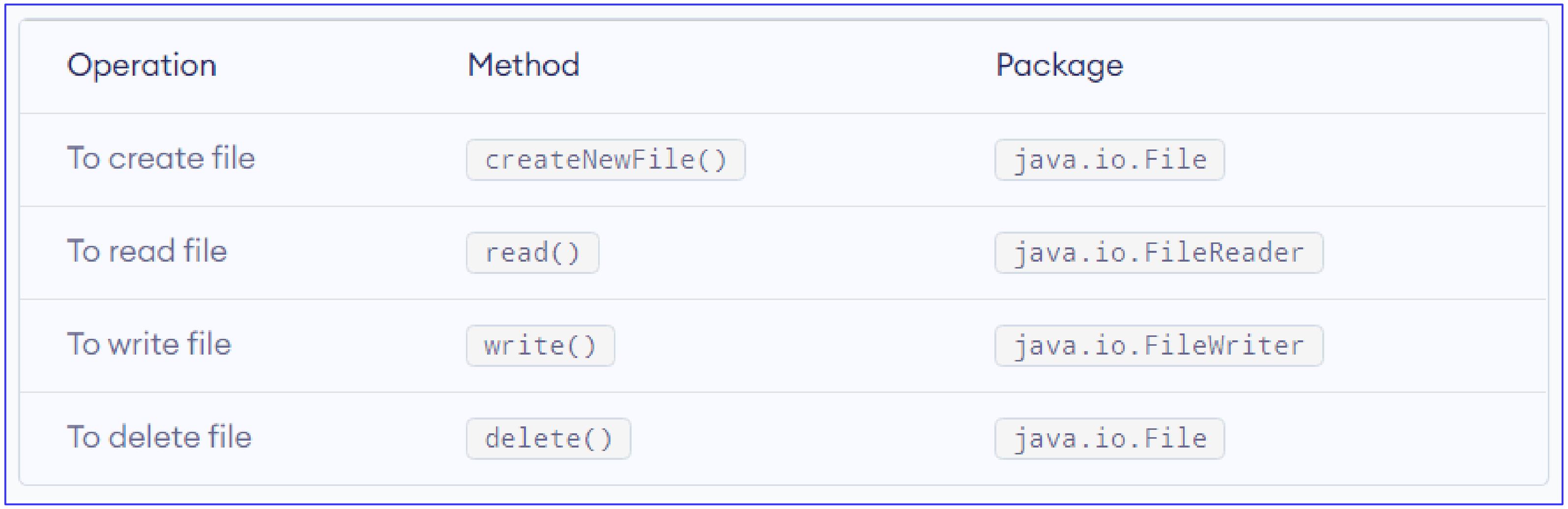


Method	Type	Description
`canRead()`	Boolean	Tests whether the file is readable or not
`canWrite()`	Boolean	Tests whether the file is writable or not
`createNewFile()`	Boolean	Creates an empty file
`delete()`	Boolean	Deletes a file
`exists()`	Boolean	Tests whether the file exists
`getName()`	String	Returns the name of the file
`getAbsolutePath()`	String	Returns the absolute pathname of the file
`length()`	Long	Returns the size of the file in bytes
`list()`	String[]	Returns an array of the files in the directory
`mkdir()`	Boolean	Creates a directory

Java Create and Write To Files

- To create a file in Java, you can use the createNewFile() method.
- This method returns a boolean value: true if the file was successfully created, and false if the file already exists.
- Note that the method is enclosed in a try...catch block.
- This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason):

JAVA File Operation Method



Operation	Method	Package
To create file	`createNewFile()`	`java.io.File`
To read file	`read()`	`java.io.FileReader`
To write file	`write()`	`java.io.FileWriter`
To delete file	`delete()`	`java.io.File`

Creating a File

```java
import java.io.File;  // Import the File class
import java.io.IOException;  // Import the IOException class to handle errors

public class CreateFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
      } else {
        System.out.println("File already exists.");
      }
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

```
File created: filename.txt
```

Writing File to a Specific Directory

- To create a file in a specific directory (requires permission), specify the path of the file and use double backslashes to escape the "\" character (for Windows).
- On Mac and Linux you can just write the path, like: /Users/name/filename.txt

```
File myObj = new File("C:\\Users\\MyName\\filename.txt");
```

Write to a File

- In the following example, we use the FileWriter class together with its write() method to write some text to the file we created in the example above.
- Note that when you are done writing to the file, you should close it with the close() method:

Example Program

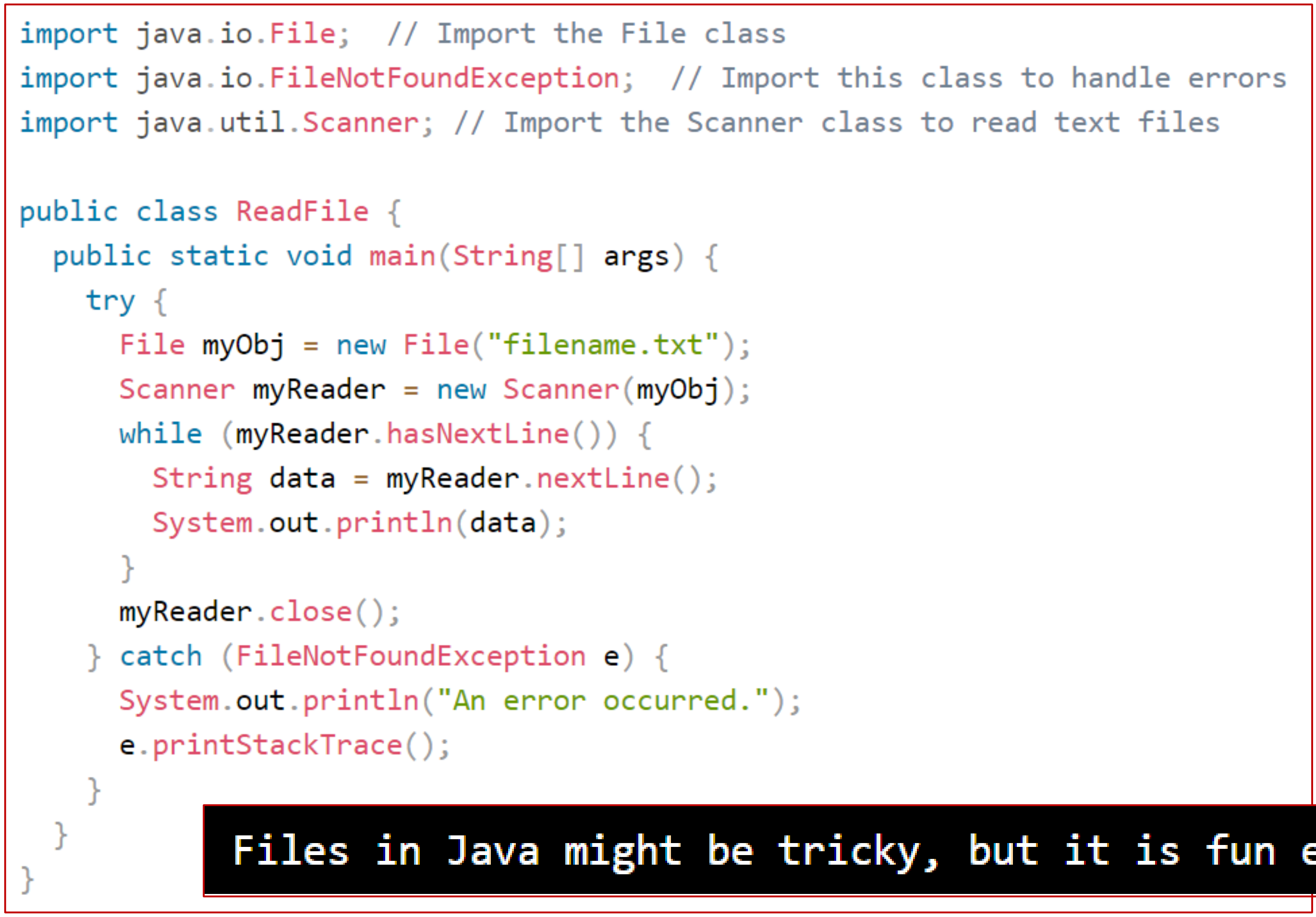
```java
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to handle errors

public class WriteToFile {
  public static void main(String[] args) {
    try {
      FileWriter myWriter = new FileWriter("filename.txt");
      myWriter.write("Files in Java might be tricky, but it is fun enough!");
      myWriter.close();
      System.out.println("Successfully wrote to the file.");
    } catch (IOException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

```
Successfully wrote to the file.
```

Read a File

- In the following example, we use the Scanner class to read the contents of the text file.



```java
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

public class ReadFile {
  public static void main(String[] args) {
    try {
      File myObj = new File("filename.txt");
      Scanner myReader = new Scanner(myObj);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("An error occurred.");
      e.printStackTrace();
    }
  }
}
```

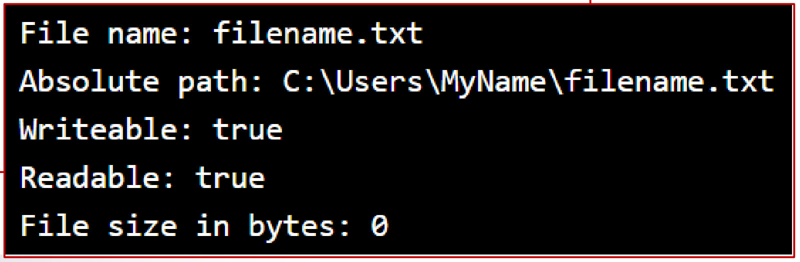
Files in Java might be tricky, but it is fun enough!

Getting File Information

- To get more information about a file, use any of the File methods:

```java
import java.io.File;  // Import the File class

public class GetFileInfo {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.exists()) {
      System.out.println("File name: " + myObj.getName());
      System.out.println("Absolute path: " + myObj.getAbsolutePath());
      System.out.println("Writeable: " + myObj.canWrite());
      System.out.println("Readable " + myObj.canRead());
      System.out.println("File size in bytes " + myObj.length());
    } else {
      System.out.println("The file does not exist.");
    }
  }
}
```



```
File name: filename.txt
Absolute path: C:\Users\MyName\filename.txt
Writeable: true
Readable: true
File size in bytes: 0
```

Deleting a File

```java
import java.io.File;  // Import the File class

public class DeleteFile {
  public static void main(String[] args) {
    File myObj = new File("filename.txt");
    if (myObj.delete()) {
      System.out.println("Deleted the file: " + myObj.getName());
    } else {
      System.out.println("Failed to delete the file.");
    }
  }
}
```

```
Deleted the file: filename.txt
```

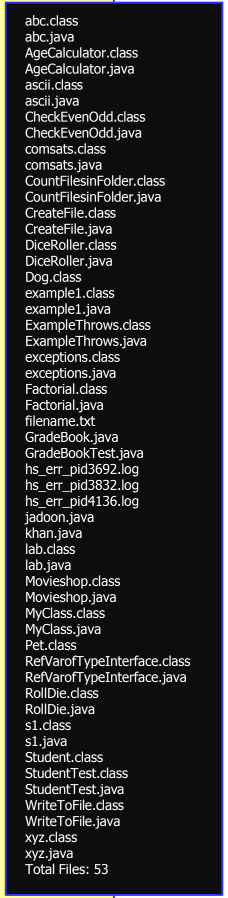
Listing all files in a Folder

```java
import java.io.File;

public class ListFilesinFolder
{
  public static void main(String[] args)
  {
    int count=0;
    // creates a file object
    File file = new File("C:\\Users\\Rab Nawaz\\Desktop\\JAVA");

    // returns an array of all files
    String[] fileList = file.list();

    for(String str : fileList)
    {
      System.out.println(str);
      count++;
    }
     System.out.println("Total Files:" +count);
  }
}
```

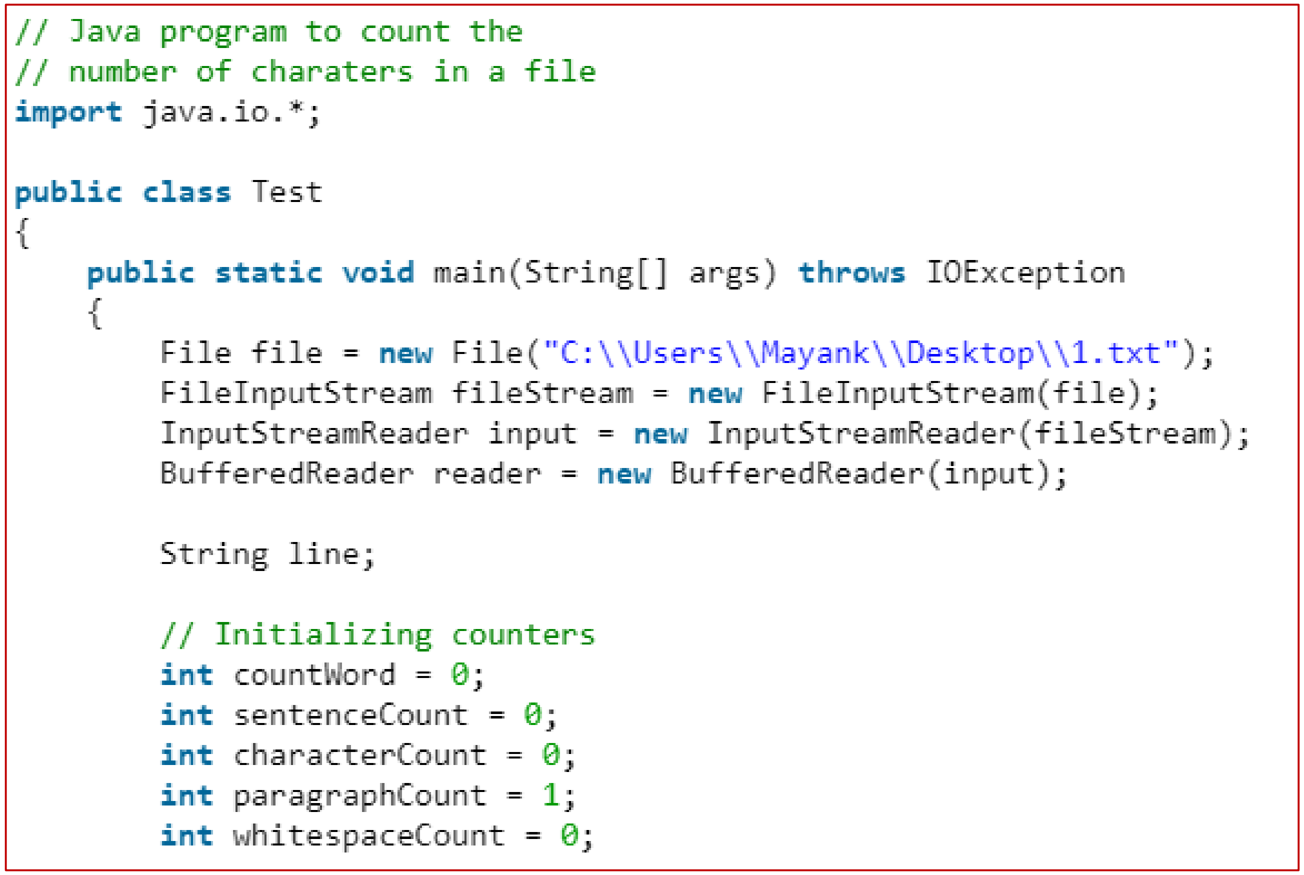


```
abc.class
abc.java
AgeCalculator.class
AgeCalculator.java
ascii.class
ascii.java
CheckEvenOdd.class
CheckEvenOdd.java
comsats.class
comsats.java
CountFilesinFolder.class
CountFilesinFolder.java
CreateFile.class
CreateFile.java
DiceRoller.class
DiceRoller.java
Dog.class
example1.class
example1.java
ExampleThrows.class
ExampleThrows.java
exceptions.class
exceptions.java
Factorial.class
Factorial.java
filename.txt
GradeBook.java
GradeBookTest.java
hs_err_pid3692.log
hs_err_pid3832.log
hs_err_pid4136.log
jadoon.java
khan.java
lab.class
lab.java
Movieshop.class
Movieshop.java
MyClass.class
MyClass.java
Pet.class
RefVarofTypeInterface.class
RefVarofTypeInterface.java
RollDie.class
RollDie.java
s1.class
s1.java
Student.class
StudentTest.class
StudentTest.java
WriteToFile.class
WriteToFile.java
xyz.class
xyz.java
Total Files: 53
```

FileInputStream

- A FileInputStream obtains input bytes from a file in a file system.
- What files are available depends on the host environment.
- FileInputStream is meant for reading streams of raw bytes such as image data.
- For reading streams of characters, consider using FileReader.

Counting Number of Characters in a file



```java
// Java program to count the
// number of charaters in a file
import java.io.*;

public class Test
{
    public static void main(String[] args) throws IOException
    {
        File file = new File("C:\\Users\\Mayank\\Desktop\\1.txt");
        FileInputStream fileStream = new FileInputStream(file);
        InputStreamReader input = new InputStreamReader(fileStream);
        BufferedReader reader = new BufferedReader(input);

        String line;

        // Initializing counters
        int countWord = 0;
        int sentenceCount = 0;
        int characterCount = 0;
        int paragraphCount = 1;
        int whitespaceCount = 0;
```

Cont…

```
// Reading line by line from the
// file until a null is returned
while((line = reader.readLine()) != null)
{
    if(line.equals(""))
    {
        paragraphCount++;
    } else {
        characterCount += line.length();

        // \\s+ is the space delimiter in java
        String[] wordList = line.split("\\s+");

        countWord += wordList.length;
        whitespaceCount += countWord -1;

        // [!?.:]+ is the sentence delimiter in java
        String[] sentenceList = line.split("[!?.:]+");

        sentenceCount += sentenceList.length;
    }
}
```

Cont...

```
        System.out.println("Total word count = " + countWord);
        System.out.println("Total number of sentences = " + sentenceCount);
        System.out.println("Total number of characters = " + characterCount);
        System.out.println("Number of paragraphs = " + paragraphCount);
        System.out.println("Total number of whitespaces = " + whitespaceCount);
    }
}
```

Output

```
Total word count = 5
Total number of sentences = 3
Total number of characters = 21
Number of paragraphs = 2
Total number of whitespaces = 7
```

Thanks