CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science Engineering

Subject Name: Java Programming
Semester: III
Subject Code: CSE201
Academic year: 2024-25

Part - 6

No.	Aim of the Practical

27.	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.
	
	 PROGRAM CODE:
	
	 `import java.io.BufferedReader;`
	
	 `import java.io.FileReader;`
	
	 `import java.io.IOException;`
	
	 `public class pra27 {`
	
	 `  public static void main(String[] args) {`
	
	 `    if (args.length == 0) {`
	
	 `      args = new String[]{"hello.txt"};`

```java
        }


        for (String fileName : args) {

            try (BufferedReader reader = new
BufferedReader(new FileReader(fileName))) {

                int lineCount = 0;

                while (reader.readLine() != null) {

                    lineCount++;

                }

                System.out.println(fileName + ": "
+ lineCount + " lines");

            } catch (IOException e) {

                System.err.println("Error reading
file " + fileName + ": " + e.getMessage());

            }

        }

    }

}
```
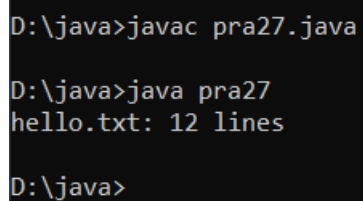
OUTPUT:

```
D:\java>javac pra27.java

D:\java>java pra27
hello.txt: 12 lines

D:\java>
```

CONCLUSION:

This program counts the number of lines in a file using Java. It reads each file specified in the command-line arguments or defaults to hello.txt if no arguments are provided. The program uses BufferedReader to read each line and increments a counter for each line read. It handles file reading errors gracefully using a try-with-resources block. The program prints the number of lines for each file processed. This showcases efficient file handling and error management in Java.

---

28. Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.
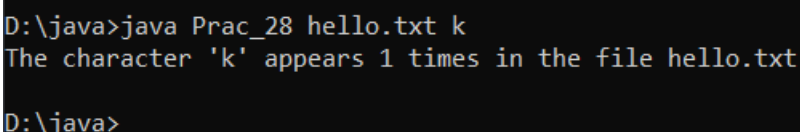
PROGRAM CODE :

```java
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;


public class Prac_28 {

    public static void main(String[] args) {

        if (args.length != 2) {

            System.out.println("Usage: java CharCount <file> <character>");

            return;

        }


        String fileName = args[0];

        char targetChar = args[1].charAt(0);
```

```java
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

        int charCount = 0;

        int c;

        while ((c = reader.read()) != -1) {

          if (c == targetChar) {

            charCount++;

          }

        }

        System.out.println("The character '" + targetChar + "' appears " + charCount + "
times in the file " + fileName);

    } catch (IOException e) {

        System.err.println("Error reading file " + fileName + ": " + e.getMessage());

    }

  }

}
```

OUTPUT:

```
D:\java>java Prac_28 hello.txt k
The character 'k' appears 1 times in the file hello.txt

D:\java>
```

CONCLUSION:
This program counts the occurrences of a specific character in a file using Java. It reads
the file character by character with BufferedReader and compares each character to the
target character. If they match, it increments a counter. The program handles file reading
errors using a try-with-resources block to ensure the reader is closed properly. It also
provides usage instructions if the required command-line arguments are not provided.
This showcases efficient character processing and error management in Java.

29	
	This program counts the occurrences of a specific character in a file using Java. It reads the file character by character with BufferedReader and compares each character to the target character. If they match, it increments a counter. The program handles file reading errors using a try-with-resources block to ensure the reader is closed properly. It also provides usage instructions if the required command-line arguments are not provided.

Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.

PROGRAM CODE:

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

public class Prac_29 {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java Prac_29 <file> <word>");
            return;
        }

        String fileName = args[0];
        String targetWord = args[1];

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            int wordCount = 0;
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");
                for (String word : words) {
                    if (word.equals(targetWord)) {
                        wordCount++;
                    }
                }
            }
            System.out.println("The word '" + targetWord + "' appears " + wordCount + "
times in the file " + fileName);
        } catch (IOException e) {
            System.err.println("Error reading file " + fileName + ": " + e.getMessage());
        }

        // Wrapper Class Example
        Integer wrapperInt = Integer.valueOf(10); // Using Integer wrapper class
        int primitiveInt = wrapperInt.intValue(); // Converting back to primitive int
```

```
        System.out.println("Wrapper Class Example: Integer value is " + wrapperInt + " and
primitive int value is " + primitiveInt);
    }
}
```

OUTPUT:

```
D:\java>javac Prac_29.java

D:\java>java Prac_29 hello.txt am
The word 'am' appears 0 times in the file hello.txt
Wrapper Class Example: Integer value is 10 and primitive int value is 10

D:\java>
```

CONCLUSION:
This program demonstrates how to count the occurrences of a specific word in a file using Java. It reads the file line by line with BufferedReader and splits each line into words. It then compares each word to the target word and increments a counter if they match. The program handles file reading errors gracefully using a try-with-resources block. It also provides usage instructions if the required command-line arguments are not provided. This showcases efficient text processing and error management in Java.

30	Write a program to copy data from one file to another file.If the destination file does not exist, it is created automatically.

PROGRAM CODE:

```java
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class Prac_30 {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java Prac_30 <source file> <destination file>");
            return;
        }

        String sourceFile = args[0];
        String destinationFile = args[1];

        try (FileInputStream fis = new FileInputStream(sourceFile);
            FileOutputStream fos = new FileOutputStream(destinationFile)) {

            byte[] buffer = new byte[1024];
            int bytesRead;

            while ((bytesRead = fis.read(buffer)) != -1) {
                fos.write(buffer, 0, bytesRead);
            }

            System.out.println("File copied successfully from " + sourceFile + " to " +
```

```
destinationFile);
    } catch (IOException e) {
        System.err.println("Error copying file: " + e.getMessage());
    }
  }
}
```

OUTPUT:

```
D:\java>javac Prac_30.java

D:\java>java Prac_30 hello.txt destination.txt
File copied successfully from hello.txt to destination.txt

D:\java>
```

CONCLUSION:

This program demonstrates how to copy data from one file to another using byte streams in Java. It reads from a source file and writes to a destination file, creating the destination file if it does not exist. The program uses FileInputStream to read bytes and FileOutputStream to write bytes. It handles errors using a try-with-resources block to ensure streams are closed properly. The program also provides usage instructions if the required command-line arguments are not provided. This showcases efficient file handling and error management in Java.

---

31 Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.

PROGRAM CODE :

```java
import java.io.*;


class Prac_31 {

    public static void main(String[] args) {

        // Demonstrate character stream

        try (FileReader fr = new FileReader("input.txt");
```

```java
    FileWriter fw = new FileWriter("output_char.txt")) {

    int c;

    while ((c = fr.read()) != -1) {

        fw.write(c);

    }

    System.out.println("Character stream copy completed.");

} catch (IOException e) {

    System.err.println("Error with character stream: " + e.getMessage());

}


// Demonstrate byte stream

try (FileInputStream fis = new FileInputStream("input.txt");

    FileOutputStream fos = new FileOutputStream("output_byte.txt")) {

    byte[] buffer = new byte[1024];

    int bytesRead;

    while ((bytesRead = fis.read(buffer)) != -1) {

        fos.write(buffer, 0, bytesRead);

    }

    System.out.println("Byte stream copy completed.");

} catch (IOException e) {

    System.err.println("Error with byte stream: " + e.getMessage());

}
```

```java
        // Use BufferedReader and BufferedWriter to read from console and write to a file

        try (BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

             BufferedWriter bw = new BufferedWriter(new FileWriter("console_output.txt")))
{

            System.out.println("Enter text (type 'exit' to finish):");

            String line;

            while (!(line = br.readLine()).equalsIgnoreCase("exit")) {

                bw.write(line);

                bw.newLine();

            }

            System.out.println("Console input written to file.");

        } catch (IOException e) {

            System.err.println("Error      with      BufferedReader/BufferedWriter:      "      +
e.getMessage());

        }

    }

}
```

OUTPUT:

```
D:\java>javac Prac_31.java

D:\java>java Prac_31
Character stream copy completed.
Byte stream copy completed.
Enter text (type 'exit' to finish):
my name is khushi
exit
Console input written to file.

D:\java>
```

CONCLUSION:

This program demonstrates the use of character and byte streams in Java. It reads from input.txt and writes to output_char.txt using character streams, and to output_byte.txt using byte streams. Additionally, it uses BufferedReader to read console input and BufferedWriter to write the input to console_output.txt. The program continues to read from the console until the user types "exit". This showcases efficient file handling and console interaction in Java.