

Aim: Implement programs related to File I/O

- 6.1 Create a csv file which will contain 10 integers in a spreadsheet. Read the file using class `java.util.Scanner` and display the sum of the numbers in the file. Handle all possible exceptions. Write a Java program to create, read and modify a file.
- 6.2 Create two objects of class `Path` viz., source and target. Perform the following operations a. Create a file at source b. Copy a file from source to target c. Move a file from source to target d. Delete a file from source e. Retrieve information about source and target

Tools used: Editor (Notepad/IntelliJ IDE), JDK and JRE

Theory:

Java file I/O

Java I/O (Input and Output) is used *to process the input and produce the output*.

Java uses the concept of a stream to make I/O operation fast. The `java.io` package contains all the classes required for input and output operations.

We can perform **file handling in Java** by Java I/O API.

The two important streams are **FileInputStream** and **FileOutputStream**

FileInputStream

This stream is used for reading data from the files. Objects can be created using the keyword **new** and there are several types of constructors available.

Following constructor takes a file name as a string to create an input stream object to read the file –

```
InputStream f = new FileInputStream("C:/java/hello");
```

Following constructor takes a file object to create an input stream object to read the file. First we create a file object using `File()` method as follows –

```
File f = new File("C:/java/hello");  
InputStream f = new FileInputStream(f);
```

Once you have *InputStream* object in hand, then there is a list of helper methods which can be used to read to stream or to do other operations on the stream.

FileOutputStream

`FileOutputStream` is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output. Here are two constructors which can be used to create a `FileOutputStream` object.

Following constructor takes a file name as a string to create an input stream object to write the file –

```
OutputStream f = new FileOutputStream("C:/java/hello")
```

Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using `File()` method as follows –

```
File f = new File("C:/java/hello");  
OutputStream f = new FileOutputStream(f);
```

Once you have *OutputStream* object in hand, then there is a list of helper methods, which can be used to write to stream or to do other operations on the stream.

Directories in Java

A directory is a **File** which can contain a list of other files and directories. You use **File** object to create directories, to list down files available in a directory. For complete detail, check a list of all the methods which you can call on File object and what are related to directories.

Creating Directories

There are two useful **File** utility methods, which can be used to create directories –

- The **mkdir()** method creates a directory, returning true on success and false on failure. Failure indicates that the path specified in the File object already exists, or that the directory cannot be created because the entire path does not exist yet.
- The **mkdirs()** method creates both a directory and all the parents of the directory.

Code:

- **Create a csv file which will contain 10 integers in a spreadsheet. Read the file using class java.util.Scanner and display the sum of the numbers in the file. Handle all possible exceptions. Write a Java program to create, read and modify a file.**

```
import java.util.*;
import java.io.*;
public class exp6_1 {
    public static void main(String[] args) {
        double sum = 0;
        List<Double> numbers = new ArrayList<>();
        String dataFilePath = "data.csv";
        String defaultData = "200,500,25,50\n100,50,25\n70,30,40,60\n40\n60,90,150\n20,40";
        Scanner inputScanner = new Scanner(System.in);
        char opted;
        File dataFile = new File(dataFilePath);

        // If file doesn't exist, creating one and writing default data to the file
        if (!dataFile.exists()) {
            System.out.println("CSV file could not be found, hence creating one !\nYou can put desired data in the file manually separated by commas");
            try {
                if (dataFile.createNewFile()) {
                    System.out.println("File created: " + dataFile.getAbsolutePath());
                    FileWriter defauFileWriter = new FileWriter(dataFile.getName());
                    defauFileWriter.write(defaultData);
                    defauFileWriter.close();
                } else {
                    System.out.println("There was a problem creating new file, try creating one manually");
                }
            } catch (Exception e) {
                System.out.println("An error occurred while writing default data to the file, try writing manually :)");
                e.printStackTrace();
            }
        }

        // Find sum of all numbers in csv file
        // All numbers in CSV file:
        displaySum(dataFile, sum, numbers);
        // Modify the numbers in csv file
        System.out.print("Do you want to write numbers to csv file? y/n: ");
        opted = inputScanner.nextLine().trim().charAt(0);
        if (opted == 'y' || opted == 'Y') {
            int n;
            double entity;
            System.out.println("How many decimal numbers do you wish to write to file(eg. 4 or 8): ");
            n = inputScanner.nextInt();
            System.out.println("Enter the numbers separated by spaces:");
            List<Double> newData = new ArrayList<>();
            while (n-- > 0) {
                entity = inputScanner.nextDouble();
                newData.add(entity);
            }
            String data = newData.toString();
            data = data.substring(1, data.length() - 1);
            System.out.println(data);
            try {
                FileWriter customWriter = new FileWriter(dataFile.getName());
                customWriter.write(data);
                customWriter.close();
                System.out.println("Your changes are succesfully written to file: " + dataFile.getAbsolutePath());
            } catch (Exception e) {
                System.out.println("xxxxxx ERROR xxxxxxxxxx::: Close the CSV file And Try Again  :)");
                e.printStackTrace();
            }
        }

        numbers.clear();

        sum = 0;

        displaySum(dataFile, sum, numbers);          // All numbers in CSV file:

        inputScanner.close();
    }

    private static void displaySum(File dataFile, double sum, List<Double> numbers) {
```

```

try {

    Scanner csvScanner = new Scanner(dataFile);

    Scanner dataScanner = null;

    while (csvScanner.hasNextLine()) {

        dataScanner = new Scanner(csvScanner.nextLine());

        dataScanner.useDelimiter(",");

        while (dataScanner.hasNext()) {

            try {

                String data = dataScanner.next().trim();

                numbers.add(Double.parseDouble(data));

                sum += Double.parseDouble(data);

            } catch (NumberFormatException ne) {

                continue;

            }

        }

        dataScanner.close();

    }

    csvScanner.close();

} catch (Exception e) {

    System.out.println(e);

    e.printStackTrace();

}

System.out.println("All numbers in CSV file:\n" + numbers.toString());

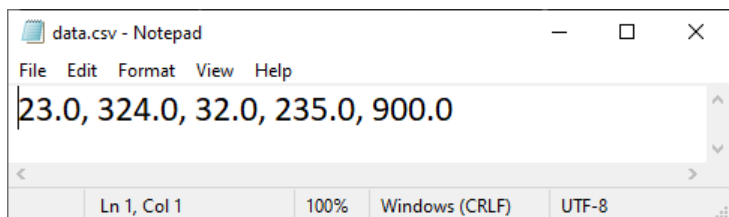
System.out.println("Sum of all numbers in CSV file: " + sum);

}

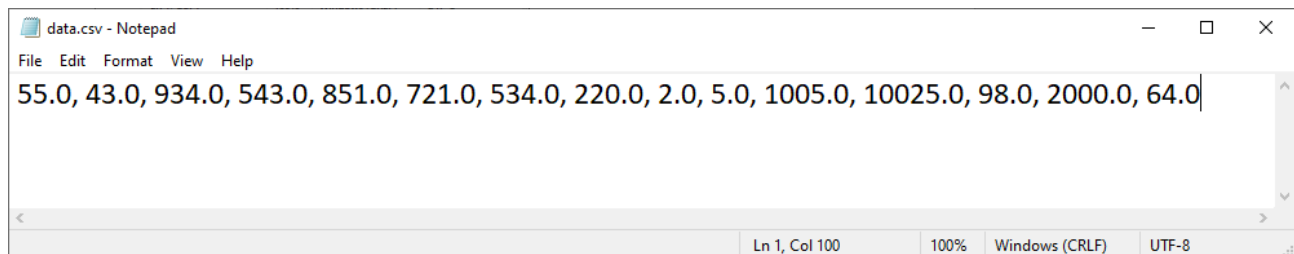
}

```

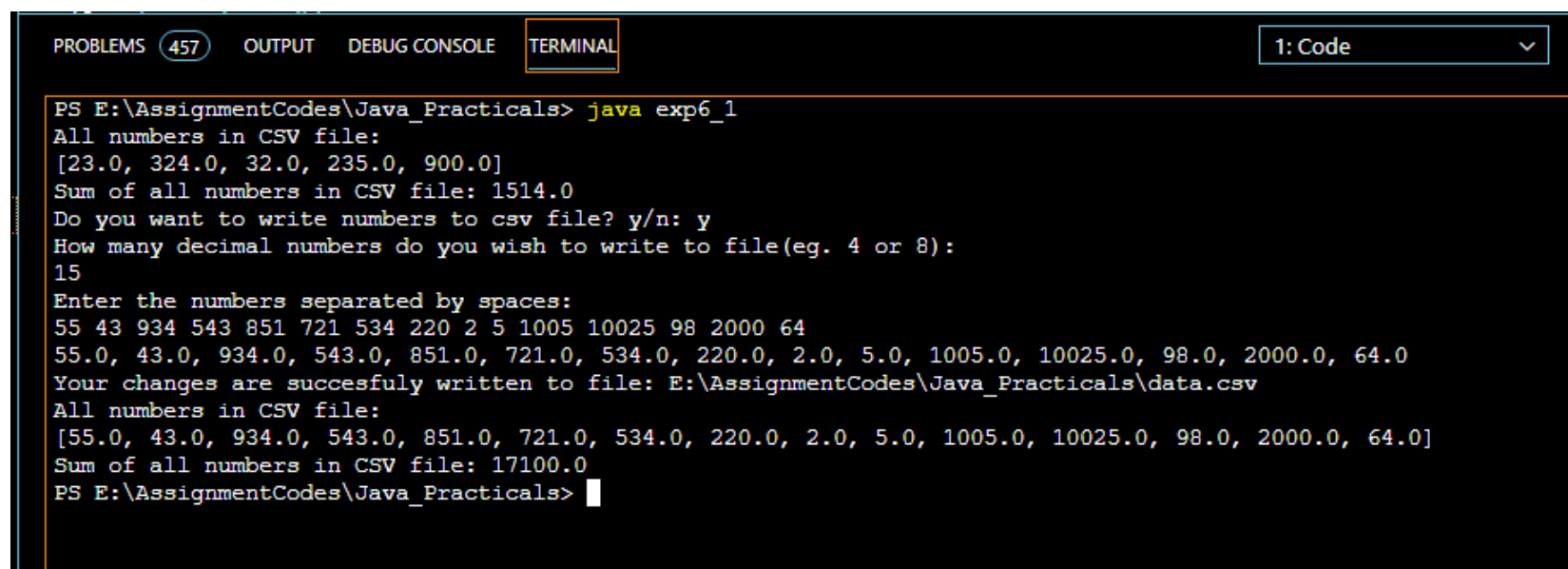
Previous CSV file:



CSV file after modifications:



Output:



- **Create two objects of class Path viz., source and target. Perform the following operations a. Create a file at source b. Copy a file from source to target c. Move a file from source to target d. Delete a file from source e. Retrieve information about source and target**

```

import java.nio.file.*;

import java.nio.file.Paths;

import java.io.*;

```

```

public class exp6_2 {

    public static void main(String[] args) {

        try{

            Path source = Paths.get("E:\\test");

            Path target = Paths.get("E:\\test\\subtest");

            String fn1=source+"\\";

            String fn2=target+"\\";

            FileWriter myWriter = new FileWriter(fn1+"filename.txt");

            myWriter.write("Files in Java might be tricky, but it is fun enough!");

            myWriter.close();

            //copy

            InputStream is = null;

            OutputStream os = null;

            File s=new File(fn1+"filename.txt");

            File d=new File(fn2+"filename.txt");
            is = new FileInputStream(s);

            os = new FileOutputStream(d);

            byte[] buffer = new byte[1024];

            int length;

            while ((length = is.read(buffer)) > 0)

                os.write(buffer, 0, length);

            is.close();
            os.close();
            .delete();

            Path temp = Files.move(Paths.get(fn1+"filename.txt"), Paths.get(fn2+"filename.txt"));

            s.delete();

            System.out.println(source+""");
            System.out.println(target+""");
            System.out.println(source.getParent()+""");
            System.out.println(target.getParent()+""");
            System.out.println(source.getRoot()+""");

            System.out.println(target.getRoot()+""");

        }catch(Exception ex){

            System.out.println(ex+""");

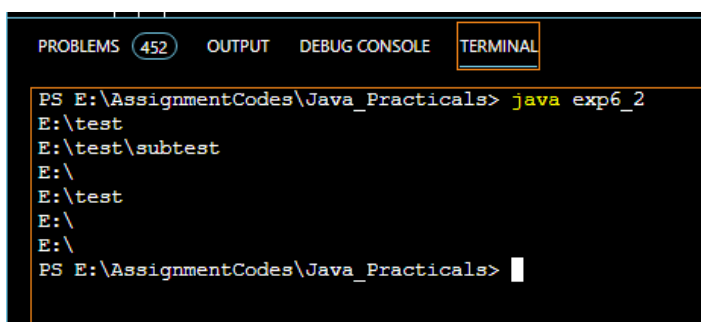
        }

    }

}

```

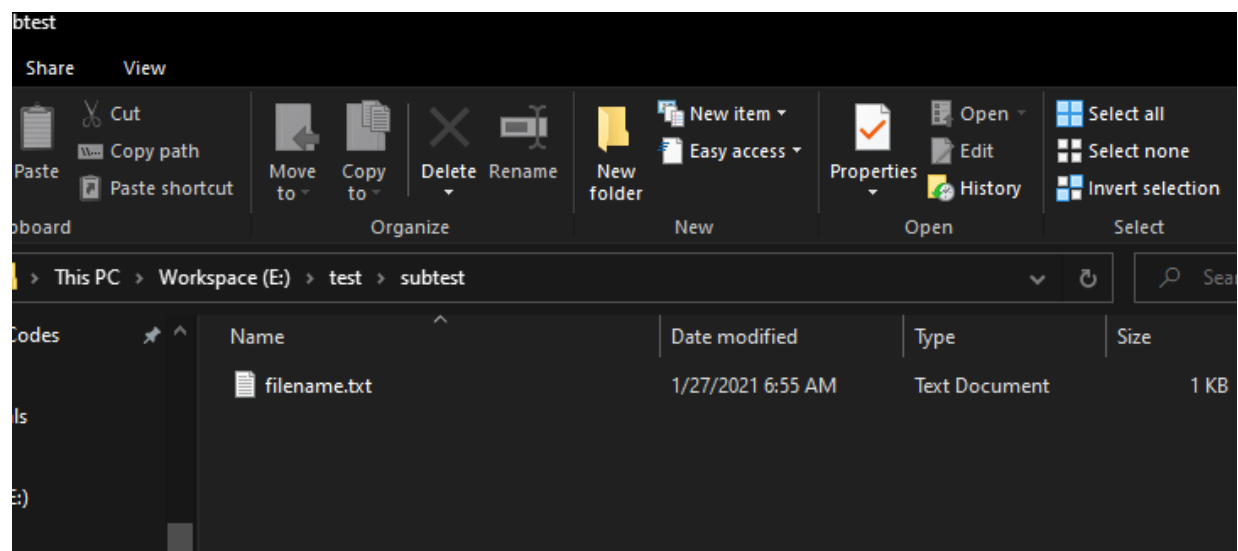
Output:



```

PS E:\AssignmentCodes\Java_Practicals> java exp6_2
E:\test
E:\test\subtest
E:\
E:\test
E:\
E:\
PS E:\AssignmentCodes\Java_Practicals>

```



Conclusion: In this experiment, we performed various File Read/write operations like editing csv files, copying/moving/deleting files, etc using FileWriter, InputStream, OutputStream, etc.