

**(Established under the Presidency University Act, 2013 of the Karnataka Act 41 of 2013)**

## CSA2022 – Advanced JAVA Programming LAB SHEET - 7

**Module 2 – Input Output Operations in Java**

**LAB SHEET - 7**

**Q2**. Write a Java program to perform the following operations with text files.

1. Display the number of characters, sentences and words present in a text file multithread.txt.
2. Display the content of the file on the screen with line number before each line.

**Solution :**

**import** java.io.\*;

**public** **class** Test {

**public** **static** **void** main(String[] args) throws Exception {

File file = **new** File("linus\\multithread.txt");

FileReader fr = **new** FileReader(file);

BufferedReader br = **new** BufferedReader(fr);

String line;

**int** wordCount = 0;

**int** characterCount = 0;

**int** sentenceCount = 0;

**int** linecount = 0;

**while** ((line = br.readLine()) != **null**) {

characterCount += line.length();

String words[] = line.split("\\s+");

wordCount += words.length;

String sentence[] = line.split("[!?.:]+");

sentenceCount += sentence.length;

System.***out***.println(++linecount +" "+line);

}

System.***out***.println("Total word count = "+ wordCount);

System.***out***.println("Total number of sentences = "+ sentenceCount);

System.***out***.println("Total number of characters = "+ characterCount);

}

}

**Operations with Binary files**

**Q3**: Write a Java program to perform the following operations with binary files.

1. Create a folder , section name as folder name in c drive by passing the folder name at run time using Scanner.
2. Create a file to write about students those who submitted above essay , file name must be “sectionname\_students.dat”.
3. Add the content to the above file as follows:

The first line is the header line, the remaining lines corresponds to rows in the table, The elements are separated by spaces.

Name Regdno Essaysubmitted

Irfan 123 yes

Manoj 124 yes

Pavan 126 no

1. Read the above file to console

**Aim:** The aim of this program is to create a folder with a name provided by the user at runtime. It checks whether the folder already exists and, if not, attempts to create it.

**Algorithm:**

1. Import the necessary libraries, including **java.io.\*** and **java.util.Scanner**.
2. Create a **CreateFolder** class with the **main** method.
3. Inside the **main** method: a. Create a **Scanner** object, **sc**, to take user input.

b. Prompt the user to enter a folder name.

c. Read the folder name entered by the user using **sc.next()**.

d. Create a **File** object, **f**, with the provided folder name.

e. Check if the folder already exists and is a directory using **exists()** and **isDirectory()**. If it does, print "already exist."

f. If the folder does not exist, attempt to create it using **mkdir()**. If creation is successful, print "successfully created."

g. If the folder cannot be created, print "cannot be created."

**Program Explanation:**

* The program is designed to allow users to create a folder with a name they enter at runtime.
* It begins by creating a **CreateFolder** class with a **main** method.
* Inside the **main** method, a **Scanner** object is created to facilitate user input.
* The program prompts the user to enter a folder name and reads the input using **sc.next()**.
* A **File** object, **f**, is created with the provided folder name.
* The program checks whether the folder already exists and is a directory. If it does, it prints "already exist."
* If the folder does not exist, the program attempts to create it using **mkdir()**. If the folder is successfully created, it prints "successfully created."
* If the folder cannot be created for any reason, it prints "cannot be created."

This program allows users to create folders on their system by providing a folder name, checks if the folder already exists, and provides feedback on the folder creation process.

Top of Form

**Program:**

a) Creating folder

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** CreateFolder

{

**public** **static** **void** main(String arr[])

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter folder name");

String foldername=sc.next();

File f=**new** File(foldername);

**if**(f.exists()&&f.isDirectory())

System.***out***.println("already exist");

**else** **if**(f.mkdir()==**true**)

System.***out***.println("successfully created");

**else**

System.***out***.println("cannot be created");

}

}

Output:

Enter folder name

5BCA2

Successfully created

**Program:**

**Aim:** The aim of this program is to create a data file with a user-provided filename within a specified folder. It writes a header containing specific information to the file.

**Algorithm:**

1. Import the necessary libraries, including **java.io.\*** and **java.util.Scanner**.
2. Create a **Main** class with the **main** method.
3. Inside the **main** method: a. Create a **Scanner** object, **sc**, to take user input.

b. Prompt the user to enter the file name and folder name.

c. Read the file name and folder name entered by the user using **sc.next()**.

d. Create a **FileOutputStream**, **output**, to write data to the file. The file is located in the specified folder and is named according to the user-provided file name.

e. Print "file created" to indicate that the file has been successfully created.

f. Create a header string containing the information "Name\tRegdno\tEssaysubmitted."

g. Convert the header string to a byte array, **arrheader**, using **getBytes()**.

h. Write the byte array to the file using the **write** method.

i. Print "file written" to indicate that the header has been written to the file.

j. Close the **FileOutputStream** to save the changes and release resources.

**Program Explanation:**

* This program allows users to create a data file within a specified folder by providing a file name and folder name.
* The program starts by creating a **Main** class with a **main** method.
* Inside the **main** method, a **Scanner** object is created for user input.
* The program prompts the user to enter a file name and a folder name. It reads the provided names using **sc.next()**.
* A **FileOutputStream** named **output** is created to write data to the file. The file is located in the specified folder and is named according to the provided file name.
* "file created" is printed to indicate that the file has been successfully created.
* A header string is defined, which contains the information "Name\tRegdno\tEssaysubmitted."
* The header string is converted to a byte array, **arrheader**, using **getBytes()**.
* The byte array is written to the file using the **write** method.
* "file written" is printed to indicate that the header has been successfully written to the file.
* Finally, the **FileOutputStream** is closed to save the changes and release resources.

Top of Form

b) Creating file

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String args[]) **throws** Exception {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter file name and folder name");

String filename=sc.next();

String foldername=sc.next();

FileOutputStream output = **new** FileOutputStream(foldername+"\\"+filename+".dat");

System.***out***.println("file created");

String header = "Name\tRegdno\tEssaysubmitted";

**byte**[] arrheader=header.getBytes();

output.write(arrheader);

System.***out***.println("file written");

output.close();

}}

Output:

Enter filename and folder name

5BCA2\_students.dat

5BCA2

**Aim:** The aim of this program is to create a data file named "5BCA2\_student.dat" and allow the user to input student records with details such as name, registration number, and submission status. The program appends these records to the file.

**Algorithm:**

1. Import the necessary libraries, including **java.io.\*** and **java.util.Scanner**.
2. Create a **Main** class with the **main** method.
3. Inside the **main** method: a. Create a **Scanner** object, **sc**, to take user input.

b. Create a **FileOutputStream**, **output**, to write data to the file. The file is named "5BCA2\_student.dat" and is located in the "5BCA2" folder. The **true** parameter indicates that data should be appended to the file if it already exists.

c. Define variables to store student record details: **name** (String), **regdno** (int), and **submitted** (char).

d. Create a **do-while** loop to allow the user to enter multiple student records.

e. Inside the loop:

* + Prompt the user to enter the student's name, registration number (as an integer), and submission status (as 'y' or 'n').
  + Read and store these details in the corresponding variables.

f. Create a string **newrecord** that combines the student's details in a formatted manner with tab separators.

* + Convert **newrecord** to a byte array, **arrheader**, using **getBytes()**.

g. Write the **arrheader** byte array to the file, which appends the new record to the existing data.

h. Prompt the user to enter 'y' to input more student records or 'n' to stop the process. Read and store the user's choice in the variable **ch**.

i. Continue the loop as long as **ch** is not equal to 'n' (indicating more records should be added).

j. Print "file written" to indicate that the records have been written to the file.

k. Close the **FileOutputStream** to save the changes and release resources.

**Program Explanation:**

* This program allows the user to input student records, including name, registration number, and submission status, and appends these records to a data file.
* The program starts by creating a **Main** class with a **main** method.
* A **Scanner** object, **sc**, is created for user input.
* A **FileOutputStream** named **output** is created to write data to the file "5BCA2\_student.dat" in the "5BCA2" folder. The **true** parameter indicates that data should be appended to the file if it already exists.
* Variables for student record details, including **name**, **regdno**, and **submitted**, are defined.
* The program uses a **do-while** loop to allow the user to input multiple student records. Inside the loop:
  + The user is prompted to enter the student's details, which are read and stored in the respective variables.
  + A formatted string, **newrecord**, is created, combining the student's details with tab separators.
  + **newrecord** is converted to a byte array, **arrheader**, using **getBytes()**, and written to the file to append the new record.
  + The user is prompted to enter 'y' to input more records or 'n' to stop.
* The loop continues as long as the user enters 'y'.
* After the loop, "file written" is printed to indicate that the records have been written to the file.
* The **FileOutputStream** is closed to save the changes and release resources.

Top of Form

**Aim:** The aim of this program is to create and update a data file named "5BCA2\_student.dat" in the "5BCA2" folder. It allows the user to input student records, including their name, registration number, and submission status, and appends these records to the existing file.

**Algorithm:**

1. Import the necessary libraries, including **java.io.\*** and **java.util.Scanner**.
2. Create a **Main** class with the **main** method.
3. Inside the **main** method: a. Create a **Scanner** object, **sc**, to take user input.

b. Create a **FileOutputStream**, **output**, to write data to the file. The file is named "5BCA2\_student.dat" and is located in the "5BCA2" folder. The **true** parameter indicates that data should be appended to the file if it already exists.

c. Define variables to store student record details: **name** (String), **regdno** (int), and **submitted** (char).

d. Create a **do-while** loop to allow the user to enter multiple student records.

e. Inside the loop:

* + Prompt the user to enter the student's name, registration number (as an integer), and submission status (as 'y' or 'n').
  + Read and store these details in the corresponding variables.

f. Create a string **newrecord** that combines the student's details in a formatted manner with tab separators.

g. Append **newrecord** to the existing **record** string, separated by newline characters.

h. Convert the updated **record** string to a byte array, **arrheader**, using **getBytes()**.

i. Write the **arrheader** byte array to the file, which appends the new record to the existing data.

j. Prompt the user to enter 'y' for more student records or 'n' to stop. Read and store the user's choice in the variable **ch**.

k. Continue the loop as long as **ch** is not equal to 'n' (indicating more records should be added).

l. Print "file written" to indicate that the records have been written to the file.

m. Close the **FileOutputStream** to save the changes and release resources.

**Program Explanation:**

* This program allows users to input student records, including name, registration number, and submission status, and appends these records to a data file.
* The program starts by creating a **Main** class with a **main** method.
* A **Scanner** object, **sc**, is created for user input.
* A **FileOutputStream** named **output** is created to write data to the file "5BCA2\_student.dat" in the "5BCA2" folder. The **true** parameter indicates that data should be appended to the file if it already exists.
* Variables for student record details, including **name**, **regdno**, and **submitted**, are defined.
* The program uses a **do-while** loop to allow the user to input multiple student records. Inside the loop:
  + The user is prompted to enter the student's details, which are read and stored in the respective variables.
  + A string **newrecord** is created, combining the student's details with tab separators.
  + **newrecord** is appended to the existing **record** string, separated by newline characters.
  + The updated **record** string is converted to a byte array, **arrheader**, using **getBytes()**.
  + The **arrheader** byte array is written to the file, which appends the new record to the existing data.
  + The user is prompted to enter 'y' for more records or 'n' to stop.
* The loop continues as long as the user enters 'y'.
* After the loop, "file written" is printed to indicate that the records have been written to the file.
* The **FileOutputStream** is closed to save the changes and release resources.

Top of Form

**Program:**

c) Add content to file

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String args[]) **throws** Exception {

Scanner sc = **new** Scanner(System.***in***);

FileOutputStream output = **new** FileOutputStream("5BCA2\\5BCA2\_student.dat",**true**);

String record=**new** String();

String name;

**int** regdno;

**char** submitted,ch='n';

**do** {

System.***out***.println("Student Name , regdno in integer, submitted or not (y/n)");

name=sc.next();

regdno=sc.nextInt();

submitted=sc.next().charAt(0);

String newrecord=record.concat("\n").concat(name).concat("\t").concat(String.*valueOf*(regdno)).concat("\t\t").concat(String.*valueOf*(submitted));

**byte**[] arrheader=newrecord.getBytes();

output.write(arrheader);

System.***out***.println("enter y for more students and n to stop");

ch=sc.next().charAt(0);

}**while**(ch!='n');

System.***out***.println("file written");

output.close();

}}

Output:

Student Name , regdno in integer, submitted or not (y/n)

Irfan 123 y

enter y for more students and n to stop

y

Student Name , regdno in integer, submitted or not (y/n)

Manoj 124 y

enter y for more students and n to stop

y

Student Name , regdno in integer, submitted or not (y/n)

Pavan 125 n

enter y for more students and n to stop

n

**Aim:** The aim of this program is to read and display the contents of a binary file named "5BCA2\_student.dat" located in the "5BCA2" folder.

**Algorithm:**

1. Create a **ReadBinaryFile** class with the **main** method.
2. Inside the **main** method: a. Create a **File** object, **file**, to represent the binary file. The file is located in the "5BCA2" folder.

b. Create a **FileInputStream**, **fis**, to read data from the binary file.

c. Create a **BufferedReader** named **br** with an **InputStreamReader** to read lines from the binary file.

d. Create a **String** variable, **line**, to store each line read from the binary file.

e. Use a **while** loop to read lines from the binary file until the end of the file is reached.

* + Inside the loop, read each line using **br.readLine()** and store it in the **line** variable.
  + Print the **line** to the console.

1. Close the **BufferedReader** and the **FileInputStream** to release resources.

**Program Explanation:**

* This program is designed to read and display the contents of a binary file named "5BCA2\_student.dat" located in the "5BCA2" folder.
* The program starts by creating a **ReadBinaryFile** class with the **main** method.
* Inside the **main** method:
  + A **File** object, **file**, is created to represent the binary file. The file is located in the "5BCA2" folder.
  + A **FileInputStream**, **fis**, is created to read data from the binary file.
  + A **BufferedReader** named **br** is created with an **InputStreamReader** to read lines from the binary file.
  + A **String** variable, **line**, is created to store each line read from the binary file.
  + A **while** loop is used to read lines from the binary file until the end of the file is reached. Inside the loop:
    - Each line is read using **br.readLine()** and stored in the **line** variable.
    - The **line** is printed to the console.
* After reading and displaying all the lines from the binary file, the program closes the **BufferedReader** and the **FileInputStream** to release resources.

Top of Form

**Program:**

d) Reading File to console

**import** java.io.\*;

**public** **class** ReadBinaryFile {

**public** **static** **void** main(String[] args) **throws** Exception {

File file = **new** File("5BCA2\\5BCA2\_student.dat");

FileInputStream fis = **new** FileInputStream(file);

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(fis));

String line;

**while** ((line = br.readLine()) != **null**) {

System.***out***.println(line);

}

}

}

Output:

Name Regdno Essaysubmitted

Irfan 123 y

Manoj 124 y

Pavan 125 n