Day 11 - 21st June 2025

File Handling

Streams

========================================================================

File handling:

=======================================================================

Task 1:

Run the below code and see the file with the given name created or not..

Run it again with “I like India” instead of “I love India”.. And see the file …

Import [java.io](http://java.io).\*;

Import java.io.IOException;

public class WriteByte

{

public static void main(String args[])

{

File f1=new File(“FileName01.txt”); \\ to create new file

FileOutputStream outfile = null;

byte Text[] = {'I',’ ‘,’'L','O','V','E',’ ‘,'I','N','D','I’,’A'};

try

{

outfile = new FileOutputStream(f1);

outfile.write(Text);

}

catch(IOException e)

{

System.out.println(e);

System.exit(-1);

}

System.out.println("Write Byte");

System.out.println("Thank You...!!!");

}

}

Task 2:

Try this code to see the output …

**Write a program which reads byte from file.**

import java.io.\*;

public class ReadingByte

{

public static void main(String args[])

{

FileInputStream infile = null;

int b;

try

{

infile = new FileInputStream("FileName01.txt");

while((b = infile.read()) != -1)

{

System.out.println((char)b);

}

infile.close();

}

catch(IOException e)

{

System.out.println("Sorry..!! File Not Found...!!!");

}

}

}

Task 3:

Taking input from the user and writing on the file…

Create  a file and see the output…

import java.io.\*;

import java.util.\*;

public class WriteByte\_1

{

public static void main(String args[]) {

FileOutputStream outfile = null;

//String s=args[0]; // to input string from command line Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

byte b1[] = s.getBytes();

try

{

outfile = new FileOutputStream("FileName02.txt");

outfile.write(b1);

}

catch(IOException e)

{

System.out.println(e);

System.exit(-1);

}

System.out.println("Write Byte");

System.out.println("Thank You...!!!");

}

}

=========================================

**Reading/writing characters**

=========================================

**FileReader** and **FileWriter**

**Task 4:**

**Write a program which creates file and writes character into that file.**

import java.io.\*;

Class CharacterWrite {

public static void main(String args[]) {

File f1=new File("FileName03.txt");

FileWriter fw = null;

try {

fw=new FileWriter(f1);

fw.write("ahmedabad \n");

fw.write(" baroda \n");

fw.close();

}

catch(FileNotFoundException e)

{

System.out.println("Sorry..!! File Not Found...!!!");

}

catch(IOException e)

{

System.out.println(e.getMessage());

}

System.out.println(“ write operation done!!”);

}

}

11.56 to 12.00

Task 5:

**Write a program which reads character from file.**

import java.io.\*;

Class Readchar

{

public static void main(String args[])

{

FileReader fr =null;

try

{

fr = new FileReader("FileName03.txt");

int ch;

while((ch = fr.read()) != -1)

{

System.out.print((char)ch);

}

System.out.println("Reading complete");

fr.close();

}

catch(FileNotFoundException e)

{

System.out.println("Sorry..!! File Not Found...!!!");

}

catch(IOException e)

{

System.out.println(e.getMessage());

}

}

}

Note:

Task 4 and task 5 talks about File Reader and Writer classes which takes characters and display characters directly.. Unlike FileOUtputstream and File Output stream (which takes bytes).

By 12.05 complete Task 5 also..

—----------------------------------------------

Task 6:

**Write a program to read one byte at a time from a file and copy it into another  file immediately**.

import java.io.\*;

Class CopyByte

{

public static void main(String args[])

{

try

{

byte b=0;

FileInputStream infile = new FileInputStream("NewFile01.txt");

FileOutputStream outfile = new FileOutputStream("NewFile05.txt");

Initialize byteread here….

while(byteread != -1)

{

b = (byte)infile.read();

outfile.write(b);

}

System.out.println("Byte Copied From in.txt to out.txt FIle ");

}

catch(FileNotFoundException e)

{

System.out.println("Sorry..!! File Not Found...!!!");

}

catch(IOException e)

{

System.out.println(e.getMessage());

}

}

}

Task 7:

Merging two files to 3rd file..

**Write a program to merge two files in third file.**

import java.io.\*;

classFileMergeDemo

{

public static void main(String args[])

{

try

{

FileInputStream file1 = new FileInputStream("NewFile01.txt");

FileInputStream file2 = new FileInputStream("NewFile02.txt");

SequenceInputStream file3 = new SequenceInputStream(file1, file2);

BufferedInputStream br1 = new BufferedInputStream(file3);

BufferedOutputStream br2 = new BufferedOutputStream(System.out);

int ch;

while((ch = br1.read())!=-1)

{

br2.write((char)ch);

}

br1.close();

br2.close();

file1.close();

file2.close();

System.out.println("Merge Two File Sucessfully ");

}

catch(IOException e)

{

System.out.println("Sorry..!! File Not Found...!!!");

}

}

}

—-------------------------------------

Task 8:

**Write an application to rename a file. Use the renameTo() method of File to  accomplish**

/\*this task. The first command line argument is the old filename and the second is  the newfilename.

\*/

import java.io.\*;

classFileRenameDemo

{

public static void main(String args[])

{

File f1 = new File(args[0]);

File f2 = new File(args[1]);

f1.renameTo(f2);

System.out.println("Rename File " +f1+" To "+f2+" Sucessfully "); }

}

Output :

Javac FileRenameDemo.java

Java FileRenameDemo input1.txt abc.txt

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Streams

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Lambda Expressions:

Task 9

import java.lang.FunctionalInterface;

// this is functional interface

@FunctionalInterface

interface MyInterface{

    // abstract method

    double getPiValue();

}

public class Main {

    public static void main( String[] args ) {

    // declare a reference to MyInterface

    MyInterface ref;

    // lambda expression

    ref = () -> 3.1415;

    System.out.println("Value of Pi = " + ref.getPiValue());

    }

}

Task 10:

Write a code to reverse  a string..

15.20 to 15.23

Task 11:

@FunctionalInterface

interface MyInterface {

    // abstract method

    String reverse(String n);

}

public class Main {

    public static void main( String[] args ) {

        // declare a reference to MyInterface

        // assign a lambda expression to the reference

        MyInterface ref = (str) -> {

            String result = "";

            for (int i = str.length()-1; i >= 0 ; i--)

            result += str.charAt(i);

            return result;

        };

        // call the method of the interface

        System.out.println("Lambda reversed = " + ref.reverse("Lambda"));

    }

}

Task 12:

Wap to create an arraylist with 5 friends names..

3 min 15.40 to 15.43

Task 13:

Wap to create a List of 5 friends names (first name and last name)

3 min 15.45 to 15.48

HInt 👍

static List<String> fullName = new ArrayList<>();

fullName.add(“Fname, LName”);

Task 14:

import java.util.ArrayList;

import java.util.List;

public class StreamMain {

    // create an object of list using ArrayList

    static List<String> places = new ArrayList<>();

    // preparing our data

    public static List getPlaces(){

        // add places and country to the list

        places.add("Nepal, Kathmandu");

        places.add("Nepal, Pokhara");

        places.add("India, Delhi");

        places.add("USA, New York");

        places.add("Africa, Nigeria");

        return places;

    }

    public static void main( String[] args ) {

        List<String> myPlaces = getPlaces();

        System.out.println("Places from Nepal:");

        // Filter places from Nepal

        myPlaces.stream()

                .filter((p) -> p.startsWith("Nepal"))

                .map((p) -> p.toUpperCase())

                .sorted()

                .forEach((p) -> System.out.println(p));

    }

}

Task 15:

Collect : Terminal operator..

Wap to accept or create a list of 5 integers and display the squares of each ..

.

Hint:

List<Integer> squareofNums = numbers.stream()

.map(num->num\*num)

.collect(Collectors.toList());

6 min 16.29 to 16.35

16.42 to 16.58 break

Task 16

Write a code to create an array list and filter the values which are odd numbers and display them..

Hint:

List<Integer> addNumbers = numbers.stream()

.filter(num -> num % 2 !=0)

.collect(Collectors.toList());

17.01 to 17.06 5 min

Task 17:

Wap to create an array list to remove duplicate values from the List.

Hint:

List<Integer> RemovDups= numbers.stream()

.distinct()

.collect(Collectors.toList());

Task 18:

Wap to run a loop / iterate()  and limit it to 20 values (1 to 2)

While displaying use for each to limit till 10 numbers.

Hint:

Stream<Integers> nums = Stream

.iterate(1, n -> n+1)

.limit(20);

Nums

.limit(10)

.foreach(System.out::println);

Task 19:

Wap to create an array List skip 15 numbers and print the output using foreach loop

HInt:

Stream<Integers> nums = Stream

.iterate(1, n -> n+1)

.limit(20);

Stream<Integer> SkipNums = nums.skip(15);

Nums.foreach(System.out::println);

17.15   ===> complete plz

Juz to know:

Terminal operators are:

Collect

Foreach

Reduce

Understand and study difference between mutable and immutable

Task 20

Mutable ⇒ changeable

Int

Collect ()

Immutable ⇒ cannot be changed

Wrapper classes–  Integer, Long ,

reduce() → will not displays more than one values —>immutable

Collect —-> displays more than one values —> mutable

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

public class ReduceExample {

    public static void main(String[] args) {

        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

        Optional<Integer> sum = numbers.stream().reduce((x, y) -> x + y);

        System.out.println("Sum of all elements: " + sum.orElse(0));

        Optional<Integer> max = numbers.stream().reduce(Integer::max);

        System.out.println("Maximum element: " + max.orElse(0));

        List<String> strings = Arrays.asList("Hello", " ", "world", "!");

        Optional<String> concatenatedString = strings.stream().reduce((x, y) -> x + y);

        System.out.println("Concatenated string: " + concatenatedString.orElse(""));

    }

}

When to use reduce and when to use collect..

Reduce will be used if you are expecting a single result from the stream (eg min, max , sum, product…)

Collect will be used if you are excepting a list of values… (list, set, map)

Home tasks:

==================================================

File Handling add ons

===================================================================================================

Buffered reader and writer — for large files to be read.

==================================================

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ReadFileExample {

    public static void main(String[] args) {

        try (BufferedReader br = new BufferedReader(new FileReader("largefile.txt"))) {

            String line;

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

                System.out.println(line);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

non buffered

import java.io.FileReader;

import java.io.IOException;

public class NonBufferedReaderExample {

    public static void main(String[] args) {

        try (FileReader fr = new FileReader("largefile.txt")) {

            int ch;

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

                System.out.print((char) ch);

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

Buffered Writer

import java.io. BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

public class BufferedWriterDemo {

   public static void main(String[] args) {

      String filePath = "example.txt";

      String content = "Hello, World!\nThis is a BufferedWriter example.";

      // Initialize BufferedWriter with a FileWriter

      try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {

         // Write content to the file

         writer.write(content);

         System.out.println("Content written to file.");

      } catch (IOException e) {

         System.err.println("An error occurred: " + e.getMessage());

      }

   }

}

Add ons — Java 8 features  nio

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

public class ImageCopier {

    public static void main(String[] args) {

        // Define the source and destination file paths

        Path sourcePath = Paths.get("path/to/your/source\_image.jpg"); // Replace with your source image path

        Path destinationPath = Paths.get("path/to/your/destination\_image.jpg"); // Replace with your desired destination path

        try {

            // Copy the file

            // REPLACE\_EXISTING option overwrites the destination file if it already exists

            Files.copy(sourcePath, destinationPath, StandardCopyOption.REPLACE\_EXISTING);

            System.out.println("Image copied successfully from " + sourcePath + " to " + destinationPath);

        } catch (IOException e) {

            System.err.println("Error copying image: " + e.getMessage());

            e.printStackTrace();

        }

    }

}

========================================================================

**Info Box**

Data structures -  a few codes - for reference

<https://drive.google.com/drive/folders/1OXACrFwF5hQ2WVSHh2gP1EZDF_PcEy_U?usp=sharing>

Java core recording — only for beginners

<https://drive.google.com/drive/folders/1j8dVg_RjjnDAayAbCxZFWMewCH99WHnq?usp=sharing>

Next week's topics: plz be prepared..

|  |  |
| --- | --- |
| Data Structure | Section Overview, What Is A Data Structure?, How Computers Store Data, Data Structures In Different Languages, Operations On Data Structures, |
| Data Structures: Arrays | Arrays Introduction, Static vs Dynamic Arrays, Implementing An Array, Strings and Arrays, Exercise: Reverse A String, Solution: Reverse A String, Exercise: Merge Sorted Arrays, Solution: Merge Sorted Arrays, |
| Hash Tables | Hash Tables Introduction, Hash Function, Hash Collisions, Hash Tables In Different Languages, Exercise: Implement A Hash Table, Solution: Implement A Hash Table, keys(), Extra: keys() Without Collision, Hash Tables VS Arrays, |
| Linked List | Linked Lists Introduction, What Is A Linked List?, Exercise: Imposter Syndrome, Exercise: Why Linked Lists?, Solution: Why Linked Lists?, What Is A Pointer?, Our First Linked List, Solution: append(), Solution: prepend(), Node Class, insert(), Solution: insert(), Solution: remove(), Doubly Linked Lists, Exercise: Doubly Linked Lists, Solution: Doubly Linked Lists, Singly VS Doubly Linked Lists, |

======================================================================