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.File;

import java.io.FileOutputStream;

import java.io.IOException;

public class task001 {

public static void main(String[] args) {

String fileName = "FileName01.txt";

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

try (FileOutputStream outfile = new FileOutputStream(fileName)) {

outfile.write(textBytes);

System.*out*.println("Write Byte " + fileName);

} catch (IOException e) {

// Print a more descriptive error message along with the stack trace

System.*err*.println("An error occurred while writing to the file: " + e.getMessage());

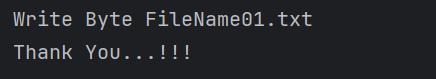
e.printStackTrace();

}

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 task002

{

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)

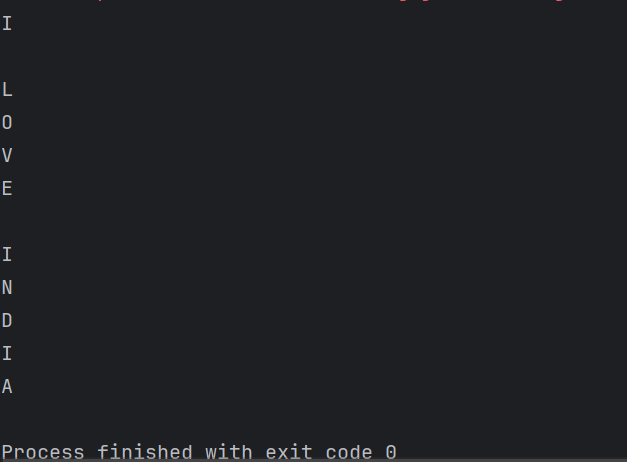
{

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

}

}

}



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 task003 {

public static void main(String args[]) {

FileOutputStream outfile = null;

// Initialize Scanner

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

System.*out*.println("Enter a string to write into the file:");

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...!!!");

// Close the output stream and scanner

try {

if (outfile != null) outfile.close();

sc.close();

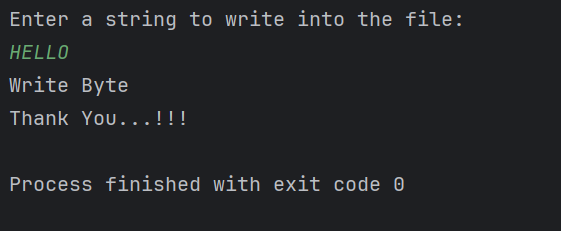
} catch (IOException e) {

System.*out*.println("Error closing resources: " + e);

}

}

}



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

**Reading/writing characters**

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

**FileReader** and **FileWriter**

**Task 4:**

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

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

import java.io.\*;

public class task004 {

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");

} catch (FileNotFoundException e) {

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

} catch (IOException e) {

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

} finally {

try {

if (fw != null) fw.close();

} catch (IOException e) {

System.*out*.println("Error while closing FileWriter: " + e.getMessage());

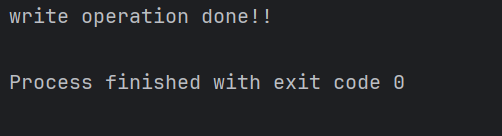
}

}

System.*out*.println("write operation done!!");

}

}



Task 5:

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

import java.io.\*;

public class task005 {

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("\nReading complete");

} catch (FileNotFoundException e) {

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

} catch (IOException e) {

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

} finally {

try {

if (fr != null) fr.close();

} catch (IOException e) {

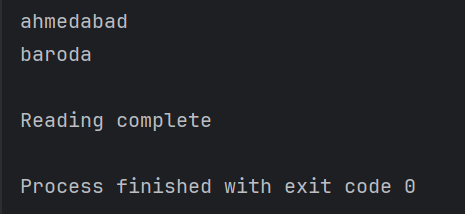
System.*out*.println("Error while closing reader: " + e.getMessage());

}

}

}

}



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.\*;

public class task006 {

public static void main(String args[]) {

FileInputStream infile = null;

FileOutputStream outfile = null;

try {

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

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

int byteread; // Declare and initialize read variable

while ((byteread = infile.read()) != -1) {

outfile.write(byteread); // Write the read byte

}

System.*out*.println("Byte Copied From FileName01.txt to NewFile05.txt");

} catch (FileNotFoundException e) {

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

} catch (IOException e) {

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

} finally {

try {

if (infile != null) infile.close();

if (outfile != null) outfile.close();

} catch (IOException e) {

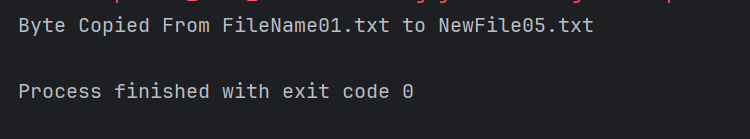
System.*out*.println("Error while closing streams: " + e.getMessage());

}

}

}

}



Task 7:

Merging two files to 3rd file..

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

import java.io.\*;

class task007 {

public static void main(String args[]) {

try {

FileInputStream file1 = new FileInputStream("FileName02.txt");

FileInputStream file2 = new FileInputStream("NewFile05.txt");

BufferedOutputStream br2 = new BufferedOutputStream(new FileOutputStream("MergedFile.txt"));

int ch;

// Write content of file1

while ((ch = file1.read()) != -1) {

br2.write(ch);

}

// Write a newline

br2.write('\n');

// Write content of file2

while ((ch = file2.read()) != -1) {

br2.write(ch);

}

file1.close();

file2.close();

br2.close();

System.*out*.println("Merged two files successfully!");

} catch (IOException e) {

System.*out*.println("Sorry..!! File Not Found or IO Error: " + e.getMessage());

}

}

}

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.

\*/

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

\*/

import java.io.\*;

public class task008 {

public static void main(String args[]) {

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

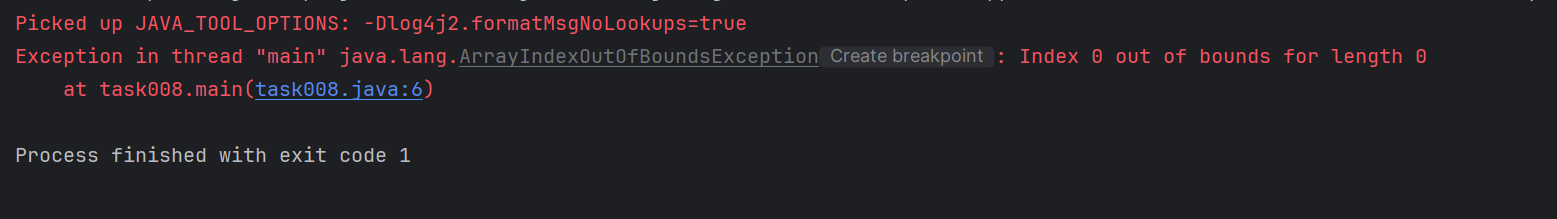
File f2 = new File(args[1]);

f1.renameTo(f2);

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

}

}



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

Streams

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

Lambda Expressions:

Task 9

import java.lang.FunctionalInterface;

// this is functional interface

@FunctionalInterface

interface MyInterface{

// abstract method

double getPiValue();

}

public class task009 {

public static void main(String[] args) {

// declare a reference to MyInterface

MyInterface ref;

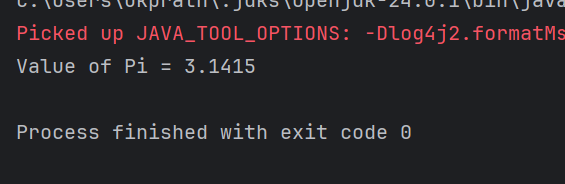
// lambda expression

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

// ReverseStringLambda

import java.util.function.Function;

public class task010{

public static void main(String[] args) {

String input = "lambda";

// Lambda to reverse a string

Function<String, String> reverse = str -> new StringBuilder(str).reverse().toString();

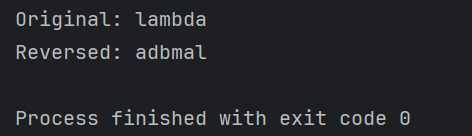
String reversed = reverse.apply(input);

System.*out*.println("Original: " + input);

System.*out*.println("Reversed: " + reversed);

}

}



Task 11:

@FunctionalInterface

interface MyInterface2 {

// abstract method

String reverse(String n);

}

public class reverse {

public static void main( String[] args ) {

// declare a reference to MyInterface

// assign a lambda expression to the reference

MyInterface2 ref = (str) -> {

String result = "";

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

result += str.charAt(i);

return result;

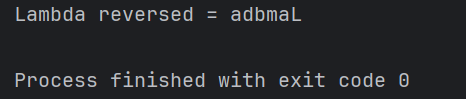
};

// 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

import java.util.ArrayList;

public class task012 {

public static void main(String[] args) {

// Create an ArrayList of String type

ArrayList<String> friends = new ArrayList<>();

// Add 5 friends' names

friends.add("Alice");

friends.add("Bob");

friends.add("Charlie");

friends.add("Diana");

friends.add("Ethan");

// Print the list

System.*out*.println("My Friends:");

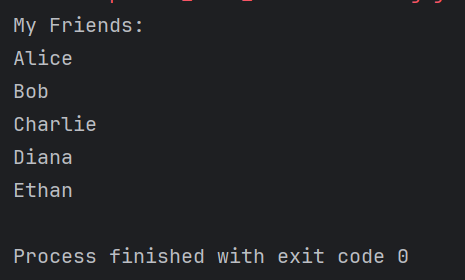
for (String name : friends) {

System.*out*.println(name);

}

}

}



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”);

import java.util.ArrayList;

import java.util.List;

public class task013 {

// Static list to store full names

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

public static void main(String[] args) {

// Add 5 friends' full names

*fullName*.add("Alice Johnson");

*fullName*.add("Bob Smith");

*fullName*.add("Charlie Brown");

*fullName*.add("Diana Miller");

*fullName*.add("Ethan Davis");

// Print full names

System.*out*.println("List of Friends:");

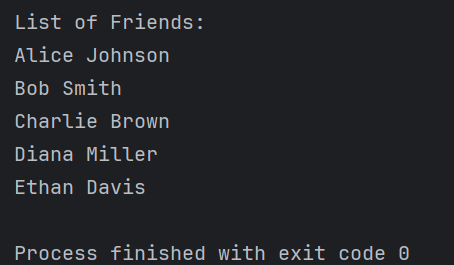
for (String name : *fullName*) {

System.*out*.println(name);

}

}

}



Task 14:

import java.util.\*;

public class task014 {

public static void main(String[] args) {

// Create a List of full names

List<String> friends = new ArrayList<>();

friends.add("Alice, Johnson");

friends.add("Alice, sachin");

friends.add("Bob, Smith");

friends.add("Charlie, Brown");

friends.add("Diana, Miller");

friends.add("Alice, dravid");

friends.add("Ethan, Davis");

System.*out*.println("Friends whose name starts with 'Alice':");

// Filter, convert to uppercase, sort, and print using Stream API

friends.stream()

.filter(name -> name.startsWith("Alice")) // Filter names starting with "Alice"

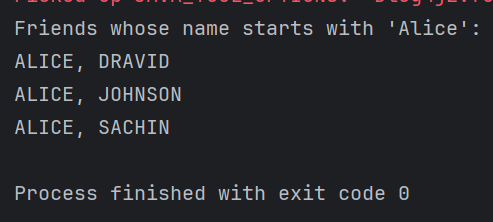
.map(String::toUpperCase) // Convert to uppercase

.sorted() // Sort alphabetically

.forEach(System.*out*::println); // Print each name

}

}



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());

import java.util.\*;

import java.util.stream.Collectors;

public class task015 {

public static void main(String[] args) {

// Step 1: Create a list of 5 integers

List<Integer> numbers = Arrays.*asList*(2, 4, 6, 8, 10);

// Step 2: Use stream to square each number and collect

List<Integer> squareOfNums = numbers.stream()

.map(num -> num \* num)

.collect(Collectors.*toList*());

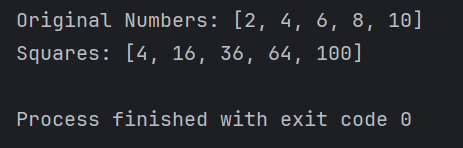
// Step 3: Print results

System.*out*.println("Original Numbers: " + numbers);

System.*out*.println("Squares: " + squareOfNums);

}

}



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());

import java.util.\*;

import java.util.stream.Collectors;

public class task016 {

public static void main(String[] args) {

// Step 1: Create a list of 5 integers

List<Integer> numbers = Arrays.*asList*(2, 3, 5, 8, 9);

// Step 2: Filter only odd numbers using stream

List<Integer> oddNumbers = numbers.stream()

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

.collect(Collectors.*toList*());

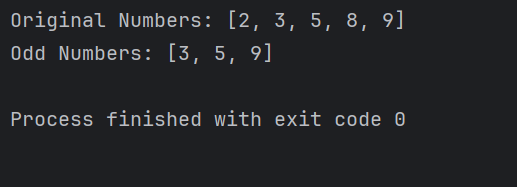
// Step 3: Print results

System.*out*.println("Original Numbers: " + numbers);

System.*out*.println("Odd Numbers: " + oddNumbers);

}

}



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());

import java.util.\*;

import java.util.stream.Collectors;

public class task017 {

public static void main(String[] args) {

// Step 1: Create a list with some duplicates

List<Integer> numbers = Arrays.*asList*(2, 4, 4, 3, 6, 8, 2, 9, 10, 10);

// Step 2: Remove duplicates and filter even numbers

List<Integer> evenNumbers = numbers.stream()

.distinct() // Remove duplicates

.filter(num -> num % 2 == 0) // Keep only even numbers

.collect(Collectors.*toList*()); // Collect result

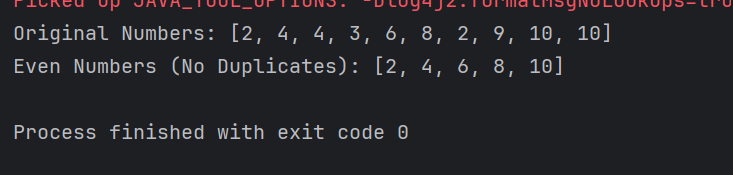
// Step 3: Print result

System.*out*.println("Original Numbers: " + numbers);

System.*out*.println("Even Numbers (No Duplicates): " + evenNumbers);

}

}



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);

import java.util.stream.Stream;

public class task018 {

public static void main(String[] args) {

// Generate numbers from 1 onwards, limit to 20 values

Stream<Integer> nums = Stream

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

.limit(20);

// From those 20, limit to first 10 and print

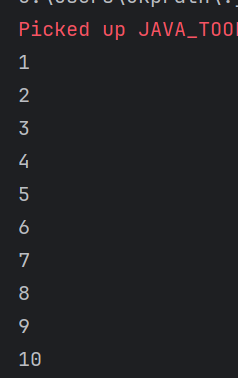
nums

.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);

import java.util.stream.Stream;

public class task019 {

public static void main(String[] args) {

// Generate numbers from 1 to 20

Stream<Integer> nums = Stream

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

.limit(20);

// Skip the first 15 numbers (print 16 to 20)

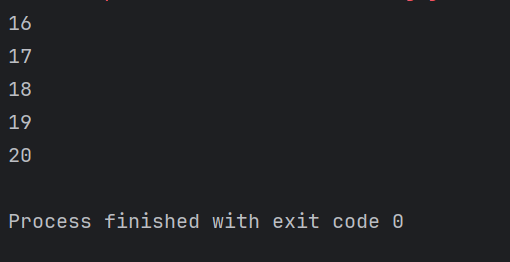
Stream<Integer> skipNums = nums.skip(15);

// Print using forEach

skipNums.forEach(System.*out*::println);

}

}



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;

import java.util.stream.Collectors;

public class task020 {

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));

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

System.*out*.println("Minimum element: " + min.orElse(0));

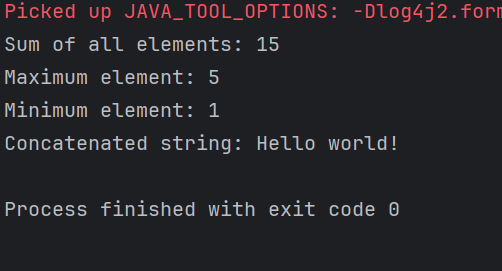
List<String> strings = Arrays.*asList*("Hello", " ", "world", "!");

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)

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

**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>