**Java Sorting and Web Data Handling Lab Documentation**

**Introduction**

This documentation outlines the functionality and usage of two Java programs developed for benchmarking sorting algorithms and handling web data. The programs are designed to showcase the efficiency of sorting algorithms and demonstrate exception handling in networking and file operations.

The first program focuses on benchmarking sorting algorithms, comparing a custom algorithm (either Selection Sort or Insertion Sort) with Java's built-in **Arrays.sort()**. It will generate random integer arrays, calculate the amount of time taken using both methods, and reports the results.

The second program involves programming with exceptions. It fetches data froma given specified web URL and saves it to a file using **InputStream** and **OutputStream**. The program demonstrates the use of try-catch blocks to handle exceptions such as **MalformedURLException**, **FileNotFoundException**, and **IOException** during the web data retrieval and file I/O processes.

**Part 1: Benchmarking Sorting Algorithms**

**Purpose**

The purpose of this program is to compare the performance of a custom sorting algorithm (either Selection Sort or Insertion Sort) with Java's built-in sorting algorithm (**Arrays.sort()**). The program generates random integer arrays, sorts them using the specified algorithms, and measures the time taken for each sorting process.

**Implementation Details**

* The program has a static size of the arrays.
* Two arrays of random integers are generated with identical content.
* The first array is sorted using a custom sorting algorithm (Selection Sort or Insertion Sort).
* The second array is sorted using **Arrays.sort() method**.
* The time taken for each sorting process is measured and printed.

**Part 2: Programming with Exceptions**

**Purpose**

This program fetches data from a given URL on run time of the web and saves it’s data to a file. It involves networking and file I/O operations, and the program handles checked exceptions using try-catch blocks.

**Implementation Details**

* The program prompts the user to enter a URL and a file name.
* It uses **URL** and **InputStream** to read data from web and **FileOutputStream** to write data to file.
* The program handles exceptions such as **MalformedURLException**, **FileNotFoundException**, and **IOException**.
* The **copyStream** method is used to copy data from the web to the file.
* Streams are closed in a **finally** block to ensure proper resource management.

**Notes and Considerations**

* Ensure an active internet connection when running Part 2 to fetch data from the web.

**Program (Part1)**

**Benchmarking Sorting Algorithms:**

import java.util.Arrays;

public class SortingCompare {

    // Change this constant to easily modify the size of the arrays

    private static final int ARRAY\_SIZE = 1000;

    public static void main(String[] args) {

        // Generate two arrays with random integers

        int[] array1 = generateRandomArray();

        int[] array2 = Arrays.copyOf(array1, array1.length);

        long startTime1 = System.currentTimeMillis();

        // calling selectionSort function with first array

        selectionSort(array1);

        long runTime1 = System.currentTimeMillis() - startTime1;

        // Time duration takes to sort the first array and show the time

        System.out.println("Selection Sort Time: " + runTime1 + " milliseconds");

        // Sort second identical array by using Arrays.sort()

        long startTime2 = System.currentTimeMillis();

        Arrays.sort(array2);

        long runTime2 = System.currentTimeMillis() - startTime2;

        // Time duration takes to sort the first array and show the time

        System.out.println("Arrays.sort() Time: " + runTime2 + " milliseconds");

        // Report the times in comments

        System.out.println("\nArray Size: " + ARRAY\_SIZE);

        System.out.println("Selection Sort Time: " + runTime1 + " milliseconds");

        System.out.println("Arrays.sort() Time: " + runTime2 + " milliseconds");

    }

    // custom Method to generate an array with random integers

    private static int[] generateRandomArray() {

        int[] array = new int[ARRAY\_SIZE];

        for (int i = 0; i < array.length; i++) {

            array[i] = (int) (Integer.MAX\_VALUE \* Math.random());

        }

        return array;

    }

    // Selection Sort implementation

    private static void selectionSort(int[] array) {

        int n = array.length;

        for (int i = 0; i < n - 1; i++) {

            int minIndex = i;

            for (int j = i + 1; j < n; j++) {

                if (array[j] < array[minIndex]) {

                    minIndex = j;

                }

            }

            int temp = array[minIndex];

            array[minIndex] = array[i];

            array[i] = temp;

        }

    }

}

**Program (Part 2)  
Programming with Exceptions**

import java.io.\*;

import java.net.\*;

public class App {

    public static void main(String[] args) {

        // Declare variables to represent InputStream and OutputStream

        InputStream inputStream = null;

        OutputStream outputStream = null;

        try {

            // Read URL and file name from the user

            // example of url \*\* https://www.gutenberg.org/files/11/11-0.txt \*\*

            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

            System.out.print("Enter the URL: ");

            String urlString = reader.readLine();

            System.out.print("Enter the file name with extension: ");

            String fileName = reader.readLine();

            // Create URL object

            URL url = new URL(urlString);

            // Get InputStream from the URL

            inputStream = url.openStream();

            // Get OutputStream for the file

            outputStream = new FileOutputStream(fileName);

            // Copy data from web to file

            copyStream(inputStream, outputStream);

            System.out.println("Download successful!");

        } catch (MalformedURLException e) {

            System.err.println("Invalid URL: " + e.getMessage());

        } catch (FileNotFoundException e) {

            System.err.println("File not found: " + e.getMessage());

        } catch (IOException e) {

            System.err.println("Error reading/writing data: " + e.getMessage());

        } finally {

            // Close streams in the finally block to ensure they are closed

            try {

                if (inputStream != null) {

                    inputStream.close();

                }

                if (outputStream != null) {

                    outputStream.close();

                }

            } catch (IOException e) {

                System.err.println("Error closing streams: " + e.getMessage());

            }

        }

    }

    // Method to copy data from InputStream to OutputStream

    private static void copyStream(InputStream in, OutputStream out) throws IOException {

        int oneByte = in.read();

        while (oneByte >= 0) {

            out.write(oneByte);

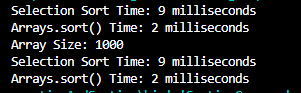
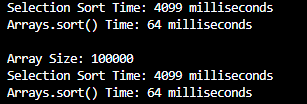
            oneByte = in.read();

        }

    }

}

**Outputs**


**The End**