**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.io.UnsupportedEncodingException;

**import** java.util.ArrayList;

/\*\*

\* QSort Program reads the input data, preferably any number(integer,float,double etc..) from file(s) through command line argument.

\* Sorts the data and stores the result into answer.txt.

\* Delimiter should be ";"

\*

\* **@author** Sanju Kurubara Budi Hall Hiriyanna Gowda

\* **@since** 09-27-2016

\*/

**public** **class** QSort {

**static** ArrayList<String> *list* = **null**;

**public** **static** **void** main(String[] args) **throws** FileNotFoundException, UnsupportedEncodingException {

**if**(args.length == 0)

{

System.***out***.println("Please enter InputFile(s) name");

System.***out***.println("Usage: java (FileName) (Input File 1) (Input File 2 - Optional) (Input File 3 - Optional)");

**return**;

}

**else** **if**(args.length > 3)

{

System.***out***.println("Please enter only 1 or 2 or 3 arguments");

**return**;

}

**else**

{

PrintWriter writer = **new** PrintWriter("answer.txt", "UTF-8");

writer.println("Sorting result:");

**for** (**int** argCount = 0; argCount < args.length; argCount++)

{

BufferedReader reader = **null**;

**try** {

//reading contents from input.txt and storing it in ArrayList

String text = **null**;

String inputFile = args[argCount];

reader = **new** BufferedReader(**new** FileReader(inputFile));

*list* = **new** ArrayList<String>();

**while** ((text = reader.readLine()) != **null**)

{

String[] resultingTokens = text.split(";");

**for** (**int** i = 0; i < resultingTokens.length; i++)

{

**if** (resultingTokens[i].trim().length() != 0)

*list*.add(resultingTokens[i].trim());

}

}

**long** startTime = System.*currentTimeMillis*();

*QuickSort*(0, *list*.size() - 1);

**long** endTime = System.*currentTimeMillis*();

**long** difference = endTime - startTime;

//Print sorted data into answer.txt file

writer.print(""+args[argCount] + " sorted: ");

**for** (**int** i=0; i< *list*.size(); i++)

{

writer.print(*list*.get(i));

**if**(i != *list*.size() - 1)

{

writer.print("; ");

}

}

writer.println();

writer.println();

writer.println("Performance Analysis");

writer.println("Input file Size Sorting Time (in milliseconds)");

writer.println(""+args[argCount]+" "+*list*.size()+" " +difference);

writer.println();

writer.println();

**if** (*list* != **null** && *list*.size() > 0)

{

*list*.clear();

}

}

**catch** (FileNotFoundException e)

{

e.printStackTrace();

}

**catch** (IOException e)

{

e.printStackTrace();

}

**finally**

{

**try**

{

**if** (reader != **null**)

{

reader.close();

}

}

**catch** (IOException e)

{

}

}

}

writer.close();

}

}

**static** **void** QuickSort(**int** start, **int** end)

{

**if** (*list* == **null** || *list*.size() == 0)

**return**;

**if** (start >= end)

**return**;

**int** r = *partition*(start, end);

*QuickSort*(start, r-1);

*QuickSort*(r+1, end);

}

**static** **int** partition(**int** start, **int** end)

{

String pivot = *list*.get(end);

**int** i = start - 1;

**int** j = end;

String tmp;

**while** (i < end)

{

//Using ParseDouble to make sure all the numbers are sortReadme.txted

// Increment i until we find greater element in the list than pivot

**while** (Double.*parseDouble*(*list*.get(++i)) < Double.*parseDouble*(pivot));

// Decrement j until we find lesser element in the list than pivot

**while** (j > 0 && Double.*parseDouble*(*list*.get(--j)) > Double.*parseDouble*(pivot));

**if** (i >= j)

**break**;

**else**

{

tmp = *list*.get(i);

*list*.set(i, *list*.get(j));

*list*.set(j, tmp);

}

}

tmp = *list*.get(i);

*list*.set(i, *list*.get(end));

*list*.set(end, tmp);

**return** i;

}

}

/\*

Readme file contents:

Software Requirements:

---------------------------------------------------

Platform: Windows 10

JDK Version: 1.7

Code Editor: JGrasp

Execution environment: Command Prompt

Brief Description

---------------------------------------------------

Program Design: Sort the input numbers using Quick sort

Algorithm used: Quick Sort

Data Structure: ArrayList, which stores numbers as string data to make sure input data can be integer,float,double etc

Program Works: Input should contain numbers(integer,float, double etc..) separated by (semicolon) or (new line)

Program Fails: Input should not contain Alphabets, special characters except (semicolon), (space) and (new line) characters.

Time Complexity: Best case: O(n\*logn)

Average case: O(n\*logn)

Worst case: O(n\*n)

Space Complexity: O(n)

General Usage Notes:

----------------------------------------------------

1. javac QSort.java

java QSort input1.txt input2.txt(optional) input3.txt(optional)

Note: 1. Path of program execution should be (src folder/ location of QSort.java) file.

2. Input files should contain numbers separated by ";"

3. answer.txt(output file) will be generated in default location i.e. the same folder where QSort.java file can be found

4. Performance analysis i.e. time taken to execute QuickSort program will be displayed after sorting the each file.

Contact Information:

-----------------------------------------------------

Name: Sanju Kurubara Budi Hall Hiriyanna Gowda

Student Id: 800953525

Email Id: skurubar@uncc.edu

Contact No: 9802988158

-----------------------------------------------------

\*/