Dylan Carlson

1161653

Lab104

09/27/18

**Recursion Class:**

/\*\*

\* This class, Recursion, includes multiple recursive methods

\* used by the client.

\*

\* @author dylca

\*/

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Recursion {

/\*\*

\* This method is passed a number and returns the sum

\* of reciprocals from 1 to n.

\*

\* @param num

\* @return

\*/

public double sumOfReciprocals(double num){

if (num == 1)

return 1.0;

return 1.0/num + sumOfReciprocals(num-1.0);

}

/\*\*

\* isabel() implements Isabel's technique. The goal is to

\* divide the array into two and add up the sums of the given sides,

\* and it finally returns the total sum of the array.

\* The array must be a power of 2.

\*

\* @param a

\* @return

\*/

public int isabel( int a[] ){

int[] b;

int total;

if ( a.length == 1) //Base case

return a[0];

else {

b = new int[ (a.length/2) ];

for (int i = 0; i< a.length/2; i++){

b[i] = a[2\*i] + a[2\*i + 1];

}

total = isabel( b );

}

return total;

}

// C:\Users\dylca\Desktop\Lab104\_Isabel.txt

/\*\*

\* loadDataFile() is used first before Isabel's technique. This is because

\* the size must be determined, and the numbers from the data file

\* must be copied to an array if it is a power of 2. It returns the sum

\* of Isabel's technique, and 0 if it is not a power of 2.

\*

\*

\* @param path

\* @return

\* @throws FileNotFoundException

\*/

public int loadDataFile( String path ) throws FileNotFoundException{

int answer = 0;

File data = new File( path );

Scanner scan;

scan = new Scanner( data );

Scanner scan2;

scan2 = new Scanner(data);

int size = 0;

while( scan.hasNext() ){

if( scan.hasNextInt() )

size++;

scan.next();

}

System.out.println("Size of Array = " + size);

if( powerOfTwo(size) ){

int[] contents = new int[size];

System.out.println("Numbers in array: ");

//data file into an array

int i = 0;

while( scan2.hasNext() ){

if( scan2.hasNextInt() ){

contents[i] = scan2.nextInt();

System.out.println( contents[i] );

i++;

}

else

scan2.next();

}

answer = isabel(contents);

}

return answer;

}

/\*\*

\* This method is used to determine if the array is a power

\* of 2, by taking the size and recursively determining if it is a power

\* of 2.

\*

\* @param n

\* @return

\*/

public boolean powerOfTwo( int n ){

boolean truth = false;

if (n == 2){

truth = true;

}

else if (n%2 == 0){

truth = powerOfTwo(n/2);

}

else if (n%2 == 1)

truth = false;

return truth;

}

/\*\*

\* list() prints out all of the files and directories

\* and files within those directories of a given path.

\*

\* @param path

\*/

public void list( String path ){

File file = new File(path);

if( file.isFile() ) //Base case

System.out.println( file );

if( file.isDirectory() ){

File[] a = file.listFiles();

if( a != null)

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

list( a[i].getPath() );

}

}

}

}

**Client Class:**

/\*\*

\* This client prompts the user using a JOptionPane and calls the

\* recursive methods from the Recursion class. It is fairly flexible

\* as to what the user enters for specific methods.

\*

\* @author Dylan Carlson

\*/

import java.io.FileNotFoundException;

import static java.lang.Double.parseDouble;

import javax.swing.JOptionPane;

public class Client {

public static void main(String[] args){

try{

menu();

}

catch (FileNotFoundException fnfe){

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

}

}

/\*\*

\* This is the main menu that the user is prompted with.

\*

\* @throws FileNotFoundException

\*/

public static void menu() throws FileNotFoundException{

String response;

String options = "A: Run method sumOfReciprocals\nB: Run Isabel's Technique\nC: Run method list\nD: Quit";

response = JOptionPane.showInputDialog( null, options, "Lab104 Algorithms", 3 );

switch ( response )

{

case "a" :

case "A" :

System.out.println("Calling sumOfReciprocals");

inputDialogA();

break;

case "b" :

case "B" :

System.out.println("Calling Isabel's Technique");

inputDialogB();

break;

case "c" :

case "C" :

System.out.println("Calling list");

inputDialogC();

break;

case "d" :

case "D" :

System.out.println("Quiting");

break;

default :

System.out.println("Please try again");

errorMessage();

menu();

}

}

/\*\*

\* This method is called if it reaches the default case

\* of the menu swith statement.

\*/

public static void errorMessage()

{

String response = "Please input a correct option";

JOptionPane.showMessageDialog(null, null, response, 0);

}

/\*\*

\* This method calls sumOfReciprocals using the data

\* the user input.

\*

\* @throws FileNotFoundException

\*/

public static void inputDialogA( ) throws FileNotFoundException

{

Recursion recursive = new Recursion();

String response;

response = JOptionPane.showInputDialog( null, "Please input an integer", "sumOfReciprocals", 3 );

System.out.println("respone = " + response );

double answer;

answer = recursive.sumOfReciprocals( parseDouble(response) );

System.out.println( "answer = " + answer );

JOptionPane.showMessageDialog(null, answer, null, 1);

menu();

}

/\*\*

\* This method is very flexible. It asks for a file path,

\* and does Isabel's technique if it exists, and keeps asking until it works.

\*

\* @throws FileNotFoundException

\*/

public static void inputDialogB() throws FileNotFoundException{

Recursion recursive = new Recursion();

String response;

response = JOptionPane.showInputDialog( null, "Please input a file path", "Isabel's Technique", 3 );

System.out.println( "response = " + response );

int answer;

try{

answer = recursive.loadDataFile( response );

System.out.println( "answer = " + answer );

if(answer != 0){

JOptionPane.showMessageDialog(null, answer, "Total", 1);

menu();

}

else{

System.out.println("Array was not a power of 2");

String options = "A: Try to enter another File \nB: Quit to menu";

response = JOptionPane.showInputDialog( null, options, "The array is not a power of 2!", 2);

switch ( response ){

case "a":

case "A":

System.out.println("Calling Isabel's Technique");

inputDialogB();

break;

case "b":

case "B":

menu();

break;

default :

System.out.println("Nothing here");

errorMessage();

inputDialogB();

}

}

}

catch (FileNotFoundException fnfe){

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

JOptionPane.showMessageDialog(null, "Please try again", "File Not Found", 0);

String options = "A: Try to enter another File \nB: Quit to menu";

response = JOptionPane.showInputDialog( null, options, "The previous file was invalid", 2);

switch ( response ){

case "a":

case "A":

System.out.println("Calling Isabel's Technique");

inputDialogB();

break;

case "b":

case "B":

menu();

break;

default :

System.out.println("Nothing here");

errorMessage();

inputDialogB();

}

}

}

/\*\*

\* This method receives the input from the user and

\* applies the list method to it, and prints it in the console.

\*

\* @throws FileNotFoundException

\*/

public static void inputDialogC()throws FileNotFoundException{

Recursion recursive = new Recursion();

String response;

response = JOptionPane.showInputDialog( null, "Please input a path", "Method C", 1 );

System.out.println("response = " + response);

recursive.list( response );

JOptionPane.showMessageDialog(null, "Please refer to the console", null, 1);

menu();

}

}

**Output:**

run:

Calling sumOfReciprocals

respone = 12

answer = 3.103210678210678

Calling Isabel's Technique

response = d

File Not Found

Calling Isabel's Technique

response = C:\Users\dylca\Desktop\Lab104\_Isabel.txt

Size of Array = 8

Numbers in array:

1

2

3

4

5

6

7

8

answer = 36

Calling list

response = C:\Users\dylca\Desktop\Circuits I

C:\Users\dylca\Desktop\Circuits I\HW1\_Circuits1\HW1-1\_Circuits1.jpg

C:\Users\dylca\Desktop\Circuits I\HW1\_Circuits1\HW1-2\_Circuits1.jpg

C:\Users\dylca\Desktop\Circuits I\HW1\_Circuits1\HW1-3\_Circuits1.jpg

C:\Users\dylca\Desktop\Circuits I\Lab 1 Basic Circuit Laws.doc

C:\Users\dylca\Desktop\Circuits I\Lab1\_Report\_Circuits.docx

Quiting

BUILD SUCCESSFUL (total time: 1 minute 10 seconds)