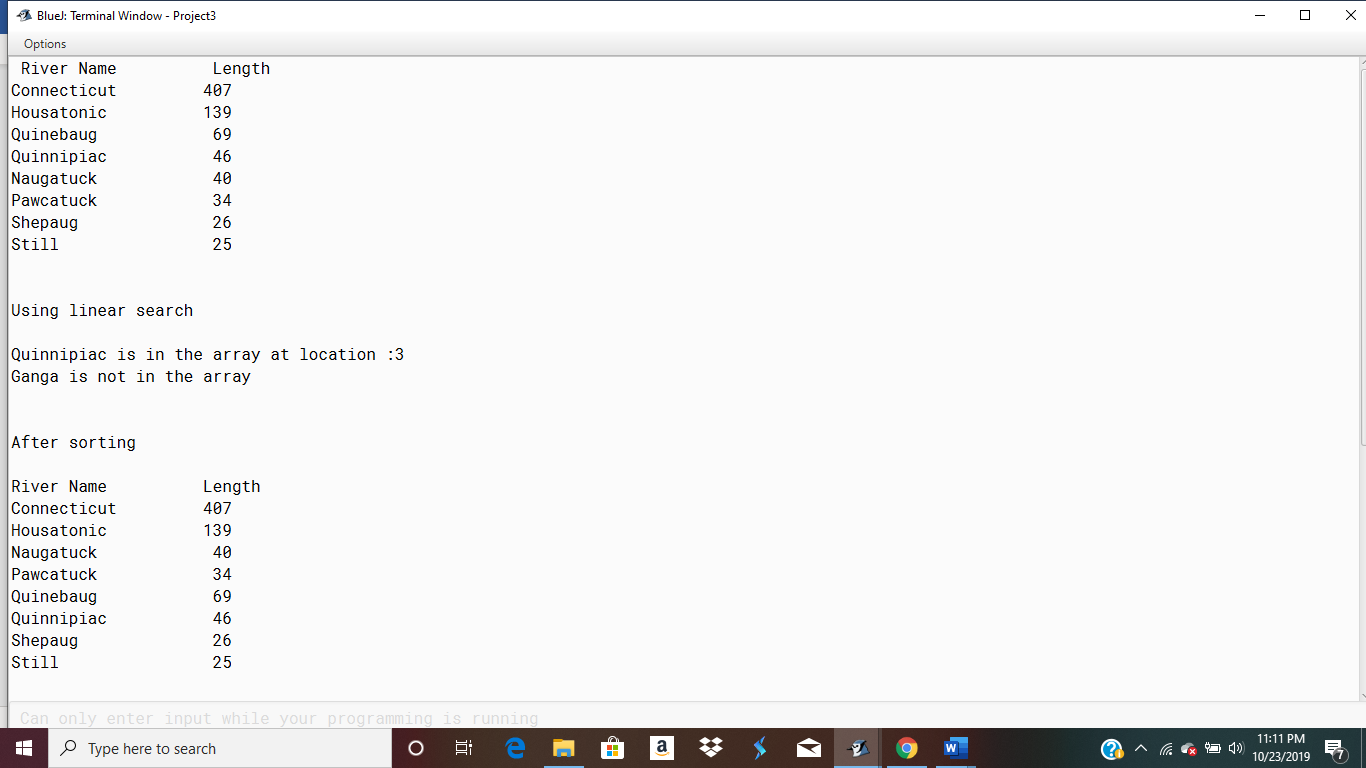
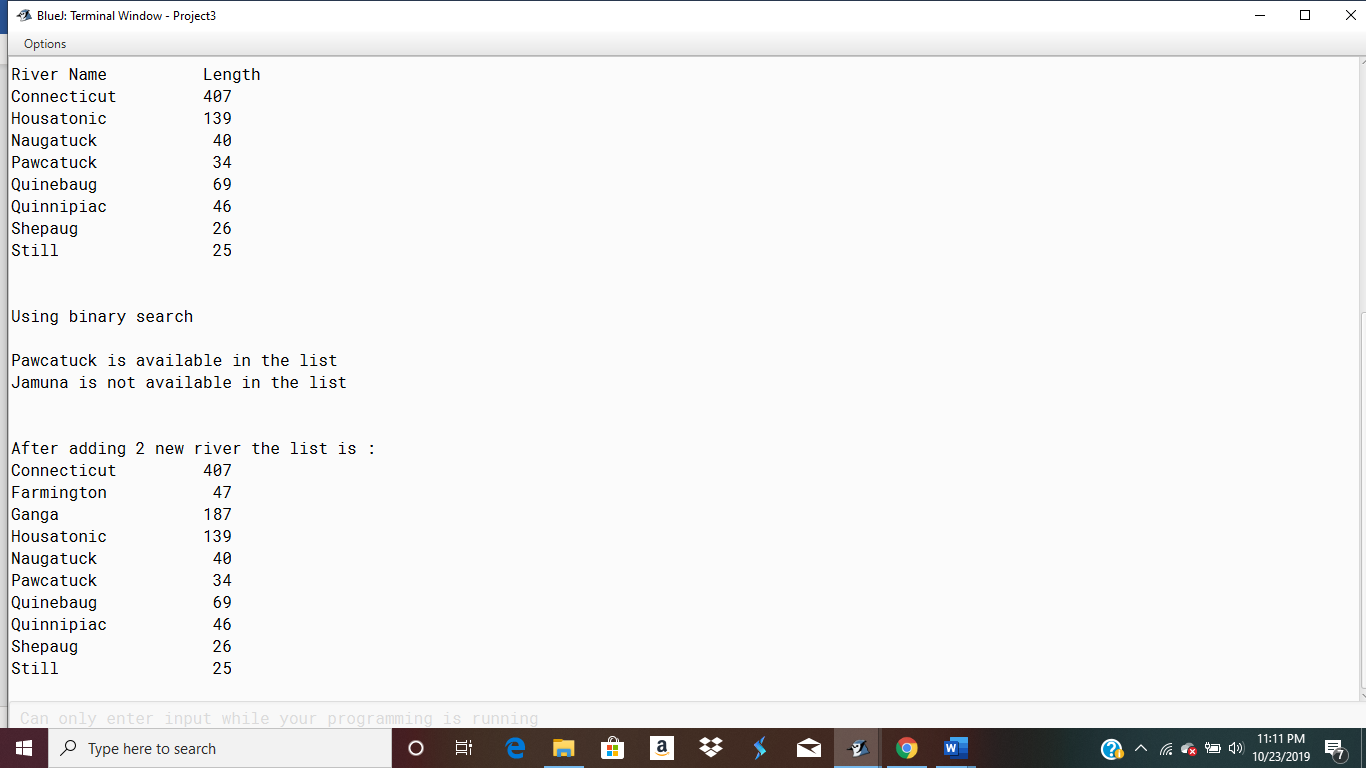
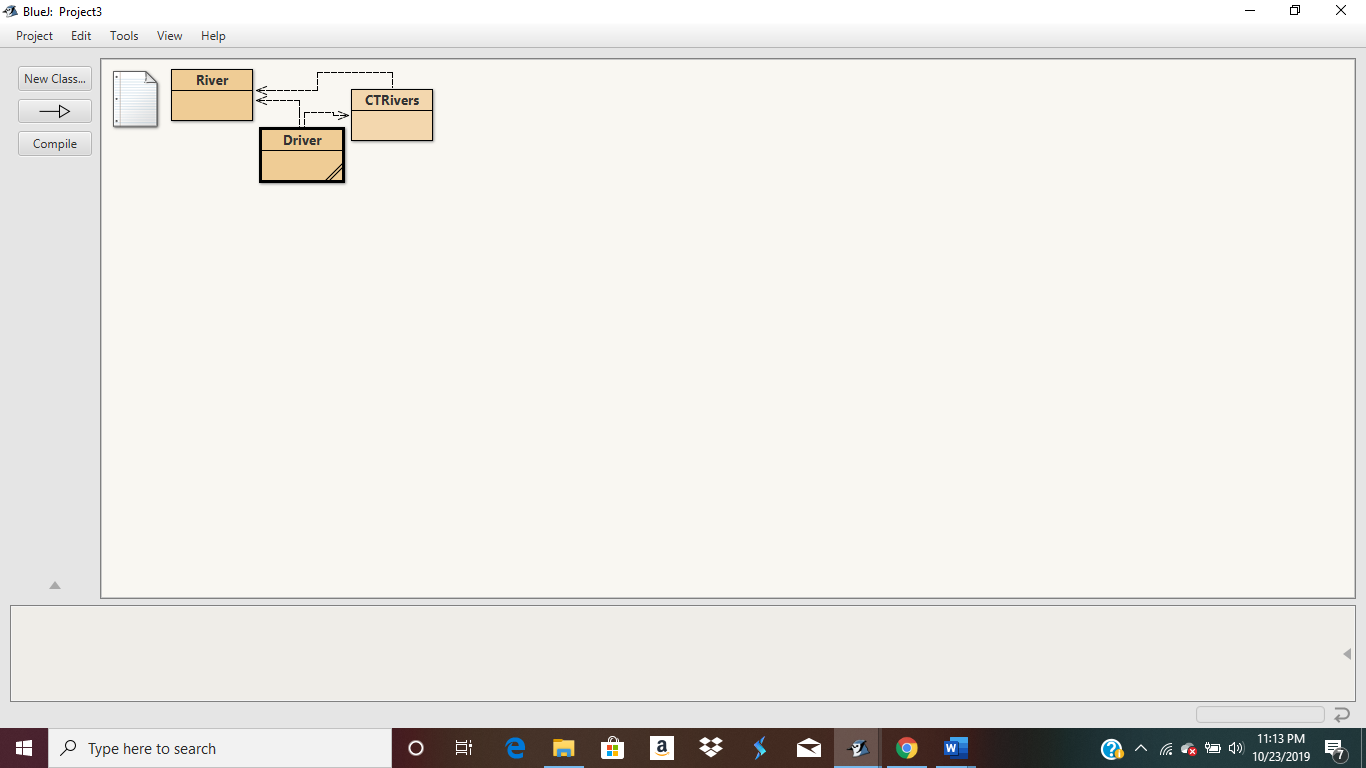
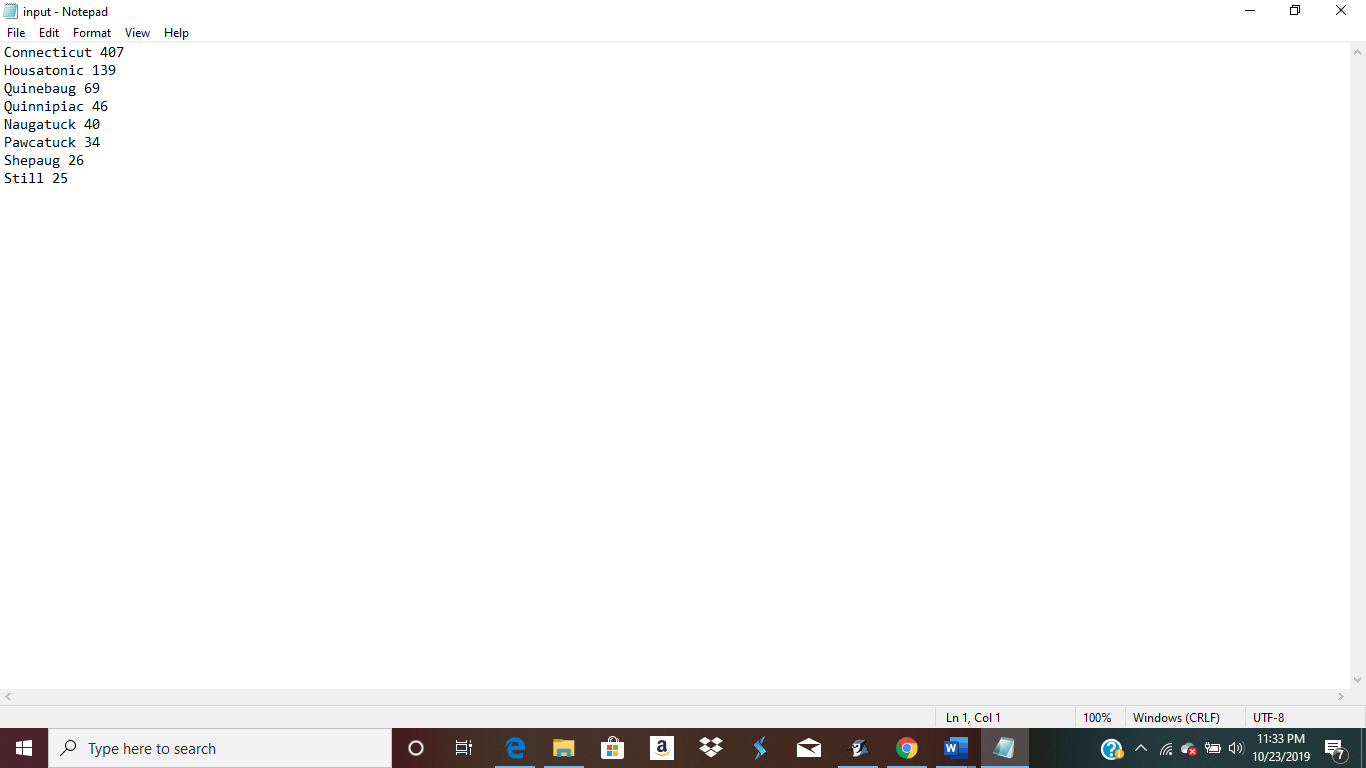
Erik Marrero









import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class Driver {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args)

{

BufferedReader reader;

River []MyList=new River[100];

CTRivers ctObject=new CTRivers();

int count=0;

try

{

reader = new BufferedReader(new FileReader("C://Users//jesic//OneDrive//Desktop//input.txt"));

String line = reader.readLine();

while (line != null)

{

String[] splited = line.split(" ");

MyList[count]=new River(splited[0],Integer.parseInt(splited[1]));

line = reader.readLine();

count++;

}

reader.close();

} catch (IOException e)

{

e.printStackTrace();

}

System.out.printf("%-7s %15s\n","River Name","Length");

ctObject.printListRec(MyList, count);

int index=ctObject.linearSearch(MyList, count, "Quinnipiac");

System.out.println("\n\nUsing linear search\n");

if(index>=0)

System.out.println("Quinnipiac is in the array at location :"+index);

index=ctObject.linearSearch(MyList, count, "Ganga");

if(index>=0)

System.out.println("Ganga is in the array at location :"+index);

else

System.out.println("Ganga is not in the array");

ctObject.sortByNameRec(MyList, count);

System.out.println("\n\nAfter sorting\n");

System.out.printf("%-7s %15s\n","River Name","Length");

ctObject.printListRec(MyList, count);

System.out.println("\n\nUsing binary search\n");

River r=ctObject.binarySearchRec(MyList, count, "Pawcatuck");

if(r==null)

System.out.println("Pawcatuck is not available in the list");

else

System.out.println("Pawcatuck is available in the list");

r=ctObject.binarySearchRec(MyList, count, "Jamuna");

if(r==null)

System.out.println("Jamuna is not available in the list");

else

System.out.println("Jamuna is available in the list");

ctObject.insertInOrder(MyList, count, new River("Farmington",47));

count++;

ctObject.insertInOrder(MyList, count, new River("Ganga",187));

count++;

System.out.println("\n\nAfter adding 2 new river the list is :");

ctObject.printListRec(MyList, count);

}

}

import java.util.ArrayList;

public class CTRivers

{

public void printListRec(River[] list, int n)

{

for(int i=0;i<n;i++)

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

}

public int linearSearch(River[] list, int n, String name)

{

for(int i=0;i<n;i++)

{

if(list[i].getName().equalsIgnoreCase(name))

return i;

}

return -1;

}

public ArrayList <River> searchRange(River[] list, int n, int min, int max)

{

ArrayList <River> riverList=new ArrayList<>();

for(int i=0;i<n;i++)

{

if(list[i].getLength()>=min && list[i].getLength()<=max)

riverList.add(list[i]);

}

return riverList;

}

public void sortByNameRec(River[] list, int n)

{

selectionSort(list,0,n);

}

//these are the helper function

private static void swap(River[] arr, int i, int j)

{

River temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

// Recursive function to perform selection sort on sub-array arr[i..n-1]

private static void selectionSort(River[] arr, int i, int n)

{

// find the minimum element in the unsorted sub-array[i..n-1]

// and swap it with arr[i]

int min = i;

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

{

// if arr[j] element is less, then it is the new minimum

if (arr[j].getName().compareTo( arr[min].getName() ) < 0)

{

min = j; // update index of min element

}

}

// swap the minimum element in sub-array[i..n-1] with arr[i]

swap(arr, min, i);

if (i + 1 < n)

{

selectionSort(arr, i + 1, n);

}

}

River binarySearchRec(River[] list, int n, String name)

{

int index=binarySearch(list, 0, n-1, name);

if(index!=-1)

return list[index];

return null;

}

private static int binarySearch(River arr[], int l, int r, String x)

{

if (r >= l) {

int mid = l + (r - l) / 2;

// If the element is present at the middle itself

if (arr[mid].getName().equals(x))

return mid;

// If element is smaller than mid, then it can only

// be present in left subarray

if ((arr[mid].getName().compareTo(x))<0)

return binarySearch(arr, l, mid - 1, x);

// Else the element can only be present in right

// subarray

return binarySearch(arr, mid + 1, r, x);

}

// We reach here when element is not present in array

return -1;

}

public void insertInOrder(River[] list, int n, River river)

{

int i=0;

for(i=0;i<n;i++)

{

if(list[i].getName().compareTo(river.getName())>0)

break;

}

for(int j=n; j > i; j--)

{

list[j] = list[j-1];

}

list[i] = river;

}

}

public class River

{

private String name;

private int length;

public River(String name, int length) {

this.name = name;

this.length = length;

}

public String getName() {

return name;

}

public int getLength() {

return length;

}

@Override

public String toString()

{

String str = String.format("%-12s %10d",name,length);

return str;

}

}