

*Programming Assignment Sheet*

|  |  |
| --- | --- |
| To: | Troy Tuckett |
| From: | Elbio Iseas |
| Class: | POS/409 |
| Date: | 9/30/14 |
| Re: | Individual Assignment for Week 3 |

|  |
| --- |
| Design: |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  | BinarySearch | | |  | |  |  |  |  |  | |  | Main Class |  | Helper Class |  | |  |  |  |  |  | |  | frmMain |  | CLIENT |  | |  |  |  |  |  | |  |  |  | lastName |  | |  |  |  | firstName |  | |  | frmMain() |  | address |  | |  |  |  | city |  | |  |  |  | state |  | |  |  |  | zipCode |  | |  |  |  | phoneNumber |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  | int countRecords(string textFile) |  | |  |  |  | Client[] readFromFile(Client[] array, string textFile) |  | |  |  |  | Client[] sortClientArray(Client[] array, int totalElements) |  | |  |  |  | int BinSearch(Client[] clients, string term) |  | |  |  |  | void loadTextBoxes( Client[] array, int index) |  | |  |  |  | boolean goodPhoneNumber(string phoneNo) |  | |  |  |  |  |  | |  |  |  |  |  |   The application in C# BinarySearch is meant to look for Clients’ information by entering their phone numbers. For this purpose, it uses a GUI (graphical user interface) to interact with end-users, a text file, a method to count the amount of Clients, a reading method to retrieve the Customers’ information from disc, a Bubble sort method to sort the file by phone number, and a method to look for the phone number in an array in memory.  The design uses two classes: the first one is the form class (frmMain), and the second one is the helper class (Client). The main class has the text boxes for the Customer’s information, a couple of buttons: Search by phone and Exit. The use of an ArrayList is for loading the file records in string elements in the previously mentioned data structure. The Bubble sort method is the classical sorting algorithm. This method receives two parameters: the first one is an array Client type, and the second parameter is the total elements of the array. It returns the sorted array when the array is sorted.  The method that reads from file returns a Client array with the Clients’ information. It receives two parameters: a Client array, and a string variable with the name of the file to read. It uses a string variable to store a line of the “comma delimited file”. It also uses an ArrayList to keep the records separately in string elements. These elements are split with a string method called Split that stores the fields in a string array that later in the method are assigned to local string variables which load the different attributes of a new Client object. The next step is to assign this new Client object to an array. These steps are inside a for loop, and when Is finished, it returns this array. The method will display messages in case of unable to read the text file, or if the text file is empty.  The binSearch method returns an integer number. It receives two parameters: a Client array, and the string term to look for (the phone number). The number returned will be -1 if the term wasn’t found, or a number starting from 0 to the max elements – 1 (the position in the array where the phone number matched with the term). It uses three variables to store the lowest index, the highest index, and the middle index of the previous two indexes. This middle index goes changing inside the while loop with the condition that it will keep the loop while the low value is less or equal to the high value. When the low value is equal or greater than the high value, then the loop will stop and the method will return a -1 because the term wasn’t found in the array, otherwise the term is found, the method will return the index where the term was found. There is a counter variable in charge of adding the amount of comparisons. |
|
| Source Program(s) : |
| // ================================================================================================  // Form1.cs  // ================================================================================================  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.IO;  //  namespace POS409DiceSimulation  {  public partial class frmMain : Form  {  public frmMain()  {  InitializeComponent();  }  private void frmMain\_Load(object sender, EventArgs e)  {  }  private void button3\_Click(object sender, EventArgs e)  {  Close();  }  private void btnWrite\_Click(object sender, EventArgs e)  {  // idea from MSDN example    string file = "rolledDice.txt";  string currentPath = Directory.GetCurrentDirectory();  string strSumOfDice = "";  StreamWriter fileWriter = new StreamWriter(file);  Random r = new Random();  int numberOne = 0;  int numberTwo = 0;  int numberSum = 0;    for (int i = 0; i < 100; i++)  {  numberOne = r.Next(1, 6);  numberTwo = r.Next(1, 6);  numberSum = numberOne + numberTwo;  strSumOfDice = numberSum.ToString();  fileWriter.WriteLine(strSumOfDice);  }  fileWriter.Close();  }  private void btnRead\_Click(object sender, EventArgs e)  {  frmResults results = new frmResults();  results.Show();  }  }  }  // ================================================================================================  // Client.cs  // ================================================================================================  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.IO;  using System.Collections;  namespace POS409BinarySearch  {    public class Client  {  string lastName;  string firstName;  string address;  string city;  string state;  string zipCode;  string phoneNumber;  public Client()  {  lastName = "";  firstName = "";  address = "";  city = "";  state = "";  zipCode = "";  phoneNumber = "";  }  public Client auxClient;  string textFile = "Customers.txt";  public int totalClients;  public Client[] clientArray;  public int comparisons;    // =============================================================================================    public int BinSearch(Client[] clients, string term)  {  comparisons = 0;  int low = 0;  int high = clients.Length - 1;  int middle = 0;  while (low <= high)  {  middle = low + ((high - low) / 2);  if (term.CompareTo(clients[middle].phoneNumber)==0)  {  comparisons++;  return middle;  }  else if (term.CompareTo(clients[middle].phoneNumber)==-1)  {  comparisons++;  high = middle - 1;  }  else  {  comparisons++;  low = middle + 1;  }  }  return -1;  }    // =============================================================================================  public int countRecords(string textFile)  {  int totalRecords = 0;  try  {  StreamReader fileReader = new StreamReader(textFile);  while ((fileReader.ReadLine()) != null)  {  totalRecords++;  }  fileReader.Close();  if (totalRecords == 0)  MessageBox.Show("File is empty!");    }  catch (Exception e)  {  MessageBox.Show("Unable to read file!");  MessageBox.Show(e.Message);  }  return totalRecords;  }  // =============================================================================================  public void loadTextBoxes( Client[] array, int index)  {  frmMain frm = new frmMain();  frm.LastName = (string)array[index].lastName;    frm.FirstName = (string)array[index].firstName;  frm.Address = (string)array[index].address;  frm.City = (string)array[index].city;  frm.State = (string)array[index].state;  frm.ZipCode = (string)array[index].zipCode;  frm.Phone = (string)array[index].phoneNumber;  }  // =============================================================================================  public Client[] readFromFile(Client[] array, string textFile)  {    ArrayList CustAList = new ArrayList();  string line;  int totalLines = 0;  try  {  StreamReader fileReader = new StreamReader(textFile);  while ((line = fileReader.ReadLine()) != null)  {  CustAList.Add(line);  }  fileReader.Close();  totalLines = CustAList.Count;    string last = "";  string first = "";  string address = "";  string city = "";  string state = "";  string zipcode = "";  string phone = "";  if (totalLines > 0)  {    string elementOfAList = "";    string[] fields;  char[] separator = new char[] {','};  for (int i = 0; i < totalLines; i++)  {    elementOfAList = (string)CustAList[i];  fields = elementOfAList.Split(separator, 7, StringSplitOptions.RemoveEmptyEntries);    last = fields[0];    first = fields[1];  address = fields[2];  city = fields[3];  state = fields[4];  zipcode = fields[5];  phone = fields[6];  Client newClient = new Client();  newClient.lastName = last;  newClient.firstName = first;  newClient.address = address;  newClient.city = city;  newClient.state = state;  newClient.zipCode = zipcode;  newClient.phoneNumber = phone;  array[i] = newClient;  }    }  else  {  MessageBox.Show("File is empty!");  }    }  catch (Exception e)  {  MessageBox.Show("Unable to read file!");  MessageBox.Show(e.Message);  }  return array;  }    // =============================================================================================    public Client[] sortClientArray(Client[] array, int totalElements)  {  Client[] arrayToSort = new Client[totalElements];  arrayToSort = array;  Client auxClient = new Client();  int result = 0;  for (int i = totalElements - 1; i > 0; i--)  {  for (int j = 0; j < totalElements - 1; j++)  {  result = arrayToSort[j].phoneNumber.CompareTo(arrayToSort[j + 1].phoneNumber);  if (result > 0)  {  auxClient = arrayToSort[j];  arrayToSort[j] = arrayToSort[j + 1];  arrayToSort[j + 1] = auxClient;  }  }  }    return arrayToSort;  }  // =============================================================================================  public Boolean goodPhoneNumber(string phoneNo)  {  Boolean OK = false;  string phNo = phoneNo.Trim();  if (phNo.Length == 10)  {  Boolean good = true;  for (int i = 0; i < 10; i++)  {  if (!(char.IsDigit(phNo, i)))  {  good = false;  break;  }  }  if (good)  {  if (Int32.Parse(phNo.Substring(0, 1)) > 0)  {  OK = true;  }  else  {  MessageBox.Show("The first digit of the phone number must be greater than zero!");  }  }  else  {  MessageBox.Show("Phone Number must be digits only!");  }  }  else  {  MessageBox.Show("Phone Number must be only digits!");  }  return OK;  }  }  } |
| Output Results: |
|  |
|  |
| // ============================================================================  // unsorted text file Customers.txt  // ============================================================================  Scott Thomas,Kristin,This is a test address for the assignment,New York,NY,10010,6469421631  Jackson,Samuel,This is a test address for the assignment,Chicago,IL,60609,3121402470  Freeman,Morgan,This is a test address for the assignment,Los Angeles,CA,90012,3106442209  Stewart,Patrick,This is a test address for the assignment,Santa Ana,CA,92702,7145600154  Stone,Emma,This is a test address for the assignment,Boston,MA,02119,6178843711  Fox,Megan,This is a test address for the assignment,Las Vegas,NV,89128,7022417986  Fay,Mia,This is a test address for the assignment,Pasadena,CA,91105,6268346618  Scarlett,Johansson,This is a test address for the assignment,Honolulu,HI,96813,8083562413  Hewitt,Jennifer Love,This is a test address for the assignment,Ogden,UT,84244,3859226036  Holmes,Katie,This is a test address for the assignment,Portland,OR,97201,5036542182  Cruise,Tom,This is a test address for the assignment,New York,NY,10007,2127441911  Hanks,Tom,This is a test address for the assignment,Albany,NY,12205,5186041497  Willis,Bruce,This is a test address for the assignment,Sacramento,CA,94207,9168178917  Stallone,Sylverster,This is a test address for the assignment,Hollywood,CA,90027,3235186031  Eastwood,Clint,This is a test address for the assignment,San Francisco,CA,94103,4152316078  DeNiro,Robert,This is a test address for the assignment,New York,NY,10012,5168042983  Parker,Sarah Jessica,This is a test address for the assignment,New York,NY,10002,6314081092  Jones,Catherine Zeta,This is a test address for the assignment,Park City,UT,94098,4354321207  Biel,Jessica,This is a test address for the assignment,Los Angeles,CA,90003,4248701987  Aniston,Jennifer,This is a test address for the assignment,New York,NY,10010,9176239876  Williams,Robin,This is a test address for the assignment,New York,NY,10009,3478897183  Sheen,Charlie,This is a test address for the assignment,Malibu,Ca,90265,4247713782  Sheen,Martin,This is a test address for the assignment,Ogden,UT,84201,8019314018  Neil,Sam,This is a test address for the assignment,San Diego,CA,92106,6193136001  Swayze,Patrick,This is a test address for the assignment,New York,NY,10006,2129135467  Washington,Denzel,This is a test address for the assignment,Pittsburg,PA,15205,4121602736  Candy,John,This is a test address for the assignment,Seattle,WA,98106,2067247796  Goodman,John,This is a test address for the assignment,South Jordan,UT,84095,3857371920  Damon,Matt,This is a test address for the assignment,Grantsville,UT,84029,4352982014  Walberg,Mark,This is a test address for the assignment,Phoenix,AZ,85003,6025372964  Diaz,Cameron,This is a test address for the assignment,Los Angeles,CA,90001,3109322074  Wilson,Owen,This is a test address for the assignment,Honolulu,HI,96813,8083915761  Gibson,Mel,This is a test address for the assignment,Santa Monica,CA,90403,3104296933  Hudson,Kate,This is a test address for the assignment,Miami,FL,33128,3055606673  Nicholson,Jack,This is a test address for the assignment,New York,NY,10011,2129968803  // ============================================================================  Description of testing   1. The program will read the data of the Clients from a text file into a data structure in memory (an array of Client objects). PASS 2. The program will sort the data in the array in ascendant order. PASS 3. The program will accept a ten digit number for the phone number to search. PASS. 4. The program will display the total comparisons taken to find the term. PASS. 5. The program will have access to the Client found. PASS. 6. The program will display the Client’s information. FAIL. 7. The program will use the phone number 503 654 2182 with no blanks in between for the search, and if found it will display the quantity of comparisons to get to the information. Comparisons : 5 . PASS. 8. The program will use the phone number 412 160 2736 with no blanks in between for the search, and if found it will display the quantity of comparisons to get to the information. Comparisons : 4 . PASS. 9. The program will use the phone number 518 604 1497 with no blanks in between for the search, and if found it will display the quantity of comparisons to get to the information. Comparisons : 5 . PASS. |
| Tested By  Elbio Iseas |