Create a Text-file Based System For Storing and Updating Teacher Records

teacher.txt file

The teacher text file (teacher.txt) has teacher records such that each attribute of a record is separated by a '|'. Each teacher record occupies a new line for easy readability.

Id1|name1|class1|section1

Id2|name2|class2|section2 ….and so on

Code

Perform() function

A menu is displayed to users to add and update teacher records and exit. The logic is contained in a while loop such that the menu is repeated until the user chooses exit.

Menu

1.Add teacher record

2.Update teacher record

3.Exit

When user chooses exit, a break statement is used to come out of the while loop.

Switch case is used to call the respective function based on users input.

Case 1 calls the addRecord() function.

Case 2 calls the updateRecord() function.

Default case outputs that the input is incorrect.

---

displayRecord() function

displayRecord() retrieves the data from the teacher.txt file and displays it.

Reading the teacher file

The entire content of the teacher file is read using File.ReadAllText() and put into a string 'strdata'. File.ReadAllText(@"D:\teacher.txt") opens the teacher.txt file, reads and closes the file.

Splitting the teacher records.

The data in 'strdata' is split such that each teacher record is separated and stored in a different index of 'rowdata'.

strdata.Split(new[] { "\r\n", "\r", "\n" }, StringSplitOptions.None) splits based on the following logic,

=>if there is a new line

=>if there is a carriage return

=>combination of carriage return and new line.

The result is rowdata[0] contains first teacher record, rowdata[1] contains second teacher record and so on.

Displaying the data

Foreach loop is used to traverse the rowdata array.

For each teacher record, each attribute(id,name,class,section) is split based on '|' using

string[] splitdata = sturecord.Split('|');

Splitdata[0] contains teacher id

Splitdata[1] contains teacher name

Splitdata[2] contains teacher class

Splitdata[3] contains teacher section .

The above details are then displayed on the console using Console.WriteLine();

---

addRecord() function

FileStream is used to write into the teacher.txt file.

A fileStream object is created in append mode since the teacher records are added onto the existing data and the object is given write access. The path of the teacher file is set in the first parameter.

FileStream fs = new FileStream("D:\\teacher.txt", FileMode.Append, FileAccess.Write);

StreamWriter is then initialised using fileStream object to write data into the teacher.txt file.

StreamWriter sw = new StreamWriter(fs);

The teacher id,name,class and section is obtained from user through console using Console.ReadLine().

The attributes are combined using "|" to distinguish them.

string str = tID + "|" + tName + "|" + tClass + "|" + tSection;

The combined string is written into the teacher file.

sw.WriteLine(str);

The file is then closed using sw.close().

DisplayRecord() is called to display the contents of the file after the changes.

---

updateRecord() function

A list of teacher record is created to perform the update operation.

List<Teacher> tlist = new List<Teacher>();

Each item of the list contains an object of teacher with attributes of the object as

Obj.id,obj.name,obj.tclass and obj.section.

The user enters the id of the teacher record to update.

The list is parsed using foreach loop .

If the record is found,

1. The new details of that record are inputted by user.
2. The old details are updated.

record.id = newId;
record.name = newName;
record.tClass = newClass;
record.section = newSec;

The updated list is overwritten into the text file.

displayRecord() is called to display the new text file.

GitHub link

apeksha7777/GradableAssesment1 (github.com)