**File Task**

**Name : Menna Alllah Hesham Nour**

**Sec. : 12**

**2013 - 2012**

**Program ::**

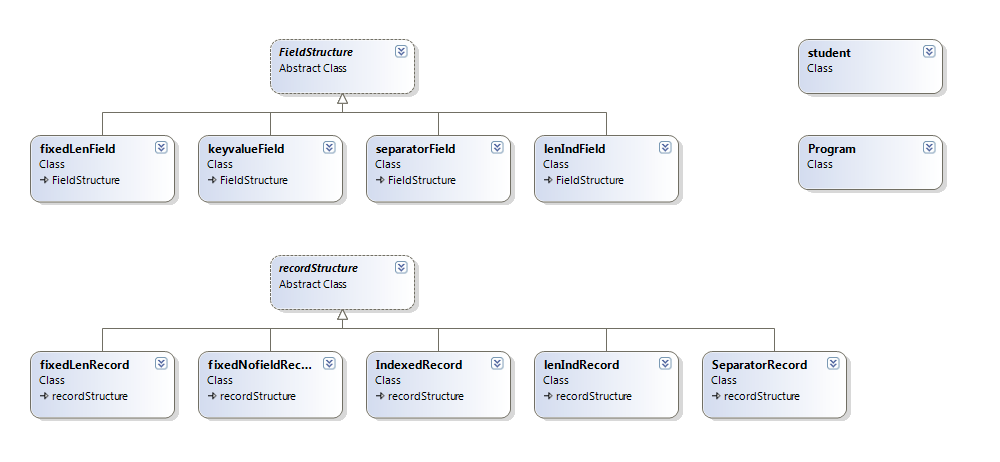
**Student ::**

**FieldStructure.cs ::**

1. **fixedLenField.cs ::**
2. **keyvalueField.cs ::**
3. **lenIndField.cs ::**
4. **separatorField.cs ::**

**RecordStructure.cs ::**

1. **FixedLenRecord.cs ::**
2. **fixedNofieldRecord.cs ::**
3. **IndexedRecord.cs ::**
4. **lenIndRecord.cs ::**
5. **SeparatorRecord.cs ::**

****

**Program ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace FileTask

{

class Program

{

static void Main(string[] args)

{

recordStructure rd = GetRecordStructureFromUser();

Console.WriteLine("What do you want to do");

Console.WriteLine("1-Enter a new student");

Console.WriteLine("2-Display File");

Console.WriteLine("3-Exit");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

student std = new student();

std.getData();

Console.WriteLine("Enter file name");

FileStream fs = new FileStream(Console.ReadLine(), FileMode.Append, FileAccess.Write);

StreamWriter sw = new StreamWriter(fs);

if (rd.keyval)

rd.WriteWzKey(sw, std);

else

rd.Write(sw, std);

sw.Close();

fs.Close();

break;

case 2:

Console.WriteLine("Enter file name");

List<student> allStudents = rd.ReadAll(Console.ReadLine());

for (int i = 0; i < allStudents.Count; i++)

{

allStudents[i].display();

}

break;

default:

break;

}

}

private static recordStructure GetRecordStructureFromUser()

{

Console.WriteLine("Press :\n1) For Fixed Lenght Record.\n2) For Length indicator Record.\n3) For Seprator Records.\n4) For Indexed Record.\n5) For Fixed number of fields.\n");

String s = Console.ReadLine();

recordStructure rd;

if (s == "1")

{

rd = new fixedLenRecord();

rd.Indexed = false;

}

else if (s == "2")

{

rd = new lenIndRecord();

rd.Indexed = false;

}

else if (s == "3")

{

rd = new SeparatorRecord();

rd.Indexed = false;

}

else if (s == "4")

{

rd = new IndexedRecord();

rd.Indexed = true;

}

else

{

rd = new fixedNofieldRecord();

rd.Indexed = false;

}

Console.WriteLine("Press :\n1) For Fixed Lenght Field.\n2) For Length indicator Field.\n3) For Seprator Field.\n4) For keyvalue Field.\n");

s = Console.ReadLine();

if (s == "1")

{

rd.id = new fixedLenField();

rd.name = new fixedLenField();

rd.address = new fixedLenField();

rd.keyval = false;

}

else if (s == "2")

{

rd.id = new lenIndField();

rd.name = new lenIndField();

rd.address = new lenIndField();

rd.keyval = false;

}

else if (s == "3")

{

rd.id = new separatorField();

rd.name = new separatorField();

rd.address = new separatorField();

rd.keyval = false;

}

else

{

rd.id = new keyvalueField();

rd.name = new keyvalueField();

rd.address = new keyvalueField();

rd.keyval = true;

}

return rd;

}

}

}

**Student ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

public class student

{

private string Id;

private string Name;

private string Address;

public string id

{

get { return Id; }

set { Id = value; }

}

public string name

{

get { return Name; }

set { Name = value; }

}

public string address

{

get { return Address; }

set { Address = value; }

}

public void display()

{

Console.WriteLine("Id:");

Console.WriteLine(this.Id);

Console.WriteLine("Name:");

Console.WriteLine(this.Name);

Console.WriteLine("Address:");

Console.WriteLine(this.Address);

}

public void getData()

{

Console.WriteLine("Enter your id:");

this.Id = Console.ReadLine();

Console.WriteLine("Enter your name:");

this.Name = Console.ReadLine();

Console.WriteLine("Enter your address:");

this.Address = Console.ReadLine();

}

}

}

**FieldStructure.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

public abstract class FieldStructure

{

public abstract string WriteField(string Field);

public abstract string writeField(string Field, string keyvalue);

public abstract string ReadField(ref string Record);

}

}

1. **fixedLenField.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class fixedLenField : FieldStructure

{

public override string WriteField(string Field)

{

if (Field.Length > 15)

{

Console.WriteLine("field lenght exceeded.\n");

Field = "";

}

else if (Field.Length < 15)

for (int i = Field.Length; i < 15; i++)

Field = Field + " ";

return Field;

}

public override string ReadField(ref string Record)

{

if (Record.Length > 15)

{

string Result = Record.Substring(0, 15);

Record.Substring(15);

for (int i = Result.Length - 1; i >= 0; i--)

if (Result[i] == ' ')

Result.Substring(0, i);

return Result;

}

return Record;

}

public override string writeField(string Field, string key)

{

return key + Field;

}

}

}

1. **keyvalueField.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class keyvalueField : FieldStructure

{

public override string writeField(string Field, string value)

{

return value + Field;

}

public override string ReadField(ref string Record)

{

if (Record.IndexOf('=') != -1)

Record.Substring(Record.IndexOf('=') + 1, Record.Length - Record.IndexOf('=') - 1);

if (Record.IndexOf('=') != -1)

{

string result = Record.Substring(0, Record.IndexOf('='));

Record = Record.Substring(Record.IndexOf('=') + 1);

return result;

}

return Record;

}

public override string WriteField(string Field)

{

return Field + "\*";

}

}

}

1. **lenIndField.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class lenIndField : FieldStructure

{

public override string WriteField(string Field)

{

char c = (char)(Field.Length + '0');

return c + Field;

}

public override string ReadField(ref string Record)

{

int length = Record[0] - '0';

string result = Record.Substring(1, length);

Record = Record.Substring(length + 1);

return result;

}

public override string writeField(string Field, string key)

{

return key + Field;

}

}

}

1. **separatorField.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class separatorField : FieldStructure

{

public override string WriteField(string Field)

{

return Field + "\*";

}

public override string ReadField(ref string Record)

{

if (Record.IndexOf('\*') != -1)

{

int length = Record.IndexOf('\*');

string result = Record.Substring(0, length);

Record = Record.Substring(length + 1);

return result;

}

return Record;

}

public override string writeField(string Field, string key)

{

return key + Field;

}

}

}

**RecordStructure.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace FileTask

{

public abstract class recordStructure

{

public FieldStructure id, name, address;

public bool keyval, Indexed;

public int IndexVal, CurrentIndexVal1, CurrentIndexVal2 ;

protected abstract string FormatRecord( string rec );

public abstract student Read( StreamReader sr);

public virtual List<student> ReadAll(string fileName)

{

FileStream file = new FileStream(fileName, FileMode.Open, FileAccess.Read);

StreamReader sr = new StreamReader(file);

List<student> result = new List<student>();

if (Indexed)

{

FileStream fs = new FileStream("Index.txt", FileMode.Open, FileAccess.Read);

StreamReader Sr = new StreamReader(fs);

if (Sr.Peek() != -1)

CurrentIndexVal1 = int.Parse(Sr.ReadLine());

while (Sr.Peek() != -1)

{

CurrentIndexVal2 = int.Parse(Sr.ReadLine());

student std = this.Read(sr);

result.Add(std);

CurrentIndexVal1 = CurrentIndexVal2;

}

}

else

{

while (sr.Peek() != -1)

{

student std = this.Read(sr);

result.Add(std);

}

}

sr.Close();

return result;

}

public void Write(StreamWriter sw, student std)

{

string record = "";

record += id.WriteField(std.id);

record += name.WriteField(std.name);

record += address.WriteField(std.address);

string RecordToWrite = this.FormatRecord(record);

sw.Write(RecordToWrite);

}

public void WriteWzKey(StreamWriter sw, student std)

{

string record = "";

record += id.writeField(std.id, "ID=");

record += name.writeField(std.name, "Name=");

record += address.writeField(std.address, "Address=");

string RecordToWrite = this.FormatRecord(record);

sw.Write(RecordToWrite);

}

protected student GetFields(string record)

{

student std = new student();

std.id = id.ReadField(ref record);

std.name = name.ReadField(ref record);

std.address = address.ReadField(ref record);

return std;

}

}

}

1. **FixedLenRecord.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class fixedLenRecord : recordStructure

{

protected override string FormatRecord(string record)

{

if ( record.Length > 45)

{

Console.WriteLine("record lenght exceeded.\n");

record = "";

}

else if ( record.Length < 45)

for (int i = record.Length; i < 15; i++)

record = record + " ";

return record;

}

public override student Read(System.IO.StreamReader sr)

{

char[] c = new char[ 45 ];

sr.Read(c, 0, 45);

String record = new String(c);

return this.GetFields(record);

}

}

}

1. **fixedNofieldRecord.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class fixedNofieldRecord : recordStructure

{

protected override string FormatRecord(string record)

{

return record + "\n";

}

public override student Read(System.IO.StreamReader sr)

{

String record = sr.ReadLine();

return this.GetFields(record);

}

}

}

1. **IndexedRecord.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace FileTask

{

class IndexedRecord : recordStructure

{

protected override string FormatRecord(string record)

{

FileStream fs = new FileStream("Index.txt", FileMode.Append, FileAccess.Write);

StreamWriter sw = new StreamWriter(fs);

if (this.IndexVal <= 0)

sw.WriteLine(this.IndexVal);

int len = record.Length;

this.IndexVal += len;

sw.WriteLine(this.IndexVal);

return record;

}

public override student Read(System.IO.StreamReader sr)

{

char[] C = new char[this.CurrentIndexVal2 - this.CurrentIndexVal1];

sr.Read(C, 0, this.CurrentIndexVal2 - this.CurrentIndexVal1);

String result = new String(C);

return this.GetFields(result);

}

}

}

1. **lenIndRecord.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class lenIndRecord : recordStructure

{

protected override string FormatRecord(string record)

{

char c = (char)(record.Length + '0');

return c + record;

}

public override student Read(System.IO.StreamReader sr)

{

String record = sr.ReadLine();

int length = record[0] - '0';

string result = record.Substring(1, length);

record = record.Substring(length + 1);

return this.GetFields(result);

}

}

}

1. **SeparatorRecord.cs ::**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace FileTask

{

class SeparatorRecord : recordStructure

{

protected override string FormatRecord(string record)

{

return record + "\n";

}

public override student Read(System.IO.StreamReader sr)

{

String record = sr.ReadLine();

return this.GetFields(record);

}

}

}