**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1  
по дисциплине **«Интернет-технологии и распределенная обработка данных»**

на тему: **«Создание простого консольного приложения»**

Выполнил: студент группы ИП-41  
Бредихин К.А.  
Принял: ст. преподаватель  
Шпетный Е.А.

Дата сдачи отчета:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

**Цель работы**: Создать консольное приложение для преобразования CSV файла в Exel, json, pdf c использованием .Net Core.

# Описание проделанной работы

Разработано консольное приложение, которое производит чтение входного файла в формате CSV и путем обработки считанных данных проводит расчеты средних величин, генерирует выходной файл в виде Excel (xlsx), PDF или JSON документа. Приложение предоставляет возможность взаимодействия с ним через консоль с помощью команд, в которых можно указать как входной файл, так и выходной файл и его тип (Excel, JSON, PDF).

Детали:

1. Входной файл содержит список студентов группы и их оценки по экзаменам за все предыдущие семестры. Оценки указаны по предметам, чтобы в процессе чтения можно было точно сопоставить предмет и оценку..
2. Обработка данных заключается в расчете среднего балла по каждому студенту, а также средней оценки по каждому предмету.
3. Выходной файл содержит в себе таблицу со всеми студентами, а также средними оценками каждого из них. Внизу таблицы приведены средние баллы по каждому предмету и по группе.
4. Один входной файл - один выходной.

**Листинг**

using System;

using System.IO;

namespace CSV

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("1. Custom path \n2. Default path");

var choise = Console.ReadKey().KeyChar;

Console.Clear();

switch (choise)

{

case '1':

Console.WriteLine("Input custom path");

var customPath = Console.ReadLine();

while (!File.Exists(customPath))

{

Console.Clear();

Console.WriteLine("Input custom path");

customPath = Console.ReadLine();

}

CustomPath(customPath);

break;

case '2':

DefaultPath();

break;

}

}

static void DefaultPath()

{

CSVTransformer cSVTransformer = new CSVTransformer();

char choise = '0';

while (choise != '5')

{

Console.WriteLine("\n\n\n1. Upload SCV File to Console \n2. Create XLSX File from CSV \n3. Create PDF File from CSV\n4. Create JSON File from CSV\n5. Exit");

choise = Console.ReadKey().KeyChar;

Console.Clear();

switch (choise)

{

case '1':

cSVTransformer.ConsoleOutput();

Console.WriteLine("SCV File Uploaded");

break;

case '2':

cSVTransformer.XLSXOutputGroupByStudent();

Console.WriteLine("XLSX File Created");

break;

case '3':

cSVTransformer.PDFOutput();

Console.WriteLine("PDF File Created");

break;

case '4':

cSVTransformer.JSONOutput();

Console.WriteLine("JSON File Created");

break;

}

}

}

static void CustomPath(string path)

{

CSVTransformer cSVTransformer = new CSVTransformer();

char choise = '0';

while (choise != '4')

{

Console.WriteLine("\n\n\n1. Upload SCV File to Console \n2. Create XLSX File from CSV \n3. Create PDF File from CSV\n4. Exit");

choise = Console.ReadKey().KeyChar;

Console.Clear();

switch (choise)

{

case '1':

cSVTransformer.ConsoleOutput(path);

Console.WriteLine("SCV File Uploaded");

break;

case '2':

cSVTransformer.XLSXOutputGroupByStudent();

Console.WriteLine("XLSX File Created");

break;

case '3':

cSVTransformer.PDFOutput();

Console.WriteLine("PDF File Created");

break;

case '4':

cSVTransformer.JSONOutput();

Console.WriteLine("JSON File Created");

break;

}

}

}

}

}

using CsvHelper;

using PdfSharp.Drawing;

using PdfSharp.Pdf;

using Root.Reports;

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Linq;

using System.Runtime.Serialization.Json;

namespace CSV

{

public class Subject

{

public string Student { get; set; }

public string SubjectName { get; set; }

public string Mark { get; set; }

public override string ToString()

{

return $"[Student:{Student}][Subject:{SubjectName}] [Mark:{Mark}]";

}

}

class CSVTransformer

{

const string PATH = @"csv\marks.csv";

IEnumerable<Subject> UploadFile(string path)

{

using (StreamReader streamReader = new StreamReader(path))

{

using (CsvReader csvReader = new CsvReader(streamReader))

{

csvReader.Configuration.HasHeaderRecord = false;

csvReader.Configuration.Delimiter = ";";

List<Subject> subject = csvReader.GetRecords<Subject>().ToList();

return subject;

}

}

}

public void ConsoleOutput(string path = PATH)

{

var subjects = UploadFile(path);

foreach (var subj in subjects)

{

Console.WriteLine(subj);

}

}

//public void XLSXOutput(string path = PATH)

//{

// var subjects = UploadFile(path);

// var row = 0;

// FileStream stream = new FileStream("marks.xls", FileMode.OpenOrCreate);

// ExcelWriter writer = new ExcelWriter(stream);

// writer.BeginWrite();

// foreach (var subject in subjects)

// {

// writer.WriteCell(row, 0, subject.Student);

// writer.WriteCell(row, 1, subject.SubjectName);

// writer.WriteCell(row++, 2, subject.Mark);

// }

// writer.EndWrite();

// stream.Close();

//}

public void XLSXOutputGroupByStudent(string path = PATH)

{

var subjects = UploadFile(path);

var row = 0;

var col = 0;

FileStream stream = new FileStream("marks.xls", FileMode.OpenOrCreate);

ExcelWriter writer = new ExcelWriter(stream);

writer.BeginWrite();

var subjList = subjects.Select(a=>a.SubjectName).Distinct().ToList();

foreach (var subjectName in subjList)

{

writer.WriteCell(row, ++col, subjectName);

}

writer.WriteCell(row, ++col, "Average");

var maxRowForAverageMark = col;

col = 0;

var studentList = subjects.GroupBy(a => a.Student).ToList();

foreach (var student in studentList)

{

writer.WriteCell(++row, col, student.Key);

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

{

++col;

if (student.Any(a => a.SubjectName == subjList[i]))

writer.WriteCell(row, col, student.FirstOrDefault(a => a.SubjectName == subjList[i]).Mark);

}

writer.WriteCell(row, maxRowForAverageMark, student.Average(a=> int.Parse(a.Mark)));

col = 0;

}

col = 0;

row++;

writer.WriteCell(row, col, "Average");

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

if (subjects.Any(a => a.SubjectName == subjList[i]))

writer.WriteCell(row, ++col, subjects.Where(a => a.SubjectName == subjList[i]).Average(a => int.Parse(a.Mark)));

writer.WriteCell(row, ++col, subjects.Average(a =>int.Parse(a.Mark)));

writer.EndWrite();

stream.Close();

}

public void PDFOutput(string path = PATH)

{

var subjects = UploadFile(path);

Report report = new Report(new PdfFormatter());

FontDef fd = new FontDef(report, "Helvetica");

FontProp fp = new FontPropMM(fd, 5);

Page page = new Page(report);

int row = 40;

foreach(var subj in subjects)

{

page.AddCenteredMM(row += 10, new RepString(fp, subj.ToString()));

}

report.Save("Marks.pdf");

}

public void JSONOutput(string path = PATH)

{

var subjects = UploadFile(path);

OutputFormatForJson Json = new OutputFormatForJson();

Json.AverageMark = subjects.Average(a => int.Parse(a.Mark));

#region addStudents

Json.Students = new List<Student>();

foreach(var studentFromSubjects in subjects.GroupBy(a => a.Student).ToList())

{

Student newStudent = new Student();

newStudent.Name = studentFromSubjects.Key;

newStudent.AverageMark = studentFromSubjects.Average(a => int.Parse(a.Mark));

newStudent.SudentsSubjects = new List<SudentsSubject>();

foreach(var studentSubj in studentFromSubjects)

{

newStudent.SudentsSubjects.Add(new SudentsSubject { Mark = int.Parse(studentSubj.Mark), Name = studentSubj.SubjectName });

}

Json.Students.Add(newStudent);

}

#endregion

#region addSubjects

Json.Subjects = new List<Subj>();

foreach (var subjectsFromStudents in subjects.GroupBy(a => a.SubjectName).ToList())

{

Subj newSubj = new Subj();

newSubj.Name = subjectsFromStudents.Key;

newSubj.AverageMark = subjectsFromStudents.Average(a => int.Parse(a.Mark));

newSubj.SubjSudents = new List<SubjSudent>();

foreach (var student in subjectsFromStudents)

{

newSubj.SubjSudents.Add(new SubjSudent { Mark = int.Parse(student.Mark), Name = student.Student });

}

Json.Subjects.Add(newSubj);

}

#endregion

DataContractJsonSerializer jsonFormatter = new DataContractJsonSerializer(typeof(OutputFormatForJson));

using (FileStream fs = new FileStream("Marks.json", FileMode.OpenOrCreate))

{

jsonFormatter.WriteObject(fs, Json);

}

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace CSV

{

class ExcelWriter

{

private Stream stream;

private BinaryWriter writer;

private ushort[] clBegin = { 0x0809, 8, 0, 0x10, 0, 0 };

private ushort[] clEnd = { 0x0A, 00 };

private void WriteUshortArray(ushort[] value)

{

for (int i = 0; i < value.Length; i++)

writer.Write(value[i]);

}

public ExcelWriter(Stream stream)

{

this.stream = stream;

writer = new BinaryWriter(stream);

}

public void WriteCell(int row, int col, string value)

{

ushort[] clData = { 0x0204, 0, 0, 0, 0, 0 };

int iLen = value.Length;

byte[] plainText = Encoding.ASCII.GetBytes(value);

clData[1] = (ushort)(8 + iLen);

clData[2] = (ushort)row;

clData[3] = (ushort)col;

clData[5] = (ushort)iLen;

WriteUshortArray(clData);

writer.Write(plainText);

}

public void WriteCell(int row, int col, int value)

{

ushort[] clData = { 0x027E, 10, 0, 0, 0 };

clData[2] = (ushort)row;

clData[3] = (ushort)col;

WriteUshortArray(clData);

int iValue = (value << 2) | 2;

writer.Write(iValue);

}

public void WriteCell(int row, int col, double value)

{

ushort[] clData = { 0x0203, 14, 0, 0, 0 };

clData[2] = (ushort)row;

clData[3] = (ushort)col;

WriteUshortArray(clData);

writer.Write(value);

}

public void WriteCell(int row, int col)

{

ushort[] clData = { 0x0201, 6, 0, 0, 0x17 };

clData[2] = (ushort)row;

clData[3] = (ushort)col;

WriteUshortArray(clData);

}

public void BeginWrite()

{

WriteUshortArray(clBegin);

}

public void EndWrite()

{

WriteUshortArray(clEnd);

writer.Flush();

}

}

}

using System;

using System.Collections.Generic;

using System.Runtime.Serialization;

using System.Text;

namespace CSV

{

[DataContract]

public class OutputFormatForJson

{

[DataMember] public List<Student> Students { get; set; }

[DataMember] public List<Subj> Subjects { get; set; }

[DataMember] public double AverageMark { get; set; }

}

[DataContract]

public class Student

{

[DataMember] public string Name { get; set; }

[DataMember] public double AverageMark { get; set; }

[DataMember] public List<SudentsSubject> SudentsSubjects { get; set; }

}

[DataContract]

public class SudentsSubject

{

[DataMember] public string Name { get; set; }

[DataMember] public int Mark { get; set; }

}

[DataContract]

public class Subj

{

[DataMember] public string Name { get; set; }

[DataMember] public double AverageMark { get; set; }

[DataMember] public List<SubjSudent> SubjSudents { get; set; }

}

[DataContract]

public class SubjSudent

{

[DataMember] public string Name { get; set; }

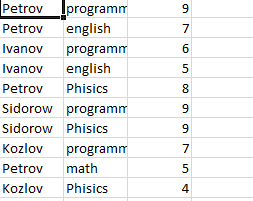
[DataMember] public int Mark { get; set; }

}

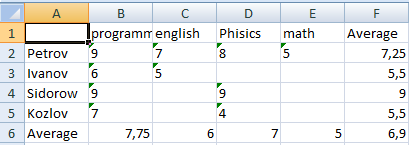
}

# Верификация и тестирование

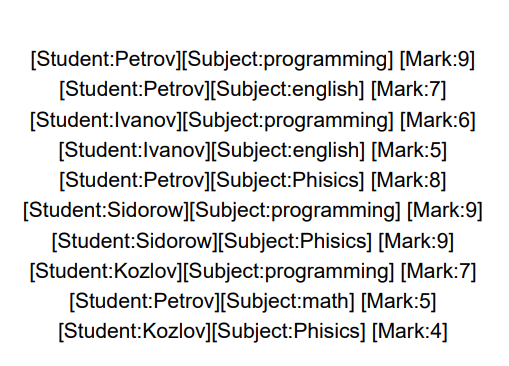
Для входных данных в CSV:



Выходные данные в Exel:



Выходные данные в Pdf:



Выходные данные в Json:

{"AverageMark":6.9,"Students":[{"AverageMark":7.25,"Name":"Petrov","SudentsSubjects":[{"Mark":9,"Name":"programming"},{"Mark":7,"Name":"english"},{"Mark":8,"Name":"Phisics"},{"Mark":5,"Name":"math"}]},{"AverageMark":5.5,"Name":"Ivanov","SudentsSubjects":[{"Mark":6,"Name":"programming"},{"Mark":5,"Name":"english"}]},{"AverageMark":9,"Name":"Sidorow","SudentsSubjects":[{"Mark":9,"Name":"programming"},{"Mark":9,"Name":"Phisics"}]},{"AverageMark":5.5,"Name":"Kozlov","SudentsSubjects":[{"Mark":7,"Name":"programming"},{"Mark":4,"Name":"Phisics"}]}],"Subjects":[{"AverageMark":7.75,"Name":"programming","SubjSudents":[{"Mark":9,"Name":"Petrov"},{"Mark":6,"Name":"Ivanov"},{"Mark":9,"Name":"Sidorow"},{"Mark":7,"Name":"Kozlov"}]},{"AverageMark":6,"Name":"english","SubjSudents":[{"Mark":7,"Name":"Petrov"},{"Mark":5,"Name":"Ivanov"}]},{"AverageMark":7,"Name":"Phisics","SubjSudents":[{"Mark":8,"Name":"Petrov"},{"Mark":9,"Name":"Sidorow"},{"Mark":4,"Name":"Kozlov"}]},{"AverageMark":5,"Name":"math","SubjSudents":[{"Mark":5,"Name":"Petrov"}]}]}

Ссылка на Github:

**Выводы**: создано консольное приложение для преобразования CSV файла в Exel, json, pdf c использованием .Net Core.