|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ДОМАШНЯЯ РАБОТА №1**

**«Паттерны проектирования и принципы ООП»**

**ДИСЦИПЛИНА: «Объектно-ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Былинка М.И. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** формированиепрактических навыков объектно-ориентированного проектирования в соответствии с принципами SOLID и использованием паттернов проектирования.

**Задачи:**

* Закрепить навыки объектно-ориентированного проектирования в соответствии с принципами SOLID
* Научиться разрабатывать программные системы слоистой архитектуры.

**Вариант №7**

В программе, разработанной в рамках выполнения лабораторной работы № 2, список студентов «жестко» задан в коде самого приложения.

В ходе выполнения домашнего задания необходимо:

* в сущность «Студент (Student)» добавить атрибут «Дата рождения (BirthDate)»;
* переработать приложение так, чтобы его «ядро» не зависело от способа получения списка студентов (источником данных может быть список в памяти, файл, база данных и проч.);
* обеспечить обработку ситуации недоступности хранилища, а также ошибок в формате хранения данных;
* спроектировать и реализовать механизм загрузки списков сущностей из файлов. Приложение не должно зависеть от конкретных имен файлов и форматов хранения;
* реализовать загрузку студентов из текстового файла «std-forma.csv» или из текстового файла «std-form-b.dat». Реализованы должны быть оба механизма, но в приложении «жестко» должно задаваться использование одного из них.

В каждой строке текстового файла «std-form-a.csv» содержится запись о студенте в следующем формате:

RegNumber,FullName,BirthDate

Т. е. перечислены значения атрибутов, разделенные символом «,». Дата рождения (поле «BirthDate») приведена в формате «dd.MM.yyyy» (здесь dd – день месяца, MM – номер месяца, yyyy – год). Пример файла, содержащего данные о 3 студентах:

CS-00345/14,Brice Lambson,22.03.1990

MM-12007/13,Andrew Peters,17.07.1989

CS-09711/13,James Gurtz,05.11.1985

В каждой строке текстового файла «std-form-b.dat» содержится запись о студенте в следующем формате:

RegNumber|FirstName|MiddleName|LastName|BirthDate

Т. е. перечислены значения атрибутов, разделенные символом «|». При этом фамилия студента хранится в поле «LastName», имя – в поле «FirstName», отчество – в поле «MiddleName». Дата рождения (поле «BirthDate») приведена в формате «yyyy-MM-dd» (здесь dd – день месяца, MM – номер месяца, yyyy – год). В этом формате последние 3 символа представляют код учебного подразделения и должны игнорироваться при загрузке. Пример файла, содержащего данные о 3 студентах:

CS-00345/14.17|Brice|S|Lambson|1990-03-22

MM-12007/13.17|Andrew|B|Peters|1989-07-17

CS-09711/13.17|James|C|Gurtz|1985-11-05

**UML-диаграмма:**



Рис. 1. UML-диаграмма

**Листинг:**

***Student.cs***

using System;

namespace LW\_2

{

public class Student

{

public Student()

{

RegNumber = "";

FullName = "";

BirthDate = DateTimeOffset.Parse("01/01/01");

}

public Student(string regNumber, string fullName)

{

RegNumber = regNumber;

FullName = fullName;

BirthDate = DateTimeOffset.Parse("01/01/01");

}

public Student(string regNumber, string fullName, DateTimeOffset birthDate)

{

RegNumber = regNumber;

FullName = fullName;

BirthDate = birthDate;

}

public string RegNumber { get; }

public string FullName { get; }

public DateTimeOffset BirthDate { get; }

}

}

***IStudentLoader.cs***

using System.Collections.Generic;

namespace LW\_2.Loaders

{

public interface IStudentLoader

{

List<Student> Load();

}

}

***IStudentFileLoader.cs***

namespace LW\_2.Loaders

{

public interface IStudentFileLoader : IStudentLoader

{

string Path { get; }

}

}

***StudentCsvLoader.cs***

using System;

using System.Collections.Generic;

using System.IO;

using System.Text.RegularExpressions;

namespace LW\_2.Loaders

{

public class StudentCSVLoader : IStudentFileLoader

{

public string Path { get; }

public StudentCSVLoader(string path) { Path = path; }

public List<Student> Load()

{

if (!File.Exists(Path))

throw new FileNotFoundException();

List<Student> students = new List<Student>();

foreach (string line in File.ReadAllLines(Path))

{

string[] data = line.Split(',');

if (data.Length != 3)

throw new FileLoadException("Not enough fields");

if (!Regex.IsMatch(data[0], @"^[A-Z]{2}-\d{5}/\d{2}$"))

throw new FormatException("Incorrect registration number");

if(!Regex.IsMatch(data[1], @"^[A-Z][a-z]+\s[A-Z][a-z]+$"))

throw new FormatException("Incorrect name");

if(!Regex.IsMatch(data[2], @"^\d{2}-\d{2}-\d{4}$"))

throw new FormatException("Incorrect birth date");

students.Add(new Student(data[0], data[1], DateTimeOffset.Parse(data[2])));

}

return students;

}

}

}

***StudentDatLoader.cs***

using System;

using System.Collections.Generic;

using System.IO;

using System.Text.RegularExpressions;

namespace LW\_2.Loaders

{

public class StudentDatLoader : IStudentFileLoader

{

public string Path { get; }

public StudentDatLoader(string path) { Path = path; }

public List<Student> Load()

{

if (!File.Exists(Path))

throw new FileNotFoundException();

List<Student> students = new List<Student>();

foreach (string line in File.ReadAllLines(Path))

{

string[] data = line.Split('|');

if (data.Length != 5)

throw new FileLoadException("Not enough fields");

if (!Regex.IsMatch(data[0], @"^[A-Z]{2}-\d{5}/\d{2}\.\d{2}$"))

throw new FormatException("Incorrect registration number");

if (!Regex.IsMatch(data[1], @"^[A-Z][a-z]+$"))

throw new FormatException("Incorrect first name");

if (!Regex.IsMatch(data[2], @"^[A-Z][a-z]\*$"))

throw new FormatException("Incorrect middle name");

if (!Regex.IsMatch(data[3], @"^[A-Z][a-z]+$"))

throw new FormatException("Incorrect last name");

if (!Regex.IsMatch(data[4], @"^\d{4}[-]\d{2}[-]\d{2}$"))

throw new FormatException("Incorrect birth date");

students.Add(new Student(data[0].Split('.')[0], data[1] + " " + data[2] + " " + data[3], DateTimeOffset.Parse(data[4])));

}

return students;

}

}

}

***UnitTest.cs***

using Microsoft.VisualStudio.TestTools.UnitTesting;

using LW\_2.Loaders;

using System.Collections.Generic;

using LW\_2;

using System;

namespace Testing

{

[TestClass]

public class Testing

{

[TestMethod]

public void TestCsvLoad()

{

StudentCSVLoader loader = new StudentCSVLoader("D:\\Students.csv");

List<Student> students = loader.Load();

Assert.AreEqual(students[0].RegNumber, "CS-00345/14");

Assert.AreEqual(students[1].FullName, "Andrew Peters");

Assert.AreEqual(students[2].BirthDate, DateTimeOffset.Parse("05.11.1985"));

}

[TestMethod]

public void TestDatLoad()

{

StudentDatLoader loader = new StudentDatLoader("D:\\Students.dat");

List<Student> students = loader.Load();

Assert.AreEqual(students[0].RegNumber, "CS-00345/14");

Assert.AreEqual(students[1].FullName, "Andrew B Peters");

Assert.AreEqual(students[2].BirthDate, DateTimeOffset.Parse("05.11.1985"));

}

}

}

**Результат:**

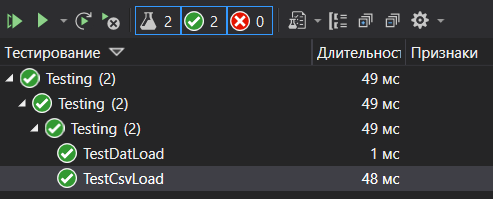


Рис. 2. Результат

**Вывод:** в ходе выполнения домашней работы были получены практические навыки реализации принципов SOLID и слоистой архитектуры, работы с файлами.

**ОСНОВНАЯ ЛИТЕРАТУРА**

1. Романенко, В.В. Объектно-ориентированное программирование: учебное пособие / В.В. Романенко: Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР).–Томск : Томский Государственный Университет Систем Управления и Радиоэлектроники, 2014.– 475 с. : ил. – Библиогр.: с. 442. : То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=480517>
2. Николаев, Е.И. Объектно-ориентированное программирование: учебное пособие / Е.И. Николаев; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждения высшего профессионального образования «Северо-Кавказский федеральный университет».– Ставрополь: СКФУ, 2015.– 225 с.: ил.– Библиогр. В кн.: То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=458133>
3. Суханов, М.В. Основы Microsoft .Net Fraework и языка программирования C#: учебное пособие / М.В. Суханов, И.В. Бачурин, И.С. Майоров: Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждения высшего профессионального образования Северный (Арктический) федеральный университет им. М.В. Ломоносова.– Архангельск: ИД САФУ, 2014 – 97 с.: схем., табл., ил.– Библиогр. В кн.– ISBN 978-5-261-00934-4: То же [Электронный ресурс]. – URL: <http://biblioclub.ru/index.php?page=book&id=312313>

**ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА**

1. Б.Албахари C# 6.0. Справочник. Полное описание языка: учеб. пособие / Б. Албахари, Дж. Албахари. – 6-е издание. М: O’reilly, 2016 г.– 1040 с.