МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Проєкт частина А**

з дисципліни « Об’єктно орієнтоване програмування »

*назва дисципліни*

на тему: «Відношення між класами:  
асоціація, агрегація, композиція, реалізація»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: доц. Вдовітченко О.В.

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

# ПОСТАНОВКА ЗАДАЧІ

1. Обрати і коротко описати нову предметну область.
2. Для обраної предметної області, з використанням інструментів для створення діаграм, за два етапи побудувати модель предметної області у вигляді діаграми класів, на якій класи/інтерфейси мають бути пов’язані різними типами відношень: асоціація, агрегація, композиція, реалізація.

* **Етап 1:** Виявлення та визначення перелічень, класів, інтерфейсів та зав’язків між ними. Бажано використовувати всі перелічені у п.2 типи відношень.
* **Етап 2:** Детальне проєктування класів та інтерфейсів: для полів/властивостей необхідно вказати модифікатор доступу, назву, тип; для методів – додатково визначити тип значення, що повертається, та список параметрів

1. Засобами середи розробки (IDE) створити каркас проєкту, для цього необхідно визначити в окремих файлах .cs всі спроєктовані у п.2 інтерфейси, класи, перелічення. Коректно реалізувати відношення між класами.

Реалізовувати властивості, конструктори і методи у класах не потрібно! Замість реального коду вони мають містити заглушки **throw new NotImplementedException().**

1. Для тестування спроєктованих класів додати проєкт з тест-класами, які повинні містити достатній для повноцінного тестування набір реалізованих unit-тестів.

Методи класів предметної області коду ще не мають, але для них вже пишемо повноцінні unit-тести (TDD-принцип розробки програми). Після запуску розроблених unit-тестів їх статус має бути failed.

1. Використання GitHub-репозиторію з коммітами є обов’язковим! Репозиторій повинен мати Readme.md файл, в який необхідно:

* додати красивий опис предметної області, для цього використовуйте заголовки, списки, рисунки та ін.);
* вставити розроблену діаграму класів.

1. Оформити звіт:

* Титульний аркуш
* Завдання
* Опис предметної області
* Проєктування діаграми класів
* Реалізація класів
* Реалізація тест-класів
* Результати запуску unit-тестів.
* Посилання на GitHub-репозиторій

ХІД РОБОТИ

Опис предметної області

Шкільна система розроблена для управління навчальним процесом у навчальному закладі. Система включає в себе основні об'єкти та взаємодії між ними, які відображають типові аспекти навчання та організації шкільного життя.

**Об'єкти системи:**

Учень:

* Властивості: Ім'я, прізвище, вік.
* Взаємодія: Учень може вивчати кілька предметів.

Предмет:

* Властивості: Назва предмету.
* Взаємодія: Предмет може мати викладача та бути вивченим декількома учнями.

Клас (Група учнів):

* Властивості: Список учнів, класний керівник.
* Взаємодія: Клас складається з учнів та має класного керівника.

Вчитель:

* Властивості: Ім'я, прізвище.
* Взаємодія: Вчитель може вести декілька предметів та мати список учнів.

**Відносини між об'єктами:**

Асоціація:

* Учень та Предмет: Учень асоційований з предметом, оскільки може вивчати кілька предметів.

Агрегація:

* Клас та Учень: Клас агрегує учнів, оскільки клас складається з учнів.

Композиція:

* Розклад та Предмет: Розклад використовує композиційне відношення з предметом, оскільки розклад може визначати конкретні предмети.

Ця шкільна система дозволяє ефективно керувати інформацією про учнів, вчителів, предмети та їх взаємодії, сприяючи організації навчання в навчальному закладі.

Проєктування діаграми класів

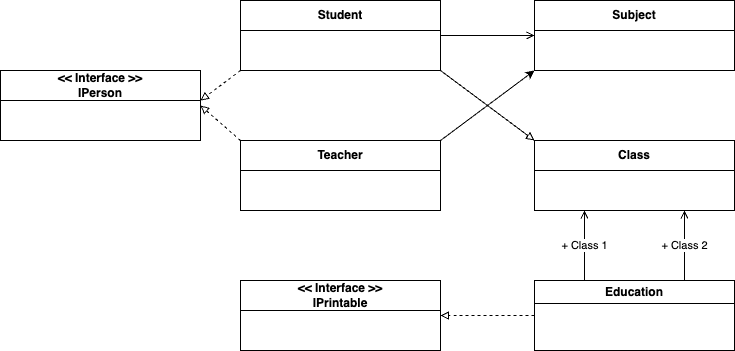


Рисунок 1 – Виявлення та визначення елементів предметної області та зв’язки між ними

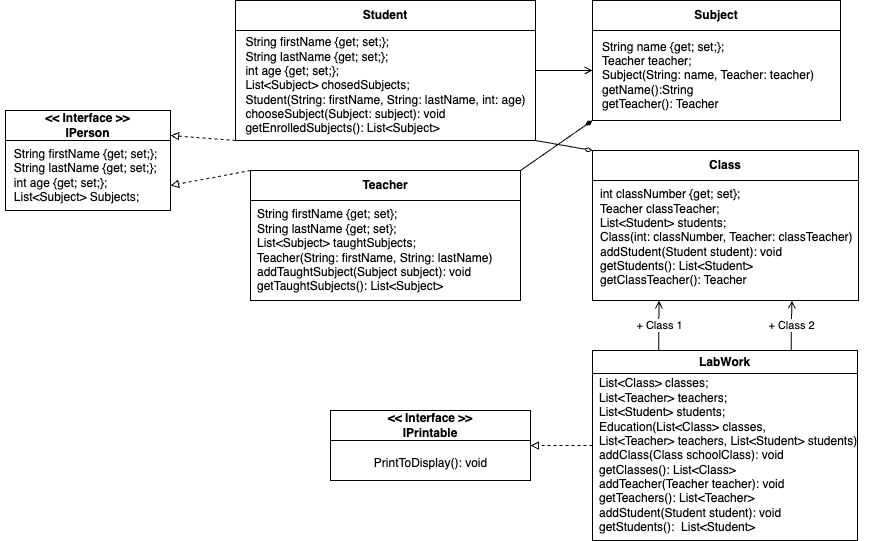


Рисунок 2 – Детальне проєктування елементів моделі предметної області

Реалізація класів програми

Детальний код класів можна переглянути в додатках А - Є.

Реалізація тест-класів

Детальний код тест-класів можна переглянути в додатках Ж-Л.

Результати запуску unit-тестів.

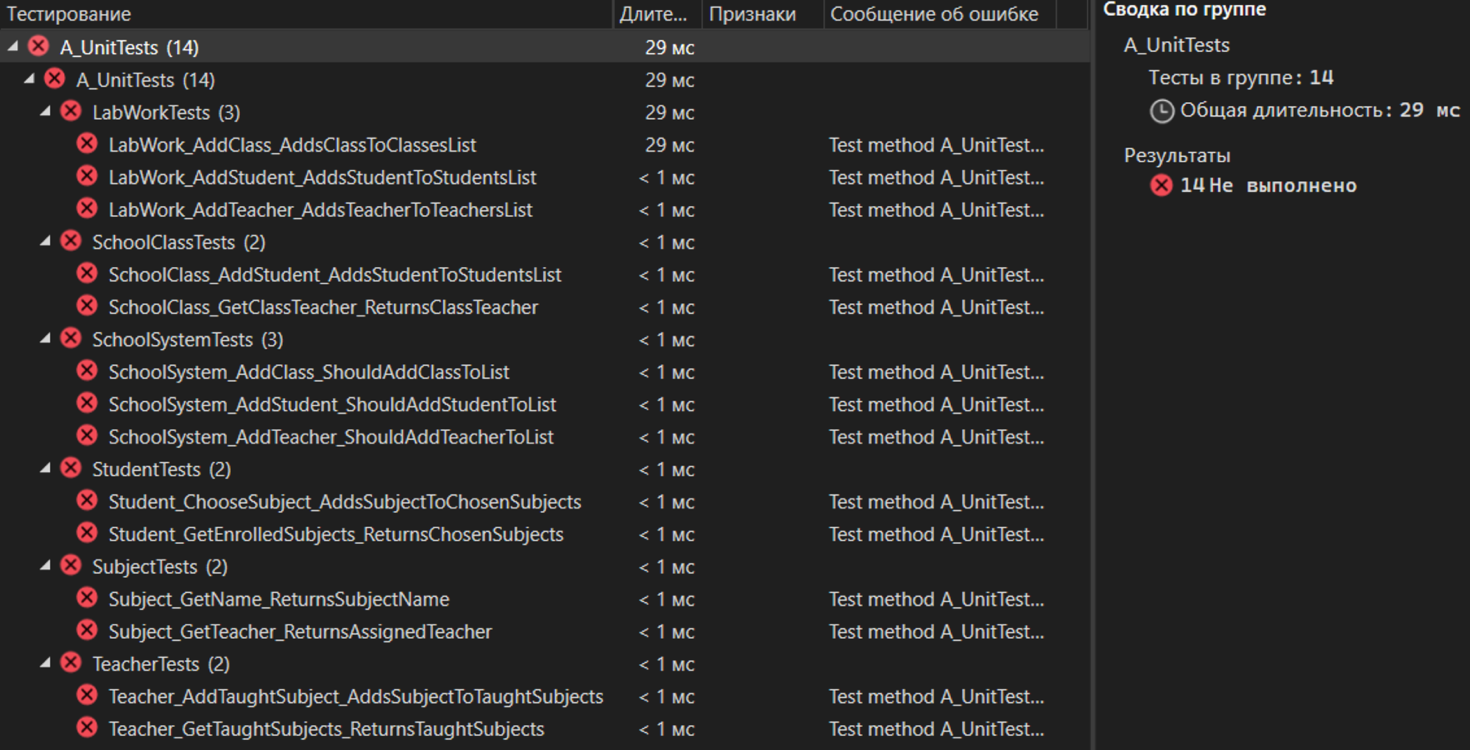


Рисунок 3 – запущені тести видали результат Fail

Посилання на GitHub-репозиторій

<https://github.com/Yarik-Za/OOP_Project-Parts/tree/Part-A>

# ВИСНОВОК

В основній програмі додано методи для збереження (серіалізації) колекції List<T> об’єктів предметної області у файли з форматом .csv (.txt) та .json. Додано також методи для читання (десеріалізації) колекції з відповідних файлів.

Меню програми було розширено новими функціями: збереженням та читанням колекції об’єктів з файлів у форматах .csv (.txt) та .json, а також можливістю очищення колекції.

Підменю для пунктів "Зберегти у файл .csv (.txt)" та "Зберегти у файл .json" реалізовано відповідно.

Підменю для пунктів "Зчитати з файлу .csv (.txt)" та "Зчитати з файлу .json" враховує наявність об’єктів у колекції List<T> на момент зчитування. Десеріалізовані об’єкти додаються до колекції лише у випадку коректного десеріалізації.

Для нових та перероблених методів були додані та кореговані unit-тести, і вони успішно пройшли всі перевірки.

Програма була детально протестована, включаючи збереження та читання з файлів у різних форматах, а також обробку різних сценаріїв вводу даних. Всі тести та етапи виконані успішно, забезпечуючи стабільну та надійну роботу програми.

***Додаток А***Машинний лістинг програми **Program.cs**

***Додаток А***Машинний лістинг класу **SchoolSystem.cs**

using Project\_part\_A.Classes;

namespace Project\_part\_A

{

public class SchoolSystem

{

private List<SchoolClass> Classes { get; }

private List<Teacher> Teachers { get; }

private List<Student> Students { get; }

public SchoolSystem(List<SchoolClass> classes, List<Teacher> teachers, List<Student> students)

{

throw new NotImplementedException();

}

public void AddClass(SchoolClass schoolClass)

{

throw new NotImplementedException();

}

public List<SchoolClass> GetClasses()

{

throw new NotImplementedException();

}

public void AddTeacher(Teacher teacher)

{

throw new NotImplementedException();

}

public List<Teacher> GetTeachers()

{

throw new NotImplementedException();

}

public void AddStudent(Student student)

{

throw new NotImplementedException();

}

public List<Student> GetStudents()

{

throw new NotImplementedException();

}

static void Main(string[] args)

{

Console.WriteLine("Hello, World!");

}

}

}

***Додаток Б***Машинний лістинг класу **LabWork.cs**

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public class LabWork : IPrintable

{

public List<SchoolClass> Classes { get; set; } = new List<SchoolClass>();

public List<Teacher> Teachers { get; set; } = new List<Teacher>();

public List<Student> Students { get; set; } = new List<Student>();

public void AddClass(SchoolClass schoolClass)

{

throw new NotImplementedException();

}

public List<SchoolClass> GetClasses()

{

throw new NotImplementedException();

}

public void AddTeacher(Teacher teacher)

{

throw new NotImplementedException();

}

public List<Teacher> GetTeachers()

{

throw new NotImplementedException();

}

public void AddStudent(Student student)

{

throw new NotImplementedException();

}

public List<Student> GetStudents()

{

throw new NotImplementedException();

}

public void PrintToDisplay()

{

throw new NotImplementedException();

}

}

}

***Додаток В***

Машинний лістинг класу **SchoolClass.cs**

namespace Project\_part\_A.Classes

{

public class SchoolClass

{

public int ClassNumber { get; set; }

public Teacher ClassTeacher { get; set; }

public List<Student> Students { get; set; } = new List<Student>();

public SchoolClass(int classNumber, Teacher classTeacher)

{

throw new NotImplementedException();

}

public void AddStudent(Student student)

{

throw new NotImplementedException();

}

public List<Student> GetStudents()

{

throw new NotImplementedException();

}

public Teacher GetClassTeacher()

{

throw new NotImplementedException();

}

}

}

***Додаток Г***Машинний лістинг класу **Student.cs**

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public class Student : IPerson

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

public List<Subject> ChosenSubjects { get; set; } = new List<Subject>();

public Student(string firstName, string lastName, int age)

{

throw new NotImplementedException();

}

public void ChooseSubject(Subject subject)

{

throw new NotImplementedException();

}

public List<Subject> GetEnrolledSubjects()

{

throw new NotImplementedException();

}

}

}

*Додаток Ґ*Машинний лістинг класу Subject.cs

namespace Project\_part\_A.Classes

{

public class Subject

{

public string Name { get; set; }

public Teacher Teacher { get; set; }

public Subject(string name, Teacher teacher)

{

throw new NotImplementedException();

}

public string GetName()

{

throw new NotImplementedException();

}

public Teacher GetTeacher()

{

throw new NotImplementedException();

}

}

}

***Додаток Д***Машинний лістинг класу **Teacher.cs**

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public class Teacher : IPerson

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

public List<Subject> TaughtSubjects { get; set; } = new List<Subject>();

public Teacher(string firstName, string lastName)

{

throw new NotImplementedException();

}

public void AddTaughtSubject(Subject subject)

{

throw new NotImplementedException();

}

public List<Subject> GetTaughtSubjects()

{

throw new NotImplementedException();

}

}

}

***Додаток Е***Машинний лістинг класу **IPerson.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project\_part\_A.Interfaces

{

public interface IPerson

{

string FirstName { get; set; }

string LastName { get; set; }

int Age { get; set; }

}

}

***Додаток Є***Машинний лістинг класу **IPrintable.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project\_part\_A.Interfaces

{

public interface IPrintable

{

void PrintToDisplay();

}

}

***Додаток Ж***Машинний лістинг класу **LabWorkUnitTest.cs**

using Project\_part\_A.Classes;

namespace A\_UnitTests

{

[TestClass]

public class LabWorkTests

{

[TestMethod]

public void LabWork\_AddClass\_AddsClassToClassesList()

{

// Arrange

var labWork = new LabWork();

var schoolClass = new SchoolClass(12, new Teacher("Mr.", "Davis"));

// Act

labWork.AddClass(schoolClass);

// Assert

Assert.IsTrue(labWork.GetClasses().Contains(schoolClass));

}

[TestMethod]

public void LabWork\_AddTeacher\_AddsTeacherToTeachersList()

{

// Arrange

var labWork = new LabWork();

var teacher = new Teacher("Mrs.", "Smith");

// Act

labWork.AddTeacher(teacher);

// Assert

Assert.IsTrue(labWork.GetTeachers().Contains(teacher));

}

[TestMethod]

public void LabWork\_AddStudent\_AddsStudentToStudentsList()

{

// Arrange

var labWork = new LabWork();

var student = new Student("Jane", "Doe", 16);

// Act

labWork.AddStudent(student);

// Assert

Assert.IsTrue(labWork.GetStudents().Contains(student));

}

}

}

***Додаток З***Машинний лістинг класу **SchoolClassUnitTest****.cs**

using Project\_part\_A.Classes;

namespace A\_UnitTests

{

[TestClass]

public class SchoolClassTests

{

[TestMethod]

public void SchoolClass\_AddStudent\_AddsStudentToStudentsList()

{

// Arrange

var schoolClass = new SchoolClass(10, new Teacher("Mrs.", "Johnson"));

var student = new Student("John", "Doe", 15);

// Act

schoolClass.AddStudent(student);

// Assert

Assert.IsTrue(schoolClass.GetStudents().Contains(student));

}

[TestMethod]

public void SchoolClass\_GetClassTeacher\_ReturnsClassTeacher()

{

// Arrange

var teacher = new Teacher("Mr.", "Smith");

var schoolClass = new SchoolClass(11, teacher);

// Act

var classTeacher = schoolClass.GetClassTeacher();

// Assert

Assert.AreEqual(teacher, classTeacher);

}

}

}

***Додаток І***Машинний лістинг класу **SchoolSystemUnitTest.cs**

using Project\_part\_A.Classes;

using Project\_part\_A;

namespace A\_UnitTests

{

[TestClass]

public class SchoolSystemTests

{

[TestMethod]

public void SchoolSystem\_AddClass\_ShouldAddClassToList()

{

// Arrange

var schoolSystem = new SchoolSystem(new List<SchoolClass>(), new List<Teacher>(), new List<Student>());

var schoolClass = new SchoolClass(10, new Teacher("Mr.", "Johnson"));

// Act

schoolSystem.AddClass(schoolClass);

// Assert

CollectionAssert.Contains(schoolSystem.GetClasses(), schoolClass);

}

[TestMethod]

public void SchoolSystem\_AddTeacher\_ShouldAddTeacherToList()

{

// Arrange

var schoolSystem = new SchoolSystem(new List<SchoolClass>(), new List<Teacher>(), new List<Student>());

var teacher = new Teacher("Mrs.", "Smith");

// Act

schoolSystem.AddTeacher(teacher);

// Assert

CollectionAssert.Contains(schoolSystem.GetTeachers(), teacher);

}

[TestMethod]

public void SchoolSystem\_AddStudent\_ShouldAddStudentToList()

{

// Arrange

var schoolSystem = new SchoolSystem(new List<SchoolClass>(), new List<Teacher>(), new List<Student>());

var student = new Student("John", "Doe", 15);

// Act

schoolSystem.AddStudent(student);

// Assert

CollectionAssert.Contains(schoolSystem.GetStudents(), student);

}

}

}

***Додаток Й***Машинний лістинг класу **StudentUnitTest.cs**

using Project\_part\_A.Classes;

namespace A\_UnitTests

{

[TestClass]

public class StudentTests

{

[TestMethod]

public void Student\_ChooseSubject\_AddsSubjectToChosenSubjects()

{

// Arrange

var student = new Student("John", "Doe", 18);

var subject = new Subject("Math", new Teacher("Mr.", "Smith"));

// Act

student.ChooseSubject(subject);

// Assert

Assert.IsTrue(student.ChosenSubjects.Contains(subject));

}

[TestMethod]

public void Student\_GetEnrolledSubjects\_ReturnsChosenSubjects()

{

// Arrange

var student = new Student("Jane", "Doe", 17);

var mathSubject = new Subject("Math", new Teacher("Ms.", "Jones"));

var englishSubject = new Subject("English", new Teacher("Mrs.", "Brown"));

student.ChooseSubject(mathSubject);

student.ChooseSubject(englishSubject);

// Act

var enrolledSubjects = student.GetEnrolledSubjects();

// Assert

Assert.AreEqual(2, enrolledSubjects.Count);

CollectionAssert.Contains(enrolledSubjects, mathSubject);

CollectionAssert.Contains(enrolledSubjects, englishSubject);

}

}

}

***Додаток К***Машинний лістинг класу **SubjectUnitTest.cs**

using Project\_part\_A.Classes;

namespace A\_UnitTests

{

[TestClass]

public class SubjectTests

{

[TestMethod]

public void Subject\_GetName\_ReturnsSubjectName()

{

// Arrange

var teacher = new Teacher("Mr.", "Johnson");

var subject = new Subject("Math", teacher);

// Act

var subjectName = subject.GetName();

// Assert

Assert.AreEqual("Math", subjectName);

}

[TestMethod]

public void Subject\_GetTeacher\_ReturnsAssignedTeacher()

{

// Arrange

var teacher = new Teacher("Ms.", "Brown");

var subject = new Subject("English", teacher);

// Act

var assignedTeacher = subject.GetTeacher();

// Assert

Assert.AreEqual(teacher, assignedTeacher);

}

}

}

***Додаток Л***Машинний лістинг класу **TeacherUnitTest.cs**

using Project\_part\_A.Classes;

namespace A\_UnitTests

{

[TestClass]

public class TeacherTests

{

[TestMethod]

public void Teacher\_AddTaughtSubject\_AddsSubjectToTaughtSubjects()

{

// Arrange

var teacher = new Teacher("Mr.", "Johnson");

var subject = new Subject("History", teacher);

// Act

teacher.AddTaughtSubject(subject);

// Assert

Assert.IsTrue(teacher.TaughtSubjects.Contains(subject));

}

[TestMethod]

public void Teacher\_GetTaughtSubjects\_ReturnsTaughtSubjects()

{

// Arrange

var teacher = new Teacher("Mrs.", "Williams");

var mathSubject = new Subject("Math", teacher);

var physicsSubject = new Subject("Physics", teacher);

teacher.AddTaughtSubject(mathSubject);

teacher.AddTaughtSubject(physicsSubject);

// Act

var taughtSubjects = teacher.GetTaughtSubjects();

// Assert

Assert.AreEqual(2, taughtSubjects.Count);

CollectionAssert.Contains(taughtSubjects, mathSubject);

CollectionAssert.Contains(taughtSubjects, physicsSubject);

}

}

}