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 Б**

з дисципліни « Об’єктно орієнтоване програмування »

*назва дисципліни*

на тему: «Успадкування. Абстрактні класи. Поліформізм»

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. Удосконалити проєкт, скорегував діаграму класів, додавши реалізацію хоча б одним класом будь-якого інтерфейсу(ів) .NET (IClonable, IComparable, IEnumereable тощо).
2. Передбачити множинну реалізацію інтерфейсів.
3. Додати хоча б один абстрактний клас, від якого успадкувати декілька нових чи вже існуючих класів. Бажано, щоб абстрактний клас мав не тільки абстрактні методи, але й абстрактні властивості.
4. Передбачити хоча б один варіант реалізації поліморфізму:

* **Варіант 1:** об’єкти похідного класу обробляються як об’єкти базового класу (в параметрах методів, в колекціях та ін.);
* **Варіант 2:** перевизначення успадкованих методів, визначення типу об’єкта під час виконання програми і виклик відповідного методу.

1. Скорегувати каркас проєкту.
2. Замінити заглушки в проєкті конкретними реалізаціями методів, властивостей тощо.
3. Запустити unit-тести, досягти проходження всіх unit-тестів.
4. Спроєктувати меню додатку, пункти меню мають відповідати предметній області і бути логічними.
5. Реалізувати основну програму (файл Program.cs) у відповідності до спроєктованого меню.
6. Виконати функціональне тестування всіх пунктів меню.
7. Оформити звіт:

* Титульний аркуш
* Завдання
* Опис предметної області
* Діаграма класів
* Демонстрація реалізації поліморфізму
* Результати запуску unit-тестів.
* Результати функціонального тестування
* Програмний код класів
* Програмний код основної програми

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.

Діаграма класів

Рисунок 1 – Згенерована діаграма класів на основі файлів програми в середовищі розробки

Демонстрація реалізації поліморфізму

У розробці було використано концепцію поліморфізму для обробки об'єктів похідного класу ElementaryStudent як об'єктів базового класу Student.

Детальний програмний код базового та похідного класів подано в додатках **Ф та Г**.

Даний підхід дозволяє передавати екземпляри класу ElementaryStudent та Student через параметри методів, зберігати їх у колекціях та інших структурах даних, якщо очікується об'єкт базового класу Student.

Приклад використання поліморфізму подано нижче:

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

// Створення об'єктів різних класів, але базового типу

Student student1 = new Student("Іван", "Іванов", 20);

ElementaryStudent elementaryStudent1 = new ElementaryStudent("Петро", "Петров", 8);

// Додавання об'єктів до колекції типу Student

studentsList.Add(student1);

studentsList.Add(elementaryStudent1);

// Виклик методів базового класу на об'єктах похідного класу

foreach (Student student in studentsList)

{

student.ChooseSubject(new Subject("Математика"));

}

Результати запуску unit-тестів.

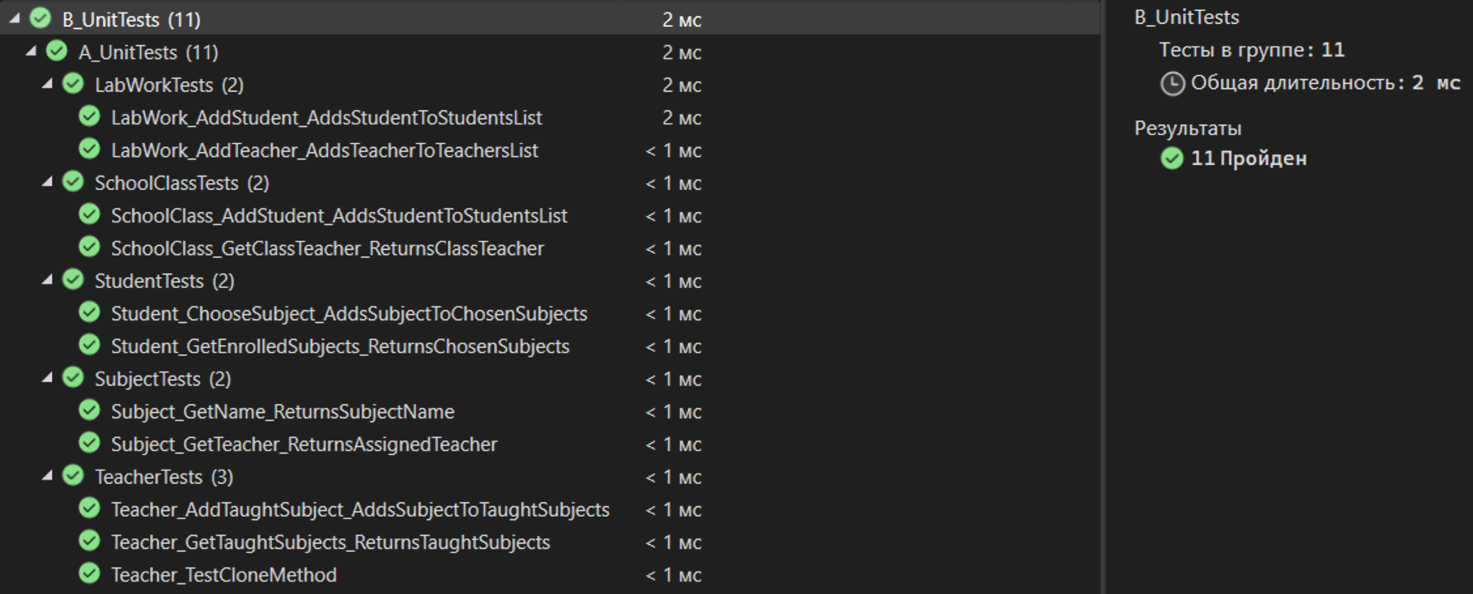


Рисунок 2 – всі тести розроблені раніше пройдені успішно

Результати функціонального тестування

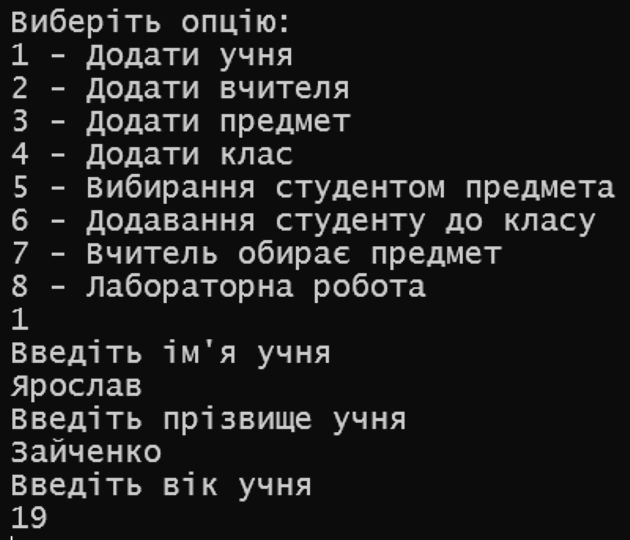


Рисунок 3 – додавання учня в програму

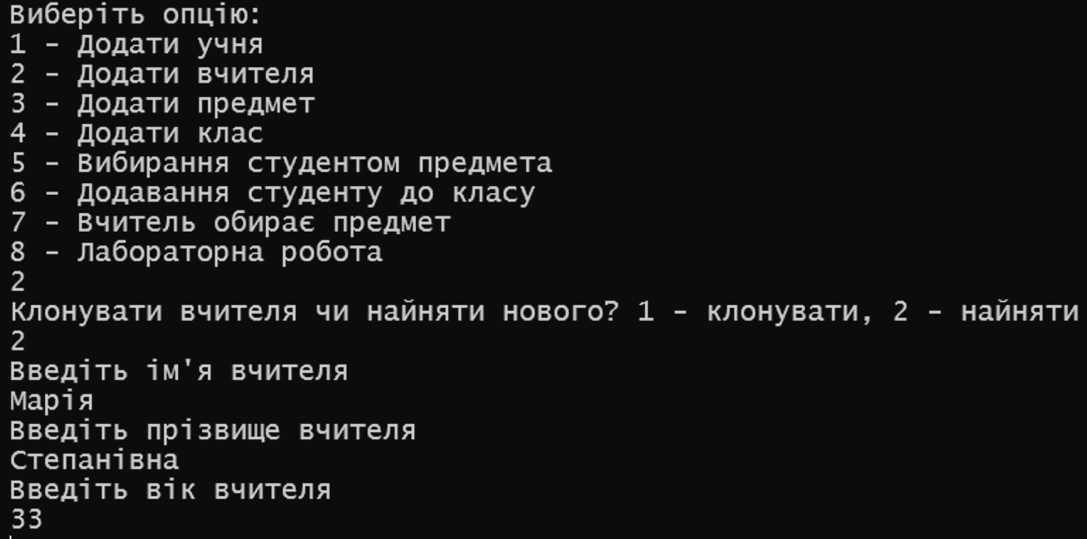


Рисунок 4 – додавання вчителя в програму

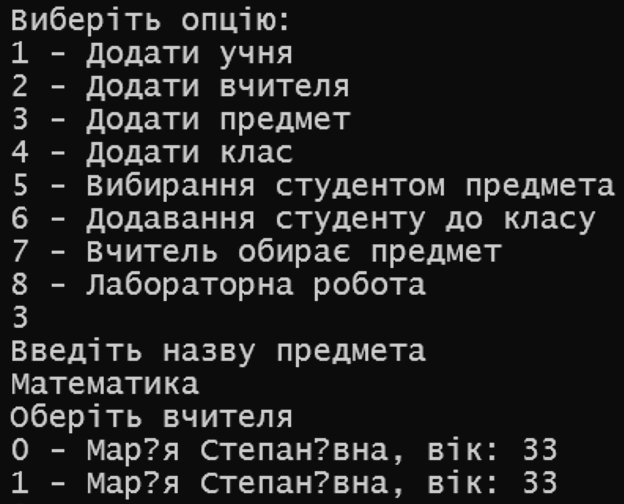


Рисунок 5 – додавання предмета в програму

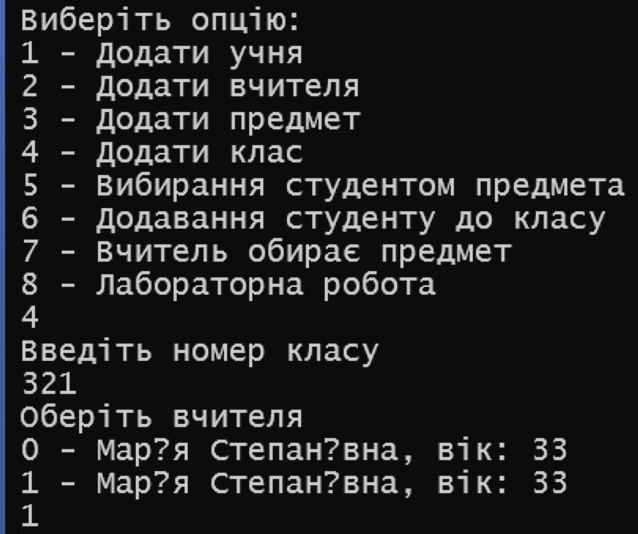


Рисунок 6 – додавання класу з вибором класного керівника в програму

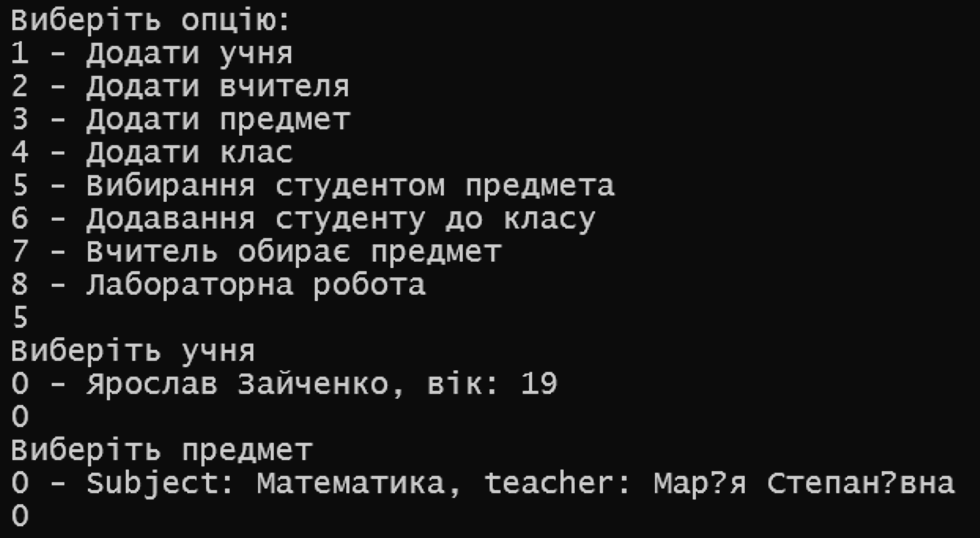


Рисунок 7 – вибір учнем предмету

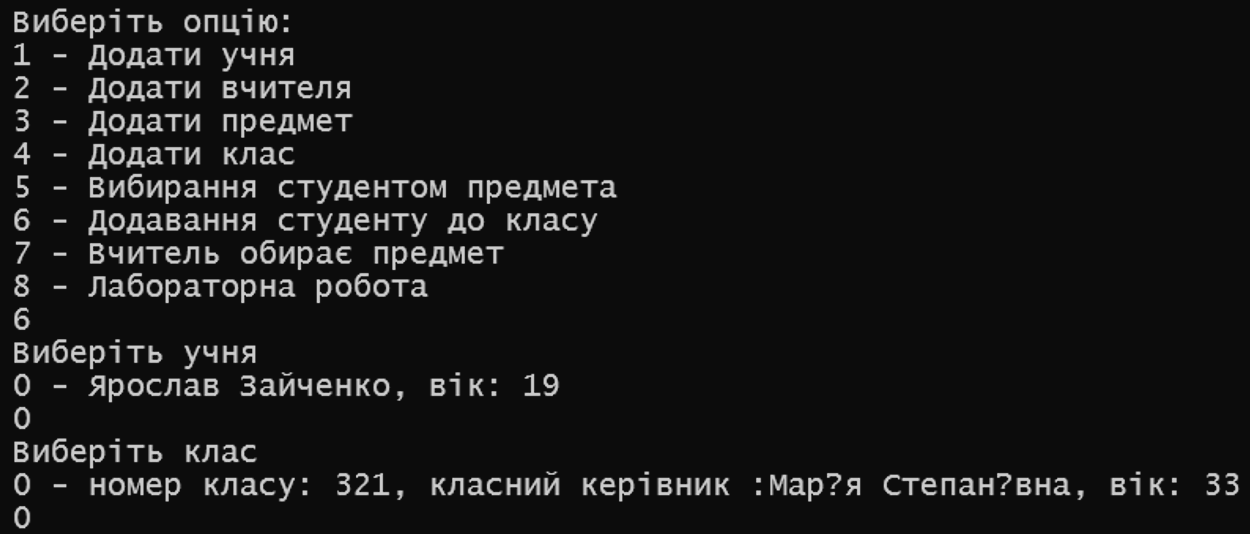


Рисунок 8 – додавання учня до класу

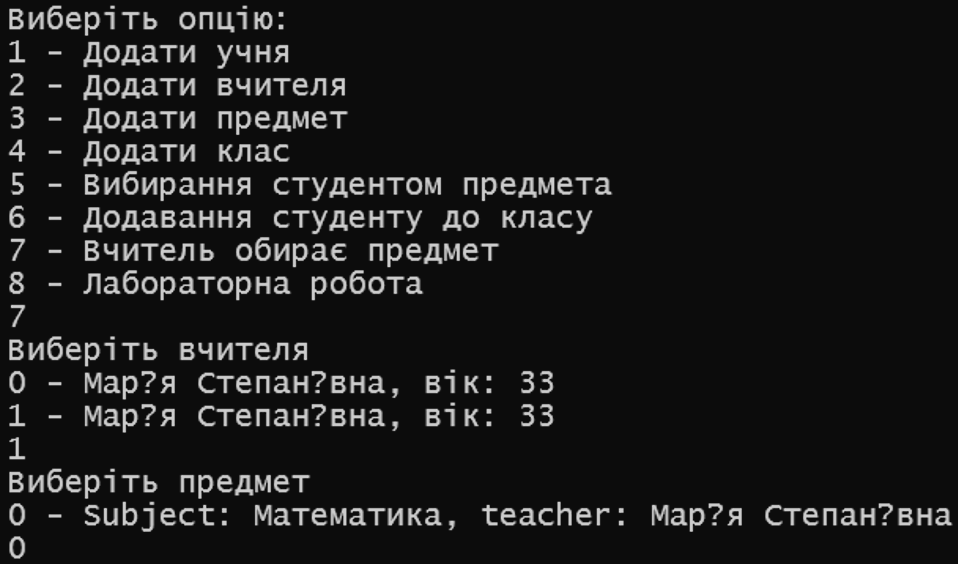


Рисунок 9 – надання предмету викладачу

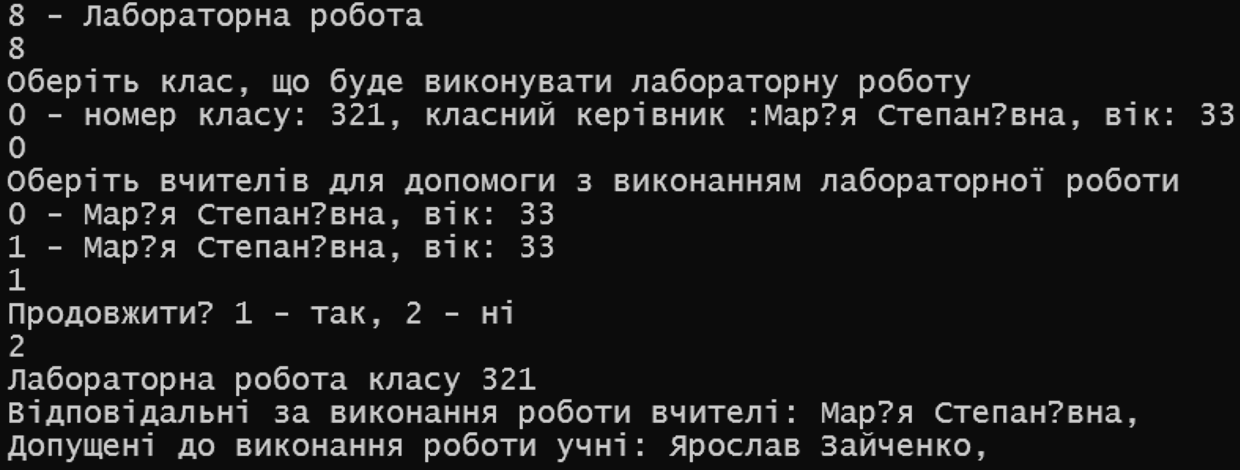


Рисунок 10 – проведення лабораторної, з вибором класу та викладача

Реалізація коду основної програми

Детальний код основної програми можна переглянути в додатку А.

Реалізація класів програми

Детальний код класів можна переглянути в додатках Б - З.

# ВИСНОВОК

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project\_part\_A

{

public class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

new SchoolSystem();

}

}

}

Важливим аспектом удосконалення була додаткова імплементація інтерфейсів .NET, таких як IClonable, IComparable, IEnumereable. Це дозволило розширити можливості класів у проєкті та використовувати стандартні механізми клонування, порівняння та перебору.

Множинна реалізація інтерфейсів стала важливим кроком у забезпеченні гнучкості та розширюваності коду. Кожен клас може тепер мати низку різних інтерфейсів, що визначає його поведінку та можливості.

Додавання абстрактного класу, який має не тільки абстрактні методи, але й абстрактні властивості, розширило базовий функціонал та стало основою для успадкування нових та існуючих класів. Це сприяло створенню єдиної архітектури з відокремленням загальної логіки.

У реалізації поліморфізму обрано варіант, де об'єкти похідного класу обробляються як об'єкти базового класу. Це надає можливість використовувати об'єкти похідного класу там, де очікується об'єкт базового класу, що робить код більш гнучким та розширюваним.

У підсумку, завдяки цим змінам, проєкт став більш структурованим, гнучким та піддається легким модифікаціям. Основна програма була спроєктована відповідно до предметної області, що забезпечує користувачам логічний та зручний інтерфейс. Усі ці кроки були важливими для досягнення мети проєкту та надання високоякісного, розширюваного та тестованого програмного продукту.

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project\_part\_A

{

public class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

new SchoolSystem();

}

}

}

***Додаток Б***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; }

private List<Subject> Subjects { get; }

public SchoolSystem()

{

Classes = new List<SchoolClass>();

Teachers = new List<Teacher>();

Students = new List<Student>();

Subjects = new List<Subject>();

MainMenu();

}

public void MainMenu()

{

try

{

Console.WriteLine("Виберіть опцію:\n1 - Додати учня\n2 - Додати вчителя\n3 - Додати предмет\n4 - Додати клас\n" +

"5 - Вибирання студентом предмета\n6 - Додавання студенту до класу\n7 - Вчитель обирає предмет\n8 - Лабораторна робота");

switch (Console.ReadLine())

{

case "1":

AddStudent();

break;

case "2":

AddTeacher();

break;

case "3":

AddSubject();

break;

case "4":

AddClass();

break;

case "5":

StudenChoosesSubject(); break;

case "6":

AddStudentToClass(); break;

case "7":

TeacherChoosesSubject(); break;

case "8":

AssignLabWork(); break;

default:

throw new Exception("Немає такого вибору");

}

}

catch (Exception ex)

{

GoToMenu(ex);

}

}

public void GoToMenu()

{

Console.ReadLine();

Console.Clear();

MainMenu();

}

public void GoToMenu(Exception exception)

{

Console.WriteLine(exception.Message);

GoToMenu();

}

public void AssignLabWork()

{

Console.WriteLine("Оберіть клас, що буде виконувати лабораторну роботу");

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

{

Console.WriteLine(i + " - " + Classes[i]);

}

SchoolClass schoolClass = Classes[int.Parse(Console.ReadLine())];

List<Student> studentstoassign = schoolClass.Students;

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

while (true)

{

Console.WriteLine("Оберіть вчителів для допомоги з виконанням лабораторної роботи");

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

{

Console.WriteLine(i + " - " + Teachers[i]);

}

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

if (!teachersassign.Contains(Teachers[pos]))

teachersassign.Add(Teachers[pos]);

else

Console.WriteLine("Такий вчитель уже залучений");

Console.WriteLine("Продовжити? 1 - так, 2 - ні");

if (Console.ReadLine() == "2")

break;

}

if (teachersassign.Count == 0 || studentstoassign.Count == 0)

throw new Exception("Недостатньо учасників для виконання лабораторної роботи");

LabWork labWork = new(schoolClass);

foreach (var student in studentstoassign)

{

labWork.AddStudent(student);

}

foreach (Teacher teacher in teachersassign)

{

labWork.AddTeacher(teacher);

}

labWork.PrintToDisplay();

GoToMenu();

}

public void AddClass()

{

if (Teachers.Count == 0)

{

throw new Exception("Немає можливих вчителів");

}

Console.WriteLine("Введіть номер класу");

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

if (GetClasses().Any(u => u.ClassNumber == number))

throw new Exception("Такий клас уже існує");

Console.WriteLine("Оберіть вчителя");

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

{

Console.WriteLine(i + " - " + Teachers[i].ToString());

}

Teacher tempteacher = Teachers[int.Parse(Console.ReadLine())];

Classes.Add(new SchoolClass(number, tempteacher));

GoToMenu();

}

public List<SchoolClass> GetClasses()

{

return Classes;

}

public void AddTeacher()

{

Console.WriteLine("Клонувати вчителя чи найняти нового? 1 - клонувати, 2 - найняти");

string result = Console.ReadLine();

if (result == "1")

{

Console.WriteLine("Оберіть вчителя");

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

{

Console.WriteLine(i + " - " + Teachers[i]);

}

Teacher tempteacher = Teachers[int.Parse(Console.ReadLine())];

Teachers.Add(tempteacher);

}

else if (result == "2")

{

Console.WriteLine("Введіть ім'я вчителя");

string firstname = Console.ReadLine();

Console.WriteLine("Введіть прізвище вчителя");

string lastname = Console.ReadLine();

Console.WriteLine("Введіть вік вчителя");

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

Teachers.Add(new Teacher(firstname, lastname, age));

}

else throw new Exception("Немає такої опції");

GoToMenu();

}

public List<Teacher> GetTeachers()

{

return Teachers;

}

public void AddStudent()

{

Console.WriteLine("Введіть ім'я учня");

string firstname = Console.ReadLine();

Console.WriteLine("Введіть прізвище учня");

string lastname = Console.ReadLine();

Console.WriteLine("Введіть вік учня");

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

if (age > 10)

Students.Add(new Student(firstname, lastname, age));

else

Students.Add(new ElementaryStudent(firstname, lastname, age));

GoToMenu();

}

public void AddStudentToClass()

{

Console.WriteLine("Виберіть учня");

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

{

Console.WriteLine(i + " - " + Students[i].ToString());

}

var tempstudent = Students[int.Parse(Console.ReadLine())];

Console.WriteLine("Виберіть клас");

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

{

Console.WriteLine(i + " - " + Classes[i].ToString());

}

Classes[int.Parse(Console.ReadLine())].AddStudent(tempstudent);

GoToMenu();

}

public void StudenChoosesSubject()

{

Console.WriteLine("Виберіть учня");

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

{

Console.WriteLine(i + " - " + Students[i].ToString());

}

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

Console.WriteLine("Виберіть предмет");

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

{

Console.WriteLine(i + " - " + Subjects[i].ToString());

}

Students[student].ChooseSubject(Subjects[int.Parse(Console.ReadLine())]);

GoToMenu();

} public void TeacherChoosesSubject()

{ Console.WriteLine("Виберіть вчителя");

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

{ Console.WriteLine(i + " - " + Teachers[i].ToString()); } int teacher = int.Parse(Console.ReadLine()); Console.WriteLine("Виберіть предмет");

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

{Console.WriteLine(i + " - " + Subjects[i]); } Teachers[teacher].AddTaughtSubject(Subjects[int.Parse(Console.ReadLine())]);

GoToMenu(); } public List<Student> GetStudents()

{ return Students;

} public void AddSubject()

{ Console.WriteLine("Введіть назву предмета");

string subjectname = Console.ReadLine();

if (GetSubjects().Any(u => u.Name == subjectname))

throw new Exception("Такий предмет уже існує");

Console.WriteLine("Оберіть вчителя");

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

{ Console.WriteLine(i + " - " + Teachers[i].ToString());

}

Teacher tempteacher = Teachers[int.Parse(Console.ReadLine())];

Subjects.Add(new Subject(subjectname, tempteacher));

GoToMenu();

}

public List<Subject> GetSubjects()

{

return Subjects;

}

}

}

***Додаток В***Машинний лістинг класу **ElementaryStudent.cs**

namespace Project\_part\_A.Classes

{

public class ElementaryStudent : Student

{

public override int Age

{

get => base.Age; set

{

if (value > 10)

{

throw new Exception("Надто дорослий для початкової школи");

}

base.Age = value;

}

}

public ElementaryStudent(string firstName, string lastName, int age) : base(firstName, lastName, age)

{

}

public override void ChooseSubject(Subject subject)

{

if (ChosenSubjects.Count > 4)

{

throw new Exception("Надто багато вибіркових дисциплін для початкової школи");

}

ChosenSubjects.Add(subject);

}

}

}

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

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public class LabWork : IPrintable

{

public SchoolClass Class { get; set; }

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

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

public LabWork(SchoolClass schoolclass)

{

Class = schoolclass;

}

public void AddTeacher(Teacher teacher)

{

Teachers.Add(teacher);

}

public List<Teacher> GetTeachers()

{

return Teachers;

}

public void AddStudent(Student student)

{

Students.Add(student);

}

public List<Student> GetStudents()

{

return Students;

}

public void PrintToDisplay()

{

Console.WriteLine("Лабораторна робота класу " + Class.ClassNumber);

string result = "";

foreach (var teacher in Teachers)

{

result += teacher.FirstName + " " + teacher.LastName + ", ";

}

Console.WriteLine("Відповідальні за виконання роботи вчителі: " + result);

result = "";

foreach (var student in Students)

{

result += student.FirstName + " " + student.LastName + ", ";

}

Console.WriteLine("Допущені до виконання роботи учні: " + result);

}

}

}

***Додаток Ґ***

Машинний лістинг класу **Person.cs**

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public abstract class Person

{

public abstract string FirstName { get; set; }

public abstract string LastName { get; set; }

public abstract int Age { get; set; }

public override string ToString()

{

return $"{FirstName} {LastName}, вік: {Age}";

}

}

}

***Додаток Д***

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)

{

ClassNumber = classNumber;

ClassTeacher = classTeacher;

}

public void AddStudent(Student student)

{

Students.Add(student);

}

public List<Student> GetStudents()

{

return Students;

}

public Teacher GetClassTeacher()

{

return ClassTeacher;

}

public override string ToString() {

string res = $"номер класу: {ClassNumber}, класний керівник :{ClassTeacher.ToString()} ";

return res;

}

}

}

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

namespace Project\_part\_A.Classes

{

public class Student : Person

{

private int age;

public override string FirstName { get; set; }

public override string LastName { get; set; }

public override int Age { get => age; set => age = value; }

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

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

{

FirstName = firstName;

LastName = lastName;

Age = age;

}

public virtual void ChooseSubject(Subject subject)

{

ChosenSubjects.Add(subject);

}

public List<Subject> GetEnrolledSubjects()

{

return ChosenSubjects;

}

}

}

*Додаток Є*Машинний лістинг класу Subject.cs

using Project\_part\_A.Interfaces;

namespace Project\_part\_A.Classes

{

public class Subject

{

public string Name { get; set; }

public Teacher Teacher { get; set; }

public Subject(string name, Teacher teacher)

{

Name = name;

Teacher = teacher;

}

public string GetName()

{

return Name;

}

public Teacher GetTeacher()

{

return Teacher;

}

public override string ToString()

{

string res= $"Subject: {Name}";

if (Teacher != null)

{

res+=$", teacher: {Teacher.FirstName} {Teacher.LastName}";

}

return res;

}

}

}

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

namespace Project\_part\_A.Classes

{

public class Teacher : Person, ICloneable

{

private int age;

public override string FirstName { get; set; }

public override string LastName { get; set; }

public override int Age

{

get => age; set

{

if (value < 23)

{

throw new Exception("Надто молодий для вчителя");

}

age = value;

}

}

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

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

{

FirstName = firstName;

LastName = lastName;

Age = age;

}

public void AddTaughtSubject(Subject subject)

{

TaughtSubjects.Add(subject);

}

public List<Subject> GetTaughtSubjects()

{

return TaughtSubjects;

}

public object Clone()

{

return new Teacher(FirstName, LastName, Age);

}

}

}

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

namespace Project\_part\_A.Interfaces

{

public interface IPrintable

{

public void PrintToDisplay();

}

}