**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра програмних систем і технологій**

**О. С. Бичков, Т.В. Ковалюк**

# Методичні вказівки до лабораторних занять з дисципліни

# «Вступ до об’єктно-орієнтованого програмування. Мова С# і платформа MS.Net Framework »

для студентів спеціальності

121 «Інженерія програмного забезпечення»

освітнього рівня «бакалавр»

Київ 2022

## Вступ

Практикум складається з двох частин: обов’язкової частини з 6 лабораторних робіт, підсумковий рейтинг яких складає 26 балів, та не обов’язкової частини – проєкту, із рейтингом 24 бали. За виконання двох МКР студент може отримати додатково 10 балів. Отже, за весь практикум студент може мати 60 балів.

Перша та друга лабораторні роботи виконуються в рамках практичних занять, на аудиторному занятті, без оформлення звітів. Індивідуальні варіанти передбачені. Перевірка виконаного завдання з лабораторних №№1, 2 здійснюється викладачем або делегується студентам, які показали гарні результати в першому семестрі, вміють самостійно опановувати теоретичний матеріал, мають досвід участі в МАН та олімпіадах, обізнані з мовою програмування С#.

Лабораторні роботи №№ 3 – 6 виконуються як традиційні лабораторні, із індивідуальним захистом роботи та оформленням звітів.

Проєкт пропонується виконувати за технологією SCRUM. Проєкт передбачає 6 версій (releases), кожна з яких оцінюється у 4 бали. Кожна версія (release) проєкту – це готовий до використання застосунок з обмеженою функціональністю відповідно до пріоритетів задач (бізнес-процесів), що узгоджуються з викладачем (product owner).

**За готовий проєкт, тобто за 6 версій, можна отримати 24 бали**. Студент самостійно визначає, на якій версії (релізу) він закінчить свою розробку. Відповідно до кількості релізів визначається сума балів за проєкт.

### Звіт з лабораторних робіт

Підготовка технічної документації на систему, програму, пристрій є важливою частиною всього комплексу робіт, які виконує розробник. Студент повинен вміти складати технічну документацію на програмне забезпечення, яке він розробляє. Тому звітам про лабораторні роботи з дисципліни «Основи програмування» приділяється особлива увага викладача. Рекомендується виконувати звіти у форматі гіпертекстового документа з гіперпосиланнями на лабораторні роботи, пункти вмісту звіту, файли або проекти, які містять код програм. Для створення гіпертекстового звіту припустимо використовувати конструктор Google Site. Студент може використовувати кросплатформні системи документування вихідного коду, наприклад, DOXYGEN (<https://www.doxygen.nl/download.html#google_vignette>) з вбудованою програмою GraphViz, які на основі вихідного коду, що містить коментарі спеціального вигляду, створюють зручну HTML документацію, з посиланнями на класи, файли, діаграми класів, викликів методів (HIPO) тощо.

#### Зміст звіту з лабораторної роботи

1. Мета роботи (береться з методичних вказівок)
2. Умова задачі (береться з методичних вказівок)
3. Аналіз задачі, теоретичні обґрунтування вибраного методу розв’язання задачі, розрахункові формули, посилання на літературні джерела
4. Графічне зображення структури програми:
   1. HIPO діаграма для звітів з лабораторних робіт №№ 1,2
   2. Діаграма класів (class diagram) для звітів з лабораторних робіт №№ 3, 4, 5, 6
   3. Діаграма діяльності (activity diagram) для звіту з лабораторних робіт №№ 5, 6
   4. Діаграма послідовності (sequence diagram) для звіту з лабораторних робіт №№ 5,6
5. Текст програми з коментарями. В Doxygen додаються автоматично через гіперпосилання.
6. Результати виконання роботи у вигляді екранних копій.
7. Аналіз достовірності результатів у вигляді контрольних розрахунків за допомогою різних інструментів (калькулятор, Excel, ручний розрахунок, online калькулятори матричних розрахунків: <https://matrixcalc.org/uk/>, базу знань і набір обчислювальних [алгоритмів](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) <https://www.wolframalpha.com/>).
8. Висновки про досягнуті в роботі результати, у т.ч. **недоопрацьовані можливости програми, напрямки її подальшого удосконалення**

**Усі звіти зі всіх лабораторних робіт в одному пакеті html документів**

# Розділ 1. Введення в мову програмування C#

## Лабораторна робота № 1 Методи введення, виведення, перетворення типів, оператори: умовний, вибору, циклу в мові C#

### Рейтинг лабораторної роботи №1

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 5 завдань та його захист | 0,5\*5 = 2,5 | 14 Лютого |
| 2 | Захист | 0,5 |  |
| 3 | Звіт | 0,5 |  |
| Разом за роботу | | 3,5 |  |

### Мета роботи

Опанувати логічні структури програмування мовою C#, оператори розгалуження, циклів, ознайомитися з методами введення та виведення даних на консоль, виконати завдання, що передбачають такі дії:

1. Прості обчислення з перетворенням типів даних. Метод Parse(), TryParse(), методи класу Convert

2. Прості арифметичні вирази, обчислення значення функцій при заданих аргументах, прості циклічні розрахунки.

3. Перетворення з арифметичного типу в тип string. Метод ToString

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію.

2. Прочитайте методичні рекомендації до лабораторної роботи та виконайте наведені в ній приклади (вони всі працездатні)

3. Зверніть увагу на те, що при введенні чисел з консолі у змінні арифметичного типу потрібно застосовувати явне приведення типу, так як неявного перетворення з типу **string** до арифметичного немає.

4. При виконанні завдань зверніть увагу на оператор **switch** (перемикач). В С# тип виразу в цьому операторі може бути не тільки переліченим (цілим та символьним), а і типу string.

### Порядок виконання роботи

1. Створити директорію Lab1, в якій буде розміщуватися проект цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного проекту

### Приклади виконання завдань

#### 1. Прості обчислення з перетворенням типів даних. Метод Parse

Розглянемо приклад створення простого консольного застосунку, який виконує прості арифметичні обчислення. Запустимо Visual Studio, виберемо тип проекту Console Application, дамо йому назву **Lab1\_1** і натиснемо кнопку **OK**.

Створимо програму, яка вводить два цілі числа, знаходить їх суму і виводить результат на консоль. Потім вводить два дійсні числа, обчислює їх частку і також виводить на консоль.

Ось таким може бути код цієї програми. Його можна просто скопіювати в VS і виконати.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_1

{

class Program

{

static void Main(string[] args)

{

//Знаходження суми цілих чисел

Console.WriteLine("Введіть перше число");

int a1 = int.Parse(Console.ReadLine());

Console.WriteLine("Введіть друге число");

int a2 = int.Parse(Console.ReadLine());

Console.WriteLine("Сума чисел a1+a2= " + (a1 + a2));

//Знаходження частки дійсних чисел

Console.WriteLine("Введіть перше число");

double d1 = double.Parse(Console.ReadLine());

Console.WriteLine("Введіть друге число");

double d2 = double.Parse(Console.ReadLine());

Console.WriteLine("Частка чисел d1/d2= " + (d1/d2));

Console.ReadKey();

}

}

}

Дані, які вводяться з консолі мають тип string (це рядки тексту). Тому введені числа потрібно перетворити в арифметичний тип. В нашому прикладі виконується явне перетворення типів з типу string в типи **int** і **double** з використанням методу Parse і неявне - при виведенні результатів обчислень на консоль.

Розглянемо ще приклад явного перетворення типу.

**Приклад**

string str = Console.ReadLine();

int i = int.Parse(str);

float f = float.Parse(str);

double d = double.Parse(str);

#### 2. Прості арифметичні вирази

Розглянемо дещо складніший приклад обчислення арифметичного виразу. Нехай нам потрібно обчислити значення поліному  *р=ax3+bx2 + cx+d* для будь-якого дійсного *x* (вводиться з консолі). Значення *a,b,c,d* є цілими і ініціалізуються в коді. Текст програми може бути таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_2

{

class Program

{

static void Main(string[] args)

{

//обчислити значення поліному р=ax^3+bx^2+cx+d

int a = 100;

int b = 20;

int c = 25;

Console.WriteLine("Введіть x");

double x = double.Parse(Console.ReadLine());

double p = a \* x \* x \* x + b \* x \* x + c \* x + 30;

Console.WriteLine("p = " + p);

Console.ReadKey();

}

}

}

#### 3. Клас Convert

Клас Convert містить статичні методи, які також можна використовувати для перетворень типів. Розглянемо приклад використання цього класу.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab1\_2

{

class Program

{

/// <summary>

/// Тестування методів класу Convert

/// </summary>

static void Main(string[] args)

{

string s;

byte b;

int n;

double x;

bool flag;

char sym;

DateTime dt;

sym = '7';

s = Convert.ToString(sym);

x = Convert.ToDouble(s);

n = Convert.ToInt32(x);

b = Convert.ToByte(n);

flag = Convert.ToBoolean(b);

x = Convert.ToDouble(flag);

s = Convert.ToString(flag);

s = "300";

n = Convert.ToInt32(s);

s = "14.09";

s = "14.09.2008";

dt = Convert.ToDateTime(s);

}

}

}

#### 4. Перетворення з арифметичного типу в тип string. Метод ToString

Так як метод ToString() визначений в базовому класі object, він має свої реалізації для всіх арифметичних типів даних. Далі наведено простий приклад з лекції 2, який демонструє застосування цього методу.

В першому варіанті використовується **явне** перетворення типу з арифметичного до типу string, а у другому - **неявне**.

Варіант 1.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_3

{

class Program

{

/// <summary>

/// Демонстрація перетворення в рядок

/// даних різного типа.

/// </summary>

static void Main(string[] args)

{

string name;

int age;

double salary;

name = "Василь Іванов";

age = 20;

salary = 2700;

string s = "Ім'я: " + name +

". Вік: " + age.ToString() +

". Зарплата: " + salary.ToString();

Console.WriteLine(s);

Console.ReadKey();

}

}

}

Варіант 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab2\_3

{

class Program

{

/// <summary>

/// Демонстрація перетворення в рядок

/// даних різного типа.

/// </summary>

static void Main(string[] args)

{

string name;

int age;

double salary;

name = "Василь Іванов";

age = 20;

salary = 2700;

string s = "Ім'я: " + name +

". Вік: " + age +

". Зарплата: " + salary;

Console.WriteLine(s);

Console.ReadKey();

}

}

}

#### 5. Створення консольних проектів з використанням операторів if

Створимо консольний проект для організації діалогу таким чином, щоб реалізувати наведений нижче алгоритм:

1. Вивести на консоль запрошення для введення імені студента.
2. Ввести своє ім'я і зберегти в текстовому рядку: string myName.
3. Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
4. Якщо щось введено, то вивести рядок привітання.
5. Вивести рядок із запитом віку
6. Якщо нічого не введено, то вивести повідомлення про це і завершити роботу.
7. Якщо вік введено, привести його значення до цілого (int myAge).
8. Якщо значення myAge<15, вивести повідомлення "Ви ще не студент. "
9. Якщо значення myAge>40, вивести повідомлення "Вчитися ніколи не пізно!"

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleHello1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введіть ім'я");

string myName = Console.ReadLine();

if (myName.Length == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

Console.WriteLine("Здрастуйте, "+ myName);

Console.WriteLine("Скільки Вам років?");

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

if (myAge == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

{

if (myAge<15)

Console.WriteLine("Ви ще не студент");

else if (myAge>40)

Console.WriteLine("Вчитися ніколи не пізно!");

else

Console.WriteLine("Ваш вік" +myAge);

}

Console.ReadKey();

}

}

}

В цьому прикладі використовується декілька операторів розгалуження **if** (останні три вкладені). Зверніть увагу, що у C# в операторі **if** діють ті самі правила синтаксису, що і в С++, якщо в тілі оператора є лише один оператор блоку, то дужки не потрібні, наприклад:

if (myName.Length == 0)

Console.WriteLine("Ви нічого не ввели, прощавайте");

else

Console.WriteLine("Здрастуйте, "+ myName);

Крім цього виконується явне перетворення типів з типу **string** в тип **int** з використанням методу Parse і неявне при виведенні віку на консоль

#### 6. Створення консольних проектів з використанням операторів switch

Розробимо простий консольний калькулятор для виконання арифметичних операцій. Вхідні дані будемо вводити з консолі, результати виводити на консоль. Приклад коду з лекції

Консольний калькулятор на 4 дії

using System;

namespace ConsoleCalculator

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

string buf;

double a, b, res;

Console.WriteLine("Введіть перший операнд:");

a = double.Parse(Console.ReadLine());

Console.WriteLine("Введіть знак операції");

char op = (char)Console.Read();

Console.ReadLine();

Console.WriteLine("Введіть другий операнд:");

b = double.Parse(Console.ReadLine());

bool ok = true;

switch (op)

{

case '+': res = a + b; break;

case '-': res = a - b; break;

case '\*': res = a \* b; break;

case '/': res = a / b; break;

default: res = double.NaN;

ok = false; break;

}

if (ok) Console.WriteLine("Результат: " + res);

else Console.WriteLine("Неприпустима операція");

}

}

}

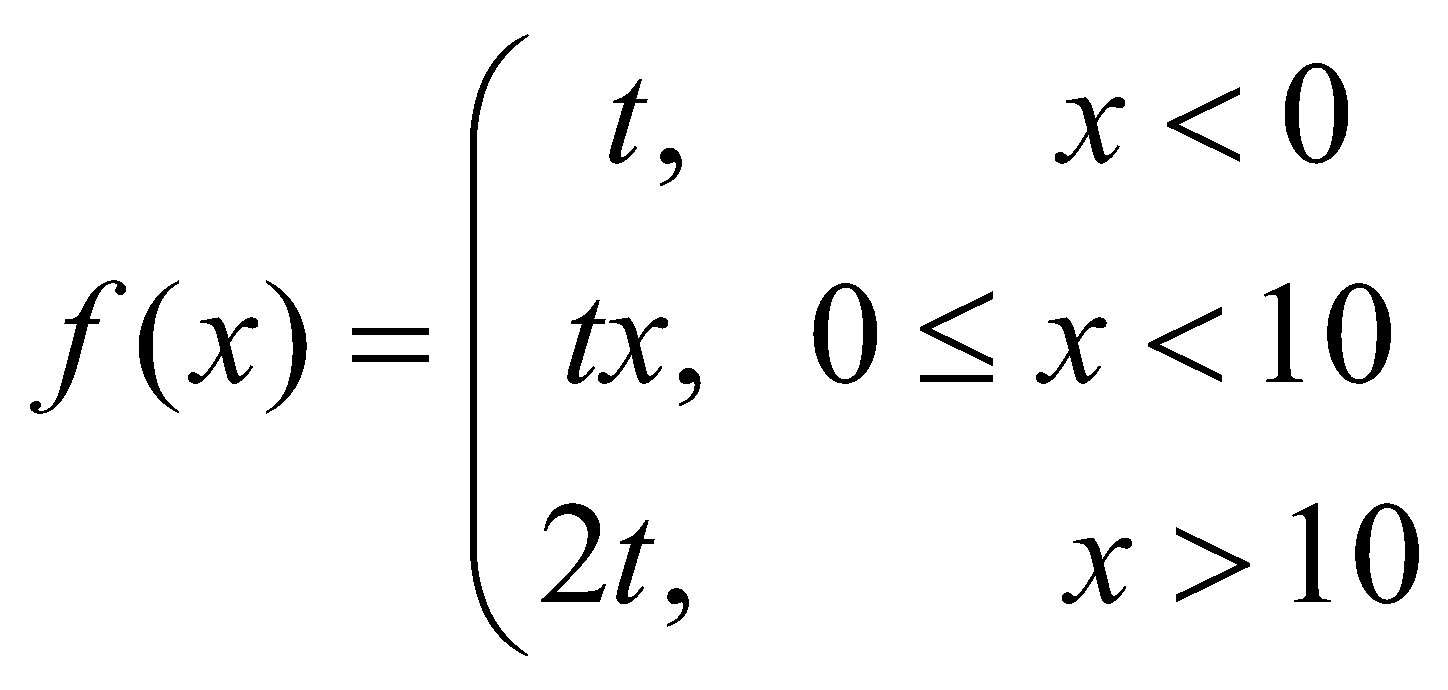
Зверніть увагу, що тип виразу в операторі **switch** є **char**. Також зверніть увагу, що перетворення з типу **string** до типу **char** повинно бути явним, тому що тип char є типом-значенням, а тип **string** – посилальним. **char op = (char)Console.Read();**

Для виходу з оператора **switch** при виконанні умови використовується оператор переходу **break**.

#### 7. Створення консольних проектів з використанням операторів циклу

##### 7.1. Цикл з передумовою while

Розглянемо приклад з лекції 4, в якому використовується цикл з передумовою. Потрібно написати програму, яка виводить для аргументу х, що змінюється в заданих границях із заданим кроком, таблицю значень наступної функції:



Назвемо xn – початкове значення аргументу, xk –кінцеве значення аргументу, dx – крок зміни аргументу, t – параметр. Усі величини є дійсні числа типу double. Програма повинна виводити таблицю, що складається з двох стовпців: значень аргументу і відповідних ним значень функції. Таблиця повинна мати заголовок.

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

double Xn = -2, Xk = 12, dX = 2, t = 2, y;

Console.WriteLine("| x | y |"); // заголовок таблиці

double x = Xn;

while (x <= Xk)

{

y = t;

if (x >= 0 && x < 10) y = t \* x;

if (x >= 10) y = 2 \* t;

Console.WriteLine("| {0,6} | {1,6} |", x, y);

x += dX;

}

Console.ReadKey();

}

}

}

Зверніть увагу, що в тілі циклу використовується два оператори **if**. На кожному кроці циклу обчислюється значення функції. Цикл завершиться, коли умова циклу не буде виконана (тобто значення x стане більше 12).

##### 7.2. Цикл з постумовою do…while

Цей тип циклу застосовується в тих випадках, коли тіло циклу необхідно обов'язково виконати хоч б один раз. В поданому нижче прикладі на консоль виводиться текст "Будете вчитися?" допоки не буде введено"y".

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

char answer;

do

{

Console.WriteLine("Будете вчитися?");

answer = (char)Console.Read();

Console.ReadLine();

} while (answer != 'y');

}

}

}

##### 7.3. Цикл з параметром (for)

Цей тип циклу використовується коли відомі границі циклу (нижня і верхня). *Приклад 4.8 з лекції 3* демонструє використання цього циклу. В цьому прикладі обчислюється сума чисел від 1 до 100. Спочатку створюється змінна цілого типу s, яка ініціалізується нулем. Потім в тілі циклу на кожному кроці до неї додається чергове число: 1,2,3,....100.

int s = 0;

for ( int i = 1; i <= 100; i++ )

s += i;

Зверніть увагу, що в тілі циклу відсутні фігурні дужки блоку. Це тому, що в циклі лише один оператор. Також зверніть увагу на використання операції інкремента:

s += i;замість присвоєння: s = s+i;

Обидва оператори дозволені в C# так само як і в C++.

Розглянемо ще один приклад використання оператора циклу з параметром.

**Постановка задачі.**

Написати метод обчислення функції sin(x), використовуючи розкладання в ряд Тейлора за формулою:

\sin x = \sum^{\infin}_{n=0} \frac{(-1)^n}{(2n+1)!} x^{2n+1}

При реалізації задачі використати цикл **for**.

Число ***x*** – це значення кута в радіанах, ***n*** – кількість членів ряду. Числа ***x, n*** ввести з консолі. Обчислення функції sin(x) за допомогою розкладання в ряд Тейлора виконати в окремому методі. Обчислення факторіала та степені числа ***х*** виконати, використовуючи рекурентні обчислення. Порівняти отримане значення функції sin(x) за рядом Тейлора із стандартним методом обчислення sin(x).

using System;

namespace TaylorSin

{

class Program

{

static double CalcSin(double x, int n)

{

//обчислення розкладання sin в ряд

double result = x; //сума членів ряду Тейлора

double item = x; //поточний член ряду Тейлора

int fact = 2; //початкове значення факторіала знаменника

for (int i = 0; i < n; i++)

{

fact \*= (fact + 1);

item \*= (-x \* x) / fact; //рекурентне обчислення члена ряду

result += item;

fact += 2; //перехід до наступного значення факторіала знаменника

}

return result;

}

static void Main(string[] args)

{

Console.WriteLine("Введіть x - кут в радіанах");

double x = double.Parse(Console.ReadLine());

Console.WriteLine("Введіть показник степеня n");

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

//виклик методу обчислення sin(x) через ряд

double mysinus = CalcSin(x, n);

//виклик методу з класу Math

double sinus = Math.Sin(x);

double delta = sinus - mysinus;

Console.WriteLine("mysinus= {0},sin={1},delta={2}", mysinus, sinus, delta);

Console.ReadKey();

}

}

}

### Варіанти завдань для лабораторної роботи № 1.

**Номер варіанту відповідає номеру прізвища студента у списку групи**

Написати в C# консольний застосунок, що реалізує завдання згідно з варіантом.

Виконання пункту 1 завдання вимагає використання методів класу Console.

Виконання пункту 2 завдання вимагає використання методів класу Math.

Виконання пункту 3 завдання вимагає використання оператору if

Виконання пункту 4 завдання вимагає використання оператору switch

Виконання пункту 5 завдання вимагає використання оператору циклу

**Студент має право проявити інішіати із застосування алгоритмів для завдань свого варіанту для удосконалення або спрощення коду.**

Не потрібно здійснювати «перевірку на дурня» через те, що подібні операції можуть здійснити такі методи, зокрема TryParce().

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислення значення поліному *p = a\*x^5 − 1 / b\*x^4+c\*x + d*. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* ввести з консолі. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі, Якщо введено не число, вивести повідомлення «Ввести число».      1. Написати функцію, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран його назву (січень,...грудень). 2. Дано натуральне число n. Обчислити добуток перших n членів ряду. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислення значення поліному *p = a\*x^4 − b\*x^3 + c\*x + d*. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* ввести з консолі. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від порядкового номера дня тижня (1,2,...7) виводить на екран його назву (понеділок,...) 2. Обчислити суму чисел в заданому у функції діапазоні. Значення чисел вводити з консолі. Якщо значення, що вводиться, виходить за межі заданого діапазону, вивести повідомлення про помилку і повторити введення. Не застосовувати масиви. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислення значення поліному p=23,5\*x^5 + 30\*x^4+10\*x+87,3. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* ввести з консолі. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від порядкового номера місяця (1,2,...12) виводить на екран пору року (зима, весна....) 2. Дано натуральне число n. Обчислити   , |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислення значення поліному p = 3,5\*x^4 + 3\*x^3+10\*x^2+8,3. Значення *a, b, c, d* – дійсні числа, які слід ініціалізувати у функції, значення змінної *x* ввести з консолі. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка в залежності від одиниці виміру часу (1,2,...24) виводить на екран частини доби (ранок, день,....) 2. Задати з консолі оцінки студента з 10 дисциплін. Обчислити загальну суму балів, найгіршу і найкращу оцінку. Не використовувати масиви. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислити площу прямокутника за заданими сторонами. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від порядкового номера кольору у спектрі (1,2,...7) виводить його назву (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий) і код RGB. 2. N суддів поставили різні оцінки одному спортсмену. Обчислити середній бал спортсмена, видаливши найменшу та найвищу суддівські оцінки. Оцінки вводити з клавіатури, Не використовуючи масиви, обраховувати суму введених значень, найменше та найбільше значення. |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Знайти довжини всіх медіан і бісектрис трикутника, якщо значення сторін трикутника введені з клавіатури. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить назву навчальної дисципліни в залежності від уподобань студента, що задаються пріоритетом 1,2,...5 дисципліни. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму для заданого х: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислити відстань від точки (x0, y0) до точки (х1, y1), значення координат яких введені з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить на консоль назву навчального закладу в залежності від середнього балу ЗНО та пріоритетів (1,2,…,5), заданих абітурієнтом. 2. Обчислити  - число Фібоначчі з номером *n*, де: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Знайти об'єм циліндра, якщо значення радіусу його основи та висоти введені з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. Обчислити значення функції в точці *х*, значення якої ввести з консолі:      1. Написати функцію, яка виводить на консоль назву університету в залежності від його консолідованим рейтингом (1,2,…,5) за 2020, що визначається інформаційним ресурсом "Освіта.ua". 2. Дано натуральне число n. Обчислити суму перших 2*n* членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислити відстань від точки (x0, y0) до прямої *ax + by + c*= 0. Значення координат точки та коефіцієнтів a, b, c прямої ввести з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка виводить на консоль назву країни в залежності від рейтингу (1,2,…,5) щастя її населення за 2020, що визначається Social Progress Index (<https://minfin.com.ua/ua/2020/10/11/53857422/> ) 2. Дано натуральне число n і дійсне число x >0. Обчислити суму перших 2n членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Знайти довжини всіх висот трикутника, якщо значення сторін a, b, c трикутника введені з консолі 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка виводить на консоль прізвище студента за його семестровим рейтингом (від 60 до 100) за результатами сесії. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Знайти об'єм конуса, якщо значення радіусу його основи та висоти введені з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви кольору та сигналів світлофора виводить на консоль назву дії, яку має виконати водій автомобіля (їхати, чекати, зупинитися, повертати, зменшити швидкість тощо). 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Обчислити площу поверхні () сфери за значенням радіусу **r** , введеним з з консолі. Результат вивести на консоль. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру ІТ-спеціальності (121, 121, …, 126) виводить на консоль її назву. 2. Дано натуральне число n і дійсне число x >0. Обчислити добуток членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Тіло починає рухатися без початкової швидкості з прискоренням *a*. Обчислити відстань, яку тіло пройде за час *t* від початку руху. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, b, с, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви місяця (січень, лютий, …) виводить на консоль його порядковий номер. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Висота ромба, проведена з вершини тупого кута, ділить сторону навпіл. Знайдіть меншу діагональ, якщо значення периметра ромба введене з клавіатури. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *a, x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви кольору у спектрі (червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий) виводить його порядковий номер і код RGB. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою меню, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати **в одному проекті Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає, чи пройде куля радіуса *r* в квадратний отвір зі стороною *a*. Дійсні значення r, a ввести з консолі. Вивести на консоль відповідне повідомлення. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними *x,* значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру року виводить на консоль назву країни, співак якої став переможцем конкурсу Євробачення, наприклад, у 2016 році перемогла Україна, 2017 рік - Португалія, і т. д. 2. Дано натуральне число n і дійсне число x >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає, чи пройде куб з ребром *a* в круглий отвір радіуса *r*. Дійсні значення *а,* *r* ввести з консолі. Вивести на консоль відповідне повідомлення. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви телеканалу виводить на консоль назву холдингу (власника), наприклад, канал «Рада», власник «Верховна Рада України», канал «1+1», холдинг «1+1 Media» і т. д. 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає, з якою швидкістю спортсмен увійде у воду, стрибаючи з *n* метрової вежі, якщо спортсмен падає з прискоренням a=9,81м/с^2, початкова швидкість v0 = 0? Значення *n* ввести з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від номеру пальця на руці виводить його назву (великий, вказівний, середній, безіменний, мізинець). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає, подібність двох трикутників за трьома сторонами, значення яких введені з консолі. Вивести на консоль відповідне повідомлення. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг та індекс рівня освіти (<https://gtmarket.ru/ratings/education-index> ). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає, подібність двох трикутників за двома сторонами та кутом між ними. Значення двох сторін та кутів трикутників увести з консолі. Вивести на консоль відповідне повідомлення. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг шкільної освітньої грамотності за версією PISA (<https://factsmaps.com/pisa-2018-worldwide-ranking-average-score-of-mathematics-science-reading/> ). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |
|  | Створити консольний застосунок мовою C#. Вхідні дані ввести з клавіатури. Результати вивести на консоль Використати методи класів **Console, Convert** в процесі введення та виведення даних. Реалізувати перераховані функції. Виклик функцій здійснити за допомогою **меню**, застосувавши оператор вибору **switch** для виклику потрібної функції. Усі завдання варіанта реалізувати в одному проекті **Console\_Lab1**.   1. Вивести на консоль власні анкетні дані: прізвище, ім'я, вік, група, курс, e-mail. Написати функцію, яка визначає приналежність точки з координатами (*x,y*) колу з радіусом *r* та центром на початку координат. Значення x,y,r ввести з консолі. 2. За даними, що введені з консолі, визначити значення виразу, використовуючи математичні функції, і вивести результат на консоль.      1. За даними x, значення яких ввести з консолі, обчислити значення функції:      1. Написати функцію, яка в залежності від назви країни виводить її рейтинг освітньої грамотності за версією PISA ((<https://factsmaps.com/pisa-2018-worldwide-ranking-average-score-of-mathematics-science-reading/>). 2. Дано натуральне число *n* і дійсне число *x* >0. Обчислити суму членів ряду: |

### Література

1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#. КНУ ім. Тараса Шевченка

2. The C# Coding Standard. Access mode: <https://github.com/hassanhabib/CSharpCodingStandard>

3. C# Coding Standards and Best Practices. Access mode: https://www.dofactory.com/csharp-coding-standards

4. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І. В., Марущак П. О. – Тернопіль: ФОП Паляниця В. А., 2020 – 320 с. Режим доступу: <https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf>

## Лабораторна робота № 2 Масиви, матриці, рядки. Методи класів Console, Convert, Random, Math в C#

### Рейтинг лабораторної роботи №2

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 8 завдань та реалізація асоціації класів | 0,5\*8=4,0  4,0+0,5 = 4,5 | 7 березня |
| 2 | Захист | 0,5 |  |
| 3 | Звіт | 0,5 |  |
| Разом за роботу | | 5,5 |  |

### Мета роботи

Навчитися працювати з масивами, матрицями, рядками відповідно до перерахованих дій:

1. Робота з масивами в C#

2. Оператор foreach

3. Генерація випадкових чисел

4. Робота з математичними функціями

5. Лінійний пошук

6. Двійковий пошук

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію.

2. Прочитайте методичні рекомендації до лабораторної роботи та виконайте наведені в ній приклади (вони всі працездатні)

3. При виконанні завдань зверніть увагу на приклад застосування циклу **foreach** в процесі обробки масивів

4. Також зверніть увагу на опис і реалізацію методу бісекцій (ділення навпіл) знаходження коренів нелінійних рівнянь.

5. Для поглибленого вивчення цього матеріалу прочитайте [1, 2].

### Порядок виконання роботи

1. Створити директорію Lab2, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного консольного проекту

### Приклади виконання завдань

#### 1. Робота з масивами в C#

Розглянемо приклад програми з лекції, яка визначає суму і кількість від'ємних елементів, а також максимальний елемент масиву, що складається з 6 цілочисельних елементів.

Тут елементи масиву ***а*** ініціалізуються при створенні масиву. Далі в циклі з параметром елементи масиву виводяться на консоль. Зверніть увагу, що для виводу використовується метод Console.Write а не Console.WriteLine, тому всі елементи виводяться в один рядок. Символ табуляції ‘\t’ в методі Console.Write() розділяє елементи при виводі.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

const int n = 6;

int[] a = new int[n] { 3, 12, 5, -9, 8, -4 };

Console.WriteLine( "Початковий масив:" );

for ( int i = 0; i < n; ++i )

Console.Write( "\t" + a[i] );

Console.WriteLine();

long sum = 0; // сума від'ємних елементів

int num = 0; // кількість від'ємних елементів

for ( int i = 0; i < n; ++i )

if ( a[i] < 0 )

{

sum += a[i];

++num;

}

Console.WriteLine( "Сума від'ємних = " + sum );

Console.WriteLine( "Кількість від'ємних = " + num );

int max = a[0]; // максимальний елемент

for ( int i = 1; i < n; ++i )

if ( a[i] > max ) max = a[i];

Console.WriteLine( "Максимальний елемент = " + max );

Console.ReadKey();

}

}

}

Розглянемо ще одну "класичну" задачу – генерація чисел Фібоначчі: послідовності чисел, яка задовольняє умовам F1 = 1; F2 = 1; Fk = Fk-1 + Fk-2 для k>2.

Цю задачу можна реалізувати з використанням масивів. В наступному прикладі кількість чисел вводиться з консолі, потім створюється одновимірний масив вказаної розмірності, який заповнюється генерованими числами.

##### Генерація чисел Фібоначчі

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введіть кількість чисел послідовності");

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

int[] fibonachi = new int[n];

fibonachi[0] = 1;

fibonachi[1] = 1;

for (int i = 2; i < n; i++)

{

fibonachi[i] = fibonachi[i - 2] + fibonachi[i - 1];

}

for (int i = 0; i < n; i++)

Console.WriteLine("fibonachi[" + i + "]=" + fibonachi[i]);

Console.ReadKey();

}

}

}

##### Генерація простих чисел. Алгоритм Ератосфена

Простим є число, яке ділиться тільки на 1 і на себе. Алгоритм пошуку послідовності простих чисел запропонував давньогрецький математик Ератосфен, і він отримав назву **Решето Ератосфена.**

**Ідея і загальний опис алгоритму.**

Є розташована в ряд за збільшенням послідовність цілих чисел. Спочатку в ній викреслюються усі числа кратні 2, окрім її самої, і так до N. Далі із списку, що вийшов, береться число, що йде за двійкою, - трійка, викреслюються усі кратні 3 числа, крім її самої. У такому вигляді алгоритм триває для частини послідовності, що залишилася, і у результаті отримаємо усі прості числа у вказаному діапазоні.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 220 |
| 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 440 |
| 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 660 |
| 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 880 |
| 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 1100 |

У таблиці наведені усі цілі числа від 2 до 100. Червоним помічені ті, які були видалені в процесі виконання алгоритму Решето Ератосфену.

Тепер розглянемо алгоритм детальніше, розбивши його на декілька частин. Отже, для пошуку простих чисел методом Решета Ератосфену треба:

1. Організувати список з чисел від 2 до N, а також логічний масив розмірністю N;
2. У вільну змінну R записати число 2;
3. Виключити всі числа кратні R, починаючи з R\*2;
4. Записати в R наступне за R не закреслене число;
5. Повторювати дії, описані в двох попередніх кроках.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace SimpleNumber

{

class Program

{

//Знаходження простих чисел. Алгоритм "Решето Ератосфена"

//Запишемо поспіль усі числа від 2 до N. Далі викреслимо з цього списку всі числа, //кратні 2, виключаючи саму двійку, потім викреслити всі числа, кратні 3,

//виключаючи саме число 3, число 4 вже викреслено, викреслюємо числа,

//кратні 5 і т.д. Продовжуємо цей процес, поки квадрат чергового числа не перевищує N.

static void Main(string[] args)

{

bool[] table = new bool[100];

int i, j;

// Відзначаємо всі числа як прості

for (i = 0; i < table.Length; i++)

table[i] = true;

// Викреслюємо зайве

for (i = 2; i \* i < table.Length; i++)

if (table[i])

for (j = 2 \* i; j < table.Length; j += i)

table[j] = false;

// Виводимо знайдене

for (i = 2; i < table.Length; i++)

{

if (table[i])

Console.WriteLine(i);

}

Console.ReadKey();

}

}

}

#### 2. Оператор foreach

Оператор **foreach** використовується для перебору елементів в масивах та інших колекціях і є новим типом циклу, реалізованому в C#. В цьому прикладі створюється *ступінчастий масив*, який заповнюється даними. За допомогою вкладених операторів циклу елементи масиву виводяться на консоль у вигляді матриці.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

static void Main(string[] args)

{

//оператор foreach і ступінчасті масиви

int[][] mas = new int[3][];

mаs[0] = new int[5] { 24, 50, 18, 3, 16 };

mas [1] = new int[3] { 7, 9, -1 };

mas [2] = new int[4] { 6, 15, 3, 1 };

Console.WriteLine("Початковий масив:");

foreach (int[] item in mas )

{

foreach (int x in item )

Console.Write("\t" + x);

Console.WriteLine();

}

Console.ReadKey();

}

}

}

#### 3. Генерація випадкових чисел

Потреба в генерації випадкових чисел часто виникає в програмуванні. В C# є *нестатичний* клас Random, методи якого дозволяють генерувати різні послідовності випадкових чисел. Нижче наведено приклад з лекції 4 в якому в методі ValsGenerator створюється об'єкт **aRand**. Далі в циклі викликається метод **aRand.Next(100),** який при кожному виклику генерує одне випадкове число від 1 до100 і призначає його черговому елементу масиву. Зверніть увагу, що метод **ValsGenerator** є статичним.

В методі **Main** створюється масив **Data,** викликається метод **ValsGenerator** для його заповнення, потім викликається метод Array.Sort(Data) для його сортування. Після чого елементи масиву виводяться в циклі на консоль.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Console\_Lab4

{

class Program

{

// генератор даних

static void ValsGenerator(int[] Vals)

{

// Random - клас для генерації випадкових чисел

Random aRand = new Random();

// заповнення масиву

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

Vals[i] = aRand.Next(100);

}

static void Main(string[] args)

{

const int N = 10;

int[] Data = new int[N];

ValsGenerator(Data);

Array.Sort(Data);

Console.WriteLine("Друк відсортованих даних");

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

Console.WriteLine("Data[" + i + "] = " + Data[i]);

Console.ReadLine();

}

}

}

Зверніть увагу, що при виводі масиву методом WriteLine()

Console.WriteLine("Data[" + i + "] = " + Data[i]);

виконується неявне приведення з типу int до типу string.

#### 4. Робота з математичними функціями

Клас Math містить методи для роботи з математичними функціями.

Розглянемо ще один приклад – використання масивів і математичних функцій в задачах обчислювальної математики.

**Постановка задачі**

Знайти дійсні корені рівняння 6x4-3x3+8x2-5=0 за методом бісекції (ділення навпіл) на відрізках [0, 1], [-1, 0].

**Алгоритм методу:**

Нехай [а,b] відрізок, на якому шукають корені. Припустимо, що функція f(x) неперервна на [а,b] і на кінцях приймає значення різних знаків *Алгоритм* методу полягає в побудові послідовності вкладених відрізків, на кінцях яких функція приймає значення різних знаків. Кожний наступний відрізок отримують діленням навпіл попереднього. Опишемо один крок ітераційного методу. Нехай на *к-*ому кроці знайдено відрізок такий, що . Знайдемо середину відрізку . Якщо,  то - корінь і задача вирішена. Якщо ні, то з двох половин відрізку вибираємо той, на кінцях якого функція має протилежні знаки:

, , якщо 

, , якщо 

*Критерій закінчення ітераційного процесу*: якщо довжина відрізку знаходження кореня менше 2, то ітерації припиняють і за значення кореня із заданою точністю приймають середину відрізку.

Код програми може бути таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace bicection

{

class Program

{ //Знайти корені нелінійного рівняння

//6x^4-3x^3+8x^2-5=0

//x1=0.74213

//x2=-0.6365

static double f(double x)

{

//рівняння, для якого шукаємо корені

double y=6\*Math.Pow(x,4)-3\*Math.Pow(x, 3)+8\*Math.Pow(x,2)-5;

return y;

}

// --------------------------------------------------------------------

static double bicect(double left, double right)

{

//метод бісекцій

double eps = 0.00001; //точність розрахунку

double center = 0;

while (right - left > eps \* 2)

{

center = (right - left) / 2 + left;

if (f(center) \* f(left) < 0) //перевірка, чи функція проходить через 0

left = center;

else

right = center;

}

return center;

}

// ----------------------------------------------------

static void Main(string[] args)

{

//метод бісекцій

double x1 = bicect(0, 1);

double x2 = bicect(-1, 0);

Console.WriteLine("Метод бісекцій");

Console.WriteLine("x1={0},x2= {1}", x1, x2);

Console.ReadKey();

}

}

}

В цій програмі є два методи: static double f(double x), в якому визначається рівняння, і метод static double bicect(double left, double right). Для обчислення ступеня в методі **f** використовується метод Math.Pow( ).

Другий метод static double bicect(double left, double right) реалізує ітераційний процес. В методі Main два рази викликається метод бісекцій для знаходження коренів на вказаних відрізках. Результат виводиться на консоль.

#### 5. Лінійний пошук в масиві

**Лінійний, послідовний пошук** — алгоритм знаходження заданого значення довільної функції на деякому відрізку.

**Формальний запис алгоритму:**

1. Визначити елемент ***key***, який шукаємо***;***

2. Встановити границі області пошуку L, R для масиву A [a];

3. Якщо аi == key – елемент знайдено;

4. Інакше – переходимо до наступного елементу.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Search

{

class Program

{

static void Main(string[] args)

{

int i, n, key, nom;

bool x = false;

Console.WriteLine("Розмір масиву");

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

int[] mas = new int[n]; //визначення масиву

Console.WriteLine("Шуканий елемент");

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

// Random - клас для генерації випадкових чисел

Random aRand = new Random();

for (i = 0; i < n; i++)

{

//формування масиву, заповнення його випадковими числами

mas[i] = aRand.Next(n);

Console.Write(mas[i] + " ");

}

for (i = 0; i < n; i++)

{ //якщо цей елемент дорівнює шуканому

if (mas[i] == key) //то x призначаємо true

{ x = true; nom = i; break; } //і виходимо з циклу

}

if (x == true)

Console.WriteLine("Елемент знайдено");

else Console.WriteLine("\nЕлемент не знайдено");

Console.ReadKey();

}

}

}

#### 6. Двійковий (бінарний) пошук

**Двійковий (бінарний) пошук** (також відомий як метод ділення навпіл і дихотомія) — класичний алгоритм пошуку елемента у відсортованому масиві.

**Формальний запис алгоритму:**

Масив ділитися на дві рівні частини, шляхом визначення першого (a), останнього (b) і середнього (c) елементів;

Середній елемент порівнюється з шуканим (s):

якщо **s < c,** останньому елементу призначається значення середнього, тим самим ділянка пошуку зменшується удвічі: b = c;

якщо **s > c,** першому елементу призначається значення середнього, і ділянка пошуку зменшується удвічі: a = c;

якщо **s = c,** елемент знайдений, і робота алгоритму завершується**.**

Якщо для перевірки не залишилося жодного елементу, алгоритм завершується, інакше виконується перехід до пункту 2.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BinarySearch

{

class Program

{

static void Main(string[] args)

{

int i, n, key, begin, end, c;

bool x = false;

Console.WriteLine("Розмір масиву");

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

int[] mas = new int[n]; //визначення масиву

Console.WriteLine("Шуканий елемент");

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

for (i = 0; i < n; i++)

{

//формування масиву, заповнення його числами кратними n (10,20,....)

mas[i] = n \* i;

Console.Write(mas[i] + " ");

}

begin = 0; end = n; //ліва і права границі масиву

while (begin < end)

{

// власне реалізація алгоритму

c = begin + (end - begin) / 2;

if (key < mas[c]) end = c;

else if (key > mas[c]) begin = c + 1;

else { x = true; break; }

}

if (x == true)

Console.WriteLine("Елемент знайдено");

else Console.WriteLine("\n Елемент не знайдено");

Console.ReadKey();

}

}

}

### Варіанти завдань для лабораторної роботи № 2

Завдання передбачає створення трьох класів з описом полів та реалізацією 8 **нестатичних методів різних класів**. Завдання 1 – 4 кожного варіанта передбачають роботу з одновимірними масивами (векторами), завдання 5, 6 кожного варіанта передбачають опрацювання матриць, завдання 7, 8 кожного варіанта реалізують операції введення та виведення обчислених результатів на консоль. Кожний клас потрібно описати в окремому файлі.

Відповідно до **SOLID принципу єдиної відповідальності** в об’єктно-орієнтованому проєктуванні клас не може мати методи, які не відповідають **одній області функціональності об’єкту цього класу**. Це означає, що класи вектор і матриця не повинні мати методи введення значень з консолі, або виведення їх на консоль. Потрібен інший клас, названий тут сервісним класом.

Взаємодія класів подана на рис. 1, де зв’язок між класами поданий **направленою асоціацією.**

// File: Vector.cs

// Author: sigma

// Purpose: Definition of Class Vector

using System;

public class Vector

{

public Service service;

}

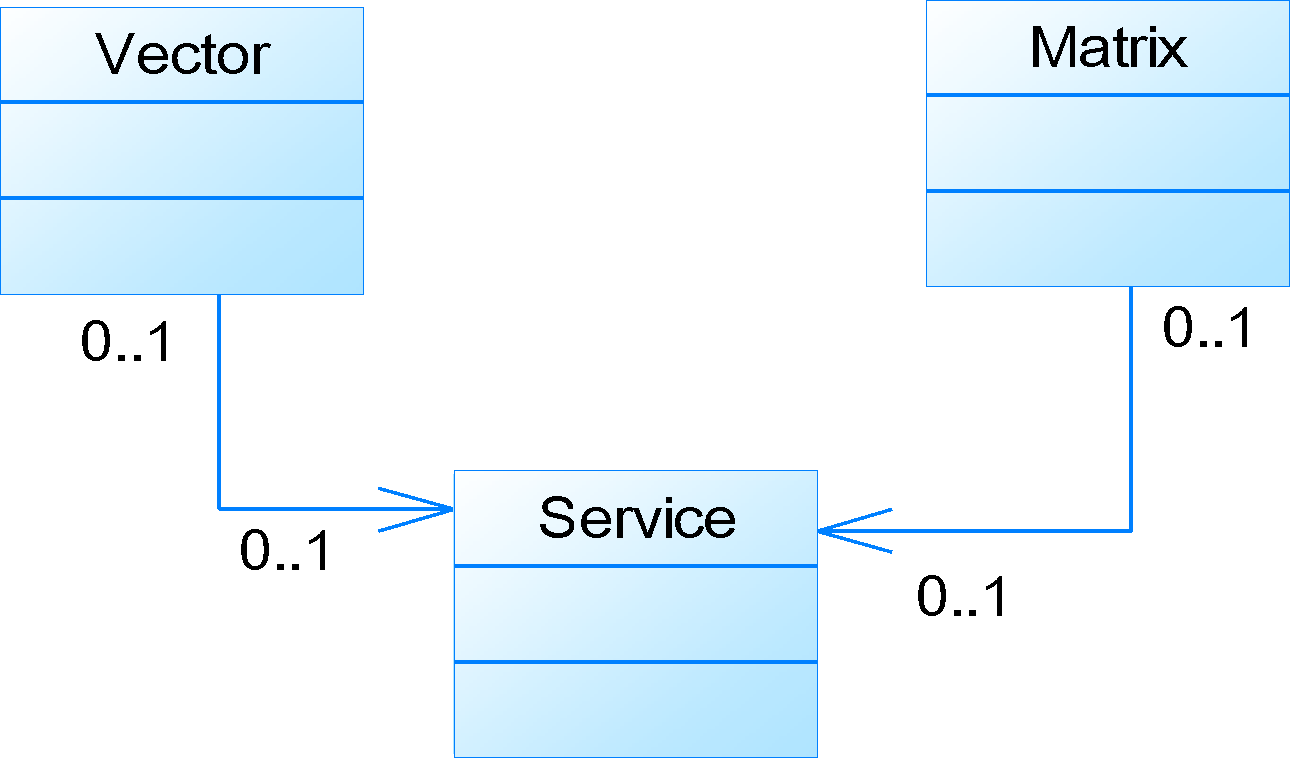


Рисунок 1 – Направлена асоціація класів. Рисунок 2 – Код, що описує асоціацію класів.

Направлена асоціація кодується мовою C# включенням асоційованого (залежного) об’єкту в клас незалежного об’єкту (рис. 2).

Ненаправлена асоціація класів подана на рис. 3.

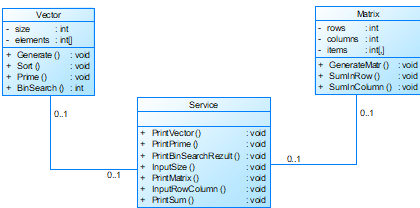


Рисунок 3. - Ненаправлена асоціація класів

При ненаправленій асоціації класів асоційовані об’єкти в інші класи, з якими вони асоціюються, не включаються. Об’єкти рівноправні і незалежні.

Забороняється використовувати властивості, методи та інтерфейси класів просторів імен [**System.Collections**](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections?view=net-8.0)**,** **System.Collections.Generic, System.Collections.ObjectModel, універсальні шаблони**, а також клас **Array**, який не належить простору імен [**System.Collections**](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections?view=net-8.0)**.**

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор слід реалізувати такі методи:**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість з консолі. 2. Відсортувати згенерований масив за спаданням значень його елементів алгоритмом швидкого сортування. Заборонено використовувати метод QuickSort() класу Array. 3. У згенерованому масиві знайти прості числа із заданого з консолі діапазону, використовуючи алгоритм Ератосфена 4. Знайти значення елемента, заданого з консолі, застосувавши нерекурсивний метод бінарного пошуку. У разі його відсутності в масиві вивести відповідне повідомлення. Використання методу BinarySearch() класу Array не дозволено.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність з консолі. 2. Визначити суму елементів заданого з консолі номеру рядка матриці та суму елементів заданого з консолі номеру стовпчика матриці.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, знайдені в масиві прості числа, шукане значення елемента масиву та його індекс в результаті бінарного пошуку. 2. Вивести на консоль елементи матриці, згенеровані в класі **Матриця**, номери рядка та стовпчика матриці, в яких обчислювалися суми елементів та значення сум. |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та інші в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість з консолі. 2. Відсортувати згенерований масив за спаданням значень його елементів алгоритмом Шелла. 3. У згенерованому масиві обчислити суму його елементів, середнє арифметичне, max елемент масиву та його індекс, використовуючи методи класу **Math**. 4. Визначити кількість повторень елемента масиву, що введений з консолі, застосувавши метод лінійного пошуку.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати матрицю, i–й рядок якої визначає номер співробітника, j-й стовпчик якої визначає номер місяця року. Кількість співробітників задати з консолі. Значення на перетині i-го рядка та j-го стовпця визначає зарплату i-го співробітника у j-му місяці. 2. У згенерованій матриці визначити загальний бюджет зарплати за рік, загальну і середню зарплату за місяць, номер якого ввести з консолі.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, обчислені в масиві суму його елементів, середнє арифметичне, max елемент масиву та його індекс, кількість повторень заданого елемента. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальний бюджет зарплати за рік, загальну і середню зарплату за вказаний з консолі місяць |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та інші в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати елементи одновимірного масиву, задавши їх кількість з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом вибору. 3. У згенерованому масиві визначити кількість парних елементів і елементів з парними індексами, застосувавши алгоритм лінійного пошуку. 4. Визначити кількість і значення елементів масиву, що кратні заданому з консолі числу, застосувавши рекурсивний метод бінарного пошуку. У разі їх відсутності вивести відповідне повідомлення. Використання методу BinarySearch() класу Array не дозволено.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень з консолі. Елемент на перетині i-го рядка та j-го стовпчика матриці означає прибуток за j-й місяць від i-го проєкту, який виконується в компанії. 2. Визначити загальний прибуток від кожного проєкту, загальний дохід компанії від усіх проєктів за всі місяці, індекс проєкту з найбільшим прибутком.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, обчислені в масиві кількість парних елементів і елементів з парними індексами, кількість і значення елементів масиву, що кратні заданому з консолі числу. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальний прибуток від кожного проєкту, загальний дохід компанії від усіх проєктів за всі місяці, індекс проєкту з найбільшим прибутком. |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом включення. 3. У згенерованому масиві визначити прості числа серед додатних елементів масиву, використовуючи алгоритм Ератосфена. 4. Знайти найбільший серед від’ємних та найменший серед додатних елементів масиву та їх індекси, застосувавши алгоритм лінійного пошуку.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень з консолі. Елемент на перетині i-го рядка та j-го стовпчика матриці означає прибуток за j-й місяць від продажу i-го товару в магазині. 2. У згенерованій матриці визначити загальний прибуток від кожного товару, загальний прибуток магазину від продажу всіх товарів за всі місяці, індекс товару, який приносить найбільший прибуток.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, знайдені прості числа серед додатних елементів масиву, найбільший серед від’ємних та найменший серед додатних елементів масиву та їх індекси. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальний прибуток магазину від кожного товару, загальний прибуток магазину від продажу всіх товарів за всі місяці, індекс товару, який приносить найбільший прибуток. |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву. .  **У класі Вектор реалізувати такі методи:**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість з консолі. 2. Відсортувати згенерований масив за зростанням значень його додатних елементів алгоритмом бульбашкового сортування. 3. Переставити елементи згенерованого масиву в порядку чергування від’ємних, додатних, нульових елементів. 4. Визначити кількість і значення елементів згенерованого масиву, що належать заданому з консолі діапазону, застосувавши метод рекурсивного бінарного пошуку. Використання методу BinarySearch() класу Array не дозволено.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі, включаючи нульове значення. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість вакцинованих людей проти Covid-19 в *i*-й країні в *j*-му місяці. Якщо значення дорівнює нулю, це означає, що в *i*-й країні в *j*-му місяці людей не вакцинували. 2. У згенерованій матриці визначити загальну кількість людей, що вакциновані в усіх країнах за весь період, країну, в якій найменша кількість вакцинованих за весь період, місяць з найбільшою кількістю вакцинованих.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після переставлення елементів, результати бінарного пошуку. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальну кількість людей, що вакциновані в усіх країнах за весь період, країну, в якій найменша кількість вакцинованих за весь період, місяць з найбільшою кількістю вакцинованих. |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати натуральні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом швидкого сортування. Заборонено використовувати метод QuickSort() класу Array. 3. Визначити, чи елементи згенерованого масиву з парними індексами є простими числами. Для визначення простих чисел використати алгоритм Ератосфена, 4. У згенерованому масиві переставити елементи так, щоб спочатку розташовувались числа, які діляться на 2, але не діляться на 3, потім числа, що діляться на 2 та на 3, потім числа, що діляться тільки на 3, потім всі інші.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елеменет на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість студентів, що проходять практику в *j*-му місяці в *i*-й компанії. 2. У згенерованій матриці визначити загальну кількість студентів, що проходили практику в усіх компаніях в заданому з консолі місяці, кількість компаній, в яких проходили практику задана з консолі кількість студентів, компанію, в якій практикувалась найбільша кількість студентів.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, елементи згенерованого масиву з парними індексами, які є простими числами, або повідомлення про їх відсутність, масив після переставлення елементів. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальну кількість студентів, що проходили практику в усіх компаніях в заданому з консолі місяці, кількість компаній, в яких проходили практику задана з консолі кількість студентів, компанію, в якій практикувалась найбільша кількість студентів. |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву  **У класі Вектор реалізувати такі методи:**   1. Згенерувати цілочислові значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом вставки. 3. Знайти елементи згенерованого масиву, що є квадратами цілих чисел, застосувавши алгоритм лінійного пошуку. 4. У згенерованому масиві знайти середнє арифметичне елементів з парними індексами.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі, включаючи нульові значення. Елеменет на перетині *i*-го рядка та *j*-го стовпчика матриці означає *i*-ту дисципліну та *j*-ту компетентність, які вони формують, і містить значення 0 або 1. На перетині *i*-го рядка та *j*-го стовпчика матриці записаний 0, якщо дисципліна не забезпечує формування компетентності студента, 1 – якщо між дисципліною і компетентністю є зв’язок. 2. У згенероаній матриці визначити, кількість дисциплін, які не забезпечують формування жодної компетентності студентів, дисципліни, які формують найбільшу кількість компетентностей, компетентності, які не забезпечені дисциплінами.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, елементи згенерованого масиву, що є квадратами цілих чисел та їх індекси, середнє арифметичне елементів з парними індексами. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, кількість дисциплін, які не забезпечують формування жодної компетентності студентів, дисципліни, які формують найбільшу кількість компетентностей, компетентності, які не забезпечені дисциплінами. |
|  |  |
|  | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву  **У класі Вектор реалізувати такі методи:**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом Шелла. 3. Знайти в згенерованому і відсортованому масиві елементи та їх індекси, які є пірамідальними числами, тобто складають послідовність чисел 1, 4, 10, 35,…, застосувавши алгоритм лінійного пошуку. Формула для обчислення *n*-го пірамідального числа: 4. У згенерованому масиві визначити усі входження в масив заданого з консолі елемента, застосувавши метод нерекурсивного бінарного пошуку. Використання методу BinarySearch() класу Array не дозволено.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість кредитів, що має *i*-та дисципліна в *j*-му навчальному році. Якщо значення елемента матриці дорівнює нулю, це означає, що i-та дисципліна не вивчається в *j*-му навчальному році. 2. У згенеровній матриці визначити загальну кількість кредитів по всіх дисциплінах за всі навчальні роки, дисципліну та навчальний рік, в якому є дисципліна з найбільшою кількістю кредитів, середню кількість кредитів в кожному навчальному році.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, елементи згенерованого масиву, що є пірамідальними числами, усі індекси шуканого елемента. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальну кількість кредитів по всіх дисциплінах за всі навчальні роки, дисципліну та навчальний рік, в якому є дисципліна з найбільшою кількістю кредитів, середню кількість кредитів в кожному навчальному році. |
| **9.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів бульбашковим алгоритмом. 3. У згенерованому масиві елементи, що розташовані між мінімальним та максимальним елементами, зписати в інвертованому порядку. 4. У згенерованому масиві визначити елементи, які є числами Фібоначчі. Кожне число Фібоначчі, починаючи з третього, є сумою двох попередніх. Перші два значення послідовності Фібоначчі дорівнюють 1, тобто числа Фібоначчі мають значення 1, 1, 2, 3, 5, 8, 13….   **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. На перетині *i*-го рядка та *j*-го стовпчика матриці записана кількість студентів, які обрані до складу студради *i*-го факультету в *j*-му навчальному році. 2. У згенерованій матриці визначити факультет і навчальний рік, в якому кількість студентів в студраді є найбільшою, сумарну кількість студентів в студраді за всі роки навчання на заданому з консолі факультеті, навчальний рік, в якому сумарна кількість студентів в студрадах усіх факультетів найменша.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, елементи згенерованого масиву, що записані в інвертованому порядку, індекси елементів згенерованого масиву, які є числами Фібоначчі. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, факультет і навчальний рік, в якому кількість студентів в студраді є найбільшою, сумарну кількість студентів в студраді за всі роки навчання на заданому з консолі факультеті, навчальний рік, в якому сумарна кількість студентів в студрадах усіх факультетів найменша. |
| **10.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи Parse(), TryParse() вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів найбільш ефективним для невеликих масивів алгоритмом. 3. Знайти в згенерованому і відсортованому масиві елементи та їх індекси, які є семикутними числами, застосувавши алгоритм лінійного пошуку. Семикутні числа складають послідовність (1, 7, 18, 34,…). Формула для обчислення *n*-го семикутного числа: 4. У згенерованому масиві визначити суму парних елементіви, які мають непарні індекси.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість заяв абітурієнтів, які подали їх до вступу на *i*-й факультет та *j*-ту спеціальність. 2. У згенерованій матриці визначити загальну кількість заяв абітурієнтів по всіх факультетах, факультет з максимальною кількістю заяв абітурієнтів, кількість заяв на задану з консолі спеціальність   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, елементи згенерованого масиву, що є семикутними числами, суму парних елементів, які мають непарні індекси. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальну кількість заяв абітурієнтів по всіх факультетах, факультет з максимальною кількістю заяв абітурієнтів, кількість заяв на задану з консолі спеціальність. |
| **11.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати додатні та від’ємні значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів алгоритмом вставки. 3. Переставити елементи згенерованого масиву так, щоб спочатку розташовувались парні числа, потім непарні, потім нульові. 4. У згенерованому масиві визначити кількість входжень кожного елемента.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість БПЛА, які поставлені в *i*-му місяці *j*-тій бригаді ЗСУ. 2. У згенерованій матриці визначити загальну кількість БПЛА за всі місяці всім бригадам ЗСУ, бригаду з максимальною кількістю БПЛА, кількість БПЛА за всі місяці заданій з консолі бригаді.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після переставлення елементів, кількість входжень кожного елемента в масиві. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, загальну кількість БПЛА за всі місяці всім бригадам ЗСУ, бригаду з максимальною кількістю БПЛА, кількість БПЛА за всі місяці заданій з консолі бригаді. |
| **12.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом змішування (Cocktail sort) <https://uk.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B7%D0%BC%D1%96%D1%88%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%D0%BC>) . 3. Знайти в згенерованому масиві два сусідні елементи, сума яких мінімальна. 4. У згенерованому масиві поміняти місцями елемент из парними і непарними індексами.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає оцінку *i*-го студента по *j*-й дисципліні. 2. Визначити студентів, які за підсумками успішності за всіма дисциплінами підпадають під відрахування, рейтинг кожного студента як середнє арифметичне по всіх дисциплінах, дисципліну, середній бал якої найбільший.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, два сусідні елементи, сума яких мінімальна, масив після перестановки елементів. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, студентів, які за підсумками успішності за всіма дисциплінами підпадають під відрахування, рейтинг кожного студента, дисципліну, середній бал якої найбільший. |
| **13.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх парну кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом вставки (Insertion Sort) 3. Для кожних двох сусідніх елементів згенерованого масиву обчислити найменший спільний дільник за алгоритмом Евкліда. 4. У згенерованому масиві здійснити пошук елемента та його індекса за алгоритмом бінарного пошуку. Значення шуканого елемента ввести з консолі. Заборонено використовувати метод BinarySearch() з класу Array   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість підручників, які *i*-й студент прочитав по *j*-й дисципліні. 2. Визначити студентів, які прочитали найбільшу кількість підручників за всіма дисциплінами, дисципліну, по якій рекомендована найменша кількість підручників, сумарну кількість підручників, які прочитали всі студенти по всіх дисциплінах   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив найменших спільних дільників, результат пошук елемента та його індекс. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, шуканих студентів, дисципліну та кількість підручників згідно з умовою п.6 |
| **14.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом сортування вибором (Selection Sort). 3. У згенерованому та відсортованому масиві видалити елементи, які менші за значення, що введене з консолі. Використати алгоритм зсуву масиву вліво. 4. У згенерованому за п.1 масиві визначити пари сусідніх елементів одного знаку та кількість таких пар.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість лабораторних робіт, виконаних *i*-м студент з *j*-ї дисципліни. 2. Визначити студентів, які виконали найбільшу кількість лабораторних робіт за всіма дисциплінами, дисципліну, по якій усі студенти виконали найменшу кількість робіт, студентів, які хоча б з однієї дисципліни не виконали жодної (в матриці записаний 0) лабораторної роботи.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після видалення елементів, пари елементів одного знаку. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, студентів, які виконали найбільшу кількість лабораторних робіт за всіма дисциплінами, дисципліну, по якій усі студенти виконали найменшу кількість робіт, студентів, які хоча б з одної дисциплінами не виконали жодної лабораторної роботи |
| **15.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом випадкового сортування (Bogosort). 3. У згенерованому та відсортованому масиві здійснити пошук елементів, сусіди яких відрізняються на задане з консолі число. 4. У згенерованому за п.1 масиві видалити усі елементи, що повторюються, використовуючи алгоритм зсуву масиву вліво.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість пропущених занять *i*-м студентом з *j*-ї дисципліни. 2. Визначити студентів, які пропустили найбільшу кількість занять за всіма дисциплінами, дисципліну, по якій пропущена всіма студентами найменша кількість занять, середнє значення пропусків всіма студентами занять з усіх дисциплін.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, шукані елементи та їх індекси, масив після видалення елементів. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, студентів, які пропустили найбільшу кількість занять за всіма дисциплінами, дисципліну, по якій пропущена всіма студентами найменша кількість занять, середнє значення пропусків всіма студентами занять з усіх дисциплін. |
| **16.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом Шелла. 3. У згенерованому та відсортованому масиві здійснити пошук елементів та їх індексів, які є числами Мерсенна () або Ферма (), застосувавши алгоритм лінійного пошуку. 4. У згенерованому за п.1 масиві переставити елементи так, щоб спочатку йшли числа Фібоначчі, потім числа Мерсенна, потім числа Ферма, потім решта.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість резюме, посланих *i*-им студентом для працевлаштування в *j*-ту компанію. 2. Визначити студентів, які послали резюме у всі компанії; компанію, в яку послали найменшу кількість резюме; студента, який не послав резюме в жодну компанію.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, шукані елементи та їх індекси, масив після переставлення елементів. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, студентів, які послали резюме у всі компанії; компанію, в яку послали найменшу кількість резюме; студента, який не послав резюме в жодну компанію. |
| **17.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення додатних і від’ємних елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом випадкового сортування (англ. Bogosort). 3. У згенерованому та відсортовному масиві переставити елементи так, щоб спочатку йшли парні елементи, якщо їх індекси у вхідному масиві парні, потім непарні елементи, якщо їх індекси у вхідному масиві непарні, потім елементи з нульовим значенням, потім решта. Якщо елементи вхідногор масиву парні, а їх індекси непарні, такі елементи у вихідному масиві віднести до решти. 4. У згенерованому за п.1 масиві обчислити суму додатних елементів з парними індексами.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає температуру повітря в *i*-ий день *j*-го місяця. 2. Визначити місяці, середня температура в яких була вище за задану з консолі; день, в якому температура була найнижча по всіх місяцях; середню температура взимку.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після переставлення елементів, суму додатних елементів з парними індексами. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, місяці, середня температура в яких була вище за задану з консолі; день, в якому температура була найнижча по всіх місяцях; середню температура взимку. |
| **18.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення додатних і від’ємних елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом сортування змішуванням (cocktail sort, shaker sort). 3. У згенерованому та відсортовному масиві переставити елементи так, щоб додатні та від’ємні числа чергувалися. 4. У згенерованому за п.1 масиві обчислити медіану елементів. Медіаною набору різних чисел називається таке число *m*, що кількість чисел, більших за *m*, дорівнює кількості чисел, менших за *m*.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість ворожих атак, які відбили ЗСУ в *i*-ий день *j*-го місяця. 2. Визначити місяці, середня кількість ворожих атак в яких була вище за задану з консолі; день, в якому кількість ворожих атак була найнижча по всіх місяцях; загальну кількість ворожих атак за всі місяці.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після переставлення елементів, медіану масиву. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, місяці, середня кількість ворожих атак в яких була вище за задану з консолі; день, в якому кількість ворожих атак була найнижча по всіх місяцях; загальну кількість ворожих атак за всі місяці. |
| **19.** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення додатних і від’ємних елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за алгоритмом Шелла 3. У згенерованому та відсортовному масиві здійснити пошук заданого з консолі елемента та його індекса. Заборонено використовувати метод **Array.BinarySearch()** 4. У згенерованому за п.1 масиві обчислити моду елементів. Модою набору чисел називається такий елемент, який зустрічається в масиві найчастіше.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість повітряних тривог в *i*-ий день *j*-го місяця. 2. Визначити місяці, середня кількість тривог в яких була вище за задану з консолі; день, в якому кількість тривог була найнижча по всіх місяцях; день, коли тривоги відсутні.   **У класі Сервіс реалізувати такі методи:**   1. Вивести команди меню на консоль з викликами методів класів **Вектор та Матриця.** 2. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, шуканий елемент та його індекс, моду масиву. 3. Вивести на консоль матрицю, згенеровану в класі **Матриця**, місяці, середня кількість тривог в яких була вище за задану з консолі; день, в якому кількість тривог була найнижча по всіх місяцях; день, коли тривоги відсутні. |
| **20** | Створити консольний проєкт мовою C#. Додати до створеного проєкту C# два створених власноруч класи: **клас Вектор** та **клас Матриця**. Кожний клас створити в окремому файлі (меню Visual Studion “Проєкт”, команда “Додати клас”). В класах **Вектор** і **Матриця** реалізувати задані в завданні методи. Згідно з SOLID принципом єдиної відповідальності для реалізації методів введення та виведення даних створіть окремий клас **Сервіс**. Зв’язок між класами - ненаправлена асоціація. Використовувати методи класів **Console**, **Convert** та методи **Parse(), TryParse()** вбудованих типів в процесі введення та виведення даних в класах **Вектор** і **Матриця неприпустимо через порушення SOLID принципу єдиної відповідальності**. Для створення елементів масиву і матриці використати методи класу **Random**. Для виконання математичних обчислень використати клас **Math.**  **Клас Вектор має такі поля:** кількість елементів масиву, значення елементів масиву.  **У класі Вектор реалізувати такі методи:**   1. Згенерувати значення елементів одновимірного масиву, задавши їх кількість та діапазон значень з консолі. 2. Відсортувати згенерований масив за зростанням значень його елементів за рекурсивним алгоритмом швидкого сортування. Забороняється використовувати метод Array.QuickSort(). 3. У згенерованому та відсортовному масиві здійснити циклічний зсув елементів на *n* елементів вправо або вліво. Значення *n* ввести з консолі. 4. У згенерованому за п.1 масиві обчислити амплітуду елементів. Амплітудою масиву називається різниця між максимальним та мінімальним значеннями елементів масиву.   **Клас Матриця має такі поля:** вимірність матриці (кількість рядків і стовпців), значення елементів матриці.  **У класі Матриця реалізувати такі методи:**   1. Згенерувати елементи матриці, задавши її вимірність та діапазон значень елементів з консолі. Елемент на перетині *i*-го рядка та *j*-го стовпчика матриці означає кількість навчальних занять по *i*-ій дисципліні в *j*-ий місяць. 2. Визначити місяці, середня кількість занять в яких була вище за задану з консолі; дисципліну, яка має найменшу кількість занять за всі місяці; дисципліну, яка не має в заданий місяць занять.   **У класі Сервіс реалізувати такі методи:**   1. Вивести на консоль згенерований в класі **Вектор** масив, масив після сортування, масив після переставлення елементів, амплітуду масиву. 2. Вивести на консоль матрицю, згенеровану в класі **Матриця**, місяці, середня кількість занять в яких була вище за задану з консолі; дисципліну, яка має найменшу кількість занять за всі місяці; дисципліну, яка не має в заданий місяць занять |

### Контрольні запитання

1. Дайте поняття класу та об’єкту
2. Яка структура класу?
3. Дайте означення масиву
4. Охарактеризуйте одновимірні масиви.
5. Охарактеризуйте багатовимірні масиви.
6. Які різновиди масивів існують в C#?
7. Як викликати відкритий метод класу через посилання на об’єкт?
8. Передача масивів в якості аргументів.
9. Передача масивів за допомогою параметрів *ref* та *out*.

### Література

* + - 1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#. КНУ ім. Тараса Шевченка

1. C# - творчість програмування. Том 1. Об’єктно-орієнтоване програмування: підручник// Під ред. Бичкова О.С. Волин. Обереги, 2024. – 292 с.
2. The C# Coding Standard. Access mode: <https://github.com/hassanhabib/CSharpCodingStandard>
3. C# Coding Standards and Best Practices. Access mode: <https://www.dofactory.com/csharp-coding-standards>
4. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І. В., Марущак П. О. – Тернопіль: ФОП Паляниця В. А., 2020 – 320 с. Режим доступу: <https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf>

# Розділ 2. Об’єктно-орієнтоване проєктування та програмування. SOLID принципи проєктування. Принципи інкапсуляції, успадкування та поліморфізму ООП

## Лабораторна робота № 3. Конструктори та аксесори класів, вкладені та часткові класи, використання текстових файлів в C#

### Рейтинг лабораторної роботи №3

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 9 завдань | 0,5\*9 = 4,5 | 21 Березня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 5,5 |  |

### Мета роботи

1. Навчитися створювати власні класи, використовуючи атрибути та методи класів, включаючи:
   1. Конструктори класів
   2. Властивості (аксесори) класів
   3. Вкладення класів
   4. Часткові класи і методи
   5. Статичні класи
   6. Текстові файли
2. Навчитися реалізовувати доступ до відкритих методів та закритих атрибутів класів.

### Методичні рекомендації до виконання лабораторної роботи

1. Прочитайте лекцію та матеріал в підручнику.
2. Прочитайте методичні вказівки до лабораторної роботи та виконайте наведені в ній приклади (вони всі працездатні).
3. При вивченні теми лекції і виконанні завдань зверніть увагу на особливості створення конструкторів класу в C#:
   * 1. конструктор не повертає значення;
     2. конструктор може бути з параметрами і без параметрів;
     3. в класі можуть бути визначені декілька конструкторів з різними списками параметрів;
     4. якщо конструктор відсутній, він створюється автоматично (конструктор за замовчуванням). Такий конструктор не має параметрів;
     5. якщо в класі визначений хоча б один конструктор, конструктор за замовчуванням автоматично системою не додається.
4. Також зверніть увагу, що для доступу до закритих полів в C# призначені спеціальні методи-властивості *get* i *set*. Якщо відсутня частина set, властивість доступна лише для читання (*read*-*only*), якщо відсутня частина get, властивість доступна лише для запису (*write-only*).
5. Для поглибленого вивчення цього матеріалу прочитайте розділи книги [1].

### Порядок виконання роботи

1. Створити директорію Lab3, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді окремих консольних проектів в одному рішенні

3. Можлива реалізація у вигляді одного консольного проекту з окремими файлами для кожного з класів з використанням меню для демонстрації роботи кожного завдання варіанту.

4. Для кожного класу передбачити окремий файл.

### Приклади виконання завдань

#### 1. Програма розрахунку рейтингу студента. Клас Student

##### Варіант 1. Конструктор з параметрами та створення об’єкту класу

Розглянемо клас *Student*, який містить відкриті поля і конструктор з параметри для їх ініціалізації. Клас містить один відкритий метод *public void StudentRating(int R)*, який виводить відповідний текст, в залежності від рейтингу студента. В методі *Main()* виконується тестування класу.

Зверніть увагу, що опис класу розміщується в просторі імен, а не класі *Program*. Опис класу рекомендується розмістити в окремому файлі – модулі класу.

using System; //файл Student.cs

namespace ex1lab3

{

/// <summary>

/// клас студент з атрибутами та методами

/// </summary>

class Student

{

public string Name; //ім'я

public int Age; // вік

public string Role; // роль

public string Faculty; //факультет

public string Group; //група

public int Course; //курс

public int Rating; //рейтинг

/// <summary>

/// конструктор з параметрами для ініціалізації полів класу

/// </summary>

/// <param name="N">ім'я студента</param>

/// <param name="A">вік</param>

/// <param name="R">роль</param>

/// <param name="F">факультет</param>

/// <param name="G">група</param>

/// <param name="C">курс</param>

public Student(string name, int age, string role, string fac, string gr, int course )

{

Name = name;

Age = age;

Role = role;

Faculty = fac;

Group = gr;

Course = course ;

}

/// <summary>

/// виведення рекомендацій студенту відповідно до його рейтингу

/// </summary>

/// <param name="R">рейтинг студента</param>

public void StudentRating(int rank)

{

Rating = rank;

if (Rating >= 80)

Console.WriteLine("Привiт вiдмiнникам");

else

if (Rating <= 30)

Console.WriteLine("Треба вчитися краще!");

else

Console.WriteLine("Можна вчитися ще краще!");

}

}

}

//======= файл Program.cs містить точку входу в програму Main()=======

using System;

namespace ex1lab3

{

class Program

{

static void Main(string[] args)

{

//дані рейтингу

./ініціалізація полів класу виконується в конструкторі з параметрами

Student newStudent = new Student("Бака", 20, "студент", "КННІ", "K-01", 3);

Console.WriteLine("Хто ви?");

Console.WriteLine("Прiзвище = " + newStudent.Name);

Console.WriteLine("Вiк= " + newStudent.Age);

Console.WriteLine("Роль= " + newStudent.Role);

Console.WriteLine("Факультет = " + newStudent.Facultet);

Console.WriteLine("група= " + newStudent.Group);

Console.WriteLine("курс= " + newStudent.Course);

Console.WriteLine("Вкажiть Ваш рейтинг?");

string r = Console.ReadLine();

newStudent.Rating = int.Parse(r);

newStudent.StudentRating(newStudent.Rating);

Console.ReadLine();

}

}

}

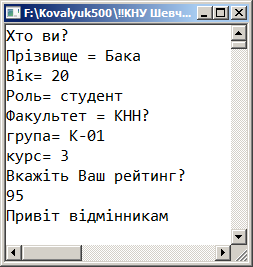


Рисунок 1 – Результат роботи програми 1

Змінимо клас, зробимо поля *Name* i *Age* закритими, а доступ до них реалізуємо через властивості *get* і *set*.

##### Варіант 2. Використання властивостей *get* і *set* замість присвоєння в конструкторі

using System; //файл Student.cs для класу Student

using System.Collections.Generic;

using System.Text;

namespace ex2Lab3

{

class Student

{

//закриті поля класу

private string name;

private int age;

//відкриті поля класу

public string Role; // роль

public string Faculty;

public string Group;

public int Course;

public int Rating;

/// <summary>

/// конструктор класу з параметрами

/// </summary>

/// <param name="F">факультет</param>

/// <param name="G">група</param>

/// <param name="C">курс</param>

/// <param name="R">роль</param>

public Student(string fac, string gr, int course, string role)

{

//конструктор з параметрами

Role = role;

Faculty = fac;

Group = gr;

Course = course;

}

/// <summary>

/// властивість для доступу до закритого поля Name через get і set для полів класу

/// </summary>

public string Name

{

get

{ return Name; }

set

{ Name = value;}

}

/// <summary>

/// властивість для доступу до закритого поля Age через get і set для полів класу

/// </summary>

public int Age

{

get

{ return age; }

set

{ age = value; }

}

/// <summary>

/// виведення повідомлень в залежності від значення рейтингу

/// </summary>

/// <param name="Rat">рейтинг студента</param>

public void StudentRating(int rat)

{

Rating = rat;

if (Rating >= 80)

Console.WriteLine("Привіт відмінникам");

else

if (Rating <= 30)

Console.WriteLine("Треба вчитися краще!");

else

Console.WriteLine("Можна вчитися ще краще!");

}

}

}

//файл Program.cs для класу Program

using System;

namespace ex2Lab3

{

class Program

{

static void Main(string[] args)

{

//дані рейтингу

//ініціалізація полів класу виконується в конструкторі з параметрами

Student newStudent = new Student("КННІ", "K-01", 3,"Студент");

Console.WriteLine("Хто ви?");

//використовуємо властивість

newStudent.Name = Console.ReadLine();

Console.WriteLine("Скiльки вам років?");

//використовуємо властивість

newStudent.Age = int.Parse(Console.ReadLine());

Console.WriteLine("Прiзвище = " + newStudent.name);

Console.WriteLine("Вiк= " + newStudent.age);

Console.WriteLine("Роль= " + newStudent.Role);

Console.WriteLine("Факультет = " + newStudent.Faculty);

Console.WriteLine("група= " + newStudent.Group);

Console.WriteLine("курс= " + newStudent.Course);

Console.WriteLine("Ваш рейтинг?");

string r = Console.ReadLine();

// привсоєння значення відкритому полю класу

newStudent.Rating = int.Parse(r);

newStudent.StudentRating(newStudent.Rating);

Console.ReadLine();

}

}

}

#### 2. Програма автоматизованого обліку банківських відомостей

Програма є автоматизованою системою обліку банківських відомостей. На кожного клієнта банку зберігаються наступні відомості:

* прізвище, ім’я, по-батькові;
* дата народження;
* паспортні дані;
* ідентифікаційний код;
* місце роботи (навчання);
* номери рахунків.

Для кожного клієнта визначимо операції:

* додати нового клієнта;
* видалити клієнта;
* змінити реквізити клієнта.

На кожному рахунку зберігається інформація про поточний баланс. З кожним рахунком можна виконувати наступні дії:

* відкриття, закриття;
* внесення грошей, зняття грошей;
* перегляд балансу.

Створимо два класи:

* класс Client для опису інформації про клієнта;
* клас Account для опису банківського рахунку

##### Клас *Client*. Варіант 1 - конструктор без параметрів, властивості класу

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Client

{

//поля класу

private string name;

private DateTime birthDate;

private string passport;

private string iD;

private string job;

private string nomAccount;

/// <summary>

/// конструктор без параметрів

/// </summary>

public Client()

{

}

}

}

Область видимості полів класу відповідно до правил має бути визначена або як **закрита**, або як **захищена**. Доступ же до полів - членів класу має бути організований або за допомогою методів, або за допомогою властивостей класу. Створимо властивості класу *Client*, які забезпечують читання і запис значень закритих полів класу.

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Client

{

//поля класу

private string name;

private DateTime birthDate;

private string passport;

private string iD;

private string nomAccount;

/// <summary>

/// конструктор без параметрів

/// </summary>

public Client()

{

//поля класу слід ініціалізувати значеннями за замовчуванням

}

/// <summary>

/// властивість класу з методами get, set для закритого поля passport

/// </summary

public string Passport

{

get

{return passport; }

set

{ passport = value; }

}

/// <summary>

/// властивість класу з методами get, set для закритого поля name

/// </summary>

public string Name

{

get

{ return name; }

set

{ name = value; }

}

/// <summary>

/// ////////////////

/// </summary>

public int Age

{

get

{

int yearNumbers; //кількість років

yearNumbers = DateTime.Now.Year - BirthDate.Year;

return yearNumbers;

}

}

/// <summary>

/// методи get і set для закритого поля BirthDate

/// </summary>

public DateTime Birthdate

{

get

{ return birthDate; }

set

{

if (DateTime.Now > value)

birthDate = value;

else

throw new Exception("Введена невірна дата народження");

}

}

/// <summary>

/// властивість класу з методами get, set для закритого поля iD

/// </summary>

public string ID\_kod

{

get { return iD; }

set

{ iD = value; }

}

/// <summary>

/// властивість класу з методами get, set для закритого поля nomAccount

/// </summary>

public string Nom\_Account

{

get { return nomAccount; }

set

{ nomAccount = value; }

}

}

}

Як видно з прикладу, властивість складається з методів *set* **і** *get*. При цьому властивість повинна містити хоч би один з методів. Метод *set* дозволяє змінювати значення поля класу, *get* − отримувати значення. У метод *set* передається значення параметра за допомогою змінної *value*. Обидва методи можуть містити довільну кількість операторів, що описують алгоритм виконання дій в процесі читання або запису значення в полі класу. У даному прикладі властивості *Passport* і *Name* дозволяють просто дістати доступ до полів класу, читаючи або встановлюючи значення відповідних змінних. Властивість *Birthdate* також призначена для читання і запису значення змінної - члена класу *BirthDate*. При цьому при читанні значення (операція *get*) відбувається просто передача значення змінної *BirthDate*, при спробі ж запису нового значення в цю змінну відбувається перевірка допустимості встановленого значення змінної. В даному випадку перевірка зводиться до порівняння нового значення дати народження з поточною датою. Якщо встановлене значення дати народження більше або дорівнює поточній даті, генерується виключення, яке не дозволяє записати нове значення в змінну, - член класу.

Властивість *age* застосовується для отримання поточного віку клієнта. Вона призначена лише для читання значення із змінної, тому містить лише метод *get*. При використанні властивості *age* відбувається обчислення поточного значення віку клієнта в роках шляхом віднімання року народження від поточного значення року.

Використання властивостей аналогічно використанню змінних. У наступному прикладі створюється об'єкт *с1* класу *Client*. Потім поля цього об'єкту заповнюються значеннями з використанням властивостей. Після цього на екран виводяться значення полів, для цього також застосовуються властивості класу:

using System;

namespace ex3Lab3

{

class Program

{

static void Main(string[] args)

{

Client c1 = new Client();

c1.Name = "Вася";

c1.Passport = "9002";

c1.Birthdate = new DateTime(2003, 08, 03);

c1.ID\_kod = "123456789";

c1.Nom\_Account = "8097";

Console.WriteLine("Имя=" + c1.name);

Console.WriteLine("Паспорт=" + c1.passport);

Console.WriteLine("Возраст=" + c1.age);

Console.WriteLine("Иден.код=" + c1.ID\_kod);

Console.WriteLine("Код счета=" + c1.Nom\_Account);

Console.ReadKey();

}

}

}

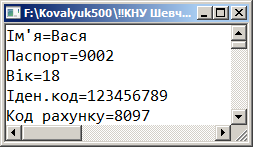


Рисунок 2 – Результат роботи програми 2

##### Клас Client. Варіант 2 - конструктор класу з параметрами

/// <summary>

/// конструктор з параметрами

/// </summary>

/// <param name="clientName">ім'я клієнта</param>

/// <param name="clientPassport">паспорт</param>

/// <param name="clientBirthDate">дата народження</param>

public Client(string clientName, string clientPassport,  
 DateTime clientBirthDate)

{

name = clientName;

passport = clientPassport;

birthdate = clientBirthDate;

}

Видно, що конструктор має три параметри. У тілі конструктора відбувається запис переданих як параметри значень у відповідні поля класу за допомогою властивостей даного класу. У випадку з датою народження це дозволяє не дублювати процедуру перевірки введеної дати, а скористатися алгоритмом, реалізованим у властивості *birthdate*.

Створимо також метод, що дозволяє змінити значення полів об'єкту класу *Client*:

/// <summary>

/// редагування полів

/// </summary>

/// <param name="clientName">ім'я клієнта</param>

/// <param name="clientPassport">паспорт</param>

/// <param name="clientBirthDate">дата народження</param>

public void EditClient(string clientName, string clientPassport,  
 DateTime clientBirthDate)

{

name = ClientName;

passport = ClientPassport;

birthdate = ClientBirthDate;

}

Як видно з прикладу, код цього методу практично повністю ідентичний конструктору з параметрами з різницею в імені, а також в типі значення, яке повертається. Звичайно, в даному випадку можна було б обійтися і використанням властивостей для зміни значень полів класу, проте, інколи буває корисно, аби такого роду зміни були реалізовані в рамках одного методу, тим більше, якщо алгоритм змін є нестандартним.

Тепер, з використанням конструктора з параметрами, можна створити і відразу ж ініціалізувати об'єкт класу *Client*:

Client c1 = new Client("Вася", "9002", new DateTime(2003, 08, 03));

Змінений код з викликом редагування полів класу:

static void Main(string[] args)

{

Client c1 = new Client("Вася", "9002", new DateTime(2003, 08, 03));

c1.ID\_kod = "123456789";

c1.Nom\_Account = "8097";

Console.WriteLine("Iм'я=" + c1.Name);

Console.WriteLine("Паспорт=" + c1.Passport);

Console.WriteLine("Вiк=" + c1.Age);

Console.WriteLine("Iден.код=" + c1.ID\_kod);

Console.WriteLine("Код рахунку=" + c1.Nom\_Account);

Console.ReadKey();

// редагування полів

c1.EditClient("Iван", "4567", new DateTime(2000, 09, 01));

//Виведення значень відредагованих полів

Console.WriteLine("Виведення значень вiдредагованих полiв");

Console.WriteLine("Iм'я=" + c1.Name);

Console.WriteLine("Паспорт=" + c1.Passport);

Console.WriteLine("Вik=" + c1.Age);

Console.ReadKey();

}

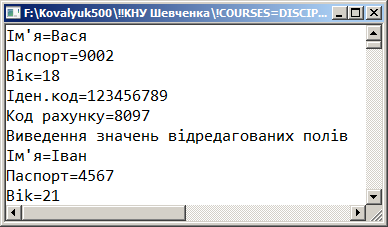


Рисунок 3 – Результат роботи програми 2, варіант 2

##### Клас Account. Варіант 1

У першому варіанті класу *Account* активно використовуватимемо поля класу. Окрім двох основних полів *credit* і *debit*, які зберігають надходження і витрати рахунку, введемо поле *balance*, яке задає поточний стан рахунку, і два поля, пов'язані з останньою виконуваною операцією. Поле *sum* зберігатиме суму грошей поточної операції, а поле *result* – результат виконання операції. Полів в класу багато, і як наслідок, в методах класу аргументів буде небагато.

class Account

{

//закриті поля класу

int debit = 0, credit = 0, balance = 0;

int sum = 0, result = 0;

/// <summary>

/// Зарахування на рахунок з перевіркою

/// </summary>

/// <param name="sum">сума, що зараховується</param>

public void PutMoney(int sum)

{

this.sum = sum;

if (sum > 0)

{

credit += sum; balance = credit - debit; result = 1;

}

else result = -1;

Mes();

}//PutMoney

/// <summary>

/// Зняття з рахунку з перевіркою

/// </summary>

/// <param name="sum">сума, що знімається</param>

public void GetMoney(int sum)

{

this.sum = sum;

if (sum <= balance)

{

debit += sum; balance = credit - debit; result = 2;

}

else result = -2;

message();

}//getMoney

/// <summary>

/// Повідомлення про виконання операції

/// </summary>

void message()

{

switch (result)

{

case 1:

Console.WriteLine("Операція зарахування грошей пройшла успішно!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case 2:

Console.WriteLine("Операцiя зняття грошей пройшла успiшно!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case -1:

Console.WriteLine("Операцiя зарахування грошей не виконана!");

Console.WriteLine("Сума має бути бiльше за нуль!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

case -2:

Console.WriteLine("Операцiя зняття грошей не виконана!");

Console.WriteLine("Сума має бути не бiльше балансу!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance);

break;

default:

Console.WriteLine("Невiдома операцiя!");

break;

}

Console.ReadLine();

}

}

Як можна бачити, лише у методів *getMoney***()** і *putMoney***()** є один вхідний аргумент. Це той аргумент, який потрібний по суті справи, оскільки лише клієнт може вирішити, яку суму він хоче зняти або покласти на рахунок. Інших аргументів в методів класу немає - вся інформація передається через поля класу. Зменшення числа аргументів призводить до підвищення ефективності роботи з методами, оскільки зникають витрати на передачу фактичних аргументів. Але при цьому ускладнюються операції роботи з вкладом, оскільки потрібно у момент виконання операції оновлювати значення полів класу. Закритий метод *Mes*() викликається після виконання кожної операції, повідомляючи про те, як пройшла операція, і інформуючи клієнта про поточний стан його балансу.

##### Клас Account. Варіант 2

Спроектуємо аналогічний клас *Account1*, який відрізняється лише тим, що у нього буде менше полів. Замість поля *balance* в класі з'явиться відповідна функція з цим же іменем, замість полів *sum* і *result* з'являться аргументи в методах, що забезпечують необхідну передачу інформації. От як виглядає цей клас:

using System;

using System.Collections.Generic;

using System.Text;

namespace ex3Lab3

{

class Account1

{

//закриті поля класу

int debit = 0, credit = 0;

/// <summary>

/// Зарахування на рахунок з перевіркою

/// </summary>

/// <param name="sum">зарахована сума</param>

public void PutMoney(int sum)

{

int res = 1;

if (sum > 0) credit += sum;

else res = -1;

message(res, sum);

} //putMoney

/// <summary>

/// Зняття з рахунку со счета з перевіркою

/// </summary>

/// <param name="sum"> сума, що знімається</param>

public void GetMoney(int sum)

{

int res = 2;

if (sum <= balance()) debit += sum;

else res = -2;

balance();

message(res, sum);

}//getMoney

/// <summary>

/// розрахунок балансу

/// </summary>

/// <returns>поточний баланс</returns>

int balance()

{

return (credit - debit);

}

/// <summary>

/// Повідомлення про виконання операції

/// </summary>

void message(int result, int sum)

{

switch (result)

{

case 1:

Console.WriteLine("Операцiя зарахування грошей пройшла успiшно!");

Console.WriteLine("Cума={0},Ваш текущий баланс={1}", sum, balance());

break;

case 2:

Console.WriteLine("Операція зняття грошей пройшла успiшно!");

Console.WriteLine("Cума={0},Ваш текущий баланс={1}", sum, balance());

break;

case -1:

Console.WriteLine("Операцiя зарахування грошей не виконана!");

Console.WriteLine("Сума має бути більше за нуль!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance());

break;

case -2:

Console.WriteLine("Операцiя зняття грошей не виконана!");

Console.WriteLine("Сума має бути не більше балансу!");

Console.WriteLine("Cума={0},Ваш поточний баланс={1}", sum, balance());

break;

default:

Console.WriteLine("Невiдома операцiя!");

break;

}

}

}//Account1

}

Порівнюючи цей клас з класом *Account*, можна бачити, що число полів скоротилося з п'яти до двох, спростилися основні методи *GetMoney*() і P*utMoney*(). Але, як плата, в класі з'явився додатковий метод *balance*(), що багато разів викликається, і в методі *message*() тепер з'явилися два аргументи. Який клас кращий? Однозначно сказати не можна, все залежить від контексту, від пріоритетів, заданих при створенні конкретної системи.

*Метод TestAccounts() класу* Program*, що тестує роботу з класами* Account *і* Account, має викликатися у методі *Main():*

using System;

namespace ex3Lab3

{

class Program

{

static void TestAccounts()

{

Account myAccount = new Account();

myAccount.PutMoney(6000);

myAccount.GetMoney(2500);

myAccount.PutMoney(1000);

myAccount.GetMoney(4000);

myAccount.GetMoney(1000);

//Аналогічна робота з класом Account1

Console.WriteLine("Новий клас i новий рахунок!");

Account1 myAccount1 = new Account1();

myAccount1.PutMoney(6000);

myAccount1.GetMoney(2500);

myAccount1.PutMoney(1000);

myAccount1.GetMoney(4000);

myAccount1.GetMoney(1000);

Console.WriteLine("Кiнець роботи");

Console.ReadLine();

}// TestAccounts

static void Main(string[] args)

{

Console.WriteLine("Тестування класiв Account i Account1 ");

TestAccounts();

}

}

}

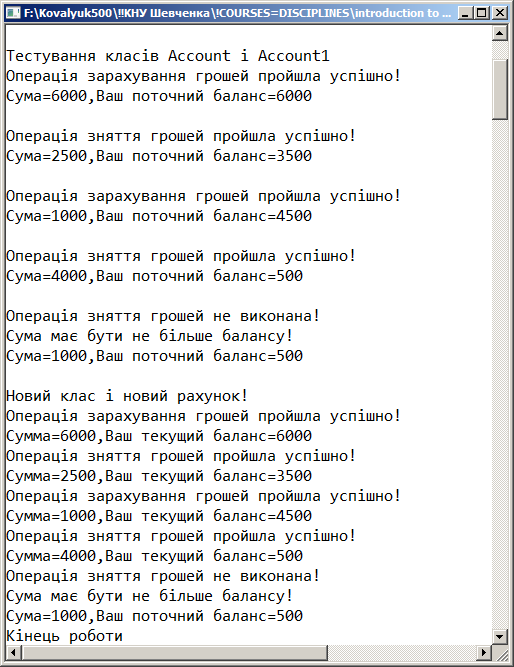


Рисунок 4 – Результат роботи програми. Клас Account. Варіант 2

#### 3. Вбудовані (вкладені) класи

У мові C# будь-який клас у своїй реалізації може містити оголошення іншого класу. Клас, що оголошується в межах фігурних дужок іншого класу, називається *вкладеним класом*. Вкладені класи можуть мати модифікатори доступу *public, protected, internal, protected internal, private* або *private protected*. Об‘єкт вкладеного класу можна оголосити у випадку, якщо вкладений клас оголошений як видимий (не як *private*).

Нехай *Outer* – ім’я класу, який містить в собі оголошення іншого класу з іменем *Inner*, *Inner* – ім’я класу, який оголошується в класі *Outer*. Якщо вкладений клас оголошено як не *private*-клас, то створення екземпляру цього класу має такий вигляд: Outer.Inner objInner = new Outer.Inner();

using System;

using System.Collections.Generic;

using System.Text;

namespace ex6Lab3

{

class Outer

{

// внутрішні змінні класу Outer

public int D;

static public int Sd;

// доступ до private-класу Inner1 з внутрішнього методу класу

public Inner GetInner()

{

Console.WriteLine("Метод GetInner() зовнiшнього класу");

Inner i1 = new Inner(); // створити екземпляр класу Inner1

// доступ до члена даних класу Inner1 через екземпляр класу

i1.d1 = 25;

// доступ до статичного члена вкладеного класу Inner1

Inner.sd1 = 30;

return i1;

}`

// private-вкладений клас Inner

public class Inner

{

public int D1;

public static int Sd1;

public string Method()

{

Console.WriteLine("Метод внутрiшнього класу ");

string str = " результат вкладеного класу Inner";

return str;

}

}

}

}

using System;

namespace ex6Lab3

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello World!");

// Використання вкладених класів Outer.Inner1, Outer.Inner2, Outer

// 1. Оголосити об'єкт класу Outer

Outer o = new Outer();

o.D = 300; // доступ до змінної через екземпляр класу Outer

Outer.Sd = 230; // доступ до статичної змінної

Console.WriteLine("Поля зовнiшнього класу: "+ o.D+" "+ Outer.Sd);

Console.WriteLine("Main:"+ o.GetInner());

// 2. Оголосити об'єкт public-класу Inner

Outer.Inner i1 = new Outer.Inner();

i1.D1 = 440;

Outer.Inner.Sd1 = 500; // доступ до статичної змінної внутрішнього класу

Console.WriteLine("Поля внутрiшнього класу " + i1.D1 + " " + Outer.Inner.Sd1);

Console.WriteLine("Main:"+i1.Method());

}

}

}

#### 4. Часткові класи (класи, що розділяються) і часткові методи

Класи можуть бути частковими. Тобто ми можемо мати кілька файлів з визначенням одного і того самого класу, і при компіляції всі ці визначення будуть скомпільовані в одне. Наприклад, визначимо в проекті два файли з кодом.

// частина 1 класу Person

public partial class Person

{

public void Move()

{

Console.WriteLine("I am moving");

}

}

// частина 2 класу Person

public partial class Person

{

public void Eat()

{

Console.WriteLine("I am eating");

}

}

Отже, два файли в проекті містить визначення одного і того самого класу *Person*, які містять два різних методу. І обидва визначені класи є **частковими**. Для цього вони визначаються з ключовим словом ***partial***.

Ключове слово *partial* вказує, що інші частини класу, структури або інтерфейсу можуть бути визначені в просторі імен. Всі частини повинні використовувати ключове слово *partial*. Для формування остаточного класу всі частини повинні бути доступні під час компіляції. Всі частини повинні мати однакові модифікатори доступу, наприклад *public*, *private* тощо.

Приклад використання методів часткових класів:

class Program

{

static void Main(string[] args)

{

Person Vadim = new Person();

Vadim.Move(); // метод першого часткового класу

Vadim.Eat(); // метод другого часткового класу

Console.ReadKey();

}

}

Часткові класи можуть містити часткові методи. Такі методи також визначаються з ключовим словом *partial*. Причому визначення часткового методу без тіла методу знаходиться в одному частковому класі, а реалізація цього самого методу – в іншому частковому класі..

public partial class Person

{

partial void DoSomething(); //оголошення часткового методу

public void OneMethod()

{

Console.WriteLine("Start");

DoSomething(); //call method of second partion class

Console.WriteLine("Finish");

}

}

public partial class Person

{

//визначення часткового методу

partial void DoSomething()

{

Console.WriteLine("I am reading a book");

}

}

Як правило, часткові методи завжди закриті (private). Виклик часткових методів здійснюється через відкриті методи

Person Ivan = new Person();

Ivan. OneMethod(); //метод викликає закритий частковий метод

#### 5. Текстові файли

В С# для роботи з файлами і потоками використовуються класи ***FileStream*, *StreamWriter***, ***StreamReader*, *FileInfo*** і інші класи.

Для введення і виведення даних не потоками, а рядками призначені класи *StringReader*, *StringWriter*.

Розглянемо приклад**.** В цьому прикладі спочатку створюється об'єкт ***f*** класу *StreamWriter*, який містить файл для виводу **output.txt**. Якщо не вказати повний шлях до файлу як нашому прикладі, то файл буде створений в директорії, де розміщується exe-файл програми: .*..\\bin\\debug\\output.txt*

Далі визначаються і ініціалізуються змінні ***int i*** та ***string s****,* які записуються у вихідний файл. Метод *Close()* закриває файл. В другій частині прикладу по черзі зчитуються рядки з файлу *input.txt,* перетворюються у числові типи і виводяться на консоль.

using System;

using System.IO; // підключення простору імен для роботи з файлами

namespace ex7Lab3

{

class Program

{

static void Main(string[] args)

{

//запис у текстовий файл

Console.WriteLine("Hello World!");

StreamWriter fout = new StreamWriter("output.txt"); // 2

//відкрити файл для запису

int i = 3;

string s1 = "Мова програмування С# - це C++++ ";

fout.WriteLine("i = " + i); // 3

fout.WriteLine("s1 = " + s1); // 4

fout.Close(); // 5

//=========читання з текстового файлу================

StreamReader fin = new StreamReader("f:\\input.txt"); // відкрити

//файл для читання

string s2 = fin.ReadLine(); // читати рядок

Console.WriteLine("s = " + s2);

char c = (char)fin.Read(); // читати символ

fin.ReadLine();

Console.WriteLine("c = " + c);

string buf = fin.ReadLine(); //читати рядок і конвертувати у ціле число

int j = Convert.ToInt32(buf);

Console.WriteLine(j);

buf = fin.ReadLine(); //читати рядок і конвертувати у число double

double x = Convert.ToDouble(buf);

Console.WriteLine(x);

buf = fin.ReadLine(); //читати рядок і конвертувати у число double

double y = double.Parse(buf);

Console.WriteLine(y);

buf = fin.ReadLine(); //читати рядок і конвертувати у ціле число

decimal z = decimal.Parse(buf);

Console.WriteLine(z);

fin.Close();

}

}

}

### Варіанти завдань для лабораторної роботи № 3

#### Інструкція з виконання лабораторного завдання

Виконання лабораторної роботи передбачає три етапи (рис.4).

Етап 1 – об’єктно-орієнтований аналіз (OOA) з виконанням об’єктно-орієнтованої декомпозиції предметної області. Об'єктно-орієнтований аналіз дозволяє чітко визначити основні сутності системи та їхню поведінку. На вході - опис предметного середовища Результат – модель предметної області у вигляді сукупності сутностей з атрибутами та операціями, що характеризують поведінку сутностей, відповідно до принципів SOLID.

Етап 2 – об’єктно-орієнтоване проєктування (OOD). На входе результат об’єктно-орієнтованої декомпозиції. Результат – UML діаграма класів, яка містить класи і взаємозв'язки між ними та UML Use Case діаграма для визначення сценарію роботи програми.

Етап 3 – об’єктно-орієнтоване програмування (OOP). На вході UML діаграма класів і UML Use Case діаграма. Результат – код програми відповідно до вимог чистого коду (гарного стилю програмування) та принципів SOLID (в цій лабораторній роботі слід дотримуватись принципів єдиної відповідальності (SRP), та принципу відкритості – закритості (OCP)).Виконання лабораторної роботи передбачає три етапи.

**Етап 1 – об’єктно-орієнтований аналіз (OOA)**.

На вході етапу ООА розробник має опис предметної області, в якій описана бізнес-логіки системи. На виході як результат ООА – модель предметної області відповідно до принципів SOLID з визначенням класів, їх атрибутів, об’єктів та методів, що визначають їх поведінку.

**Етап 2 – об’єктно-орієнтоване проєктування (OOD).**

На вході етапу OOD модель предметної області як результат етапу ООА. На виході етапу OOD – *UML діаграма класів* з переліком усіх класів, атрибутів, операцій (методів) та взаємозв'язків між класами. Для лабораторної роботи 3 рекомендується встановлювати **асоціативні** (направлені і ненаправлені) зв’язки між класами. Кратність зв’язків **один до одного**. **Не використовуйте зв’язки типу успадкування, агрегація, композиція.**

**Етап 3 – об’єктно-орієнтоване програмування (OOP)**.

На вході етапу OOP маємо UML діаграму класів (результат етапу OOD). На виході етапу OOP– код класів відповідно до вимог чистого коду (гарного стилю програмування) та принципів SOLID (в цій лабораторній роботі слід дотримуватись принципів *єдиної відповідальності (SRP), та принципу відкритості – закритості (OCP)*). Сценарій роботи програми реалізується у функції Main() або розробник реалізує його в окремому класі. Для розробки сценарію роботи програми необхідно розробити UML діаграму варіантів використання (Use Case Diagram)

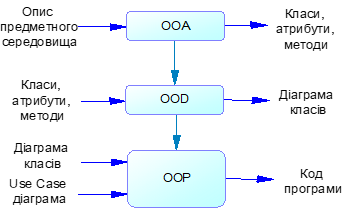


Рисунок 4 – Етапи об'єктно-орієнтованого підходу до розробки ПЗ

**Для забезпечення SOLID принципу відкритості – закритості (OCP) потрібно створити 4 версії коду лабораторної роботи №3**. **Кожна версія коду в окремому проєкті:**

- версія 1 включає пункти 1,2,3,4,5,6. Дедлайн зараховується по версії 1.

- версія 2 включає код версії 1 та пункт 7 завдання;

- версія 3 включає код версії 2 та пункт 8 завдання;

- версія 4 включає код версії 3 та пункт 9 завдання.

**Для забезпечення SOLID принципу єдиної відповідальності (SRP)** слід визначити методи, які властиві зазначеним об’єктам предметної області. Методи, які не властиві відповідним об’єктам, слід перенести в інші класи, визначивши їх самостійно. Для опрацювання полів класів, які не задіяні в заданих предметною областю методах класів, їх слід або вилучити, або задати методи, які ці поля використовуватимуть самостійно. Якщо студент вважає для зручності роботи користувача з програмою використовувати меню, для цього слід розробити окремий клас Menu. Не ускладнюйте інші класи, включаючи меню у вигляді метода в клас.

На етапі **об’єктно-орієнтоване програмування (OOP)** потрібно розробити код відповідно до пунктів завдання.

**Пункт 1** завдання передбачає розробку класів відповідно до опису предметної області, що містять закриті поля.

**Пункти 2 та 3** завдання передбачають розробку конструкторів за замовчування, з параметрами (та конструктора копії) в створених класах.

**Пункт 5 завдання** – розробка в створених класах відкритих методів за сценарієм роботи програми (UML Use Case diagram). Алгоритми методів студент розробляє самостійно. В завданні алгоритм може бути не описаний.

**Пункт 6 завдання** – створення сервісного класу, в який включити відкриті методи для читання з консолі, виведення результатів на консоль, методів запису та читання з текстових файлів.

**Пункт 7 завдання** – доповнення класів вбудованими (вкладеними класами). Рекомендується у вигляді другої версії лабораторної роботи.

**Пункт 8 завдання** – доповнення другої версії програми частковими класами та методами. Зробити у вигляді третьої версії лабораторної роботи.

**Пункт 9 завдання** – доповнення третьої версії новим статичним класом.

В методі *Main*() класу *Program* продемонструвати виклики усіх методів усіх класів. Усі значення, що розраховуються, записувати до *текстових файлів,* методи їх обробки включити в сервісний клас, оскільки ці методи не властиві об’єктам предметної області.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**. **Номер варіанта визначається за порядковим номером студента в журналі групи.**

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є викладач і студент. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 1.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 1.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Викладач | Джерело навчального контенту; координує дисципліну; виконує оцінювання студентів | Ім’я викладача Назва дисципліни Навчальне навантаження (години) Кількість студентів | Збільшити кількість студентів  Зменшити кількість студентів Змінити обсяг навчального навантаження  Поставити оцінку студенту Передати матеріал Записати оцінку в журнал | | Студент | Отримувач знань і навчального контенту; виконує завдання; формує власний рейтинг | Ім’я студента Назва дисципліни Список робіт з оцінками Обсяг виконаних робіт | Додати оцінку Переглянути оцінки  Розрахувати рейтинг Завантажити матеріал | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один викладач** працює з **одним студентом**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (викладач і студент) для реалізації своїх дій.  Розробити таблицю зв’язків між сутностями за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про викладача і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітній процес 1».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Teacher,** **Student, Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.1.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну станів об’єктів відповідно до табл.1.1.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Student** вбудований (вкладений) клас Дипломний\_проект (**DiplomaProject**) з додатковими сутностями згідно з табл. 1.3:  Таблиця 1.3. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Дипломний проект | Інтегрує знання студента й оцінюється викладачем | Назва теми,  Кількість реалізованих алгоритмів,  Складність теми.  Оцінка  Ім’я керівника | Вибрати тему з переліку.  Визначити складність теми.  Визначити оцінку |   **Алгоритм вибору теми** такий: перелік тем подати текстовим файлом, з консолі ввести ключові слова, читати рядки файлу в пам’ять, шукати входження ключових слів в прочитані рядки, вивести на консоль рядок, в якому знайдені задані ключові слова.  **Алгоритм визначення оцінки** дипломного проекту (ДП): порахувати кількість реалізованих в ДП методів, задати складність кожного методу, визначити складність усього ДП, визначити співвідношення складності ДП та балів, порахувати оцінку в балах.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Student**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод вибору теми, в інший файл – метод розрахунку оцінки за дипломний проект.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас Наукова стаття (**ScientificPaper**), включивши в нього функцію (на вибір студента) з варіанта 1 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є кафедра і студент. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 2.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 2.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Кафедра | Організаційна одиниця, яка адмініструє напрям підготовки та облік студентів | Назва кафедри  Кількість студентів  Напрям підготовки  Перелік дисциплін  Максимальна кількість студентів | Збільшити кількість студентів в межах максимальної кількості та з відповідним напрямом підготовки  Зменшити кількість студентів  Додати дисципліну, що відповідають напрямку  Видалити дисципліну | Дія дозволена, якщо поточна кількість < максимальної кількості  Додати дисципліну можна, якщо вона відповідає напрямку підготовки кафедри  Видалення можливе, якщо дисципліна присутня у списку | | Студент | Учасник освітнього процесу, чия успішність вимірюється на основі оцінок | Ім’я студента  Напрям підготовки  Список оцінок  Рівень навчального навантаження  Номер залікової книжки | Визначити рейтинг успішності  Переглянути оцінки  Додати оцінку | Додати оцінку можна, якщо вона не перевищує рейтинг в 100 балів | | Сервіс | Технічний посередник: забезпечує обмін, введення /виведення та збереження даних | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |  |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра навчає** **одного студента**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і студент) для реалізації своїх дій.  Розробити таблицю зв’язків між сутностями за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів.  На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента.  Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітній процес 2».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Department,** **Student, Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.2.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.2.1.  **Алгоритм методів збільшення та зменшення кількості студентів**, які навчаються на кафедрі такий: згенерувати оцінку з акредитації кафедри (A або B, або E, або F): оцінка А – збільшення кількості студентів на 20%, оцінка В – кількість студентів не змінюється, оцінка Е - кількість студентів зменшується на 10%, оцінка F - повідомлення про ліквідацію ліцензії та зменшення кількості студентів на 50%.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Student** вбудований (вкладений) клас КонкурснаРобота (**СompetitiveWork**) з додатковими сутностями згідно з табл. 2.2:  Таблиця 2.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Конкурсна робота студента | Творчий результат студента, який оцінюється в межах конкурсу і відображає індивідуальну активність | Назва роботи  Тематика конкурсу  Дата подання  Оцінка / рейтинг  Статус (подано, прийнято, відхилено)  Призове місце  Автор (студент) | Подати роботу  Визначити відповідність теми роботи тематиці конкурсу  Оцінити роботу  Призначити статус  Присвоїти місце  Переглянути деталі |   **Алгоритм «Подати роботи»**. Студент вводить назву роботи, спеціальність. Фіксується поточна дата як дата подання. Статус роботи встановлюється як "подано". Робота зв’язується з автором (студентом). Перевіряється, чи студент не подав іншу роботу (у моделі 1:1). Якщо умови виконані — запис зберігається.  **Алгоритм перевірки відповідності теми роботи тематиці конкурсу**. Перелік тематик конкурсу подати рядками в текстовому файлі, читаючи рядки з файлу, шукати збіги ключових слів з назви конкурсної роботи з прочитаним рядком з файлу, вивести на консоль рядки файлу, в яких знайдені збіги ключових слів з назви роботи з тематикою конкурсу.  Алгоритми інших дій студент розробляє самостійно.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Student**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод перевірки відповідності назви конкурсної роботи тематиці конкурсу, в інший файл – метод розрахунку балів за конкурсну роботу.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **СreativeWork**, включивши в нього функцію (на вибір студента) з варіанта 2 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є кафедра і викладач. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 3.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 3.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Кафедра | Керує ресурсами: планує навантаження, розподіляє дисципліни, забезпечує відповідність нормам | Назва кафедри  Загальне річне навантаження (год) кафедри  Поточна кількість викладачів  Перелік дисциплін  Мінімальний обсяг дисципліни (90 год)  Максимальне навантаження на одного викладача (600 год) | Додати викладача (з перевіркою ліміту)  Видалити викладача  Розподілити навантаження між викладачами  Переглянути розподіл навантаження | Додати викладача можливо, тільки якщо на нього припадає не більше 600 год.  Додати викладача можливо, якщо його спеціалізація відповідає профілю дисципліни.  Кафедра не може створювати дисципліни з меншим навантаженням ніж 90 годин | | Викладач | Реалізує навчальну, наукову та методичну діяльність в межах нормативного навантаження | ПІБ  Спеціалізація  Загальне річне навантаження (год) викладача  Кількість дисциплін  Кількість наукових статей  Кількість методичних робіт  Кількість організаційних робіт | Визначити рейтинг викладача  Перевірити перевищення навантаження  Додати наукову/методичну роботу | рейтинг викладача =  статті × 10 + методичні роботи × 5 + організаційні роботи × 3 | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |  |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один викладач працює на одній кафедрі** (модель 1:1). **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і викладач) для реалізації своїх дій.  Розробити таблицю зв’язків між сутностями за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітній процес 3».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Department,** **Teacher, Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.3.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.3.1.  **Алгоритм методу розподілу навантаження.** Згенерувати число, що означає загальне навантаження кафедри в діапазоні від 6000 до 10000 годин. Перевірити можливості розподілу. Обчислити базовий розподіл як загальне навантаження кафедри / кількість викладачів і порівняння його з мінімальним обсягом дисципліни. Перевірити, що не порушені норми (навантаження викладача <= 600), не порушений мінімальний обсяг дисципліни (обсяг >= 90), загальне навантаження кафедри <= кількість викладачі\*600.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Teacher** вбудований (вкладений) клас Керівництво конкурсними роботами (**ContestWorkManage**r) з додатковою сутністю згідно з табл. 3.2:  Таблиця 3.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Конкурсна робота студента | Відображає процес супроводу та оцінки конкурсної роботи викладачем | Назва конкурсної  Тема конкурсу  Статус (прийнята / відхилена)  Рейтинг роботи  Призове місце, отримане конкурсною роботою (1,2, 3). Значення 0 - робота не попала в номінацію призових.  Викладач-керівник | Подати роботу  Визначити відповідність теми роботи тематиці конкурсу  Оцінити роботу  Призначити статус  Присвоїти місце  Переглянути деталі |   Приклад деяких алгоритмів. **Алгоритм «Подати роботи»**. Викладач вводить назву роботи, спеціальність. Фіксується статус роботи встановлюється як «подано».  **Алгоритм перевірки відповідності теми роботи тематиці конкурсу**. Перелік тематик конкурсу подати рядками в текстовому файлі, читаючи рядки з файлу, шукати збіги ключових слів з назви конкурсної роботи з прочитаним рядком з файлу, вивести на консоль рядки файлу, в яких знайдені збіги ключових слів з назви роботи з тематикою конкурсу.  **Алгоритм розрахунку рейтингу конкурсної роботи:** згенерувати 3 цілих числа в діапазоні від 0 до 10 (числа свідчать про наукову новизну, практичну цінність, якість оформлення), Сума балів від 25 до 30 свідчить про перше призове місце, сума балів від 20 до 24 – друге призове місце, сума балів від 15 до 19 – третє призове місце, сума балів менше за 15 – робота в номінацію не попала .  Алгоритми інших дій студент розробляє самостійно.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Teacher**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод перевірки відповідності назви конкурсної роботи тематиці конкурсу, в інший файл – метод розрахунку рейтингу конкурсної роботи.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас Міжнародний проект **InterProject**, включивши в нього функцію (на вибір студента) з варіанта 3 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує модель державно-приватного партнерства, в якій учасниками моделі є ІТ-компанія і кафедра. ІТ-компанія створює кафедру. Кафедра укладає партнерство з факультетом. Факультет делегує навчальне навантаження кафедрі. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 4.1 подані ролі, атрибути та операції сутностей моделі державно-приватного партнерства  Таблиця 4.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | ІТ-компанія | Комерційна організація, яка створює або інтегрує освітні підрозділи для підготовки фахівців | Назва компанії  Сфера діяльності  Закріплена кафедра  Партнерський факультет  Якість практичної підготовки студентів | Створити кафедру  Закріпити її за напрямом  Погодити співпрацю з факультетом  Покращення якості практичної підготовки студентів | Наявність договору / меморандуму про співпрацю із ЗВО.  Затвердження програм практик.  Визнання компанії як бази практики / стажування. | | Факультет | Академічна одиниця ЗВО, яка співпрацює з ІТ-компанією для передачі частини підготовки фахівців | Назва факультету  Декан  Кількість кафедр  Загальне річне навчальне навантаження  Перелік напрямів  Партнерська ІТ-компанія  Якість освітнього процесу | Затвердити список напрямів підготовки Визначити обсяг годин на рік, Розподілити річне навантаження  між кафедрами  Погодити співпрацю з ІТ-компанією  Контролювати якість освітнього процесу |  | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, навчальний план, напрям підготовки тощо |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один факультет співпрацює з однією ІТ-компанією** (модель 1:1). **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і викладач) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів.  На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента.  Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітній процес 3».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Faculty,** **ITCompany, Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.4.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.4.1.  **Алгоритм «Погодити співпрацю з ІТ-компанією».** Компанія ініціює запит (формат взаємодії, перелік напрямів, вигоди) про співпрацю,звертаючись до факультету. факультет реєструє звернення, оцінює пропозиції (перевіряє відповідність тематики компанії напряму підготовки факультету, практичну цінність, ресурси). Факультет підписує угоду/меморандум/наказ, якщо схвалена пропозиція, повідомлення про відмову, якщо пропозиція не схвалена.  **Алгоритм «Контролювати якість освітнього процесу».** Визначити перелік критеріїв якості (дисципліни, кваліфікація викладачі, метод. забезпечення, бажання студентів). Порівняти дані за критеріями з нормативами (можна задати випадковим чином). Формувати висновки, визначивши значення «відповідає критеріям», «має суттєві недоліки», «має несуттєві недоліки». Рішення записується у файл.  **Алгоритм «Покращити якість підготовки студентів».** Показник якості підготовки студентів (індекс KPI) розраховувати як *Індекс KPI = ((Факт – База) / (Норма – База)) \* 100%,* де *Факт* – фактичні результати роботи; *База* – допустиме мінімальне значення показника. Нижче базового рівня – відсутність результату; *Норма* – плановий рівень, те, що студент повинен виконувати обов'язково, нижче – студент не впорався зі своїми обов'язками; *Ціль* – значення, до якого потрібно прагнути, наднормативний показник, що дозволяє поліпшити результати.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**   1. Додати до класу **Faculty** вбудований (вкладений) клас Кафедра (**Department**) з додатковою сутністю згідно з табл. 4.2:   Таблиця 4.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Кафедра | Внутрішній підрозділ факультету, що реалізує навчальне навантаження у співпраці з факультетом | Назва кафедри  Спеціалізація  Завідувач кафедри  Кількість викладачів  Перелік дисциплін з обсягом годин  Річне навчальне навантаження  Партнерський факультет | Сформувати навчальний план (перелік дисциплін та їх години)  Призначити викладачів  Погодити напрям навчання | Призначити викладача на дисципліну можливо, якщо його спеціалізація відповідає профілю дисципліни.  Напрям навчання погоджений, якщо є відповідні кадри |   **Алгоритм «Сформувати навчальний план»**. Отримати від факультету загальне навчальне навантаження. Скласти список дисциплін з обсягом годин. Визначити послідовність їх викладання та розбити дисципліни по роках та семестрах. Критерії: в семестрі 30 кредитів (900 годин). В року 60 кредитів (1800 годин). Затвердити на факультеті.  Алгоритми інших дій студент розробляє самостійно.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Faculty**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод формування навчального плану, в інший файл – метод призначення викладачів на дисципліну  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **CompanyProject** (проект ІТ-компанії), включивши в нього функцію (на вибір студента) з варіанта 4 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує інституційно-адміністративна модель, в якій сутностями моделі є університет і факультет. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 5.1 подані ролі, атрибути та операції сутностей інституційно-адміністративна моделі.  Таблиця 5.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Університет | Головна академічна установа, що координує факультети, формує рейтинг і бюджет | Назва університету  Річний бюджет  Рейтинг (за Osvita.ua)  Кількість факультетів  Кількість студентів, наукових праць, міжнародних проєктів, ЗНО вступників | Розрахувати рейтинг університету за версією Osvita.ua  Розрахувати розмір річного фінансування за рейтингом  Додати факультет  Вилучити факультет | Рейтинг оновлюється один раз на рік.  рейтинг = f(середній бал ЗНО, кількість публікацій, кількість студентів, кількість факультетів \* середня ефективність)  Фінансування залежить від позиції у рейтингу: чим вище — тим більша частка бюджету. | | Факультет | Академічна одиниця ЗВО, яка співпрацює з ІТ-компанією для передачі частини підготовки фахівців | Назва факультету  Декан  Кількість кафедр  Кількість спеціальностей  Кількість студентів  Пов’язаний університет | Збільшити/зменшити кількість кафедр  Додати/видалити спеціальність  Збільшити/зменшити кількість студентів | Кількість кафедр = кількість спеціальностей.  На кожну спеціальність має припадати ≥ 200 студентів. Кількість спеціальностей залежить від кількості студентів | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один університет має один факультет** (модель 1:1). **Сервіс** не є учасником інституційно-адміністративної моделі, є лише інструментом, який використовують інші сутності (університет, факультет) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Інституційно-адміністративна модель».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **University,** **Faculty,** **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.5.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.5.1.  **Алгоритм розрахунку рейтингу університету за версією Osvita.ua** (https://osvita.ua/vnz/rating/25752/): консолідований бал університету визначається як сума балів за рейтингами «SCОPUS», «ТОП200 Україна», «Бал ЗНО на контракт». Враховуються показники: кількість наукових праць, студентів, міжнародних проєктів, ЗНО вступників по кожному факультету, що є ефективністю факультету .  *Рейтинг = f(середній бал ЗНО, кількість публікацій, кількість студентів, кількість факультетів \* середня ефективність усіх факультетів)*  Бали за різні номінації отримати в результаті генерації псевдовипадкових цілих чисел в діапазоні від 0 до 200 методами класу Random або взяти реальні.  **Алгоритм розрахунку розміру річного фінансування університет:** здійснюється відповідно до рейтингу університету та кількості студентів, враховуючи, що підготовка одного бюджетного студента в рік коштує від $2000 до $6000. Якщо університет у **топ-10,** він отримує **100% базового фінансування**. Якщо нижче — коефіцієнт пропорційно знижується.  *Фінансування = Базовий\_фонд \* Коефіцієнт\_рейтингу – (Адміністративні\_витрати + Кількість\_факультетів \* умовна\_витрата\_на\_структуру),*  де *Базовий\_фонд* - загальний обсяг коштів, який може бути виділений університету з держбюджету або спонсорських джерел; *Коефіцієнт\_рейтингу* - число (0.0 – 1.0 або 0% – 100%), що визначається на основі рейтингу університету; *Адміністративні\_витрати* (20 - 30% від бюджету університету) - постійні витрати на управлінський апарат, *Кількість\_факультетів* визначається в процесі роботи програми; умовна\_витрата\_на\_структуру - середня сума, яка витрачається на забезпечення роботи одного факультету.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Faculty** вбудований (вкладений) клас Стартап\_Інкубатор (**StartupIncubator**) факультету як додаткову сутність згідно з табл. 5.2:  Таблиця 5.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Стартап-Інкубатор | Інституція підтримки інноваційної активності студентів | Назва інкубатора,  Кількість менторів, Кількість активних проєктів,  Кількість залучених студентів,  Список стартапів  Обсяг інвестицій, які інкубатор отримає для впровадження своїх старапів | Додати студентський стартап  Призначити ментора  Оцінити результативність  Розрахувати рейтинг результативності  Вибрати найкращий стартап проєкт. |  |   **Алгоритм методу «Вибрати найкращий стартап проєкт».** Вважаємо, що є 10 проектів і 5 експертів, кожний з яких ставить проєкту оцінку. В якості оцінок проєкту виступають псевдовипадкові числа в діапазоні від 1 до 10. В результаті отримаємо матрицю вимірністю 5\*10 (кількість експертів (рядки)\*кількість проєктів (стовпці)). Для кожного проєкту знаходимо суму балів (сума елементів по стовпчиках), потім розраховуємо середнє арифметичний бал для проєкту, поділивши сумарний бал проєкту на кількість експертів. Сортуємо середні бали за зростанням. Проєкт, який має найменший середній бал, вважатимемо найкращим.  **Алгоритм методу «Рейтинг результативності студентів в Стартап-Інкубаторе»**. Для кожного студента визначаємо долю (значення від 0 до 1) його участі в кожному проєкті за допомогою генератору псевдовипадкових чисел. Отриману долю участі множимо на обсяг інвестицій, сортуємо отримані значення за спаданням. Студент з найвищим показником є найрезультативнішим.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **University**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод розрахунок рейтингу університету, в інший файл – метод розрахунку розміру річного фінансування університету.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **StartupProject** (Стартап проєкт), включивши в нього функцію (на вибір студента) з варіанта 5 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує освітнє середовище, учасниками якого є кафедра і ІТ-компанія. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 6.1 подані ролі, атрибути та операції сутностей освітнього середовища.  Таблиця 6.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Кафедра | Освітній підрозділ ЗВО | Назва Якість освітньої програми Кількість викладачів Кількість дисциплін Кількість креативних студентів Кількість реальних проєктів | Покращити якість освітньої програми Залучити фахівців з компанії Збільшити кількість креативних студентів | Студент бере участь у ≥ 1 реальному проєкті.  Студент створив хоча б 1 власну розробку / стартап.  Студент отримав ≥ 90 балів за інноваційні курси.  Студент бере участь у наукових дослідженнях, хакатонах, олімпіадах | | ІТ-компанія | Роботодавець, партнер кафедри, джерело практичного досвіду. | Назва Кількість співробітників Прибуток Прийнято випускників на роботу Реальний проєкт | Найняти випускників Збільшити прибуток |  | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра співпрацює з однією ІТ-компанією** (модель 1:1). **Сервіс** не є учасником освітнього середовища, є лише інструментом, який використовують інші сутності (кафедра, ІТ-компанія) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів.  На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента.  Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Інституційно-адміністративна модель».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Department,** **Company** та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.6.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.6.1.  **Алгоритм методу «Покращити якість освітньої програми».** Якщо якість освітньої програми не відповідає вимогам (наприклад, <100%), залучити фахівців з компанії, додати нові дисципліни, оновити силабуси дисциплін, збільшити кількість викладачів, залучити студентів до виконання проєктів, збільшити якість освітньої програми на кілька відсотків.  **Алгоритм методу «Збільшити кількість креативних студентів».** Визначити поточний рівень креативності студентів (порахувати кількість студентів, які беруть участь у реальних проєктах, створили власну розробку/стартап, отримали ≥ 90 балів за інноваційні курси, беруть участь у наукових дослідженнях, олімпіадах). Визначити проєкти для залучення студентів, задати кількість студентських нагород. Розрахувати нову кількість креативних студентів.  **Алгоритм методу «Збільшити прибуток компанії»**. Розрахувати прибуток: *Додатковий прибуток = Кількість співробітників \* коефіцієнт,* де *коефіцієнт* - це множник, який відображає внесок одного співробітника у прибуток компанії (наприклад, згенерувати число від 1000 до 20000, що означатиме, в середньому кількість грн в місяць прибутку від одного працівника). Додати додатковий прибуток до загального прибутку компанії. Зафіксувати нове значення прибутку (запис у файл).  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Company** вбудований (вкладений) клас ІТ акселератор (**ITAccelerator**) як додаткову сутність згідно з табл. 6.2:  Таблиця 6.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | ІТ акселератор | Сприяти розвитку стартапів, залучати студентів та випускників до підприємницької діяльності | Назва  Кількість проектів, участь в яких беруть викладачі і студенти;  Кількість студентів, які навчаються в акселераторі;  Обсяг інвестицій, які акселератор передає на кафедру.  Рівень інноваційності | Додати студентський стартап  Призначити ментора  Оцінити результативність стартапу  Розрахувати рейтинг студентів для конкурсу в акселератор;  Вибрати найкращий стартап проєкт. |   **Алгоритм методу «Вибрати найкращий стартап проєкт».** Вважаємо, що є 10 проектів і 5 експертів, кожний з яких ставить проєкту оцінку. В якості оцінок проєкту виступають псевдовипадкові числа в діапазоні від 1 до 10. В результаті отримаємо матрицю вимірністю 5\*10 (кількість експертів (рядки)\*кількість проєктів (стовпці)). Для кожного проєкту знаходимо суму балів (сума елементів по стовпчиках), потім розраховуємо середнє арифметичний бал для проєкту, поділивши сумарний бал проєкту на кількість експертів. Сортуємо середні бали за зростанням. Проєкт, який має найменший середній бал, вважатимемо найкращим.  **Алгоритм методу «Рейтинг результативності студентів в Стартап-Інкубаторі»**. Для кожного студента визначаємо долю (значення від 0 до 1) його участі в кожному проєкті за допомогою генератору псевдовипадкових чисел. Отриману долю участі множимо на обсяг інвестицій, сортуємо отримані значення за спаданням. Студент з найвищим показником є найрезультативнішим.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Department**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод покращення якості освітньої програми, в інший файл – метод розрахунку обсягу інвестицій на кафедру від стартапів:  *Інвестиції на кафедру = ∑ (Інвестиції від і-го стартапу× Коефіцієнт\_повернення),* де  *Інвестиції від і-го стартапу* = сумі коштів, залучених і-им стартапом, коефіцієнт повернення — це відсоток від прибутку компанії, який йде на підтримку освітньої бази кафедри.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **StartupProject** (Стартап проєкт), включивши в нього функцію (на вибір студента) з варіанта 6 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує освітнє середовище, учасниками якого є студент та його індивідуальна навчальна програма. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 7.1 подані ролі, атрибути та операції сутностей освітнього середовища.  Таблиця 7.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Студент | Учасник освітнього процесу, який навчається за програмою | ПІБ студента,  Список оцінок (0–10), Рейтинг (середній бал), Розмір стипендії, Закріплена навчальна програма | Розрахувати рейтинг  Розрахувати стипендію  Показати слабкі місця у навчанні | Слабкі місця: аналіз оцінок по предметах та виявлення предметів з оцінками < 60 балів (незадовільно)  Стипендія = базова (державна) стипендія, якщо рейтинг ≥ 80, 0, якщо рейтинг < 80, підвищена на кілька %, якщо рейтинг > 90. | | Навчальна програма | Формалізований освітній маршрут, який визначає зміст і обсяг навчання | Назва програми, Кількість кредитів, Список дисциплін, Перелік результатів навчання,  Рівень (бакалавр, магістр),  Якість (оцінка) | Визначити якість навчальної програми  Визначити складність навчальної програми  Оцінити відповідність навчальної програми ринку праці. |  | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент навчається тільки за однією навчальною програмою** (модель 1:1). **Сервіс** не є учасником освітнього середовища, є лише інструментом, який використовують інші сутності (студент, навчальна програма) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітня модель».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Student,** **Curriculum** та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.6.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.7.1.  **Алгоритм методу «Визначити якість навчальної програми».** Визначити критерії якості навчальної програми: актуальність дисциплін, практична спрямованість, методичне забезпечення, працевлаштування випускників, задоволеність студентів і задати їх кількісні оцінки. Визначити середню оцінку. Класифікувати результат, наприклад за 5-ти бальною шкалою: 4.5–5.0: Висока якість, 3.5–4.4: Добра, 2.5–3.4: Посередня, <2.5: Низька.  **Алгоритм методу «Визначити складність навчальної програми».** Визначити показники складності навчального матеріалу. наприклад, обсяг інформації, абстрактність, математизація, відсоток теорії у порівнянні з практикою, швидкість засвоєння, відсоток самостійної роботи у порівнянні з аудиторної, складність мислення, запам’ятовування. Для кожного показника задати бал (0..5). Обчислити середнє, дати інтерпретацію по діапазонам середнього балу (висока, низька, середня).  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Student** вбудований (вкладений) клас **WorkInCompany** (Робота в компанії)) як додаткову сутність згідно з табл. 7.2:  Таблиця 7.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Робота в ІТ-компанії | Професійна активність студента поза університетом | Назва компанії  Кількість проектів, які виконує студент;  Обсяг освітньої програми, яка збігається за змістом з роботою студента. | Розрахувати обсяг навчальної програми, який можна зарахувати студенту як результат його роботи в компанії в процесі виконання виробничих проектів;  Сформувати рекомендації щодо удосконалення індивідуальної навчального програми для наближення її до профілю роботи студента.  Оцінити вплив на професійну підготовку. |   **Алгоритм методу «Розрахувати обсяг навчальної програми для зарахування за роботу в компанії».** Задати кількість годин, які студент відпрацював на роботі, а також тематику роботи. знайти дисципліни з подібною тематикою. Перевірити результати навчання. Оцінити, наскільки % збігається робота в компанії з освітньою програмою. Не більше 25% загального обсягу ОП може бути зараховано через роботу.  **Алгоритм методу «Сформувати рекомендації щодо удосконалення індивідуальної навчальної програмиу»**. Розробити масив дисциплін за індивідуальною програмою. Сформувати профіль діяльності в компанії (посада, навички, інструменти, проєкти). Побудувати порівняльну таблицю індивідуальної програми та профілю студента. Визначити, що покривається навчанням, що використовується на роботі, але відсутнє в програмі. Згенерувати рекомендації: додати дисципліни, замінити дисципліни на нові, залучити до проєктів за реальними завданнями.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **WorkInCompany**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Розрахувати обсяг навчальної програми для зарахування за роботу в компанії», в інший файл – метод «Сформувати рекомендації щодо удосконалення індивідуальної навчального програми».  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **IndividWork** (індивідуальна робота), включивши в нього функцію (на вибір студента) з варіанта 7 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує банківське середовище, учасниками якого є клієнт і банкомат. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам банківського середовища операцій. В таблиці 8.1 подані ролі, атрибути та операції сутностей банківського середовища.  Таблиця 8.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Клієнт | Власник грошей, користувач банкомата | Ім’я, Прізвище, Номер картки, PIN-код,  Баланс | Перевірити PIN-код,  Зняти кошти,  Поповнити рахунок,  Перевірити баланс | Якщо баланс ≥ сума зняття, можна зняти кошти  Якщо перевірено PIN-код, можна перевіряти баланс | | Банкомат | Спеціальний пристрій, за допомогою якого можна керувати готівкою зі свого банківського рахунку | ID банкомата, Локація,  Доступні кошти (сума готівки, яка є в банкоматі),  Стан (працює / вимкнений / немає готівки) | Прийняти карту клієнта,  Видати готівку,  Показати залишок,  Заблокувати банківську картку через неправильний пін-код  Прийняти готівку | Якщо стан АТМ активний, можна працювати.  Якщо сума ≤ доступні кошти банкомата, можна видати готівку.  Карта клієнта автентифікована | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт користується одним банкоматом** (модель 1:1). **Сервіс** не є учасником банківського середовища, є лише інструментом, який використовують інші сутності (клієнт, банкомат) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів.  На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента.  Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Банківська модель».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи**,** **CClient,** **ATM** та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.6.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.7.1.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **ATM** вбудований (вкладений) клас **Bank** (Банк)) як додаткову сутність згідно з табл. 8.2:  Таблиця 8.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Банк | Установа, що володіє банкоматом та обслуговує клієнта | Назва банку,  Адреса  Ідентифікатори клієнта  Кількість грошей на рахунку клієнта | Оновити баланс клієнта  Збільшити, зменшити кількість грошей на рахунку клієнта;  Переказ грошей з картки клієнта, зокрема на оплата комунальних платежів з картки клієнта. |   **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **ATM**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Блокувати банківську картку», в інший файл – метод «Показати залишок на картці».  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **BankCard** (банківська картка), включивши в нього функцію (на вибір студента) з варіанта 8 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує освітнє середовище, учасниками якого є студент та його студентський парламент. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам освітнього середовища операцій. В таблиці 9.1 подані ролі, атрибути та операції сутностей освітнього середовища.  Таблиця 9.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Студент | Учасник освітнього процесу, який може бути членом студ. парламенту | ПІБ студента,  Курс,  Рейтинг (середній бал),  Стипендія  Статус участі в парламенті (бажаючий, кандидат, член)  Ідеї студента | Розрахувати рейтинг  Подати заявку на участь у парламенті  Взяти участь у голосуванні  Ініціювати пропозицію / ідею  Розрахувати розмір стипендії | Для участі в студ. парламенті: курс ≥ 2,  рейтинг ≥ 80.  Студент є членом студ. парламенту, тоді голосує, ініціює ідеї.  При рейтингу >90 стипендія підвищена | | Студентський парламент | Представницький орган самоврядування | Назва парламенту, Рівень (факультет / університет),  Склад (кількість членів) парламенту (назви департаментів),  Статус активності, Назва проєкту  Кількість проєктів  Обсяг стипенд. фонду | Прийняти нового члена  Взяти участь у голосуванні  Оприлюднити проєкт / ініціативу  Формувати рекомендації щодо заохочення студентів або їх відрахування Призначити  стипендії студентам. | Відрахування за неуспішність за значенням рейтингу, який < 60.  Стипендія за умови рейтингу >80/ | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент може бути членом одного студ. парламенту** (модель 1:1). **Сервіс** не є учасником освітнього середовища, є лише інструментом, який використовують інші сутності (студент, студ. парламент) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Освітня модель».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Student,** **Student**P**arliament** та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.6.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів відповідно до табл.9.1.  **Алгоритм методу «Подати заявку на участь у парламенті».** Якщо виконані умови (курс >=2, рейтинг >80, студент не є членом парламенту), тоді створити заявку (файл), передати об’єкту студ. парламенту, отримати підтвердження або відмову, оновити статус студента.  **Алгоритм методу «Взяти участь у голосуванні»**. Перевірити умови: студент є членом студ. парламенту, засідання активне, студент ще не голосував, вивести питання на голосування, прийняти відповідь (За / Проти / Утримався), зафіксувати відповідь (у файлі/на консолі).  **Алгоритм методу «Призначити стипендію студенту»**. Визначити рейтинги студентів, визначити обсяг стипендіального фонду, визначити студентів, яким слід призначити соціальну стипендію, та студентів, яким слід призначити стипендію за результати навчання (рейтинг не менше 80 балів). Для студентів-відмінників= стипендія підвищена.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **StudentParliament** вбудований (вкладений) клас **StudCouncilCampus** (Студентська Рада студмістечка ) як додаткову сутність згідно з табл. 9.2:  Таблиця 9.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Студентська рада студмістечка | Орган студентського самоврядування, відповідальний за побут і проживання | Назва ради,  Склад,  Адреса гуртожитку, Кількість мешканців, Плани благоустрою, Контактні особи | Розглядати звернення мешканців гуртожитку  Перевірити дотримання умов проживання студентів.  Організувати захід у гуртожитку  Формувати списки студентів на поселення у гуртожиток  Реагувати на порушення умов проживання (виселення, догана тощо). | Аналіз потреб студентів та можливостей кампусу.  Перевірка умов дотримання студентами  правил внутрішнього розпорядку, |   **Алгоритм методу «Перевірити дотримання умов проживання студентів».** Отримати скаргу мешканця гуртожитку, або звіт чергового по гуртожитку, або виявленний інцидент під час перевірки. Реєструвати інцидент (дата, місце, дані студента). Перевірити достовірність (опитати свідків, перегляд відео з камер спостереження, оцінка наслідків). Класифікувати порушення (суттєве, несуттєве, грубе). Визначити міру наказання (попередження, догана, виселення). Зафіксувати інцидент в БД (запис у файл). Видати повідомлення (вивести на консоль).  **Алгоритм методу «Формувати списки студентів на поселення у гуртожиток»**. Отримати та зареєструвати заяви (ПІБ, курс, соц. статус (пільговик, не пільговик), адреса постійного проживання). Визначити пріоритет (з інших міст, сироти, особи з інвалідністю, відмінники. наявність рекомендацій тощо). Врахувати наявну кількість місць. Створити та відсортувати список претендентів за пріоритетністю (запис у файл). Створити резервний список. Оприлюднити список (вивести на консоль).  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас StudentParliament, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Формувати списки студентів на поселення у гуртожиток», в інший файл – метод «Перевірити дотримання умов проживання студентів».  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **StudentCouncilPresident**(голова студ. ради), включивши в нього функцію (на вибір студента) з варіанта 9 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує середовище транспортної логістики, учасниками якої є водій та автомобіль. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам транспортної логістики операцій. В таблиці 10.1 подані ролі, атрибути та операції сутностей середовища транспортної логістики.  Таблиця 10.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Водій | Керує автомобілем, відповідає за його стан і рух | Ім’я, Прізвище, Номер посвідчення, Категорія прав, Стаж керування автомобілем,  Стан водія (допущений/ні)  Маршрут руху. | Вчинити порушення ПДР  Повідомити про ДТП або технічну несправність  Заявити про потребу заміни авто  Контроль фізичного стану водія | ПДР порушені, якщо маршрут не зареєстрований або стан водія поганий і він мав бути недопущений.  Виявлено проблему під час руху  Авто втратило придатність у результаті ДТП | | Автомобіль | Об'єкт керування і обліку | Марка,  Модель,  Рік випуску,  Номер авто,  Технічний стан (справний, несправний),  Статус (в ремонті, незареєстрований, списаний, знятий з реєстрації тощо),  Ступінь пошкодження | Пройти технічний огляд  Пройти реєстрацію  Поламатися внаслідок ДТП  Розрахувати вартість ремонту  Бути заміненим | Відповідно до графіку тех. огляду  Внесено до БД, має чинні документи  ДТП зафіксовано, ушкодження перевищують/ не перевищують критичний рівень  Вартість ремонту відповідно до переліку ушкоджень  Визнано неремонтопридатним або ремонт перевищує допустимі витрати | | Сервіс | Технічний посередник: забезпечує обмін, введення /виведення, збереження даних | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один водій керує одним автомобілем** (модель 1:1). **Сервіс** не є учасником транспортної логістики, є лише інструментом, який використовують інші сутності (водій та автомобіль) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Транспортна логістика».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Driver** (водій) та **Car** (автомобіль)та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.6.1).   2 Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 10.1.  **Алгоритм методу «Вчинити порушення ПДР».** Перевірити статус водія. Якщо статус неактивний, зафіксувати порушення і зняти водія з маршруту. Перевірити наявність зареєстрованого маршруту. Якщо маршрут незареєстрований, зафіксувати порушення та накласти на водія санкції. Зареєструвати тип порушення (перевищення швидкості, проїзд на червоне світло тощо). Додати запис до особової справи водія. Якщо порушення спричинило поломку авто, повідомити про несправність.  **Алгоритм методу «Повідомити про несправність».** Отримати номер авто. Зафіксувати несправність авто (записати до файлу). Оцінити серйозність поломки (можна ремонтувати, критична поломка, авто непридатне до ремонту тощо). створити запит на заміну авто (за потреби) або на ремонт. Поставити водія та авто у чергу на заміну або ремонт.  **Алгоритм методу «Розрахувати вартість ремонту»**. Визначити деталі, що пошкоджені. Визначити вартість кожної пошкодженої деталі та вартість ремонтних робіт.  *Вартість ремонту\_деталі = (Ціна деталі - Знижка) + (Години роботи × Ціна години);*  *Знижка = Базова ціна \* (Коефіцієнт\_амортизації );*  *Коефіцієнт\_амортизації = Рік\_амортизації × Норма\_зносу\_на\_рік*  *Рік\_амортизації* — кількість повних років з моменту випуску автомобіля.  *Норма\_зносу\_на\_рік* — стандартна втрата вартості авто за рік (%)  *Загальна сума = ∑ Вартість ремонту\_деталі*  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Car** вбудований (вкладений) клас **CarEngine** (двигун) як додаткову сутність згідно з табл. 10.2:  Таблиця 10.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Двигун | Центральний технічний компонент автомобіля | Тип,  Об’єм, Потужність, Стан,  Пробіг, Температура, Середні витрати пального | Розрахувати витрати пального на задану довжину маршруту, якість дороги, клімат.  Розрахувати тривалість маршруту з урахуванням заданої швидкості, запасу пального, параметрів двигуна.  Запустити двигун.  Перевірити температуру.  Вимкнути двигун. | Стан = справний, температура в нормі.  Якщо стан ≠ «справний» → завершити з повідомленням «неможливо запустити». Якщо температура > критичної → завершити, вивести попередження.  Якщо всі умови виконано → встановити стан: **«**працює**»**. |   **Алгоритм методу «Розрахувати витрати пального».**  *Фактична витрата = Базова витрата × Коеф. дороги × Коеф. клімату.*  *Витрати пального = (Фактична витрата / 100) × довжина маршруту.*  Тут *Базова витрата* - витрати пального на 100 км, яку визначає виробник як стандартна вимога (задати самостійно). *Коеф. дороги* - множник, який відображає вплив типу дорожнього покриття та умов руху на витрати пального, наприклад, автотраса - коеф.=1, міський режим - коеф. = 1.2, бездоріжжя - коеф.=1.5.  *Коеф. клімату* - множник, який враховує вплив зовнішньої температури та кліматичних умов на роботу двигуна й споживання пального, наприклад, нормальні умови - коеф. = 1, холод - коеф. = 1.1, спека - коеф. =1.2.  **Алгоритм методу «Розрахувати тривалість маршруту»**.  *макс\_дальність = (запас\_пального × 100) / середня\_витрата*  *дійсна\_відстань = min(довжина\_маршруту, макс\_дальність)*  *тривалість поїздки\_час = дійсна\_відстань / середня\_швидкість*  *середня\_витрата\_пального* - це кількість пального (в літрах), яку автомобіль витрачає на 100 км шляху в середньому за певний період або за умов реального руху.  *витрата\_пального*  = (заправлено літрів / пройдено км) × 100.  *середня\_швидкість -* усереднене значення швидкості руху автомобіля на певному маршруті з урахуванням зупинок, світлофорів, поворотів, заторів.  *середня\_швидкість* = загальна відстань (км) / загальний час у дорозі (год)  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Двигун**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Розрахувати витрати пального на задану довжину маршруту, якість дороги, клімат», в інший файл – метод. «Розрахувати тривалість маршруту з урахуванням заданої швидкості, запасу пального, параметрів двигуна»,  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Passenger** (пасажир), включивши в нього функцію (на вибір студента) з варіанта 10 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує середовище наукової роботи, сутностями якої є конференція і автор доповіді на конференції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам наукової роботи операцій. В таблиці 11.1 подані ролі, атрибути та операції сутностей наукового середовища.  Таблиця 11.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Конференція | Захід для представлення наукових доповідей | Назва,  Дата  проведення,  Місце,  Напрямки  конференції,  Гранична дата подачі,  Статус  (активна/архів) | Реєструвати учасників  Отримати доповідь  Рецензувати доповідь  Відхилити доповідь  Прийняти доповідь  Друкувати доповідь  Перевірити  відповідність теми напрямку конференції | Прийом відкрито  Тема доповіді відповідає напрямку  Доповідь рецензована  Доповідь прийнята, якщо рецензія позитивна  Доповідь відхилена, якщо рецензія негативна | | Автор  доповіді | Учасник конференції, який готує і подає наукову доповідь | Ім’я, Прізвище,  Науковий ступінь,  Тема доповіді,  Обсяг тексту,  Дата подачі  Статус (студент, аспірант викладач) | Подати доповідь  Отримати рецензію  Внести правки до тексту доповіді  Розрахувати вартість участі в конференції | Досягнутий дедлайн  Тема доповіді відповідає конференції  Форматування відповідає вимогам | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один автор подає доповідь на одну конференцію** (модель 1:1). **Сервіс** не є учасником наукової роботи, є лише інструментом, який використовують інші сутності (конференція та автор) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області «Наукова робота».  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Conference** (Конференція), **Author** (Автор)та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.11.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 11.1.  **Алгоритм методу «Отримати доповідь».** Перевірити статус конференції: якщо статус “не активна”, відхилити заявку, повідомити актора. Якщо дата вийшла за межі Deadline, відхилити подання доповіді. Перевірити відповідність теми доповіді напрямку конференції. Перевірити технічні вимоги (обсяг, формат, структура доповіді, шаблон). Якщо технічні умови виконано, зареєструвати автора і доповідь (записати дані у файл), надіслати підтвердження автору (виведення на консоль повідомлення).  **Алгоритм перевірки відповідності теми напрямкам конференції.** Сформувати масив (список) назв напрямків конференції. Визначити масив (список) ключових слів доповіді. Знайти співпадіння ключових слів доповіді зі словами з назв напрямків конференції. Можна використати попередньо створений словник ключових слів. Якщо збіги є, то відповідність встановлена, інакше автору надсилається повідомлення про відхилення його доповіді через невідповідність тематиці конференції.  **Алгоритм методу «Рецензувати доповідь».** Призначити рецензента. визначити результати рецензування (випадковим чином сформувати відповідь - прийняти без змін, прийняти з правками, відхилити). Записати рішення до файлу конференції. Повідомити автора - вивести на консоль повідомлення.  **Алгоритм методу «Розрахунок вартості участі в конференції».** Задати базову вартість участі (згенерувати випадкове число від 500 до 1000 грн.) в залежності від форми участі (очна, дистанційна). Задати знижки за статус автора (студент, аспірант, викладач, інші): *Знижка = Базова\_вартість × Коеф\_знижки. Коеф\_знижки =* 0 (викладач) - 0,5 (студент). Задати надбавки за додаткові опції (друкований збірник доповідей, сертифікат, переклад тощо). Задати надбавки за перевищення обсягу доповіді. Додати вартість проїзду для очної участі. Додати оплату проживання для очної участі.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Author** вбудований (вкладений) клас **Paper** (стаття) як додаткову сутність згідно з табл. 11.2:  Таблиця 11.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Стаття | Об’єкт наукової подачі, що оцінюється та публікується | Назва статті  Кількість сторінок. слів  Кількість посилань на релевантні джерела  Мова статті | Перевірити відповідність вимогам  Розрахувати вартість перекладу на англійську  Визначити рівень унікальності | вартість перекладу= кількість слів/сторінок \* ціна перекладу одного слова/ однієї сторінки |   **Алгоритм методу «Перевірити відповідність оформлення статті вимогам конференції».** У діалозі запитати про наявність розділів: мета, постановка проблеми, обґрунтування результатів, висновки, рекомендації, список посилань. Якщо вказані розділи присутні у статті, то вивести повідомлення про відповідність статті вимогам, інакше - повідомлення про невідповідність статті вимогам.  **Алгоритм методу «Визначити рівень унікальності статті».** Очистити текст від списку літератури, анотації, цитат (видалити з файлу). Розбити текст (файл) на блоки. Для кожного блоку знайти в Internet або інших базах статей збіги. Визначити % збігів. Якщо збігів <=20%, текст статті унікальний, інакше текст запозичений. Сформувати звіт з переліком фрагментів зі збігами.  *% унікальності = 100% – (% фрагментів зі збігами).*  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Paper**, подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Перевірити відповідність вимогам», в інший файл – метод « Визначити рівень унікальності».  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Subject** (предмет статті), включивши в нього функцію (на вибір студента) з варіанта 11 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує середовище практико-орієнтованого навчання, сутностями якої є студент і біржа практик. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам практико-орієнтованого навчання операцій. В таблиці 12.1 подані ролі, атрибути та операції сутностей практико-орієнтованого навчання.  Таблиця 12.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Студент | Користувач біржі, який шукає та проходить практику | ПІБ,  Факультет, Спеціальність,  Курс, Середній бал, Очікуваний тип практики (виробнича, переддипломна тощо) | Подати заявку на практику  Оцінити практику  Підтвердити завершення  Вибрати місце практики | Реєстрація на біржі завершена  Облікові дані актуальні | | Біржа практик | Платформа для розміщення та призначення місць практики | Назва біржі,  Перелік компаній, Перелік доступних практик,  Статус заявки,  Статус співпраці | Прийняти заявку студента  Призначити місце практики  Надіслати звітні документи  Показати місця практики (компаній) | Є вільні місця за профілем (спеціальність, факультет. курс) студента  Заявка перевірена | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент подає запит на одну біржу практики** (модель 1:1). **Сервіс** не є учасником практико-орієнтованого навчання, є лише інструментом, який використовують інші сутності (студент і біржа практик) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області практико-орієнтованого навчання.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Student** (Студент), **PracticeExchange** (Біржа практик та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.11.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 12.1.  **Алгоритм методу «Подати заявку на практику».** Отримати дані профілю студента: курс, спеціальність, середній бал. Перевірити дані. Якщо перевірка не пройдена → зупинити алгоритм. Показати студенту список доступних місць практики. Студент обирає місце (позицію). Перевірити відповідність позиції спеціальності, курсу. Якщо позиція недоступна → повернути повідомлення. Якщо доступна, створити заявку та зберегти дані: ID студента, ID позиції, дата подачі. Задати статус *"Очікує підтвердження".* Зберегти заявку.  **Алгоритм методу «Оцінити практику».** Отримати заявку студента. Перевірити статус заявки = *«Завершено».* Якщо ні — припинити дію. Показати форму оцінювання. Студент вводить числову оцінку (1–5), текстовий коментар (опціонально). Перевірити, чи оцінка в допустимому діапазоні? Якщо так, зберегти оцінку та коментар, змінити статус заявки на *«Оцінено»,* повернути повідомлення про успішну оцінку. Якщо ні — видати повідомлення про помилку введення.  **Алгоритм методу «Прийняти заявку студента».** Отримати заявку за ID. Перевірити статус заявки = *«Очікує підтвердження».* Якщо статус ≠ «Очікує», припинити обробку. Отримати обраний напрям/позицію практики. Перевірити, чи є вільні місця для обраної позиції та чи відповідає студент профілю позиції (спеціальність, курс)? Якщо умови виконано, тоді змінити статус заявки на *«Прийнята»,* зменшити кількість вільних місць. Якщо умови не виконано: змінити статус заявки на *«Відхилена»,* вказати причину відмови. Зберегти заявку.  **Алгоритм методу «Призначити місце практики».** Отримати заявку. Перевірити статус заявки = *«Прийнята».* Якщо ні — завершити роботу. Створити масив (список) компаній за напрямком заявки. Обрати компанію з вільною позицією. Для пошуку компанії застосовувати метод бінарного пошуку в масиві рядків, якими є назви компаній та їх спеціалізації. Закріпити студента за цією позицією. Змінити статус заявки на *«У процесі».* Зберегти призначення. Повідомити студента та компанію.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **PracticeExchange** вбудований (вкладений) клас **ІТCompany**, з якою біржа має договір про співпрацю, як додаткову сутність згідно з табл. 12.2:  Таблиця 12.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | ІТ-компанія | Організація-партнер, що надає місця для проходження практики | Назва компанії,  Галузь,  Кількість вільних місць,  Статус співпраці (Активна / Призупинена / Завершена | Створити пропозицію  Прийняти студента  Надати характеристику студенту  Завершити практику | Є активний договір з біржею  Позиція відкрита  Кількість місць > 0 |   **Алгоритм методу «Створити пропозицію практики».** Перевірити статус співпраці з біржею. Якщо статус ≠ *«Активна»*, завершити дію, повернути повідомлення. Компанія задає дані: назва позиції (наприклад, «Frontend Intern»), напрямок (галузь, ключові слова), опис обов’язків, вимоги до студента (курс, спеціальність, мін. середній бал), кількість місць, тривалість практики. Присвоїти новій пропозиції статус *«Відкрита»*. Зберегти пропозицію в БД (файл). Збільшити кількість активних позицій компанії. Повернути підтвердження успішного створення  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **PracticeExchange,** подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити методи сутності Біржа практики (табл. 12.1), в інший файл – методи класу ІТ-компанія (табл. 12.2).  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Rezume** (резюме), включивши в нього функцію (на вибір студента) з варіанта 12 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
|  | **Опис предметної області**. Існує середовище ріелторських послуг, сутностями якої є агенство нерухомості та клієнт. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам практико-орієнтованого навчання операцій. В таблиці 13.1 подані ролі, атрибути та операції сутностей в середовищі ріелторських послуг.  Таблиця 13.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Агентство нерухомості | Посередник, що підбирає, пропонує та супроводжує операції з нерухомістю | Назва агентства,  Список об’єктів,  Ліцензія,  Статус співпраці | Прийняти запит клієнта  Шукати об’єкт відповідно до запиту клієнта  Оцінити вартість нерухомості  Підготувати договір  Закрити угоду | Є активна ліцензія  Клієнт подав запит  Є релевантні об’єкти в базі | | Клієнт | Особа, яка звертається по послуги з оренди чи купівлі нерухомості | Ім’я, Прізвище,  Тип запиту (купівля/оренда), Бюджет,  Бажаний район,  Статус запиту (новий, активний, завершений) | Шукати агенство нерухомості  Подати запит  Переглянути пропозиції  Укласти/відхилити угоду | Подано коректний запит  Агентство обслуговує клієнта  Об’єкт доступний | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт подає запит в одне агенство нерухомості** (модель 1:1). **Сервіс** не є учасником практико-орієнтованого навчання, є лише інструментом, який використовують інші сутності (клієнт та агенство нерухомості) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області ріелторських послуг  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **RealEstateAgency** (Агентство нерухомості), **Client** (клієнт) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.13.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 13.1.  **Алгоритм методу «Прийняти запит клієнта».** Отримати запит (кількість кімнат, місця розташування будинку, вартість квартири, первинний-вторинний ринок, тип (елітне, бізнес-клас, економ-клас) тощо) зі статусом «Новий». Перевірити наявність релевантних об’єктів у базі. Якщо є відповідність, змінити статус запиту на «У процесі». Якщо немає — залишити статус без змін, надіслати повідомлення клієнту. Можна клієнту призначити персонального аналітика, який супроводжуватиме процес до укладання договору.  **Алгоритм методу «Оцінити вартість нерухомості.** Отримати заявку студента. Перевірити статус заявки = *«Завершено».* Якщо ні — припинити дію. Показати форму оцінювання. Студент вводить числову оцінку (1–5), текстовий коментар (опціонально). Перевірити, чи оцінка в допустимому діапазоні? Якщо так, зберегти оцінку та коментар, змінити статус заявки на *«Оцінено»,* повернути повідомлення про успішну оцінку. Якщо ні — видати повідомлення про помилку введення.  **Алгоритм методу «Розрахувати вартість нерухомості».**  *Вартість=Площа×БазоваЦінаЗаМ2×КоефіцієнтЛокації×КоефіцієнтТипу×КоефіцієнтРемонту+(Вартість×СумаКоефіцієнтівДодатковихФакторів)*  *КоефіцієнтЛокації -* від 2 до 0.5 (центр, престижний район, спальний, околиця, проблемний)? розраховується як частка СередняЦінаВРайоні​ / СередняЦінаПоМісту.  *КоефіцієнтТипуНерухомості = СередняЦінаЗаТип / СередняЦінаЖитловоїКвартири*  *КоефіцієнтРемонту = СередняЦінаЗаСтан / СередняЦінаНовобудови*  Додаткові фактори можна визначити як надбавку або знижку у відсотках до вартості:  *СумаКоефіцієнтівДодатковихФакторів = Наявність балкона+Наявність ліфта+Гарний Вид +Парковка або гараж - Перший або останній поверх - Шумний район.*  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **RealEstateAgency** вбудований (вкладений) клас **Rieltor** (ріелтор), який супроводжуватиме клієнта, як додаткову сутність згідно з табл. 13.2:  Таблиця 13.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Рієлтор | Представник агентства, який супроводжує клієнта протягом угоди | Назва компанії,  Ім’я, Прізвище, Спеціалізація,  Кількість угод,  Статус доступності - Доступний / Зайнятий | Прийняти запит  Супроводжувати клієнта  Оформити угоду | Рієлтор доступний, спеціалізація відповідає типу об’єкта |   **Алгоритм методу «Супроводжувати клієнта».** Отримати закріпленого клієнта: перевірити, чи клієнт прив’язаний до поточного рієлтора. якщо «ні» — завершити метод із повідомленням про помилку. Завантажити активний запит клієнта: перевірити статус запиту = «*У процесі обробки»* або «*Активний»*; якщо статус недопустимий — завершити. Отримати список релевантних об’єктів, відфільтрувавши об’єкти бази за параметрами запиту (типи запиту, район, бюджет тощо). Надіслати клієнту пропозиції для перегляду. Клієнт робить вибір об’єктів для перегляду. Якщо об’єкти не обрані — завершити метод. Імітувати перегляд об’єктів. Після перегляду отримати рішення «*Влаштовує* / *Не влаштовує*. Якщо клієнт обрав об’єкт, змінити статус запиту на «*Очікує укладення угоди».* Зберегти ID об’єкта як о*»*браний. Якщо відмовився, позначити об’єкт як переглянутий. Повторювати дії, доки не буде знайдено об’єкт або клієнт відмовився від подальшого пошуку.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **PraRealEstateAgency,** подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Шукати об’єкт відповідно до запиту клієнта*»*, в інший - метод «Оцінити вартість нерухомості*»*  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Apartment** (квартира), включивши в нього функцію (на вибір студента) з варіанта 13 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **14.** | **Опис предметної області**. Існує середовище Академічна мобільність, сутностями якої є студент та відділ академічної мобільності Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам практико-орієнтованого навчання операцій. В таблиці 14.1 подані ролі, атрибути та операції сутностей в середовищі академічної мобільності.  Таблиця 14.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Студент | Здобувач освіти, учасник академ. мобільності | ПІБ,  Освітній рівень,  Середній бал,  Наукові здобутки  Статус заявки | Розрахувати рейтинг студента,  Подати заявку,  Отримати результат розгляду заявки,  Підписати угоду | Обрана програма існує, мотиваційний лист подано,  Заявка розглянута, Заявка схвалена | | Відділ академічної мобільності | Організатор і координатор програм | Назва програми,  Мінімальний середній бал Кількість учасників,  Список (масив) ЗВО-партнерів | Розглянути заявку,  Шукати університет  Схвалити заявку,  Відхилити заявку,  Оформити документи, | Середній бал студента відповідає вимогам  Рейтинг студента відповідає вимогам  Немає місць або студент не відповідає вимогам, тоді заявка відхилена | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент подає заявку в один відділ на одну програму** (модель 1:1). **Сервіс** не є учасником академічної мобільності, є лише інструментом, який використовують інші сутності (студент і відділ академічної мобільності) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області академічної мобільності.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Student** (Студент), **AcademicMobility** (відділ академічної мобільності) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.14.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 14.1.  **Алгоритм методу «Розрахувати рейтинг студента».** Рейтинг студенту включає середній бал за 4 сесії, бали за наукову діяльність студента (кількість статей – 5 балів за кожну статтю, участь в конференціях і конкурсах – по 4 бали за кожну подію тощо) та бали за знання іноземної мови. Кількість оцінок в кожну сесію генерувати псевдовипадковими цілими числами в діапазоні від 60 до 100.  **Алгоритм методу «Подати заявку».** Перевірити, чи студент ще не подавав заявку. Якщо вже подавав, вивести повідомлення про неможливість повторної подачі. Якщо ні, створити заявку, вказавши програму академ. мобільності, додавши мотиваційний лист та розрахований рейтинг студента. Встановити статус заявки як «Очікує розгляду». Надіслати заявку до відділу академічної мобільності, вивести повідомлення «Заявка подана»  **Алгоритм методу «Розглянути заявку».** Перевірити, чи заявка ще не була розглянута. Перевірити рейтинг студента. Якщо нижче мінімального порогу, відхилити заявку. Перевірити, чи є вільні місця у програмі академ. мобільності. Якщо немає, відхилити заявку. Якщо всі умови виконані, схвалити заявку. Оновити статус заявки.  **Алгоритм методу «Шукати університет».** Отримати від студента його спеціальність, курс, рейтинг, бажану країну для мобільності. Відкрити список університетів, з якими підписані угоди про академічну мобільність. Фільтрувати університети за критеріями: споріднена спеціальність, бажана країна. Перевірити, чи є на поточний момент відкриті програми для обміну. Якщо знайдені університети → повернути список університетів. Якщо список порожній → вивести повідомлення: «На жаль, немає університетів, що відповідають вашим критеріям».  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **AcademicMobility** вбудований (вкладений) клас **InterProgram** (міжнародні програми), який супроводжуватиме клієнта, як додаткову сутність згідно з табл. 14.2:  Таблиця 14.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Міжнародна програма | Представлення програми для академічної мобільності між університетами | Назва програми  Спеціальність  Країна  Тривалість програми (місяці)  Університет-партнер | Перевірити наявність місць у програмі  Провести конкурс серед претендентів на участь у міжнародних програмах | Якщо місць на програму немає, конкурс не проводиться |   **Алгоритм методу «Провести конкурс».** Перевірити наявність доступних місць: якщо кількість місць = 0, припинити процес відбору та повідомити всіх претендентів про закриття набору. Сформувати список претендентів: перевірити, чи всі заявки відповідають вимогам, видалити ті, що не відповідають вимогам. Оцінити претендентів за критеріями: визначити вагу кожного критерію (наприклад, середній бал – 30%, наукова діяльність – 30%, рівень англ. мови – 20%, мотиваційний лист – 20%) та уточнити рейтинг студента:  *Рейтинг=(Середній бал×0.3)+(Оцінка за мову×0.3)+(наукова діяльність×0.3)+(рівень англ. мови×0.2)+(Оцінка мотиваційного листа×0.2)*  Відсортувати студентів за рейтингом у порядку спадання. Взяти з початку відсортованого списку топ-N студентів, де N – кількість доступних місць. Додати їх до списку відібраних студентів. Якщо кількість заявок перевищує N, сформувати список резервних студентів, які не пройшли основний конкурс. Претенденти, які увійшли до списку відібраних студентів, отримують статус «Прийнято». Ті, хто залишився у резервному списку, отримують статус «Резерв». Решта отримують статус «Відхилено». Надіслати результати всім учасникам конкурсу.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **AcademicMobility,** подавши його як частковий, поділивши клас на дві частини кожна в окремому файлі: в один файл включити метод «Шукати університет*»*, в інший - метод «Провести конкурс*»*  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **ResearchWork** (наукова робота), включивши в нього функцію (на вибір студента) з варіанта 14 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **15.** | **Опис предметної області**. Існує середовище Таксі-сервіс, сутностями якої є клієнт і таксіі Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам Таксі-сервіс операцій. В таблиці 15.1 подані ролі, атрибути та операції сутностей в середовищі Таксі-сервіс.  Таблиця 15.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Клієнт | Замовник послуги таксі | Ім'я,  Контактний номер, Місце посадки,  Місце призначення,  Спосіб оплати | Замовити таксі (вказати місце посадки, місце призначення)  Оплатити поїздку (сума, спосіб оплати)  Скасувати замовлення | є доступний автомобіль, клієнт підтвердив замовлення | | Таксі | Виконавець замовлення, перевізник | номер таксі,  статус (вільне/зайняте), тариф за км,  тариф за хвилину  вартість оплати  відстань | Прийняти замовлення (місце посадки, місце призначення).  Розрахувати вартість поїздки (відстань, тариф)  Визначити оптимальний маршрут поїздки | Авто вільне, відстань до клієнта в межах допустимого.  Тариф встановлений, відстань визначена. | | Сервіс | Технічний посередник: забезпечує обмін введення/виведення збереження даних | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт робить одне замовлення, отримує одне таксі** (модель 1:1). **Сервіс** не є учасником Таксі-сервіс є лише інструментом, який використовують інші сутності (клієнт і таксі) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області Таксі-сервіс.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Client** (Клієнт) та **Taxi** (таксі) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.15.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 15.1.  **Алгоритм методу «Замовити таксі».** Отримати від клієнта інформацію про місце посадки та місце призначення. Перевірити коректність введених даних. Якщо місце посадки або призначення не задано - вивести повідомлення про помилку. Передати дані про замовлення у систему таксі. Очікувати підтвердження прийняття замовлення. Якщо замовлення підтверджено - повідомити клієнта про деталі таксі (номер авто, ім'я водія, час прибуття). Якщо замовлення не підтверджено - запропонувати новий запит.  **Алгоритм методу «Оплатити поїздку».** Перевірити, чи поїздка завершена. Якщо поїздка ще триває - повідомити про неможливість оплати. Отримати значення суму для оплати. Вибрати спосіб оплати (готівка/карта). Якщо вибрана карта - перевірити доступність коштів. Якщо коштів недостатньо - вивести повідомлення про необхідність обрати інший спосіб оплати. Провести оплату. Підтвердити успішне завершення платежу та видати чек.  **Алгоритм методу «Розрахувати вартість поїздки».**  *Вартість=Базовий тариф+(Відстань×Тариф за км)+(Час у дорозі×Тариф за хвилину)+Надбавки*, де  *Базовий тариф* – фіксована сума за подачу авто. *Відстань* – кількість кілометри від початкової до кінцевої точки. *Тариф за км* – вартість 1 км поїздки, *Час у дорозі* – загальний час руху в хвилинах. *Тариф за хвилину* – додаткова вартість за час у дорозі. *Надбавки* – додаткові платежі (нічний тариф, години пік, погодні умови тощо).  **Алгоритм методу «Визначити оптимальний маршрут».** Отримати координати початкової та кінцевої точки маршруту. Надіслати запит до картографічного сервісу для отримання доступних маршрутів. Визначити оптимальний маршрут за критеріями: найменша відстань, найкоротший час у дорозі з урахуванням трафіку, мінімальна кількість світлофорів та зупинок. Якщо є затори або обмеження на маршруті, перерахувати альтернативний маршрут, оцінити додатковий час у дорозі. Вибрати маршрут з найкращими показниками. Передати маршрут водію для навігації.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **TaxiService** вбудований (вкладений) клас **Car** (автомобіль), на якому їхатиме клієнт, як додаткову сутність згідно з табл. 15.2:  Таблиця 15.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Автомобіль | Засіб пересування | Марка,  Модель,  Пробіг,  Витрата пального  Запас пального | Перевірити технічний стан, Визначити витрати пального,  Оцінити залишковий ресурс | автомобіль закріплений за таксі, автомобіль у справному стані |   **Алгоритм методу «Визначити витрати пального».** Отримати загальний пробіг автомобіля (взяти із одометра чи з бортового комп’ютера за наЯвністю). Визначити базову ставку витрат (наприклад, 0.1 грн за 1 км пробігу). Якщо пробіг менше 100 000 км, розрахувати витрати за формулою: *витрати = 0.1 \* пробіг.* Якщо пробіг перевищує 100 000 км, розрахувати за формулою: *витрати = 0.2 \* пробіг.* Зафіксувати значення витрат (вивести на консоль).  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Taxi**, подавши його як частковий, поділивши його на два файли: в один файл включити метод розрахунок вартості виконання замовлення, в інший файл – метод розрахунок оптимального маршруту руху.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Сity** (місто), включивши в нього функцію (на вибір студента) з варіанта 15 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **16.** | **Опис предметної області**. Існує середовище фінансових послуг, сутностями якої є клієнт і банк. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам середовище фінансових послуг операцій. В таблиці 16.1 подані ролі, атрибути та операції сутностей в предметній області фінансових послуг.  Таблиця 16.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Клієнт | Фізична або юридична особа, яка користується фінансовими послугами банку | ПІБ,  паспорт,  ІПН,  номер рахунку,  баланс рахунку, кредитний рейтинг | Відкрити рахунок  Закрити рахунок  Поповнити рахунок  Зняти кошти  Отримати кредит | Клієнт має надати необхідні документи | | Банк | Фінансова установа, що надає банківські послуги | Назва банку  перелік послуг, процентні ставки, кредитні ліміти | Створити рахунок клієнта  Відкрити депозит  Надати кредит  Обробити платіж  Нарахувати відсотки по кредиту/депозиту | Клієнт має відповідати вимогам банку  Клієнт має позитивний кредитний рейтинг  На рахунку є активний кредит | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу  Шлях до файлу  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт робить одне замовлення, отримує одне таксі** (модель 1:1). **Сервіс** не є учасником фінансових послуг, є лише інструментом, який використовують інші сутності (клієнт і банк) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області фінансових послуг..  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Client** (Клієнт), **Bank** (банк) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.16.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 16.1.  **Алгоритм методу «Поповнити рахунок / Зняти кошти».** Отримати номер рахунку та визначити суму для поповнення/зняття. Перевірити, чи існує рахунок. Перевірити, що сума на рахунку достатня для зняття.  Додати суму до поточного балансу рахунку/ Відняти суму зі стану рахунку. Оновити дані про баланс у базі банку. Підтвердити успішне поповнення рахунку / зняття коштів.  **Алгоритм методу «Надати кредит клієнту».** Отримати номер рахунку клієнта та суму кредиту. Перевірити кредитний рейтинг клієнта. Переконатися, що сума не перевищує встановлений кредитний ліміт. Додати суму кредиту до балансу клієнта. Оновити кредитну історію в базі банку. Підтвердити видачу кредиту.  **Алгоритм методу «Нарахувати відсотки по кредиту».** Отримати номер рахунку клієнта. Перевірити, чи на рахунку є активний кредит. Розрахувати відсотки за встановленою ставкою.  *Відсотки = Сума кредиту×Ставка×Кількість місяців​/12 -* за формулою простих відсотків  *Загальний борг = Сума кредиту×(1+Ставка / Кількість періодів)^(Кількість періодів) -* за формулою складних відсотків.  Відняти початкову суму кредиту, щоб отримати тільки нараховані відсотки. Додати отримані відсотків до загального боргу клієнта. Оновити інформацію про борг у базі банку. Повідомити клієнта про нараховані відсотки.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Bank** вбудований (вкладений) клас **Website** (сайт банку), для роботи з електронним банкінгом, як додаткову сутність згідно з табл. 16.2:  Таблиця 16.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Сайт банку | Онлайн-платформа для надання банківських послуг | URL  Сервіси  Кількість користувачів на день | Контроль залишків на поточних рахунках,  Оформлення депозиту,  Оплата комунальних послуг,  Переказ коштів на рахунки | Оформлення депозиту доступне тільки для клієнтів із рахунком  Оплата комунальних послуг можлива лише при достатньому балансі |   **Алгоритм методу «Оплата комунальних послуг».** Авторизація клієнта: отримати від клієнта логін і пароль, перевірити коректність введених даних, вивести повідомлення про помилку при невірному введенні. Вибір послуги для оплати: відобразити список доступних комунальних послуг (електроенергія, газ, вода тощо), отримати від клієнта вибір послуги, запросити реквізити для оплати (номер особового рахунку, суму). Перевірка балансу: отримати поточний баланс рахунку клієнта, якщо баланс менший за суму платежу – вивести повідомлення про недостатні кошти та припинити операцію. Формування та підтвердження платежу: відобразити клієнту деталі платежу (сума, отримувач, комісія), запросити підтвердження оплати, якщо підтвердження не отримано – скасувати операцію. Виконання платежу: відняти суму платежу (з урахуванням комісії) з рахунку клієнта, згенерувати квитанцію платежу. Надіслати клієнту повідомлення про успішну оплату. Оновити загальну звітність банку. Вивести повідомлення про успішне завершення оплати.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Bank**, подавши його як частковий, поділивши на два файли: в один файл включити метод видачі кредитів під відсотки, в інший файл – метод нарахування відсотків за кредитами.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас Національний банк України, включивши в нього функцію (на вибір студента) з варіанта 16 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **17.** | **Опис предметної області**. Існує середовище медичних послуг, сутностями якої є пацієнт і лікар Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам середовища медичних послуг операцій. В таблиці 17.1 подані ролі, атрибути та операції сутностей у предметній області медичних послуг.  Таблиця 17.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Пацієнт | Фізична особа, яка отримує медичну допомогу | ПІБ,  дата народження, медична картка,  історія хвороби,  поточні скарги | Записатися на прийом,  Отримати діагноз, Отримати призначення лікування | Запис на прийом можливий, якщо є вільний час у лікаря. Діагноз встановлюється після огляду лікарем. Призначення лікування можливе тільки після встановлення діагнозу. | | Лікар | Фізична особа, яка надає медичні послуги | ПІБ,  спеціалізація,  стаж роботи,  графік прийому,  список пацієнтів,  доступ до медичних карт | Провести прийом пацієнта, Встановити діагноз, Призначити лікування | Прийом можливий, якщо пацієнт записаний.  Діагноз встановлюється після обстеження пацієнта. Лікування призначається після встановлення діагнозу. | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один пацієнт, один лікар** (модель 1:1). **Сервіс** не є учасником медичних послуг, є лише інструментом, який використовують інші сутності (пацієнт і лікар) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області медичних послуг.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Patient** (пацієнт) **Doctor** (Лікар) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.17.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 17.1.  **Алгоритм методу «Записатися на прийом».** Пацієнт обирає спеціалізацію лікаря. Відображається список доступних лікарів за спеціалізацією. Пацієнт обирає лікаря зі списку. Відображається розклад прийому лікаря. Пацієнт обирає зручну дату та час прийому. Система перевіряє наявність вільного часу. Якщо час доступний, запис підтверджується. Якщо час зайнятий, пацієнту пропонуються інші варіанти. Пацієнт отримує підтвердження запису.  **Алгоритм методу «Отримати діагноз».** Пацієнт приходить на прийом до лікаря. Лікар проводить огляд та опитування пацієнта. Якщо необхідно, лікар призначає додаткові обстеження. Пацієнт проходить необхідні обстеження (аналізи, рентген тощо). Лікар аналізує результати обстежень. Лікар встановлює діагноз. Пацієнт отримує інформацію про свій діагноз.  **Алгоритм методу «Провести прийом пацієнта».** Лікар переглядає список записаних пацієнтів.Лікар викликає наступного пацієнта в кабінет. Лікар проводить опитування пацієнта щодо скарг. Лікар вносить інформацію про скарги та огляд у медичну картку. Якщо потрібно, лікар призначає додаткові аналізи та обстеження. Лікар завершує прийом пацієнта.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **Patient** вбудований (вкладений) клас **MedicalCard** (медична картка**)**, як додаткову сутність згідно з табл. 17.2:  Таблиця 17.2. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Медична картка | Архів медичних даних пацієнта | Номер картки,  дані пацієнта, історія хвороб, призначення лікування | Оновити медичну історію  Додати результати аналізів  Переглянути історію хвороб | Є нові діагнози або результати обстежень  Пацієнт пройшов обстеження |   **Алгоритм методу «Додати результати аналізів».** Отримати медичну картку пацієнта за номером. Перевірити, чи існує картка. Якщо ні, вивести повідомлення про помилку та завершити виконання. Переконатися, що аналізи проводилися (перевірити наявність відповідного направлення або запиту). Додати результати аналізів до медичної картки. Зберегти зміни в базі даних. Підтвердити успішне оновлення.  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **Doctor**, подавши його як частковий, поділивши його на два файли: в один файл включити метод встановлення діагнозу хвороби, в інший файл – метод призначення лікування.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Illness** (хвороба), включивши в нього функцію (на вибір студента) з варіанта 17 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **18.** | **Опис предметної області**. Існує середовище мережі користувачів, сутностями якої є соціальна мережа та користувач Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам мережі користувачів операцій. В таблиці 18.1 подані ролі, атрибути та операції сутностей у предметній області медичних послуг.  Таблиця 18.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Соціальна мережа | Платформа для взаємодії користувачів | Назва мережі  Кількість користувачів  Спрямованість мережі (для науковців, для розваг тощо) | Додати користувача до мережі  Видалити користувача з мережі  Публікувати контент  Створити групи в мережі  Заблокувати користувача  Шукати користувачів | Користувач має бути зареєстрованим для видалення.  Мережа повинна бути активною для створення групи. | | Користувач | Фізична особа, яка комунікує через мережу | Ім’я  e-mail  Пароль  Статус користувача (активний/неактивний)  Кількість друзів | Зареєструвати користувача  Оновити профіль  Додати/видалити контент до мережі  Додати/видалити друзів  Ініціація пошуку користувачів |  | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками мережі користувачів: **один користувач, одна мережа** (модель 1:1). **Сервіс** не є учасником мережі користувачів, є лише інструментом, який використовують інші сутності (користувач і соціальна мережа) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області користування соціальними мережами.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **SocialNetwork** (Соціальна мережа), **User** (Користувач) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.18.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 18.1.  **Алгоритм методу «Додати контент до мережі».** Перевірити, чи має користувач право додавати контент, якщо ні — зупинити виконання і вивести повідомлення про відмову, якщо так — продовжити. Запитати користувача тип контенту (текст, фото, відео тощо). Ввести контент (текст, фото, відео або інші дані). Перевірити контент на відповідність політикам мережі (правилам, цензурі, тощо), якщо контент порушує правила — повідомити користувача та не додавати контент, інакше — продовжити. Додати контент до відповідної категорії: якщо це особистий пост — додати до стрічки новин користувача, якщо це публікація у групі — додати до обраної групи. Повідомити користувача про успішне додавання контенту.  **Алгоритм методу «Створити групу в мережі».** Перевірити, чи має користувач права на створення груп. Якщо користувач не має прав — зупинити виконання і повернути повідомлення про відмову. Запитати у користувача назву групи. Перевірити, чи не існує вже групи з такою назвою, якщо група з такою назвою вже існує — повідомити користувача і зупинити виконання. Запитати у користувача додаткові налаштування для групи: тип групи (відкрита чи закрита), опис групи, кількість адміністраторів або учасників. Створити групу в базі даних: призначити адміністратора (користувача, який створив групу), додати початкових учасників (якщо є). Повідомити користувача про успішне створення групи.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **SocialNetwork** вбудований (вкладений) клас **Friends** (друзі), як додаткову сутність згідно з табл. 18.2:  Таблиця 18.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Друзі | Користувачі, які можуть обмінюватися контентом | FirstName та LastName, e-mail друга  Статус дружби | Надіслати запит на дружбу  Підтвердити запит на дружбу  Відправити повідомлення другу  Заблокувати небажаного “друга” |   **Алгоритм методу «Надіслати запит на дружбу».** Перевірити, чи ініціатор дружби і отримувач запиту ще не є друзями (статус дружби не підтверджений). Якщо не є друзями, створити запис запиту на дружбу між ініціатором та отримувачем зі статусом "надіслано". Показати ініціатору повідомлення "Запит на дружбу надіслано". Показати отримувачу запит на дружбу зі статусом "надіслано".  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **SocialNetwork**, подавши його як частковий, поділивши його на два файли: в один файл включити метод створити групу, в інший файл – метод додати контент  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Internet**, включивши в нього функцію (на вибір студента) з варіанта 18 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **19.** | **Опис предметної області**. Існує транспортна інфраструктура міста, сутностями якої є само місто і транспорт. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам транспортної інфраструктури операцій. В таблиці 19.1 подані ролі, атрибути та операції сутностей у предметній області медичних послуг.  Таблиця 19.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Місто | Організує транспортну систему, контролює її ефективність | Назва міста  Кількість жителів  Площа міста  Кількість транспортних одиниць | Розрахувати щільність транспорту  Оцінити навантаження на інфраструктуру | Площа > 0  Кількість транспорту > 0 | | Транспорт | Забезпечує перевезення в межах міста | Тип транспорту (автобус, трамвай, метро)  Максимальна пасажиромісткість  Кількість рейсів на день | Розрахувати перевізну здатність  Оцінити ефективність маршруту | Рейси > 0  Пасажиромісткість > 0 | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками транспортної інфраструктури один до одного: **одне місто, один транпорт** (модель 1:1). **Сервіс** не є учасником транспортної інфраструктури, є лише інструментом, який використовують інші сутності (місто і транспорт) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області транспортна інфраструктура.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **City** ( місто), **Тransport** (транспорт) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.19.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 19.1.  Таблиця 19.2. Розрахункові формули для алгоритмів транспортної інфраструктури   |  |  |  |  | | --- | --- | --- | --- | | № | Показник | Формула | Пояснення | | 1 | Пасажиромісткість ТЗ | Пасажиромісткість = Сидячі місця+ Стоячі місця | Кількість пасажирів, яких може вмістити один транспортний засіб за один рейс | | 2 | Перевізна здатність ТЗ (за день) | Перевізна\_здатність = Пасажиромісткість × Кількість\_рейсів\_на\_день | Скільки пасажирів перевозить один ТЗ за день | | 3 | Потреба в перевезеннях | Потреба = Кількість\_жителів × Частка\_користувачів\_транспорту | Скільки людей користуються громад. транспортом щодня | | 4 | Кількість транспорту (фактична) | Кількість\_ТЗ = Потреба / Перевізна\_здатність | Скільки транспорту потрібно для фактичного перевезення пасажирів | | 5 | Кількість транспорту (за нормативом) | Кількість\_ТЗ = Кількість\_жителів / Норматив\_пасажирів\_на\_1\_ТЗ | Кількість ТЗ, що відповідає плановим стандартам обслуговування населення | | 6 | Щільність транспорту в місті | Щільність = Кількість\_ТЗ / Площа\_міста | Скільки одиниць транспорту припадає на 1 км² площі міста | | 7 | Навантаження на одну транспортну одиницю | Навантаження = Кількість\_жителів / Кількість\_ТЗ | Скільки жителів обслуговує одна одиниця транспорту |   6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **City** вбудований (вкладений) клас **Infrastructure**  (інфраструктура), як додаткову сутність згідно з табл. 19.3:  Таблиця 19.3. Додаткові сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умови виконання дій** | | Інфраструктура | Структура маршрутів і їхня доступність для міських районів | Кількість маршрутів,  Середня довжина маршруту,  Рівень покриття міста - % міської території, яку обслуговує транспорт | Розрахувати щільність покриття  Оцінити ефективність мережі | Щільність = Кількість\_маршрутів / Площа\_міста  Ефективність = Рівень\_покриття / Середня\_довжина\_маршруту |   **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **City**, подавши його як частковий, поділивши його на два файли: в один файл включити методи класу **Infrastructure**, в інший файл – методи класу **City**.  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **TransporStop** (зпинка транспорту) включивши в нього функцію (на вибір студента) з варіанта 19 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |
| **20.** | **Опис предметної області**. Існує цифрова платформа для молоді, сутностями якої є молодь і цифрова платформа. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності Сервіс для реалізації не притаманних участникам цифрової платформи операцій. В таблиці 20.1 подані ролі, атрибути та операції сутностей у предметній області медичних послуг.  Таблиця 20.1. Основні сутності предметної області   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | **Умова виконання дії** | | Молодь | Учасник цифрового ринку, що створює кар’єру через платформу | Ім’я,  Вік,  Навички,  Рейтинг,  Кількість проєктів  Профіль | Додати навичку  Оцінити досвід (виконання проєкту)  Розрахувати рейтинг | Є хоча б один завершений проєкт або фідбек | | Цифрова платформа | Середовище, яке надає інструменти для навчання і роботи | Назва платформи,  Кількість учасників,  Доступні проєкти,  Назва курсу | Призначити проєкт  Згенерувати звіт про активність  Алгоритм підбору  Порекомендувати курс | Є активний профіль молодої людини | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення даних, зберігає дані | Формат виводу Шлях до файлу Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу | У файл потрібно записати значення атрибутів сутностей, все, що буде виведено на консоль, усі розрахункові значення |   Здійснити **об’єктно-орієнтований аналіз** і визначити класи, поля і методи. Спрощена взаємодія між учасниками цифрової кар’єри: **один учасник цифрового ринку, одна цифрова платформа** (модель 1:1). **Сервіс** не є учасником цифрового кар’єрного ринку, є лише інструментом, який використовують інші сутності (молодь і цифрова платформа) для реалізації своїх дій.  **Розробити таблицю зв’язків між сутностями** за зразком:   |  |  |  |  | | --- | --- | --- | --- | | Сутність 1 | Сутність 2 | Тип зв’язку (1:1) | Опис зв’язку між сутностями |   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування** і побудувати діаграму класів. На основі діаграми класів розробити **код класів** мовою C# для обробки даних про кафедру і студента. Розробити **сценарій роботи програми** і побудувати Use Case діаграму.  На основі Use Case діаграми написати код для реалізації сценарію роботи програми у функції Main() програмного моделювання предметної області “Цифрова кар’єра молоді”.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Створити класи **Youth** (молодь), **DigitalPlatform** (цифрова платформа) та **Service**. Кожен клас має бути створений в окремому файлі командою *Project → Add class.* Визначити в класах закриті поля, що відповідають атрибутам предметної област (табл.20.1).   2. Визначити в класах конструктор за замовчуванням (без параметрів для ініціалізації полів класів нульовими та пустими (для типу *string*) значеннями).  3. Визначити в класах конструктори з параметрами для ініціалізації полів класів початковими значеннями та конструктор копії для створення нових об'єктів як копії існуючого об'єкта.  4. Визначити в класах відкриті властивості (get, set) для доступу до закритих полів та зміни значень ініціалізованих в конструкторах полів класів.  5. Визначити в класах відкриті методи для реалізації поведінки об'єктів через зміну їх станів (полів) відповідно до табл. 20.1.  **Алгоритм методу «Розрахувати рейтинг (на основі виконаних проєктів)».** Отримати список усіх проєктів, у яких брала участь особа. Відфільтрувати лише завершені проєкти з оцінкою. Якщо таких проєктів немає — встановити рейтинг = 0. Для кожного проєкту взяти значення Оцінка (наприклад, від 1 до 10 або у %), визначити коефіцієнт складності проєкту (за потреби). Обчислити середню оцінку:  *рейтинг = (сума оцінок \* коефіцієнти складності) / кількість проєктів*  Зберегти рейтинг у профіль користувача. Оновити статус особи (наприклад: "Новачок", "Стабільний", "Топ").  **Алгоритм методу «Призначити проєкт».** Отримати активний профіль молодої особи. Перевірити, чи не виконує вона інший проєкт у цей момент. Якщо так — призначення неможливе (вийти з алгоритму). Зчитати список навичок користувача, поточний рейтинг. Завантажити базу доступних проєктів зі статусом *«вільний»*. Для кожного проєкту перевірити відповідність за сферою/тематикою, перевірити мінімальний рейтинг. Відсортувати відповідні проєкти за релевантністю (співпадіння навичок, рівень складності). Вибрати найбільш відповідний проєкт. Призначити користувача як виконавця. Оновити статус проєкту → *«У роботі».* Повідомити учасника про призначення.  6. У функції Main() реалізувати сценарій роботи програми відповідно до Use Case діаграми.  **Версія 2 (Версія 1 + п.7) в окремому проєкті**  7. Додати до класу **DigitalPlatform** вбудований (вкладений) клас **Project** (проєкт), як додаткову сутність згідно з табл. 20.2:  Таблиця 20.2. Додаткові сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Проєкт | Практичне завдання, яке виконує молодь, отримуючи досвід і оцінку | Назва,  Сфера,  Складність,  Статус виконання,  Оцінка | Призначити виконавця  Змінити статус  Оцінити результат |   **Алгоритм методу «Призначити виконавця».** Отримати ID кандидата (молодої особи), якого призначають виконавцем. Перевірити статус проєкту: якщо статус ≠ "вільний" або "очікує виконавця", завершити метод з повідомленням: *"Проєкт недоступний для призначення"*. Перевірити, чи користувач не зайнятий іншим активним проєктом: якщо так, завершити: *"Користувач не може бути призначений на кілька проєктів одночасно"*. Призначити виконавця: створити зв’язок між проєктом і користувачем (наприклад, запис у таблиці виконання\_проєкту), записати дату призначення. Оновити статус проєкту: встановити статус = "У роботі", зафіксувати ID\_виконавця у проєкті. Ініціювати логіку старту: встановити дедлайн, створити або активувати трекер виконання (прогрес виконавця). Надіслати повідомлення виконавцю: підтвердження призначення, посилання на бриф (інформація про проєкт), вимоги, дедлайн, контакт координатора. Вивести повідомлення про призначення  **Версія 3 (Версія 2 + п.8) в окремому проєкті.**  8. Модифікувати клас **DigitalPlatform** , подавши його як частковий, поділивши його на два файли: в один файл включити метод Призначити проєкт, в інший файл – метод Порекомендувати курс  **Версія 4 (Версія 3 + п.9) в окремому проєкті.**  9. Додати до проекту новий статичний клас **Mentor** (ментор, координатор), включивши в нього функцію (на вибір студента) з варіанта 20 лабораторної роботи 2, зробивши її статичною.  **Алгоритми методів, які не описані в завданні, студент розробляє самостійно.** |

### Література

1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#. КНУ ім. Тараса Шевченка
2. C# - творчість програмування. Том 1. Об’єктно-орієнтоване програмування: підручник// Під ред. Бичкова О.С. Волин. Обереги, 2024. – 292 с.
3. The C# Coding Standard. Access mode: <https://github.com/hassanhabib/CSharpCodingStandard>
4. C# Coding Standards and Best Practices. Access mode: <https://www.dofactory.com/csharp-coding-standards>
5. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І. В., Марущак П. О. – Тернопіль: ФОП Паляниця В. А., 2020 – 320 с. Режим доступу: <https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf>

## Лабораторна робота №4 Успадкування класів в мові програмування C#: конструктори базових та похідних класів, інтерфейси, абстрактні класи

### Рейтинг лабораторної роботи №4

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 6 завдань | 0,5\*9 = 4,5 | 25 Березня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 5,5 |  |

### Мета роботи

1. Навчитися повторному використанню коду через спадкування класів
2. Навчитися створювати конструктори базових та похідних класів
3. Побудувати ієрархію класів відповідно до варіанту
4. Навчитися розробляти оголошення та реалізацію інтерфейсів, використовувати стандартні інтерфейси.
5. Навчитися працювати з абстрактними класами

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекцію та методичні вказівки до цієї лабораторної роботи.

2. Виконайте і проаналізуйте всі наведені в лекції приклади.

3. Зверніть увагу, що в С# заборонено множинне успадкування класів, тобто кожний похідний клас може мати тільки одного предка (базовий клас).

4. Якщо в базовому класі є конструктор з параметрами, то для його використання в похідному класі необхідно використовувати конструкцію *...(...):base(...){...}*, яка розташовується в оголошенні конструктора похідного класу між заголовком конструктора і тілом.

5. Для поглибленого вивчення цього матеріалу прочитайте розділи 3, 4 книги [2] та розділ 11 книги [3].

### Порядок виконання роботи

1. Створити директорію Lab4, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді одного консольного проекту з окремими файлами для кожного з класів з використанням меню для демонстрації роботи кожного завдання варіанту.

3. Для кожного класу передбачити окремий файл.

### Приклади виконання завдань

Спадкування — один з трьох фундаментальних принципів об'єктно-орієнтованого програмування, оскільки саме завдяки йому можливо створення ієрархічних класифікацій. Використовуючи спадкування, можна створити загальний клас, який визначає характеристики, властиві безлічі пов'язаних елементів. Цей клас потім може бути успадкований іншими, вузькоспеціалізованими класами з додаванням до кожного з них своїх унікальних особливостей. В мові С# клас, який успадковується, називається базовим. Клас, який успадковує базовий клас, називається похідним. Отже, похідний клас — це спеціалізована версія базового класу. У похідний клас, що успадковує всі змінні, методи, властивості, оператори і індексатори, визначені в базовому класі, можуть бути додані унікальні елементи.

#### Оголошення базових і похідних класів

Нехай є клас ***Person***, який описує людину;

class Person

{

private string name;

public string Name

{

get { return name; }

set { name = value; }

}

public void Display()

{

Console.WriteLine(name);

}

}

Але раптом нам став потрібен клас, що описує співробітника підприємства, наприклад, клас ***Employee***. Оскільки цей клас буде реалізовувати той самий функціонал, що і клас ***Person***, так як співробітник – це також і людина, то було б раціонально зробити клас ***Employee*** похідним (або спадкоємцем, або підкласом) від класу ***Person***, який, в свою чергу, називається базовим класом або батьком (або суперкласом). Для класу ***Employee*** базовим є ***Person***, і тому клас ***Employee*** успадковує всі ті самі властивості, методи, поля, які є в класі ***Person***. Єдине, що не передається при спадкуванні, це конструктори базового класу.

class Employee : Person

{

public void Display()

{

Console.WriteLine(\_name);

}

}

Таким чином, спадкування реалізує ставлення ***is-a*** (є), об'єкт класу ***Employee*** також є об'єктом класу ***Person***:

static void Main(string[] args)

{

Person p = new Person { Name = "Nikita" };

p.Display();

p = new Employee { Name = "Kolya" };

p.Display();

Console.Read();

}

За замовчуванням усі класи успадковуються від базового класу ***Object***, навіть якщо ми явно не встановлюємо успадкування. Тому вище означені класи ***Person***і ***Employee***крім своїх власних методів, також матимуть і методи класу *O****bject: ToString(), Equals(), GetHashCode() і GetType)****.*

Усі класи за замовчуванням можуть успадковуватися. Однак є ряд обмежень:

* не підтримується множинне спадкування, клас може успадковуватися тільки від одного класу;
* при створенні похідного класу треба враховувати тип доступу до базового класу. Тип доступу до похідному класу повинен бути таким самим, як і у базового класу, або більш строгим. Тобто, якщо базовий клас має тип доступу **internal**, то похідний клас може мати тип доступу **internal** або **private**, але **не public;**
* однак слід також враховувати, що якщо базовий і похідний клас знаходяться в різних збірках (проектах), то в цьому випадку похідний клас може успадковувати тільки від класу, який має модифікатор **public;**
* якщо клас оголошений з модифікатором **sealed**, то від цього класу не можна успадковувати і створювати похідні класи.

#### 2. Доступ до членів базового класу з класу-спадкоємця

Похідний клас може мати доступ тільки до тих членів базового класу, які визначені з модифікаторами ***private*** ***protected*** (якщо базовий і похідний клас знаходяться в одній збірці), ***public***, ***internal*** (якщо базовий і похідний клас знаходяться в одній збірці), ***protected*** і ***protected*** ***internal***.

class Employee : Person

{

public void Display()

{

Console.WriteLine(Name);

}

}

#### 3. Конструктори, ключове слово base

Тепер додаємо в класи конструктори. Клас ***Person*** має конструктор, який встановлює властивість ***Name***. Оскільки клас ***Employee*** успадковує і встановлює ту саму властивість ***Name***, логічно було б не писати по сто разів код установки, а якось викликати відповідний код класу ***Person***. До того ж властивостей, які треба встановити в конструкторі базового класу, і параметрів може бути набагато більше. За допомогою ключового слова ***base*** можна звернутися до базового класу. У нашому випадку в конструкторі класу ***Employee*** нам треба встановити ім'я і компанію. Але ім'я ми передаємо на установку в конструктор базового класу, тобто в конструктор класу ***Person***, за допомогою виразу ***base*** *(****name****)*.

class Person

{

public string Name { get; set; }

public Person(string str) //конструктор базового класу

{

Name = str;

}

public void Display()

{

Console.WriteLine(Name);

}

}

class Employee : Person

{

public string Company { get; set; }

//конструктор похідного класу

public Employee(string str, string company): base(str)

{

Company = company;

}

}

static void Main(string[] args)

{

Person p = new Person("Nikita");

p.Display();

Employee emp = new Employee("Kolya", "Microsoft");

emp.Display();

Console.Read();

}

Конструктори не передаються похідному класу при спадкуванні. І якщо в базовому класі не визначений конструктор за замовчуванням без параметрів, а тільки конструктори з параметрами (як у випадку з базовим класом ***Person***), то в похідному класі ми обов'язково повинні викликати один з цих конструкторів через ключове слово ***base***.

#### 4. Порядок виклику конструкторів

При виклику конструктора класу спочатку відпрацьовують конструктори базових класів і тільки потім конструктори похідних. Наприклад, візьмемо такі класи:

class Person

{

string name;

int age;

public Person(string name)

{

this.name = name;

Console.WriteLine("Person(string name)");

}

public Person(string name, int age) : this(name)

{

this.age = age;

Console.WriteLine("Person(string name, int age)");

}

}

class Employee : Person

{

string company;

public Employee(string name, int age, string company) : base(name, age)

{

this.company = company;

Console.WriteLine("Employee(string name, int age, string company)");

}

}

При створенні об’єктуа Employee:

Employee worker = new Employee("Kolya", 22, "Microsoft");

отримаємо наступний консольний вивід:

Person(string name)

Person(string name, int age)

Employee(string name, int age, string company)

У підсумку ми отримуємо наступний ланцюг викликів методів.

Спочатку викликається конструктор **Employee (string name, int age, string company)**. Він делегує виконання конструктору **Person (string name, int age)**

Викликається конструктор **Person (string name, int age)**, який сам поки не виконується і передає виконання конструктору **Person (string name)**

Викликається конструктор **Person (string name)**, який передає виконання конструктору класу **System.Object**, так як це базовий за замовчуванням клас для **Person**.

Виконується конструктор **System.Object.Object()**, потім виконання повертається конструктору **Person (string name)**

Виконується тіло конструктора **Person (string name)**, потім виконання повертається конструктору **Person (string name, int age)**

Виконується тіло конструктора **Person(string name, int age)**, потім виконання повертається конструктору **Employee(string name, int age, string company)**

Виконується тіло конструктора **Employee (string name, int age, string company)**. В результаті створюється об'єкт **Employee**

#### 5 Інтерфейси

Інтерфейс являє посилальний тип, який може оголошувати деякий функціонал, тобто набір методів і властивостей без їх реалізації. Потім цей функціонал реалізують класи і структури, які застосовують дані інтерфейси.

##### Оголошення інтерфейсу

Для оголошення інтерфейсу використовується ключове слово ***interface***. Як правило, назви інтерфейсів в C# починаються з великої літери **I**, наприклад, ***IComparable****,* ***IEnumerable та IEnumerator*** (так звана угорська нотація), однак це не обов'язкова вимога, а більше стиль програмування.

Що може визначати інтерфейс? В цілому інтерфейси можуть визначати такі сутності:

* методи;
* властивості;
* індексатори;
* події;
* статичні поля і константи (починаючи з версії C# 8.0).

Однак інтерфейси **не можуть** визначати нестатичні змінні. Наприклад, найпростіший інтерфейс, який визначає всі ці компоненти:

interface IMovable

{

const int minSpeed = 0; // константа мінімальна швидкість

// статична змінні

static int maxSpeed = 60; // максимальна швидкість

void Move(); // метод руху

string Name { get; set; } // властивість назва

delegate void MoveHandler(string message); //делегат для події

event MoveHandler MoveEvent; // подія руху

}

В даному випадку визначено інтерфейс ***IMovable***, який представляє певний об'єкт, що рухається. Методи і властивості інтерфейсу можуть не мати реалізації, в цьому вони зближуються з абстрактними методами і властивостями абстрактних класів. В даному випадку інтерфейс оголошує метод ***Move***, який буде представляти деяке пересування. Він не має реалізації. Те саме стосується властивості ***Name***. Якщо його члени – методи і властивості не мають модифікаторів доступу, то за замовчуванням цей доступ є ***public***, так як мета інтерфейсу - оголошення функціоналу для реалізації його класом. Це стосується також і констант і статичних змінних, які в класах і структурах за замовчуванням мають модифікатор ***private***. В інтерфейсах вони мають за замовчуванням модифікатор ***public***.

interface IMovable

{

public const int minSpeed = 0; // мінімальна швидкість

private static int maxSpeed = 60; // максимальна швидкість

public void Move(); // метод руху

protected internal string Name { get; set; } // властивість назва

public delegate void MoveHandler(string message); // делегат для події

public event MoveHandler MoveEvent; // подія руху

}

##### Застосування інтерфейсів

Не можна створювати об'єкти інтерфейсу безпосередньо за допомогою конструктора, як наприклад, в класах: Інтерфейс призначений для реалізації в класах і структурах. Наприклад, маючи інтерфейс **IMovable**, який-небудь клас або структура можуть застосувати даний інтерфейс:

using System;

namespace HelloApp

{

interface IMovable

{

void Move();

}

class Person : IMovable

{

public void Move()

{

Console.WriteLine("Людина рухається");

}

}

struct Car : IMovable

{

public void Move()

{

Console.WriteLine("Машина їде");

}

}

class Program

{

static void Action(IMovable movable)

{

movable.Move();

}

static void Main(string[] args)

{

Person person = new Person();

Car car = new Car();

Action(person);

Action(car);

Console.Read();

}

}

}

У даній програмі визначено метод ***Action***(), який в якості параметра приймає об'єкт інтерфейсу ***IMovable***. На момент написання коду ми можемо не знати, що це буде за об'єкт - якийсь клас або структура. Єдине, в чому ми можемо бути впевнені, що цей об'єкт обов'язково реалізує метод ***Move***() і ми можемо викликати цей метод. Іншими словами, інтерфейс – це контракт, що якийсь певний тип обов'язково реалізує деякий функціонал.

Інтерфейси мають ще одну важливу функцію: C# не підтримує множинне успадкування, тобто можна успадкувати клас тільки від одного класу. Інтерфейси дозволяють частково обійти це обмеження, оскільки в C# клас може реалізувати відразу кілька інтерфейсів. Усі реалізовані інтерфейси вказуються через кому:

using System;

namespace HelloApp

{

interface IAccount

{

int CurrentSum { get; } // Поточна сума на рахунку

void Put(int suma); // Покласти гроші на рахунок

void Withdraw(int suma); // зняти гроші з рахунку

}

interface IClient

{

string Name { get; set; }

}

class Client : IAccount, IClient //клас реалізує два інтерфейси

{

int sum; // сума грошей

public string Name { get; set; }

public Client(string name, int suma)

{

Name = name;

sum = sum;

}

public int CurrentSum { get { return sum; } }

public void Put(int sum) { sum += suma; }

public void Withdraw(int sum)

{

if (sum >= suma)

{

sum -= suma;

}

}

}

class Program

{

static void Main(string[] args)

{

Client client = new Client("Igor", 200);

client.Put(30);

Console.WriteLine(client.CurrentSum); //230

client.Withdraw(100);

Console.WriteLine(client.CurrentSum); //130

Console.Read();

}

}

}

##### Стандартні інтерфейси NET

|  |  |  |
| --- | --- | --- |
| **Інтерфейс** | **Методи** | **Опис** |
| IComparable | CompareTo() | Задає метод порівняння об'єктів за принципом більше або менше. Метод повертає результат порівняння двох об'єктів - поточного і переданого йому в якості параметра. |
| IComparer | Compare() | Метод повертає результат порівняння двох об'єктів, переданих йому в якості параметрів. |
| ICloneable | MemberwiseClone()  Clone() | Клонування – створення копії об'єкта.. Якщо необхідно скопіювати в іншу область пам'яті поля об'єкта, можна скористатися методом MemberwiseClone (), який будь-який об'єкт успадковує від класу Оbject. Це називається поверхневим клонуванням. Для створення повністю незалежних об'єктів необхідно глибоке копіювання, коли в пам'яті створюється дублікат всього дерева об'єктів, тобто об'єктів, на які посилаються поля об'єкта, поля полів тощо. Об'єкт, що має власні алгоритми клонування, повинен оголошуватися як похідний інтерфейсу ICloneable і перевизначати його єдиний метод Clone (). |
| IEnumerable | GetEnumerator() | Задає послідовність перебору елементів об'єкта, повертає об'єкт типу IEnumerator |
| IEnumerator | GetEnumerator(), Current, MoveNext(), Reset(). | Властивість Current повертає поточний елемент об'єкта, метод MoveNext () просуває нумератор на наступний елемент об'єкта, метод Reset () встановлює нумератор в початок перегляду. |

#### 6. Абстрактні класи

Абстрактний клас схожий на звичайний клас. Він також може мати змінні, методи, конструктори, властивості. Єдине, що при визначенні абстрактних класів використовується ключове слово ***abstract***. Головна відмінність полягає в тому, що не можна використовувати конструктор абстрактного класу для створення його об'єкта.

Навіщо потрібні абстрактні класи? Припустимо, в нашій програмі для банківського сектора ми можемо визначити дві основні сутності: клієнт банку і співробітник банку. Кожна з цих сутностей має свою специфіку, наприклад, для співробітника треба визначити його посаду, а для клієнта – суму на рахунку. Відповідно клієнт і співробітник становитимуть окремі класи ***Client*** та ***Employee***. У той самий час обидві ці сутності можуть мати щось спільне, наприклад, ім'я та прізвище, якусь іншу загальну функціональність. І цю загальну функціональність краще винести в якийсь окремий клас, наприклад, ***Person***, який описує людини. Тобто класи ***Employee*** (співробітник) і ***Client*** (клієнт банку) будуть похідними від класу ***Person***. І так як всі об'єкти в нашій системі представлятимуть або співробітника банку, або клієнта, то безпосередньо ми від класу ***Person*** створювати об'єкти не будемо. Тому має сенс зробити його абстрактним:

**abstract** class Person

{

public string Name { get; set; }

public Person(string name)

{

Name = name;

}

public void Display()

{

Console.WriteLine(Name);

}

}

class Client : Person

{

public int Sum { get; set; } // сума на рахунку

public Client(string name, int sum) : base(name)

{

Sum = sum;

}

}

class Employee : Person

{

public string Position { get; set; } // посада

public Employee(string name, string position): base(name)

{

Position = position;

}

}

Потім ми зможемо використовувати ці класи:

Client client = new Client("Oleg", 500);

Employee employee = new Employee("Boris", "Apple");

client.Display();

employee.Display();

### Варіанти завдань для лабораторної роботи № 4

Виконання лабораторної роботи №4 передбачає три етапи аналогічно лабораторній роботі №3 (рис. 5).

**Етап 1** – об’єктно-орієнтований аналіз (OOA) предметної області з розширенням бізнес-логіки завдань лаб.роботи №3 на додаткові бізнес-задачі, що визначені в завданнях лабораторної роботи №4. Об'єктно-орієнтований аналіз дозволяє доповнити сутності предметної області лабораторної роботи №3 та їхню поведінку новими особливостями. На вході – предметні сутності версії 1 згідно із завданнями лаб. роботи №3, опис додаткових сутностей та їх поведінки згідно із завданнями лаб. роботи №4. Результат – модель предметної області у вигляді сукупності сутностей з атрибутами та операціями, що характеризують поведінку сутностей, відповідно до принципів SOLID та опису предметної області лабораторної роботи №4.

**Етап 2** – об’єктно-орієнтоване проєктування (OOD). На входе результат об’єктно-орієнтованої декомпозиції (ООА). Результат – UML діаграма класів, яка містить класи і взаємозв'язки між ними та UML Use Case діаграма для визначення сценарію роботи програми.

**Етап 3** – об’єктно-орієнтоване програмування (OOP). На вході UML діаграма класів і UML Use Case діаграма. Результат – код програми відповідно до вимог чистого коду (гарного стилю програмування) та принципів SOLID.

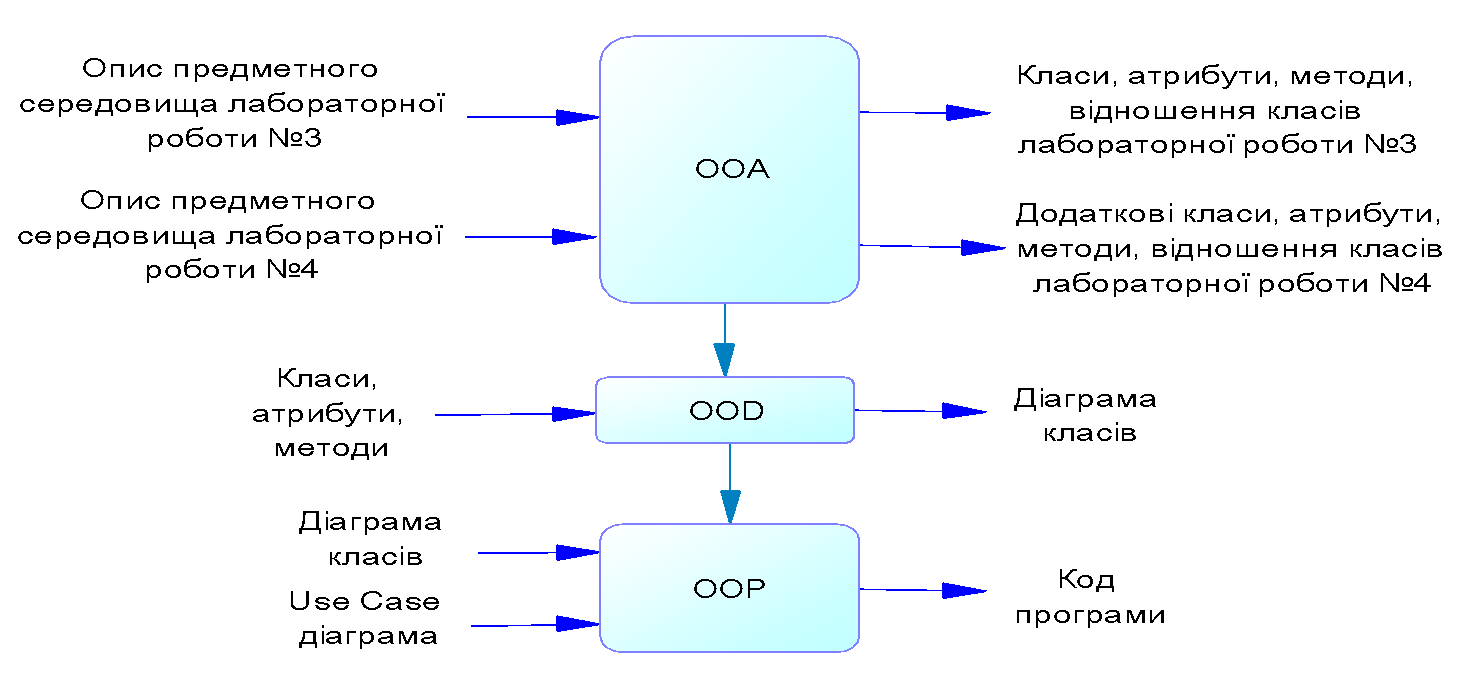


Рисунок 5 – Етапи об'єктно-орієнтованого підходу до розробки ПЗ для лабораторної роботи №4.

У цій лабораторній роботі слід дотримуватись **SOLID принципів єдиної відповідальності (SRP), принципу відкритості – закритості (OCP) та принцип підстановки Лісков (LSP).**

**Принцип єдиної відповідальності (SRP)** - кожен клас має відповідати тільки за одну область функціональності і мати тільки одну задачу або відповідальність.

**Принципу відкритості – закритості (OCP) -** класи повинні бути відкритими для розширення, але закритими для модифікації. Це означає, що поведінка класу може бути змінена без зміни його вихідного коду, шляхом додавання нового коду, а не модифікації існуючого.

**Принцип підстановки Лісков (LSP) -** класи-нащадки повинні мати можливість замінити батьківські класи без порушення функціональності**.**

Виконання лабораторної роботи передбачає чотири версії коду, кожна версія в окремому проєкті Visual Studio.

В методі *Main*() класу *Program* продемонструвати виклик усіх методів усіх класів. Усі значення, що розраховуються, записувати до *текстових файлів* методам класу **Service.** Також у файл записувати *протокол роботи програми*, тобто дані, що виводяться на консоль.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**.

**Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є узагальнена сутність «**Людина**». Сутності «**Викладач**» і «**Студент**» є нащадками від узагальненої сутності «Людина» і успадковують від неї спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. За бажанням використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4 1.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.1.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Людина | Базовий клас для всіх учасників навчального процесу | Ім’я,  Назва дисципліни | Отримати ім’я особи Змінити ім’я Отримати назву дисципліни Змінити назву дисципліни | | Викладач | Джерело навчального контенту; координує дисципліну; виконує оцінювання студентів | Ім’я викладача (успадковано) Назва дисципліни (успадковано) Навчальне навантаження (години) Кількість студентів | Збільшити кількість студентів  Зменшити кількість студентів Змінити обсяг навчального навантаження  Поставити оцінку студенту Передати матеріал студенту Записати оцінку в журнал | | Студент | Отримувач знань і навчального контенту; виконує завдання; формує власний рейтинг | Ім’я студента (успадковано) Назва дисципліни (успадковано) Список робіт з оцінками Обсяг виконаних робіт | Додати оцінку Переглянути оцінки  Розрахувати рейтинг Отримати матеріал від викладача | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один викладач** працює з **одним студентом**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (викладач і студент) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про викладача і студента згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   * + - 1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4.       2. Створити новий клас **Person** (базовий клас), модифікувати класи **Teacher** та **Student** з лабораторної роботи №3, зробивши їх похідними від Person. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл.4.1.1).       3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **Person**, зробивши його **protected** для заборони створення об’єкту класу **Person** як не релевантного предметній області.       4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**.       5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів.       6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.1.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **Person** абстрактним. Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.   **Версія 3 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **Person**, від якого успадковуються класи **Teacher** та **Student**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.   **Версія 4 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1. Створити окремий клас **StudentGroup**, який в якості поля має масив (список) об'єктів класу **Student**. В класі **StudentGroup** реалізувати: * інтерфейс *IComparable* для порівняння студентів за рейтингом методом *CompareTo*(); * інтерфейси *IComparer* для порівняння студентів не тільки за рейтингом, але і за обсягом виконаних робіт методом *Compare*(); * інтерфейс *IEnumerable* та *IEnumerator* для виведення на консоль списку студентів, впорядкованих за рейтингом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є узагальнена сутність «**Освітній суб’єкт**». Сутності «**Кафедра**» і «**Студент**» є нащадками від узагальненої сутності «**Освітній суб’єкт**». «**Кафедра**» - це організаційний освітній суб’єкт;«**Студент**» - це індивідуальний освітній суб’єкт. Ці суб’єкти успадковують від узагальненої сутності «**Освітній суб’єкт**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Через бажання використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4.1.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.2.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітній суб’єкт | Базова сутність для уніфікації обробки напряму підготовки | Ім’я сутності,  Назва напряму підготовки | Отримати ім’я  Змінити ім’я Отримати назву напряму Змінити назву напряму | | Кафедра | Організаційна одиниця, яка адмініструє напрям підготовки та облік студентів | Назва кафедри (успадковано як назва сутності) Назва напряму підготовки (успадковано) Поточна кількість студентів Макс. кількість студентів Перелік дисциплін за напрямом підготовки | Збільшити кількість студентів  Зменшити кількість студентів Додати дисципліну (за напрямом) Видалити дисципліну | | Студент | Учасник освітнього процесу, чия успішність вимірюється на основі оцінок | Ім’я студента (успадковано як назва сутності) Назва напряму підготовки (успадковано) Список оцінок Рівень навантаження. | Додати оцінку з дисципліни Переглянути оцінки  Розрахувати рейтинг | | Сервіс | Технічний посередник: забезпечує обмін даними, введення/виведення, збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра** працює з **одним студентом**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і студент) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру і студента згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   * + - 1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4.       2. Створити новий клас **EducationalEntity** (базовий клас), модифікувати класи **Department** та **Student** з лабораторної роботи №3, зробивши їх похідними від **EducationalEntity**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.1.1).       3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **EducationalEntity**, зробивши його **protected** для заборони створення об’єкту класу **EducationalEntity** як не релевантного предметній області.       4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**.       5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів.       6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.2.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **EducationalEntity** абстрактним. Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.   **Версія 3 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **EducationalEntity**, від якого успадковуються класи **Department** та **Student**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.   **Версія 4 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1. Створити окремий клас **CatalogDepartment**, який в якості поля має масив об'єктів класу **Department**. В класі **CatalogDepartment** реалізувати: * інтерфейс *IComparable* для порівняння кафедр за кількістю студентів методом *CompareTo*(); * інтерфейс *IComparer* для порівняння кафедр не тільки за кількістю студентів, але і за кількістю дисциплін (перелік дисциплін); * інтерфейси *IEnumerable та IEnumerator* для виведення на консоль список кафедр, впорядкованих за кількістю студентів (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є узагальнена сутність «**Освітня одиниця**». Сутності «**Кафедра**» і «**Викладач**» є нащадками від узагальненої сутності «**Освітня одиниця**». «**Кафедра**» - це організаційний освітній суб’єкт;«**Викладач**» - це індивідуальний освітній суб’єкт. Ці суб’єкти успадковують від узагальненої сутності «**Освітня одиниця**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Через бажання використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4.3.1 подані ролі, атрибути та операції сутностей освітнього процесу.  Таблиця 4.3.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітня одиниця | Базова сутність для уніфікації обробки даних про викладача | Назва одиниці Загальне річне навантаження (години) Перелік дисциплін | Отримати назву  Змінити назву Додати / видалити дисципліну Отримати / оновити загальне навантаження | | Кафедра | Організаційна одиниця, яка керує ресурсами, планує і розподіляє навантаження | Назва кафедри (успадковано як назва одиниці) Загальне річне навантаження (успадковано) Перелік дисциплін (успадковано)  Поточна кількість викладачів Список викладачів | Додати викладача (з перевіркою, що його навантаження не перевищує 600 годин) Видалити викладача Розподілити загальне навантаження між викладачами Переглянути розподіл | | Викладач | Виконує навчальну, наукову та методичну діяльність | ПІБ (успадковано як назва) Загальне річне навантаження (успадковано)  Спеціалізація викладача Кількість дисциплін К-сть наукових / методичних / орг. робіт | Додати наукову або методичну роботу Перевірити перевищення допустимого навантаження (норма 600 год) Визначити рейтинг за продуктивністю | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Рейтинг викладача може обчислюватись, наприклад, так:  *Рейтинг = (К-сть наукових робіт × 2 + К-сть методичних × 1.5 +К-сть орг. × 1) / (Загальне навантаження / 100)*  Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: на **одній кафедрі** працює **один викладач**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і викладач) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру і викладача згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   * + - 1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4.       2. Створити новий клас **EducationalUnit** (базовий клас), модифікувати класи **Department** та **Teacher** з лабораторної роботи №3, зробивши їх похідними від **EducationalUnit**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.3.1).       3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **EducationalUnit**, зробивши його **protected** для заборони створення об’єкту класу **EducationalUnit** як не релевантного предметній області.       4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**.       5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів.       6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.3.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **EducationalUnit** абстрактним. Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.   **Версія 3 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **EducationalUnit**, від якого успадковуються класи **Department** та **Teacher**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.   **Версія 4 в окремому проєкті.**   * + - 1. Зробити копію проєкту версії 1. Створити окремий клас **CatalogTeacher** який в якості поля має масив об'єктів класу **Teacher**. В класі **CatalogTeacher** реалізувати: * інтерфейс *IComparable* для порівняння кафедр за рейтингом викладачів методом *CompareTo*(); * інтерфейс *IComparer* для порівняння кафедр не тільки за рейтингом викладачів, але й за їх річним навантаженням методом *Compare*(); * інтерфейси *IEnumerable та IEnumerator* для виведення на консоль списку викладачів, впорядкованих за рейтингом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є узагальнена сутність «**Організаційна структур**а». Сутності «**ІТ-компанія**» і «**Факультет**» є нащадками від узагальненої сутності «**Організаційна структур**а». Ці суб’єкти успадковують від узагальненої сутності «**Організаційна структура**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Через бажання використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4.4.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.4.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Організаційна структура | Базова сутність для суб’єктів взаємодії у сфері підготовки фахівців | Назва структури Сфера або напрям діяльності Якість процесу Партнерська структура Перелік напрямів або кафедр | Змінити назву Змінити якість Підтвердити/змінити партнерство Додати напрям або кафедру | | ІТ-компанія | Комерційна структура, яка створює або інтегрує кафедри для покращення практичної підготовки | Сфера діяльності (успадковано) Якість практичної підготовки (успадковано від якості процесу)  Закріплена кафедра Партнерський факультет | Закріпити кафедру за напрямом (сферою діяльності) Погодити співпрацю з факультетом Покращити якість практичної підготовки | | Факультет | Академічна одиниця, що бере участь у підготовці фахівців спільно з ІТ-компаніями | Назва факультету (успадковано) Перелік напрямів (успадковано)  Якість освітнього процесу (успадковано)  Кількість кафедр Загальне річне навантаження Партнерська ІТ-компанія | Затвердити напрями Визначити річний обсяг годин Розподілити навантаження між кафедрами Погодити співпрацю з ІТ-компанією Контролювати якість освіти | | Сервіс | Технічний посередник для обміну, введення/ виведення,збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна компанія** співпрацює з **одним факультетом**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (компанія і факультет) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про компанію і факультет згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **OrganizationalStructure** (базовий клас), модифікувати класи **Faculty** та **ITCompany** з лабораторної роботи №3, зробивши їх похідними від **OrganizationalStructure**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.4.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **OrganizationalStructure**, зробивши його **protected** для заборони створення об’єкту класу **OrganizationalStructure** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.4.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **OrganizationalStructure** абстрактним. Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.   **Версія 3 в окремому проєкті.**   1. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **OrganizationalStructure**, від якого успадковуються класи **Faculty** та **ITCompany**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.   **Версія 4 в окремому проєкті.**   1. Зробити копію проєкту версії 1. Створити окремий клас **CatalogFaculties**, який в якості поля має масив об'єктів класу **Faculty**. В класі **CatalogFaculties** реалізувати:  * інтерфейс *IComparable* для порівняння факультетів за кількістю кафедр методом *CompareTo*(); * інтерфейс *IComparer* для порівняння факультетів не тільки за кількістю кафедр, але й за їх річним навантаженням; * інтерфейси *IEnumerable та IEnumerator* для виведення на консоль список факультетів, впорядкованих за кількістю викладачів(реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **5.** | **Опис предметної області**. Існує інституційно-адміністративна модель, в якій сутностями моделі є узагальнена сутність «**Академічна установа**». Сутності «**Університет**» і «**Факультет**» є нащадками узагальненої сутності «**Академічна установа**». Ці суб’єкти успадковують від узагальненої сутності «**Академічна установа**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4.5.1 подані ролі, атрибути та операції сутностей інституційно-адміністративної моделі.  Таблиця 4.5.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Академічна установа | Базова сутність для всіх освітніх організаційних одиниць | Назва установи Кількість студентів Кількість підрозділів (кафедр/факультетів) Список спеціальностей або факультетів | Змінити назву Змінити кількість студентів Змінити кількість підрозділів Додати / видалити спеціальність або підрозділ | | Університет | Головна академічна установа, що координує факультети, формує рейтинг і бюджет | Назва університету (успадковано) К-сть факультетів (успадковано) К-сть студентів (успадковано)  К-сть наукових праць К-сть міжнародних проєктів Середній бал ЗНО вступників  Рейтинг (Osvita.ua)  Річний бюджет | Розрахувати рейтинг за версією Osvita.ua Розрахувати бюджет на основі рейтингу Додати / вилучити факультет | | Факультет | Академічна одиниця, що бере участь у підготовці фахівців спільно з ІТ-компаніями | Назва факультету (успадковано) Кількість студентів (успадковано)  Кількість кафедр Кількість спеціальностей Пов’язаний університет | Збільшити / зменшити кількість кафедр Додати / видалити спеціальність Змінити кількість студентів | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один університет має** **один факультет.** **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (університет і факультет) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про університет і факультет згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **AcademicInstitution** (базовий клас), модифікувати класи **University** та **Faculty** з лабораторної роботи №3, зробивши їх похідними від **AcademicInstitution**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.5.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **AcademicInstitution**, зробивши його **protected** для заборони створення об’єкту класу **AcademicInstitution** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.5.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **AcademicInstitution** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.*   **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **AcademicInstitution**, від якого успадковуються класи **University** та **Faculty**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **CatalogUniversity**, який в якості поля має масив об'єктів класу **University**. В класі **CatalogUniversity** реалізувати:   * інтерфейс *IComparable* для порівняння університетів за рейтингом методом *CompareTo*(); * інтерфейс *IComparer* для порівняння університетів не тільки за рейтингом, але й за їх річним бюджетом; * інтерфейси *IEnumerable та IEnumerator* для виведення на консоль списку університетів, впорядкованих за рейтингом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **6.** | **Опис предметної області**. Існує освітнє середовище, в якому сутностями моделі є узагальнена сутність «**Функціональна структура**». Сутності «**Університет**» і «**Факультет**» є нащадками узагальненої сутності «**Функціональна структура**». Ці суб’єкти успадковують від узагальненої сутності «**Функціональна структура**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача, слід створити окрему сутність Меню (не обов’язково). В таблиці 4.6.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.6.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Функціональна структура | Базова сутність, що виконує проєкти, взаємодіє з фахівцями, формує якість і вплив | Назва структури Кількість працівників Кількість реальних проєктів | Змінити назву Додати працівника Збільшити кількість проєктів | | Кафедра | Освітній підрозділ ЗВО | Назва (успадковано) Кількість викладачів як працівників (успадковано) Кількість дисциплін Кількість креативних студентів Якість освітньої програми | Покращити якість освітньої програми Залучити фахівців з компанії Збільшити кількість креативних студентів | | ІТ-компанія | Роботодавець, партнер кафедри, джерело практичного досвіду | Назва (успадковано) Кількість працівників (успадковано) Кількість реальних проєктів (успадковано)  Прибуток Кількість найнятих випускників | Найняти випускників Збільшити прибуток Збільшити проєктну активність | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Формат виводу Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра - одна ІТ-компанія.** **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (кафедра і ІТ-компанія) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру і компанію згідно зі сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **FunctionalStructure** (базовий клас), модифікувати класи з лабораторної роботи №3, зробивши їх похідними від **FunctionalStructure**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас (не обов'язково). Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.6.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **FunctionalStructure**, зробивши його **protected** для заборони створення об’єкту класу **FunctionalStructure** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.6.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **FunctionalStructure** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.*  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **FunctionalStructure**, від якого успадковуються класи **Department** та **Company**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **CatalogCompanies**, який в якості поля має масив об'єктів класу **Company**. В класі **CatalogCompanies** реалізувати:   * інтерфейс *IComparable* для порівняння компаній за прибутком методом *CompareTo*(); * інтерфейс *IComparer* для порівняння університетів не тільки за прибутком, але й за кількістю проєктів; * інтерфейси *IEnumerable та IEnumerator* для виведення на консоль списку компаній, впорядкованих за прибутком (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **7** | **Опис предметної області**. Існує предметна область у вигляді освітнього середовища, в якому сутностями моделі є узагальнена сутність «**Освітній елемент**». Сутності «**Студент**» і «**Навчальна програма**» є нащадками узагальненої сутності «**Освітній елемент**». Ці суб’єкти успадковують від узагальненої сутності «**Освітній елемент**» спільні атрибути (Назва) та операції. Оскільки дві похідні сутності різнопланові — одна репрезентує **людину** , інша — **освітній продукт/структуру**, вводиться додатковий базовий атрибут «**Тип освітнього елемента**», значенням якого є «людина» або «структура». Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача, слід створити окрему технічну сутність Меню (не обов’язково), яка не входить в предметну область. В таблиці 4.7.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.7.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітній елемент | Базова сутність, що узагальнює освітні елементи | Назва освітнього елемента  Тип освітнього елемента (людина/структура) | Отримати назву  Змінити назву  Отримати тип Змінити тип | | Студент | Похідна сутність - учасник освітнього процесу, який навчається за програмою | ПІБ (успадковано як назва)  Тип (успадковано як людина)  Список оцінок (0–10),  Рейтинг (середній бал),  Розмір стипендії,  Назва навчальної програми | Розрахувати рейтинг  Розрахувати стипендію  Показати слабкі місця у навчанні за аналізом оцінок | | Навчальна програма | Похідна сутність - освітній маршрут студента | Назва програми (успадковано як назва),  Тип (успадковано як структура)  Кількість кредитів,  Список дисциплін,  Перелік результатів навчання,  Рівень (бакалавр, магістр),  Якість навчальної програми (оцінка) | Визначити якість навчальної програми  Визначити складність навчальної програми  Оцінити відповідність навчальної програми ринку праці | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент - одна навчальна програма.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності ( студент і навчальна програма) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента і навчальну програму за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **EducationalElement** (базовий клас), модифікувати класи з лабораторної роботи №3, зробивши їх похідними від **EducationalElement**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас (не обов'язково). Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.7.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **EducationalElement**, зробивши його **protected** для заборони створення об’єкту класу **EducationalElement** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.7.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**   1. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **EducationalElement** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування.*   **Версія 3 в окремому проєкті.**   1. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **EducationalElement**, від якого успадковуються класи **Student** та **Curriculum**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.   **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **CatalogCurriculum**, який в якості поля має масив об'єктів класу **Curriculum**. В класі **CatalogCurriculum** реалізувати:   * інтерфейс *IComparable* для порівняння навчальних програм за їх якістю методом *CompareTo*(); * інтерфейс *IComparer* для порівняння навчальних програм не тільки за якістю, але й за кількістю дисциплін методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль списку навчальних програм, впорядкованих за кількістю дисциплін (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*) (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **8** | **Опис предметної області**. Існує предметна область у вигляді банківської мережі, в якій сутностями моделі є узагальнена сутність «**Банківська мережа**». Сутності «**Клієнт**» і «**Банкомат**» є нащадками узагальненої сутності «**Банківська мережа**». Ці суб’єкти успадковують від узагальненої сутності «**Банківська мережа**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.8.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.8.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Банківська мережа | Узагальнена сутність, що представляє взаємодію об'єктів банківської системи | Ідентифікатор об’єкта в мережі,  Статус активності | Авторизувати,  Завершити сесію | | Клієнт | Власник грошей, користувач банкомата | Ідентифікатор клієнта (успадковано),  Статус активності (успадковано як «активний», «блокований», «неавторизований»)  Ім’я, Прізвище,  Номер картки,  PIN-код,  Баланс | Перевірити PIN-код,  Зняти кошти,  Поповнити рахунок,  Перевірити баланс | | Банкомат | Спеціальний пристрій, за допомогою якого можна керувати готівкою зі свого банківського рахунку | Ідентифікатор банкомата (успадковано),  Статус активності (успадковано як «працює», «вимкнений», «немає готівки»)  Локація,  Доступні кошти (сума готівки, яка є в банкоматі),  Стан (працює / вимкнений / немає готівки) | Прийняти карту клієнта,  Видати готівку,  Показати залишок,  Заблокувати банківську картку через неправильний пін-код  Прийняти готівку | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт – один банкомат.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (клієнт і банкомат) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта і банкомат за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **BankingNetwork** (базовий клас), модифікувати класи з лабораторної роботи №3, зробивши їх похідними від **BankingNetwork**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас (не обов'язково). Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.8.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **BankingNetwork**, зробивши його **protected** для заборони створення об’єкту класу **BankingNetwork** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.8.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **BankingNetwork** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **BankingNetwork**, від якого успадковуються класи **Client** та **ATM**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **ATMNetwork**, який в якості поля має масив (список) об'єктів класу **ATM**. В класі **ATMNetwork** реалізувати:   * інтерфейс *IComparable* для порівняння банкоматів за обсягом доступних коштів методом *CompareTo*(); * інтерфейс *IComparer* для порівняння банкоматів не тільки за обсягом доступних коштів, але й за станом методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль списку банкоматів, впорядкованих за обсягом готівки (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **9** | **Опис предметної області**. Існує предметна область у вигляді освітнього середовища, в якому сутностями моделі є узагальнена сутність «**Самоврядування**». Сутності «**Студент**» і «**Студентський парламент**» є нащадками узагальненої сутності «**Самоврядування**». Ці суб’єкти успадковують від узагальненої сутності «**Самоврядування**» спільні атрибути (назва, статус тощо) та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача, слід створити окрему технічну сутність Меню (не обов’язково), яка не входить в предметну область. В таблиці 4.9.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.9.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Самоврядування | Узагальнена сутність як спільнота студентів, що організовують студентське життя | Назва або ПІБ,  Рівень участі,  Статус активності | Оновити статус, Подати/опрацювати ініціативу/пропозицію | | Студент | Учасник освітнього процесу, який може бути членом студ. парламенту | ПІБ студента (успадковано як назва),  Рівень (успадковано як Рівень участі – індивідуальний учасник )  Статус активності (успадкований бажаючий, кандидат, член парламенту)  Курс,  Рейтинг (середній бал),  Стипендія  Ідеї студента | Розрахувати рейтинг  Подати заявку на участь у парламенті  Взяти участь у голосуванні  Ініціювати пропозицію / ідею  Розрахувати розмір стипендії | | Студентський парламент | Представницький орган самоврядування | Назва парламенту (успадковано як назва),  Рівень (успадковано як Рівень участі – факультет, університет), Статус активності (успадковано – активний, неактивний),  Склад (кількість членів) парламенту (назви департаментів),  Назва  проєкту  Кількість проєктів  Обсяг стипенд. фонду | Прийняти нового члена  Взяти участь у голосуванні  Оприлюднити проєкт / ініціативу  Формувати рекомендації щодо заохочення  студентів або їх відрахування  Призначити стипендії студентам.  Відхилити заявку  Оприлюднити результати голосування | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент – один студентський парламент.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності ( студент і студентський парламент) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента і студентський парламент за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **SelfGovernment** (базовий клас), модифікувати класи **Student,** **Student**P**arliament** з лабораторної роботи №3, зробивши їх похідними від **SelfGovernment**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас (не обов'язково). Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.9.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **SelfGovernment**, зробивши його **protected** для заборони створення об’єкту класу **SelfGovernment** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.9.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **SelfGovernment** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **SelfGovernment**, від якого успадковуються класи **Student** та **Student**P**arliament**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetStud**P**arliament**, який в якості поля має масив (список) об'єктів класу **Student**P**arliament**. В класі **SetStud**P**arliament** реалізувати:   * інтерфейс *IComparable* для порівняння студентських парламентів за кількістю членів методом *CompareTo*(); * інтерфейс *IComparer* для порівняння студентських парламентів не тільки за кількістю членів, але й за кількістю проєктів методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку студентських парламентів, впорядкованих за кількістю членів (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **10** | **Опис предметної області**. Існує предметна область у вигляді транспортної логістики, в якій сутностями моделі є узагальнена сутність «**Об’єкт транспортної системи**». Сутності «**Водій**» і «**Автомобіль**» є нащадками узагальненої сутності «**Об’єкт транспортної системи**». Ці суб’єкти успадковують від узагальненої сутності «**Об’єкт транспортної системи**» спільні атрибути (ID, Назва тощо) та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача, слід створити окрему технічну сутність Меню (не обов’язково), яка не входить в предметну область. В таблиці 4.10.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.10.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Об’єкт транспортної системи | Узагальнена сутність яка забезпечує рух, керування, контроль або обслуговування. | ID   * + Назва (ідентифікатор)   + Тип об’єкта *(водій / транспортний засіб)*   + Статус активності *(активний, не допущений, неактивний, зареєстрований, незареєстрований)*   + Поточний стан *(нормальний / критичний / потребує уваги)* | Оновити статус Надіслати повідомлення про інцидент Зареєструвати об'єкт Ініціювати перевірку стану | | Водій | Керує автомобілем, відповідає за його стан і рух | Ім’я, Прізвище (успадковано як ID та назва),  Стан водія (успадковано як статус активності)  Номер посвідчення,  Категорія прав,  Стаж керування автомобілем,  Маршрут руху. | Вчинити порушення ПДР  Повідомити про ДТП або технічну несправність  Заявити про потребу заміни авто  Контроль фізичного стану водія | | Автомобіль | Об'єкт керування і обліку | Модель (успадковано як назва),  Номер авто (успадковано як ID),  Технічний стан (успадковано як поточний стан)  Статус (успадковано як Статус активності – «в ремонті», «не зареєстрований», «списаний», «знятий з реєстрації» тощо),  Рік випуску,  Ступінь пошкодження | Пройти технічний огляд  Пройти реєстрацію  Поламатися внаслідок ДТП  Розрахувати вартість ремонту  Бути заміненим | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один водій – один автомобіль.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (водій та автомобіль) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про водія і автомобіль за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **TransportSystemObject** (базовий клас), модифікувати класи **Driver** (водій) та **Car** (автомобіль) з лабораторної роботи №3, зробивши їх похідними від **TransportSystemObject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності слід зробити меню як окремий клас (не обов'язково). Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.10.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **TransportSystemObject**, зробивши його **protected** для заборони створення об’єкту класу **TransportSystemObject ernment** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.10.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **TransportSystemObject** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **TransportSystemObject**, від якого успадковуються класи **Driver** та **Car**. Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfCar**, який в якості поля має масив (список) об'єктів класу **Car**. В класі **SetOfCar** реалізувати:   * інтерфейс *IComparable* для порівняння автомобілів за роком випуску методом *CompareTo*(); * інтерфейс *IComparer* для порівняння автомобілів не тільки за роком випуску, але й за ступенем пошкодження методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку автомобілів, впорядкованих за роком випуску (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **11** | **Опис предметної області**. Існує предметна область у вигляді наукової роботи, в якій сутностями моделі є узагальнена сутність «**Об’єкт конференції**». Сутності «**Конференція**» і «**Автор доповіді**» є нащадками узагальненої сутності «**Об’єкт конференції**». Ці суб’єкти успадковують від узагальненої сутності «**Об’єкт конференції**» спільні атрибути (ID, Назва тощо) та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.11.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.11.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Об’єкт конференції | Узагальнена сутність яка забезпечує контроль і керування конференцією та авторами. | ID,  Назва,  Статус,  Дата події / подачі | Оновити статус,  Перевірити на відповідність критеріям,  Ініціювати комунікацію | | Конференція | Захід для представлення наукових доповідей | ID (успадковано)  Назва конференції (успадковано як назва),  Дата проведення (успадковано як Дата події),  Статус (активна/архів) (успадкована як Статус)  Місце,  Напрямки  конференції,  Гранична дата подачі, | Реєструвати учасників  Отримати доповідь  Рецензувати доповідь  Відхилити доповідь  Прийняти доповідь  Друкувати доповідь  Перевірити  відповідність теми напрямку конференції | | Автор доповіді | Учасник конференції, який готує і подає наукову доповідь | ID (успадковано),  ПІБ (успадковано як Назва),  Дата подачі (успадковано як Дата події / подачі)  Статус (студент, аспірант викладач) (успадковано як Статус)  Науковий ступінь,  Тема доповіді,  Обсяг тексту, | Подати доповідь  Отримати рецензію  Внести правки до тексту доповіді  Розрахувати вартість участі в конференції | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один автор – одна конференція.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (автор і конференція) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про автора і конференцію за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **ConferenceObject** (базовий клас), модифікувати класи **Conference** (Конференція), **Author** (Автор) з лабораторної роботи №3, зробивши їх похідними від **ConferenceObject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.11.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **ConferenceObject**, зробивши його **protected** для заборони створення об’єкту класу **ConferenceObject** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.11.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **ConferenceObject** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **ConferenceObject**, від якого успадковуються класи **Conference** (Конференція), **Author** (Автор). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfAuthor**, який в якості поля має масив (список) об'єктів класу **Author**. В класі **SetOfAuthor** реалізувати:   * інтерфейс *IComparable* для порівняння авторів за обсягом тексту доповідей методом *CompareTo*(); * інтерфейс *IComparer* для порівняння авторів не тільки за обсягом тексту доповіді, але й за датою подачі методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку авторів, впорядкованих за статусом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **12** | **Опис предметної області**. Існує предметна область практико-орієнтованого навчання, в якій сутностями моделі є узагальнена сутність «**Старт кар’єри**». Сутності «**Студент**» і «**Біржа практик**» є нащадками узагальненої сутності «**Старт кар’єри**». Ці суб’єкти успадковують від узагальненої сутності «**Старт кар’єри**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.12.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 4.12.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Старт кар’єри | Узагальнена сутність для учасників і керуючих елементів системи | Назва об'єкта,  Тип (студент / компанія),  Статус активності (активний / неактивний, відхилений, виключений, очікування підтвердження, завершив участь ) | Отримати/змінити ідентифікатор  Отримати/змінити тип  Отримати/Змінити статус, | | Студент | Користувач біржі, який шукає та проходить практику | ПІБ (успадковано як назва)  Тип (успадковано як «студент»)  Статус активності (успадковано як: «активний / неактивний, відхилений, виключений, підтверджений, завершив участь»)  Факультет, Спеціальність,  Курс, Середній бал, Вид практики (виробнича, проєктна, переддипломна тощо) | Подати заявку на практику  Оцінити практику  Підтвердити завершення  Вибрати місце практики | | Біржа практик | Платформа для розміщення та призначення місць практики | Назва біржі (успадковано як назва),  Тип (успадковано як «компанія»)  Статус активності(успадковано як «активна, неактивна, очікує оновлення, завершена» )  Перелік компаній,  Перелік доступних практик,  Статус заявки,  Статус співпраці | Прийняти заявку студента  Призначити місце практики  Надіслати звітні документи  Показати місця практики (компаній) | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент – одна біржа практик.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності ( студент і біржа практик) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента і біржу практик за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **CareerStar** (базовий клас), модифікувати класи **Student** (Студент), **PracticeExchange** (Біржа практик) з лабораторної роботи №3, зробивши їх похідними від **CareerStar**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.12.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **CareerStar**, зробивши його **protected** для заборони створення об’єкту класу **CareerStar** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.12.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **CareerStar** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **CareerStar**, від якого успадковуються класи **Student** (Студент), **PracticeExchange** (Біржа практик). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfStudent**, який в якості поля має масив (список) об'єктів класу **Student**. В класі **SetOfStudent** реалізувати:   * інтерфейс *IComparable* для порівняння студентів за середнім балом методом *CompareTo*(); * інтерфейс *IComparer* для порівняння студентів не тільки за середнім балом, але й за типом практики методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку студентів, впорядкованих за спеціальностями (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **13** | **Опис предметної області**. Існує предметна область ріелторських послуг, в якій сутностями моделі є узагальнена сутність «**Сторони угоди**». Сутності «**Клієнт**» і «**Агенство нерухомості**» є нащадками узагальненої сутності «**Сторони угоди**». Ці суб’єкти успадковують від узагальненої сутності «**Сторони угоди**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.13.1 подані ролі, атрибути та операції сутностей ріелторських послуг.  Таблиця 4.13.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Сторони угоди про нерухомість | Узагальнена сутність для учасників угоди про нерухомість | Назва,  Тип сторони (агентство / клієнт), Статус участі (активна / тимчасова / завершена | Зареєструвати у системі (ініціалізувати атрибути), Активувати профіль (змінити статус участі),  Отримати / змінити атрибути, Надіслати запит / відповідь (визначити тип сторони угоди) | | Агентство нерухомості | Посередник, що підбирає, пропонує та супроводжує операції з нерухомістю | Назва агентства (успадковано як назва),  Тип сторони (успадковано як «агенство»)  Статус участі (успадковано)  Список об’єктів,  Ліцензія, | Прийняти запит клієнта  Шукати об’єкт відповідно до запиту клієнта  Оцінити вартість нерухомості  Підготувати договір  Закрити угоду | | Клієнт | Особа, яка звертається за послугами з оренди чи купівлі нерухомості | Ім’я, Прізвище (успадковано як назва)  Тип сторони (успадковано як «Клієнт»)  Статус участі (успадковано)  Тип запиту (купівля/оренда), Бюджет,  Бажаний район,  Статус запиту (новий, активний, завершений) | Шукати агенство нерухомості  Подати запит  Переглянути пропозиції  Укласти/відхилити угоду | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт – одне агенство нерухомості.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (клієнт та агенство нерухомості) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта та агенство нерухомості, за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **PartyToAgreement** (базовий клас), модифікувати класи **RealEstateAgency** (Агентство нерухомості), **Client** (клієнт) з лабораторної роботи №3, зробивши їх похідними від **PartyToAgreement**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.13.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **PartyToAgreement**, зробивши його **protected** для заборони створення об’єкту класу **PartyToAgreement** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.13.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **PartyToAgreement** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **PartyToAgreement**, від якого успадковуються класи **RealEstateAgency** (Агентство нерухомості), **Client** (клієнт). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfClient**, який в якості поля має масив (список) об'єктів класу **Client**. В класі **SetOfClient** реалізувати:   * інтерфейс *IComparable* для порівняння клієнтів за бюджетом методом *CompareTo*(); * інтерфейс *IComparer* для порівняння клієнтів не тільки за бюджетом, але й за типом запиту методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку клієнтів, впорядкованих за бюджетом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **14** | **Опис предметної області**. Існує предметна область академічної мобільності, в якій сутностями моделі є узагальнена сутність «**Сторона програми обміну**». Сутності «**Студент**» і «**Відділ академічної мобільності**» є нащадками узагальненої сутності «**Сторона програми обміну**». Ці суб’єкти успадковують від узагальненої сутності «**Сторона програми обміну**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.14.1 подані ролі, атрибути та операції сутностей предметної області академічної мобільності.  Таблиця 4.14.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Сторона програми обміну | Узагальнена сутність для процесу академічної мобільності | Ідентифікатор (*код для однозначної ідентифікації сторін обміну*),  Тип учасника *(студент / відділ)*,  Статус взаємодії *(активний, завершений, скасований)* | Ініціювати участь у програмі (ініціалізувати значення атрибутів),  Отримати/змінити тип учасника  Отримати/змінити статус заявки про взаємодію | | Студент | Здобувач освіти, учасник академ. мобільності | Ідентифікатор (успадковано),  Тип учасника (успадковано як “студент”)  Статус заявки (успадковано як Статус взаємодії)  ПІБ,  Освітній рівень (бакалавр, магістр),  Середній бал,  Наукові здобутки | Розрахувати рейтинг студента,  Подати заявку,  Отримати результат розгляду заявки,  Підписати угоду | | Відділ академічної мобільності | Організатор і координатор програм обміну | Ідентифікатор (успадковано),  Тип учасника (успадковано як “відділ”)  Статус взаємодії (успадковано)  Назва програми,  Мінімальний середній бал Кількість учасників,  Список (масив) ЗВО-партнерів | Розглянути заявку,  Шукати університет  Схвалити заявку,  Відхилити заявку,  Оформити документи, | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один студент – один відділ академічної мобільності.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності ( студент і відділ академічної мобільності) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента та відділ академічної мобільності за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **ExchangeProgramSide** (базовий клас), модифікувати класи **Student** (Студент), **AcademicMobility** (відділ академічної мобільності) з лабораторної роботи №3, зробивши їх похідними від **ExchangeProgramSide**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.14.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **ExchangeProgramSide**, зробивши його **protected** для заборони створення об’єкту класу **ExchangeProgramSide** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.14.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **ExchangeProgramSide** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **ExchangeProgramSide**, від якого успадковуються класи **Student** (Студент), **AcademicMobility** (відділ академічної мобільності). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfStudent**, який в якості поля має масив (список) об'єктів класу **Student**. В класі **SetOfStudent** реалізувати:   * інтерфейс *IComparable* для порівняння студентів за середнім балом методом *CompareTo*(); * інтерфейс *IComparer* для порівняння студентів не тільки за середнім балом, але й за науковими здобутками методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку студентів, впорядкованих за середнім балом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **15** | **Опис предметної області**. Існує предметна область Таксі-сервіс, в якій сутностями моделі є узагальнена сутність «**Сервіс таксі**». Сутності «**Клієнт**» і «**Таксі**» є нащадками узагальненої сутності «**Сервіс таксі**». Ці суб’єкти успадковують від узагальненої сутності «**Сервіс таксі**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.15.1 подані ролі, атрибути та операції сутностей послуг таксі.  Таблиця 4.15.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Таксі сервіс | Узагальнена сутність для процесу організації поїздки | Ідентифікатор замовлення (*код для однозначної ідентифікації поїздки*  Тип учасника *(клієнт / таксі)*,  Статус замовлення *(створене, активне, завершене, скасоване, немає машини)* | Ініціювати поїздку (ініціалізувати значення атрибутів),  Отримати/змінити тип учасника  Отримати/змінити статус замовлення | | Клієнт | Замовник послуги таксі | Ідентифікатор замовлення (успадковано),  Тип учасника (успадковано як "клієнт")  Статус замовлення (успадковано)  Ім'я клієнта, Контактний номер,  Місце посадки,  Місце призначення,  Спосіб оплати | Замовити таксі (вказати місце посадки, місце призначення)  Оплатити поїздку (сума, спосіб оплати)  Скасувати замовлення | | Таксі | Виконавець замовлення, перевізник | Ідентифікатор замовлення (успадковано)  Тип учасника (успадковано як "таксі")  Статус замовлення (успадковано)  Номер таксі,  Тариф за км, Тариф за хвилину  Вартість оплати, Відстань | Прийняти замовлення (місце посадки, місце призначення).  Розрахувати вартість поїздки (відстань, тариф)  Визначити оптимальний маршрут поїздки | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт – одне таксі.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності ( клієнт і таксі) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта та таксі за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **TaxiService** (базовий клас), модифікувати класи **Client** (Клієнт) та **Taxi** (таксі) з лабораторної роботи №3, зробивши їх похідними від **TaxiService** . Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.15.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **TaxiServic**, зробивши його **protected** для заборони створення об’єкту класу **TaxiService** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.15.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **TaxiService** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **TaxiService**, від якого успадковуються класи **Client** (Клієнт) та **Taxi** (таксі). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfTaxi**, який в якості поля має масив (список) об'єктів класу **Taxi**. В класі **SetOfTaxi** реалізувати:   * інтерфейс *IComparable* для порівняння таксі за тарифом за км методом *CompareTo*(); * інтерфейс *IComparer* для порівняння таксі не тільки за тарифом за км, але й за вартістю оплати методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку таксі, впорядкованих за вартістю оплати (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **16** | **Опис предметної області**. Існує предметна область фінансових послуг, в якій сутностями моделі є узагальнена сутність «**Фінансовий суб’єкт**». Сутності **«Клієнт»** і **«Банк»** є нащадками узагальненої сутності «**Фінансовий суб’єкт**». Ці суб’єкти успадковують від узагальненої сутності «**Фінансовий суб’єкт**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.16.1 подані ролі, атрибути та операції сутностей фінансових послуг.  Таблиця 4.16.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Фінансовий суб’єкт | Узагальнена роль для учасників фінансової взаємодії | Ідентифікатор,  Тип фінансового суб’єкта (Клієнт / Банк),  Статус взаємодії (активний / неактивний / завершений) | Завершити операцію  Ініціювати фінансову операцію (ініціалізувати значення атрибутів),  Отримати/змінити тип фінансового суб’єкта  Отримати/змінити статус взаємодії | | Клієнт | Фізична або юридична особа, яка користується фінансовими послугами банку | Ідентифікатор (успадковано)  Тип суб’єкта (успадковано як "Клієнт")  Статус взаємодії (успадковано)  ПІБ,  Паспорт, ІПН,  Номер рахунку,  Баланс рахунку,  Кредитний рейтинг | Відкрити рахунок  Закрити рахунок  Поповнити рахунок  Зняти кошти  Отримати кредит | | Банк | Фінансова установа, що надає банківські послуги | Ідентифікатор (успадковано),  Тип суб’єкта (успадковано як "Банк"), Статус взаємодії (успадковано)  Назва банку  Перелік послуг,  Процентні ставки,  Кредитні ліміти | Створити рахунок клієнта  Відкрити депозит  Надати кредит  Обробити платіж  Нарахувати відсотки по кредиту/депозиту | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт – один банк.** **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (клієнт і банк) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта та банк за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **FinancialSubject**(базовий клас), модифікувати класи **Client** (Клієнт) та **Bank** (Банк) з лабораторної роботи №3, зробивши їх похідними від **FinancialSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.16.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **FinancialSubject**, зробивши його **protected** для заборони створення об’єкту класу **FinancialSubject** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.16.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **FinancialSubject** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **TaxiService**, від якого успадковуються класи **Client** (Клієнт) та **Bank** (Банк). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfClient**, який в якості поля має масив (список) об'єктів класу **Client**. В класі **SetOfClient** реалізувати:   * інтерфейс *IComparable* для порівняння клієнтів за кредитним рейтингом методом *CompareTo*(); * інтерфейс *IComparer* для порівняння клієнтів не тільки за кредитним рейтингом, але й за балансом рахунку методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку клієнтів, впорядкованих за кредитним рейтингом (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **17** | **Опис предметної області**. Існує предметна область медичних послуг, в якій сутностями моделі є узагальнена сутність **«Медичний суб’єкт**». Сутності **«Пацієнт»** і **«Лікар»** є нащадками узагальненої сутності **«Медичний суб’єкт**». Ці суб’єкти успадковують від узагальненої сутності **«Медичний суб’єкт**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам медичного середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.17.1 подані ролі, атрибути та операції сутностей фінансових послуг.  Таблиця 4.17.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Медичний суб’єкт | Узагальнена роль для осіб, що беруть участь у медичному процесі | Ідентифікатор,  ПІБ,  Тип медичного суб’єкта (Пацієнт /Лікар),  Статус взаємодії (активний / завершений / очікує) | Завершити операцію  Ініціювати фінансову операцію (ініціалізувати значення атрибутів),  Отримати/змінити тип фінансового суб’єкта  Отримати/змінити статус взаємодії | | Пацієнт | Фізична особа, яка отримує медичну допомогу | Ідентифікатор (успадковано),  ПІБ (успадковано),  Тип суб’єкта (успадковано як “Пацієнт”,  Статус взаємодії (успадковано)  Дата народження,  Медична картка,  Історія хвороби,  Поточні скарги | Записатися на прийом,  Отримати діагноз, Отримати призначення лікування | | Лікар | Фізична особа, яка надає медичні послуги | Ідентифікатор (успадковано),  ПІБ (успадковано),  Тип суб’єкта (успадковано як “Лікар”,  Статус взаємодії (успадковано  Спеціалізація,  Стаж роботи,  Графік прийому,  Список пацієнтів,  Доступ до медичних карт | Провести прийом пацієнта,  Встановити діагноз, Призначити лікування | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один пацієнт – один лікар.** Тоді *список пацієнтів* можна не використовувати. **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (пацієнт і лікар) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про пацієнта та лікаря за сценарієм роботи версії 1 лабораторної роботи №3. Для спрощення коду не створюйте класи для “медичної картки”, та “історії хвороби”. Задайте їх у вигляді даних типу **string**.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **MedicalSubject** (базовий клас), модифікувати класи **Patient** (Пацієнт) **Doctor** (Лікар) з лабораторної роботи №3, зробивши їх похідними від **MedicalSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.17.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **MedicalSubject**, зробивши його **protected** для заборони створення об’єкту класу **MedicalSubject** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.17.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **MedicalSubject** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **MedicalSubject**, від якого успадковуються класи **Patient** (Пацієнт) і **Doctor** (Лікар). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfPatient**, який в якості поля має масив (список) об'єктів класу **Patient**. В класі **SetOfPatient** реалізувати:   * інтерфейс *IComparable* для порівняння пацієнтів за кількістю поточних скарг методом *CompareTo*(); * інтерфейс *IComparer* для порівняння пацієнтів не тільки за кількістю поточних скарг, але й за датою народження методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку пацієнтів, впорядкованих за датою народження (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **18** | **Опис предметної області**. Існує предметна область мережі користувачів, в якій сутностями моделі є узагальнена сутність **«Суб’єкт мережі**». Сутності **«Соціальна мережа»** і **«Користувач»** є нащадками узагальненої сутності **«Суб’єкт мережі**». Ці суб’єкти успадковують від узагальненої сутності **«Суб’єкт мережі**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам медичного середовища операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.18.1 подані ролі, атрибути та операції сутностей мережі користувачів.  Таблиця 4.18.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Суб’єкт мережі | Узагальнена роль для учасників взаємодії в мережі | Назва ,  Тип суб’єкта мережі (Соціальна мережа /Користувач),  Статус активності (активний / неактивний) | Ініціювати взаємодію  Отримати дані про взаємодію  Завершити взаємодію | | Соціальна мережа | Платформа для взаємодії користувачів | Тип суб’єкта мережі (успадковано як “Соціальна мережа”),  Статус активності (успадковано)  Назва мережі (успадковано як Назва))  Кількість користувачів  Спрямованість мережі (для науковців, для розваг тощо) | Додати користувача до мережі  Видалити користувача з мережі  Публікувати контент  Створити групи в мережі  Заблокувати користувача  Шукати користувачів | | Користувач | Фізична особа, яка комунікує через мережу | Тип суб’єкта мережі (успадковано як “Користувач”),  Статус активності (успадковано)  Ім’я користувача (успадковано як Назва)  e-mail  Пароль  Кількість друзів | Зареєструвати користувача  Оновити профіль  Додати/видалити контент до мережі  Додати/видалити друзів  Ініціювати пошук користувачів | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **один користувач – одна мережа**. **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (користувач і мережа) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про користувача і мережу за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **NetworkSubject** (базовий клас), модифікувати класи **SocialNetwork** (Соціальна мережа), **User** (Користувач) з лабораторної роботи №3, зробивши їх похідними від **NetworkSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.18.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **NetworkSubject**, зробивши його **protected** для заборони створення об’єкту класу **NetworkSubject** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.18.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **NetworkSubject** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **NetworkSubject**, від якого успадковуються класи **SocialNetwork** (Соціальна мережа), **User** (Користувач). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfUser**, який в якості поля має масив (список) об'єктів класу **User**. В класі **SetOfUser** реалізувати:   * інтерфейс *IComparable* для порівняння користувачів за кількістю друзів методом *CompareTo*(); * інтерфейс *IComparer* для порівняння користувачів не тільки за кількістю друзів, але й за його контентом методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку користувачів, впорядкованих за кількістю друзів (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **19** | **Опис предметної області**. Існує предметна область інфраструктура міста, в якій сутностями моделі є узагальнена сутність **«Транспортна інфраструктура**». Сутності **«Місто»** і **«Транспорт»** є нащадками узагальненої сутності **«Транспортна інфраструктура**». Ці суб’єкти успадковують від узагальненої сутності **«Транспортна інфраструктура**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам інфраструктури міста операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.19.1 подані ролі, атрибути та операції сутностей мережі користувачів.  Таблиця 4.19.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Транспортна інфраструктура | Узагальнений суб’єкт міської транспортної взаємодії | Ідентифікатор  Назва суб’єкту  Статус активності (активний / неактивний)  Тип суб’.єкту (місто / транспорт) | Ініціювати взаємодію (встановити значення статусу “активний” )  Перевірити стан суб’єкту  Завершити дію (змінити статус на “неактивний”) | | Місто | Організує транспортну систему, контролює її ефективність | Назва міста (успадковано як Назва суб’єкту)  Статус (успадковано)  Тип (успадковано як "місто")  Кількість жителів  Площа міста  Кількість транспортних одиниць | Розрахувати щільність транспорту  Оцінити навантаження на інфраструктуру | | Транспорт | Забезпечує перевезення в межах міста | Ідентифікатор (успадковано)  Назва (успадковано як назва транспорту/маршруту),  Статус активності (успадковано),  Тип (успадковано як транспорт: автобус, трамвай, метро)  Максимальна пасажиромісткість  Кількість рейсів на день | Розрахувати перевізну здатність  Оцінити ефективність маршруту | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одне місто – один тип транспорту**. **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (місто і транспорт) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про місто та транспорт за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **TransportInfrastructure** (базовий клас), модифікувати класи **City** (місто), **Тransport** (транспорт) з лабораторної роботи №3, зробивши їх похідними від **TransportInfrastructure**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.19.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **TransportInfrastructure**, зробивши його **protected** для заборони створення об’єкту класу **TransportInfrastructure** як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.19.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **TransportInfrastructure** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **TransportInfrastructure**, від якого успадковуються класи **City** (місто), **Тransport** (транспорт). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfTransport**, який в якості поля має масив (список) об'єктів класу **Тransport** . В класі **SetOfTransport** реалізувати:   * інтерфейс *IComparable* для порівняння транспорту за перевізною здатністю методом *CompareTo*(); * інтерфейс *IComparer* для порівняння транспорту не тільки за перевізною здатністю, але й за щільністю транспорту в місті методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку видів транспорту, впорядкованих за кількістю транспортних засобів (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |
| **20** | **Опис предметної області**. Існує предметна область цифрового простору, в якій сутностями моделі є узагальнена сутність **«Суб’єкт цифрового простору**». Сутності **«Молодь**» і **«Цифрова платформа»** є нащадками узагальненої сутності **«Суб’єкт цифрового простору**». Ці суб’єкти успадковують від узагальненої сутності **«Суб’єкт цифрового простору**» спільні атрибути та операції. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам цифрового простору операцій. Якщо розробник бажає використовувати меню для зручності користувача (не обов’язково), слід створити окрему технічну сутність Меню, яка не входить в предметну область. В таблиці 4.20.1 подані ролі, атрибути та операції сутностей мережі користувачів.  Таблиця 4.20.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Суб’єкт цифрового простору | Узагальнена сутність для суб’єктів цифрової взаємодії | Назва / Ім’я,  Статус активності (активний / неактивний)  Тип учасника (молодь / платформа) | Ініціювати взаємодію,  Перевірити статус,  Завершити дію | | Молодь | Учасник цифрового ринку, що створює кар’єру через платформу | Ім’я (успадковано як назва),  Статус активності (успадковано),  Тип учасника (успадковано як “ молодь”)  Вік,  Навички,  Рейтинг,  Кількість проєктів  Профіль | Додати навичку  Оцінити досвід (виконання проєкту)  Розрахувати рейтинг | | Цифрова платформа | Середовище, яке надає інструменти для навчання і роботи | Назва платформи (успадковано як назва),  Статус активності (успадковано),  Тип учасника (успадковано як “платформа”)  Кількість учасників,  Доступні проєкти,  Назва курсу | Призначити проєкт  Згенерувати звіт про активність  Алгоритм підбору проєкту  Рекомендувати курс | | Сервіс | Технічний посередник: забезпечує обмін, введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками освітнього середовища: **одна платформа – один представник молоді**. **Сервіс (Меню)** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (молодь і платформа) для реалізації своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про молодь і платформу за сценарієм роботи версії 1 лабораторної роботи №3.  **Версія 1 (пункти завдання 1,2,3,4,5,6).**   1. Скопіювати проєкт версії 1 лабораторної роботи №3 і перейменувати його для версії 1 лабораторної роботи №4. 2. Створити новий клас **SubjectDigitalSpace** (базовий клас), модифікувати класи **Youth** (молодь), **DigitalPlatform** (цифрова платформа) з лабораторної роботи №3, зробивши їх похідними від **SubjectDigitalSpace**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №3. Не додавайте в клас **Service** меню команд. При необхідності (не обов'язково) слід зробити меню як окремий клас. Кожен клас має бути створений в окремому файлі*.* В класах потрібно визначити закриті поля, що відповідають атрибутам предметної област (табл. 4.19.1). 3. Визначити конструктор базового класу (без параметрів або з параметрами) для ініціалізації полів класу **SubjectDigitalSpace**, зробивши його **protected** для заборони створення об’єкту класу **SubjectDigitalSpace**я як не релевантного предметній області. 4. Визначити конструктори похідних класів з параметрами для ініціалізації полів похідних класів з посиланнями на конструктор базового класу через ключове слово **base**. 5. Визначити методи –аксесори (властивості) для доступу до закритих полів похідних класів. 6. Визначити методи похідних класів, що демонструють операції сутностей відповідно до табл. 4.19.1. Якщо не модифікуються алгоритми методів лабораторної роботи №3, реалізація їх залишається без змін в лабораторній роботі №4.   **Версія 2 в окремому проєкті.**  7. Зробити копію проєкту версії 1 лабораторної роботи №4. Зробити базовий клас **SubjectDigitalSpace** абстрактним. *Довести, що не можна створювати об’єкти абстрактного класу, але збережена можливість успадкування*.  **Версія 3 в окремому проєкті.**  8. Зробити копію проєкту версії 2. Реалізувати трирівневу ієрархію класів у вигляді інтерфейсу, який реалізується похідним (не абстрактним) класом **SubjectDigitalSpace**, від якого успадковуються класи **Youth** (молодь), та **DigitalPlatform** (цифрова платформа). Продемонструвати доступ до методів, що реалізовані в похідних класах, через **посилання на інтерфейс**.  **Версія 4 в окремому проєкті.**  9. Зробити копію проєкту версії 1. Створити окремий клас **SetOfDigitalPlatform**, який в якості поля має масив (список) об'єктів класу **DigitalPlatformi**. В класі **SetOfDigitalPlatform** реалізувати:   * інтерфейс *IComparable* для порівняння платформ за кількістю учасників методом *CompareTo*(); * інтерфейс *IComparer* для порівняння транспорту не тільки кількістю учасників, але й за кількістю проєктіві методом *Compare*(); * інтерфейси *IEnumerable* та *IEnumerator* для виведення на консоль переліку платформ, впорядкованих за кількістю учасників (реалізувати методи *MoveNext*(), *Reset*(), *GetEnumerator*(), властивість *Current*). |

### Контрольні запитання

* 1. Що таке успадкування в C#? Які його основні переваги?
  2. Чим відрізняється клас від інтерфейсу в C#?
  3. Чи можна успадкувати клас від декількох базових класів? Обгрунтуйте?
  4. Як реалізується множинна спадковість в C#?
  5. У чому полягає різниця між abstract та interface?
  6. Які модифікатори доступу до членів класу ви знаєте? Як вони впливають на успадкування?
  7. Що означає “перевизначення методу” в контексті успадкування?
  8. Чи можна створити об’єкт інтерфейсу? Якщо ні, то навіщо вони потрібні?

### Література

1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#. КНУ ім. Тараса Шевченка
2. C# - творчість програмування. Том 1. Об’єктно-орієнтоване програмування: підручник// Під ред. Бичкова О.С. Волин. Обереги, 2024. – 292 с.
3. The C# Coding Standard. Access mode: <https://github.com/hassanhabib/CSharpCodingStandard>
4. C# Coding Standards and Best Practices. Access mode: <https://www.dofactory.com/csharp-coding-standards>
5. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І. В., Марущак П. О. – Тернопіль: ФОП Паляниця В. А., 2020 – 320 с. Режим доступу: <https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf>

## Лабораторна робота №5 Поліморфізм методів і операцій, віртуальні функції, перевантаження операторів, індексатори в мові програмування C#

### Рейтинг лабораторної роботи №5

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 5 завдань | 0,5\*7=3,5 | 10 квітня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 4,5 |  |

### Мета роботи:

1. Навчитися здійснювати перевантаження методів і операцій (поліморфізм методів і операцій) застосовувати віртуальні функції

2. Навчитися використовувати індексатори.

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекцію та методичні вказівки до цієї лабораторної роботи.

2. Реалізуйте та проаналізуйте всі приклади з лекції.

3. При роботі над лекцією і виконанні завдань зверніть увагу, що в C# є **дві** різні реалізації перевантаження методів. В першому випадку в одному класі може бути декілька методів з **однаковими іменами**. При цьому вони повинні повертати дані різного типу і/або мати різну кількість параметрів.

В другому випадку перевантаження методів використовується при спадкуванні класів через механізм віртуальних методів.

Крім перевантаження методів в C# можна створювати методи зі змінною кількістю параметрів. При цьому останнім параметром повинен бути масив.

4. Для поглибленого вивчення цього матеріалу прочитайте [2].

### Порядок виконання роботи

1. Створити директорію Lab5, в якій будуть розміщуватися проекти цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді окремих консольних проєктів – кожне завдання варіанту в окремому проєкту. Передбачити опис кожного класу в окремому файлі.

3. Можна виконати завдання свого варіанту у вигляді одного консольного проекту з окремими файлами для кожного з класів з використанням меню для демонстрації роботи кожного завдання варіанту.

### Приклади виконання завдань

Під час виконання об'єкти похідного класу можуть оброблятися як об'єкти базового класу в таких місцях, як параметри методу і колекції або масиви. Коли виникає поліморфізм, оголошений тип об'єкта перестає відповідати своєму типу під час виконання.

#### 1. Перевантаження методів

Використання декількох методів з одним і тим самим **іменем**, але різними типами **параметрів** називається *перевантаженням методів.*

Компілятор визначає, який саме метод потрібно викликати за типом фактичних параметрів. Наприклад, нижче наведено декілька реалізацій метода *max* з лекції 8, який повертає найбільше значення для різних типів і кількості параметрів.

// Повертає найбільше з двох цілих:

int max(int а, int b)

// Повертає найбільше з трьох цілих:

int max(int а, int b, int з)

// Повертає найбільше першого параметра і довжини другого:

int max(int а, string b)

// Повертає найбільше другого параметра і довжини першого:

int max(string b, int а)

...

Console.WriteLine(max( 1, 2 ));

Console.WriteLine(max( 1, 2, 3 ));

Console.WriteLine(max( 1, "2" ));

Console.WriteLine(max( "1", 2 ));

Багато методів, які реалізовані в мові C# перевантажені.

#### 2. Віртуальні члени класу

Базові класи можуть визначати і реалізовувати *віртуальні методи*, а похідні класи - перевизначати їх, тобто надавати своє власне визначення і реалізацію. Під час виконання, коли клієнт викликає метод, CLR виконує пошук типу об'єкта під час виконання і викликає перезапис віртуального методу. У вихідному коді можна викликати метод в базовому класі і забезпечити виконання версії методу, що відноситься до похідному класу.

Похідний член повинен використовувати ключове слово *override*, яке вказує, що метод призначений для участі у віртуальному виклику. Віртуальні методи і властивості дозволяють похідним класам розширювати базовий клас без необхідності використовувати реалізацію методу базового класу.

public class BaseClass

{

public virtual void DoWork() { }

public virtual int WorkProperty

{

get { return 0; }

}

}

public class DerivedClass : BaseClass

{

public override void DoWork() { }

public override int WorkProperty

{

get { return 0; }

}

}

#### 3. Приховування членів базового класу новими членами

Якщо ви хочете, щоб похідний клас мав член з тим самим іменем, що і член в базовому класі, можна використовувати ключове слово *new*, щоб приховати член базового класу. Ключове слово *new* вставляється перед типом значення, що повертається, члена класу, що заміщає.

public class BaseClass

{

public void DoWork() { WorkField++; }

public int WorkField;

public int WorkProperty

{

get { return 0; }

}

}

public class DerivedClass : BaseClass

{

public new void DoWork() { WorkField++; }

public new int WorkField;

public new int WorkProperty

{

get { return 0; }

}

}

Доступ до прихованих членам базового класу можна здійснювати з клієнтського коду приведенням примірника похідного класу до примірника базового класу

DerivedClass B = new DerivedClass();

B.DoWork(); // Calls the new method.

BaseClass A = (BaseClass)B;

A.DoWork(); // Calls the old method.

#### 4. Захист віртуальних членів від перевизначення похідними класами

Похідний клас може зупинити віртуальне успадкування, оголосивши перевизначення як *запечатаний*. Для зупинки спадкування в оголошення члена класу потрібно вставити ключове слово *sealed* перед ключовим словом *override*. Прикладом є наступний код:

public class A

{

public virtual void DoWork() { }

}

public class B : A

{

public override void DoWork() { }

}

public class C : B

{

public sealed override void DoWork() { }

}

Запечатані методи можна замінити похідними класами за допомогою ключового слова *new*, як показано в наступному прикладі:

public class D : C

{

public new void DoWork() { }

}

#### 5. Доступ до віртуальних членів базового класу з похідних класів

Похідний клас, який замінив або перевизначив метод або властивість, може отримати доступ до методу або властивості базового класу за допомогою ключового слова *base*.

public class Base

{

public virtual void DoWork() {/\*...\*/ }

}

public class Derived : Base

{

public override void DoWork()

{

//Perform Derived's work here

//...

// Call DoWork on base class

base.DoWork();

}

}

#### 6. Перевантаження операторів

Поряд з методами ми можемо також перевантажувати оператори. Перевантаження операторів тісно пов'язана з перевантаженням методів. Для перевантаження оператора служить ключове слово *operator*, що визначає операційний метод, який, в свою чергу, визначає дію оператора щодо свого класу. Існують дві форми операторних методів (*operator*): одна для унарних операторів, інша ‒ для бінарних. Нижче наведена загальна форма для кожного різновиду цих методів:

// Загальна форма перевантаження унарного оператора.

public static возвращаемый\_тип **operator** op(тип\_параметра операнд)

{

// операції

}

// Загальна форма перевантаження бінарного оператора.

public static возвращаемый\_тип **operator** op(тип\_параметра1 операнд1,

тип\_параметра2 операнд2)

{

// операції

}

##### Перевантаження бінарних операторів

Для бінарних операторів (+, -, \*, /, %, ==, !=, >, < тощо) тип значення, що повертається співпадає з класом, в якому перевантажується оператор, в якості параметрів використовуються об’єкти того самого класу, що й тип значення, що повертається. Зверніть увагу на те, що операційні методи повинні мати обидва специфікатори типу *public і static,* оскільки застосовуються до усіх об’єктів *.*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class MyArr

{

// Координати точки в тривимірному просторі

public int x, y, z;

public MyArr(int x = 0, int y = 0, int z = 0)

{

this.x = x;

this.y = y;

this.z = z;

}

// Перевантажуємо бінарний оператор +

public static MyArr operator +(MyArr obj1, MyArr obj2)

{

MyArr arr = new MyArr();

arr.x = obj1.x + obj2.x;

arr.y = obj1.y + obj2.y;

arr.z = obj1.z + obj2.z;

return arr;

}

}

class Program

{

static void Main(string[] args)

{

MyArr Point1 = new MyArr(1, 12, -4);

MyArr Point2 = new MyArr(0, -3, 18);

Console.WriteLine("Координати першої точки: " +

Point1.x + " " + Point1.y + " " + Point1.z);

Console.WriteLine("Координати другої точки: " +

Point2.x + " " + Point2.y + " " + Point2.z + "\n");

MyArr Point3 = Point1 + Point2;

Console.WriteLine("\nPoint1 + Point2 = "

+ Point3.x + " " + Point3.y + " " + Point3.z);

Console.ReadLine();

}

}

}

##### 6.2 Перевантаження унарних операторів

Унарні оператори (++, --, -) перевантажуються так само як і бінарні. Головна відмінність полягає в тому, що у них є лише один операнд. Для унарних операторів параметр позначає операнд, що передається.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class MyArr

{

// Координати точки в тривимірному просторі

public int x, y, z;

public MyArr(int x = 0, int y = 0, int z = 0)

{

this.x = x;

this.y = y;

this.z = z;

}

// Перевантажуємо унарний оператор -

public static MyArr operator -(MyArr obj1)

{

MyArr arr = new MyArr();

arr.x = -obj1.x;

arr.y = -obj1.y;

arr.z = -obj1.z;

return arr;

}

// Перевантажуємо унарний оператор ++

public static MyArr operator ++(MyArr obj1)

{

obj1.x += 1;

obj1.y += 1;

obj1.z += 1;

return obj1;

}

}

class Program

{

static void Main(string[] args)

{

MyArr Point1 = new MyArr(1, 12, -4);

Console.WriteLine("Координати точки: " +

Point1.x + " " + Point1.y + " " + Point1.z);

MyArr Point2 = -Point1;

Console.WriteLine("-Point1 = "

+ Point2.x + " " + Point2.y + " " + Point2.z);

Point2++;

Console.WriteLine("Point2++ = "

+ Point2.x + " " + Point2.y + " " + Point2.z);

Console.ReadLine();

}

}

|

#### 7. Індексатори

Індексатори дозволяють *індексувати об'єкти* і звертатися до даних за індексом. Фактично за допомогою індексаторів ми можемо працювати з об'єктами як з масивами. За формою вони нагадують властивості зі стандартними блоками *get* і *set*, які повертають і присвоюють значення. Формальне визначення індексатора:

Тип\_що\_повератється this [Тип параметр1, ...]

{

    get { ... }

    set { ... }

}

На відміну від властивостей індексатор не має назви. Замість нього вказується ключове слово *this*, після якого в квадратних дужках йдуть параметри. Індексатор повинен мати як мінімум *один параметр*.

class Person

{

public string Name { get; set; }

}

class People

{

Person[] data;

public People()

{

data = new Person[5];

}

// індексатор

public Person this[int index]

{

get

{ return data[index]; }

set

{ data[index] = value;}

}

}

Після цього ми можемо працювати з об'єктом *People* як з набором об'єктів *Person*:

class Program

{

static void Main(string[] args)

{

People people = new People();

people[0] = new Person { Name = "Tom" };

people[1] = new Person { Name = "Bob" };

Person tom = people[0];

Console.WriteLine(tom?.Name);

Console.ReadKey();

}

}

Слід враховувати, що індексатор *не може бути статичним* і застосовується тільки до *примірника класу*. Але при цьому індексатори можуть бути *віртуальними* й *абстрактними* і можуть перевизначитися в похідних класах.

### Варіанти завдань для лабораторної роботи № 5

Виконання лабораторної роботи №5 передбачає три етапи аналогічно лабораторним роботам №3, №4 (рис. 6).

**Етап 1** – об’єктно-орієнтований аналіз (OOA) предметної області з розширенням бізнес-логіки завдань лаб. роботи №4 на додаткові бізнес-задачі, що визначені в завданнях лабораторної роботи №5. Об'єктно-орієнтований аналіз дозволяє доповнити сутності предметної області лабораторної роботи №4 та їхню поведінку новими особливостями. На вході – предметні сутності версії 1 згідно із завданнями лаб. роботи №4 та опис додаткових сутностей та їх поведінки згідно із завданнями лаб. роботи №5. Результат – модель предметної області у вигляді сукупності сутностей з атрибутами та операціями, що характеризують поведінку сутностей, відповідно до принципів SOLID та опису предметної області лабораторної роботи №5.

**Етап 2** – об’єктно-орієнтоване проєктування (OOD). На вході – результат об’єктно-орієнтованої декомпозиції та аналізу (ООА). Результат – UML діаграма класів, яка містить класи і взаємозв'язки між ними та UML Use Case діаграма для визначення сценарію роботи програми.

**Етап 3** – об’єктно-орієнтоване програмування (OOP). На вході – UML діаграма класів і UML Use Case діаграма. Результат – код програми відповідно до вимог чистого коду (гарного стилю програмування) та принципів SOLID.

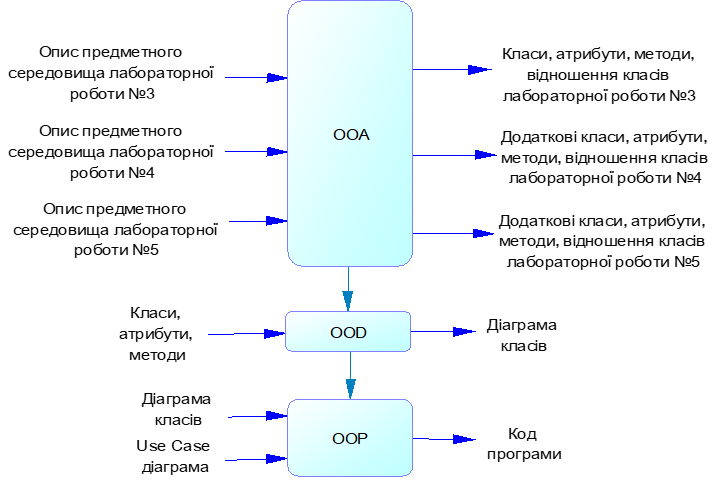


Рисунок 6 – Етапи об'єктно-орієнтованого підходу до розробки ПЗ для лабораторної роботи №5.

У цій лабораторній роботі слід дотримуватись **SOLID принципу єдиної відповідальності (SRP), принципу відкритості – закритості (OCP), принципу підстановки Лісков (LSP).**

**Принцип єдиної відповідальності (SRP)** - кожен клас має відповідати тільки за одну область функціональності і мати тільки одну задачу або відповідальність.

**Принципу відкритості – закритості (OCP) -** класи повинні бути відкритими для розширення, але закритими для модифікації. Це означає, що поведінка класу може бути змінена без зміни його вихідного коду, шляхом додавання нового коду, а не модифікації існуючого.

**Принцип підстановки Лісков (LSP) -** класи-нащадки повинні мати можливість замінити батьківські класи без порушення функціональності**.**

Виконання лабораторної роботи передбачає три версії коду, кожна версія в окремому проєкті Visual Studio.

В методі *Main*() класу *Program* продемонструвати виклик усіх методів усіх класів.

Усі значення, що розраховуються, записувати до *текстових файлів* методами класу **Service.** Також у файл записувати *протокол роботи програми*, тобто дані, що виводяться на консоль.

**Студент має право додати додаткові поля та методи в класи завдань свого варіанту**.

**Номер варіанта визначається за порядковим номером в журналі.**

|  |  |
| --- | --- |
| **Номер варіанта** | **Зміст завдання** |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є узагальнена сутність «**Людина**». Сутності «**Викладач**» і «**Студент**» є нащадками від узагальненої сутності «**Людина**» і успадковують від неї спільні атрибути та операції. Комунікація сутностей викладача і студента здійснюється через сутність «**Дипломний проєкт**» з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій, зокрема робота з консоллю та файлами. У разі потреби використання меню для зручності користувача, доцільно створити окрему сутність «**Меню»** (не обов’язково). В таблиці 5 1.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.1.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Людина | Базовий клас для всіх учасників навчального процесу | Ім’я,  Назва дисципліни | Отримати ім’я особи  Змінити ім’я  Отримати назву дисципліни  Змінити назву дисципліни | | Викладач | Джерело навчального контенту; координує дисципліну; виконує оцінювання студентів | Ім’я викладача (успадковано)  Назва дисципліни (успадковано)  Навчальне навантаження (години)  Кількість студентів | Збільшити кількість студентів  Зменшити кількість студентів  Змінити обсяг навчального навантаження  Поставити оцінку студенту  Передати матеріал студенту  Записати оцінку в журнал | | Студент | Отримувач знань і навчального контенту; виконує завдання; формує власний рейтинг | Ім’я студента (успадковано)  Назва дисципліни (успадковано)  Список робіт з оцінками  Обсяг виконаних робіт | Додати оцінку Переглянути оцінки  Розрахувати рейтинг Отримати матеріал від викладача | | Дипломний проєкт | Інтегрує знання студента й оцінюється викладачем | Назва теми,  Кількість реалізованих алгоритмів,  Складність теми.  Оцінка  Ім’я керівника | Вибрати тему з переліку.  Визначити складність теми.  Визначити оцінку | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Формат виводу  Шлях до файлу (ім’я файлу)  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Викладач**» і «Д**ипломний проєкт**» та «**Студент**» і «Д**ипломний проєкт**» асоціативний кратності один до одного. Між сутностями «**Викладач»** і«**Студент»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **один викладач** керує **одним дипломним проєктом**. **Один студент** виконує **один дипломний проєк**т. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (викладач і студент) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про викладача і студента згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **Person** (абстрактний базовий клас), **Teacher** і **Student -** похідні класи від **Person**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.1.1). *.*  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **Person** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Teacher**, **Student** як **override**.  3.  Додати клас **DiplomaProject** (Дипломний проект) з лаб. роб. №3, звязавши його асоціаціями з класами **Teacher** і **Student**. Включити в клас **DiplomaProject** атрибути і методи згідно з табл. 5.1.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Teacher**, **Student**, **DiplomaProject** статичний поліморфізм операторів, додавши методи перевантаження бінарних операторів (+, -, ==, != , >, < ): збільшити (зменшити) навчальне навантаження на задану кількість годин, збільшити (зменшити) кількість студентів на задану величину, порівняти викладачів за певними атрибутами, порівняти студентів за рейтингом (>, <), додати (видалити) задану оцінку студенту, порівняти студентів за певними атрибутами (==, !=), збільшити (зменшити) кількість реалізованих в дипломі алгоритмів, порівняти два дипломних проєктів за оцінкою. У випадку перевантаження операцій == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Teacher**, **Student, DiplomaProject** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість годин навчального навантаження викладача, збільшити (зменшити) рейтинг студента, збільшити (зменшити) рівень складності диплому тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **StudentGroup**, атрибутом якого зробити масив (список) об’єктів класу **Student**, використавши індексатори для доступу до заданих за індексом студентів.  7.  Додати в клас **DiplomaProject** конструктор копії. За допомогою конструктора копії створити копію диплома з тими самими атрибутами, що й оригінал, але без оцінки. |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є  узагальнена сутність «**Освітній суб’єкт**». Сутності «**Кафедра**» і «**Студент**» є нащадками від узагальненої сутності «**Освітній суб’єкт**». «**Кафедра**» - це організаційний освітній суб’єкт;«**Студент**» - це індивідуальний освітній суб’єкт. Ці суб’єкти успадковують від узагальненої сутності «**Освітній суб’єкт**» спільні атрибути та операції. Комунікація сутностей кафедри і студента здійснюється через сутність «**Конкурсна робота студента**» з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.1.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.2.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітній суб’єкт | Базова сутність для уніфікації обробки напряму підготовки | Ім’я сутності,  Назва напряму підготовки | Отримати ім’я  Змінити ім’я  Отримати назву напряму  Змінити назву напряму | | Кафедра | Організаційна одиниця, яка адмініструє напрям підготовки та облік студентів | Назва кафедри (успадковано як назва сутності)  Назва напряму підготовки (успадковано)  Поточна кількість студентів  Макс. кількість студентів  Перелік дисциплін за напрямом підготовки | Збільшити кількість студентів  Зменшити кількість студентів  Додати дисципліну (за напрямом)  Видалити дисципліну | | Студент | Учасник освітнього процесу, чия успішність вимірюється на основі оцінок | Ім’я студента (успадковано як назва сутності)  Назва напряму підготовки (успадковано)  Список оцінок  Рівень навантаження. | Додати оцінку з дисципліни  Переглянути оцінки  Розрахувати рейтинг | | Конкурсна робота студента | Творчий результат студента, який оцінюється в межах конкурсу і відображає індивідуальну активність | Назва роботи  Тематика конкурсу  Дата подання  Оцінка / рейтинг  Статус (подано, прийнято, відхилено)  Призове місце  Автор (студент) | Подати роботу  Визначити відповідність теми роботи тематиці конкурсу  Оцінити роботу  Призначити статус  Присвоїти місце  Переглянути деталі | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Формат виводу  Шлях до файлу (ім’я файлу)  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями **«Кафедра»** і «**Конкурсна робота**» та **«Студент»** і «**Конкурсна робота**» асоціативний кратності один до одного. Між сутностями **«Кафедра»** і«**Студент»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра** керує виконанням **однієї конкурсної роботи**. **Один студент** виконує **одну конкурсну роботу**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (викладач і студент) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру, студента і конкурсну роботу згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **EducationalEntity** (абстрактний базовий клас), **Department** і **Student -** похідні класи від **EducationalEntity**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.2.1). *.*  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **Person** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Department**, **Student** як **override**.  3.  Додати клас **СompetitiveWork** (Конкурсна робота) з лаб. роб. №3, звязавши його асоціаціями з класами **Department** і **Student**. Включити в клас **СompetitiveWork** атрибути і методи згідно з табл. 5.1.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Department**, **Student, СompetitiveWork** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) кількість дисциплін на задану величину, збільшити (зменшити) кількість студентів на задану величину, порівняти кафедри за певними атрибутами, порівняти студентів за рейтингом (>, <), додати (видалити) задану оцінку студенту, порівняти студентів за певними атрибутами (==, !=), збільшити (зменшити) рейтинг конкурсної роботи, порівняти дві конкурсні роботи за номером призового місця. У випадку перевантаження операцій == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Department**, **Student, СompetitiveWork** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість студентів на кафедрі, збільшити (зменшити) рейтинг студента, підняти (опустити) конкурсну роботу за призовим місцем тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **StudentGroup**, атрибутом якого зробити масив (список) об’єктів класу **Student**, використавши індексатори для доступу до заданих за індексом студентів.  7.  Додати в клас **СompetitiveWork** конструктор копії. За допомогою конструктора копії створити копію конкурсної роботи з тими самими атрибутами, що й оригінал, але без оцінки та призового місця. |
|  | **Опис предметної області**.  Існує освітнє середовище, в якому учасниками освітнього процесу є  узагальнена сутність «**Освітня одиниця**». Сутності «**Кафедра**» і «**Викладач**» є нащадками від узагальненої сутності «**Освітня одиниця**». «**Кафедра**» - це організаційний освітній суб’єкт;«**Викладач**» - це індивідуальний освітній суб’єкт. Ці суб’єкти успадковують від узагальненої сутності «**Освітня одиниця**» спільні атрибути та операції. Комунікація сутностей кафедри і викладач здійснюється через сутність «**Конкурсний проєкт викладача** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.3.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.3.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітня одиниця | Базова сутність для уніфікації обробки даних про викладача | Назва одиниці  Загальне річне навантаження (години)  Перелік дисциплін | Отримати назву  Змінити назву  Додати / видалити дисципліну  Отримати / оновити загальне навантаження | | Кафедра | Організаційна одиниця, яка керує ресурсами, планує і розподіляє навантаження | Назва кафедри (успадковано як назва одиниці)  Загальне річне навантаження (успадковано)  Перелік дисциплін (успадковано)  Поточна кількість викладачів  Список викладачів | Додати викладача (з перевіркою, що його навантаження не перевищує 600 годин)  Видалити викладача  Розподілити загальне навантаження між викладачами  Переглянути розподіл | | Викладач | Виконує навчальну, наукову та методичну діяльність | ПІБ (успадковано як назва)  Загальне річне навантаження (успадковано)  Спеціалізація викладача  Кількість дисциплін  К-сть наукових / методичних / орг. робіт | Додати наукову або методичну роботу  Перевірити перевищення допустимого навантаження (норма 600 год)  Визначити рейтинг за продуктивністю | | Конкурсний проєкт викладача | Відображає процес супроводу та оцінки конкурсного проєкту | Назва конкурсного проєкту  Тема конкурсу  Статус (прийнята / відхилена)  Рейтинг роботи  Призове місце, отримане конкурсною роботою (1,2, 3). Значення 0 - робота не попала в номінацію призових.  Викладач-виконавець | Подати роботу  Визначити відповідність теми роботи тематиці конкурсу  Оцінити роботу  Призначити статус  Присвоїти місце  Переглянути деталі | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу)  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями **«Кафедра»** і «**Конкурсний проєкт**» та «**Викладач**»і «**Конкурсний проєкт**» асоціативний кратності один до одного. Між сутностями **«Кафедра»** і«**Викладач»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра** керує виконанням **одного конкурсного проєкту**. **Один викладач** виконує **один конкурсний проєкт**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (викладач і кафедра) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру, викладача і конкурсний проєкт згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **EducationalUnit** (абстрактний базовий клас), **Department** і **Teacher -** похідні класи від **EducationalUnit**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.3.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **EducationalUnit** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Department**, **Teacher** як **override**.  3.  Додати клас **ContestWorkManage**r (Конкурсна робота) з лаб. роб. №3, звязавши його асоціаціями з класами **Department** і **Teacher**. Включити в клас **ContestWorkManage**r атрибути і методи згідно з табл. 5.3.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Department**, **Teacher, ContestWorkManage**r статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) кількість дисциплін на задану величину, збільшити (зменшити) кількість річне навантаження на задану величину, порівняти кафедри за певними атрибутами, порівняти викладачів за рейтингом (>, <), додати (видалити) кількість дисциплін у викладача, порівняти викладачів за певними атрибутами (==, !=), збільшити (зменшити) рейтинг конкурсного проєкту, порівняти два конкурсних проєкти за номером призового місця. У випадку перевантаження операцій == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Department**, **Teacher, ContestWorkManage**r статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість дисциплін на кафедрі, збільшити (зменшити) рейтинг викладача, підняти (опустити) конкурсний проєкт за призовим місцем тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **CatalogTeacher**, атрибутом якого зробити масив (список) об’єктів класу **Teacher**, використавши індексатори для доступу до заданих за індексом викладачів.  7. Додати в клас **ContestWorkManage**r конструктор копії. За допомогою конструктора копії створити копію конкурсного проєкту з тими самими атрибутами, що й оригінал, але без оцінки (рейтингу) та призового місця. |
|  | **Опис предметної області**. Існує освітнє середовище, в якому учасниками освітнього процесу є  узагальнена сутність «**Організаційна структур**а». Сутності «**ІТ-компанія**» і «**Факультет**» є нащадками від узагальненої сутності «**Організаційна структур**а». Ці суб’єкти успадковують від узагальненої сутності «**Організаційна структура**» спільні атрибути та операції. Комунікація сутностей  «**ІТ-компанія**» і «**Факультет**» здійснюється через сутність **«Кафедра»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.4.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.4.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Організаційна структура | Базова сутність для суб’єктів взаємодії у сфері підготовки фахівців | Назва структури  Сфера або напрям діяльності  Якість процесу  Партнерська структура  Перелік напрямів або кафедр | Змінити назву  Змінити якість  Підтвердити/змінити партнерство  Додати напрям або кафедру | | ІТ-компанія | Комерційна структура, яка створює або інтегрує кафедри для покращення практичної підготовки | Сфера діяльності (успадковано)  Якість практичної підготовки (успадковано від якості процесу)  Закріплена кафедра  Партнерський факультет | Закріпити кафедру за напрямом (сферою діяльності)  Погодити співпрацю з факультетом  Покращити якість практичної підготовки | | Факультет | Академічна одиниця, що бере участь у підготовці фахівців спільно з ІТ-компаніями | Успадковані атрибути:  Назва факультету  Перелік напрямів  Якість освітнього процесу  Кількість кафедр Загальне річне навантаження Партнерська ІТ-компанія | Затвердити напрями Визначити річний обсяг годин Розподілити навантаження між кафедрами Погодити співпрацю з ІТ-компанією Контролювати якість освіти | | Кафедра | Внутрішній підрозділ факультету, що реалізує навчальне навантаження у співпраці з факультетом та співпрацює з ІТ-компанією | Назва кафедри  Спеціалізація  Завідувач кафедри  Кількість викладачів  Перелік дисциплін з обсягом годин  Річне навчальне навантаження  Партнерський факультет | Сформувати навчальний план (перелік дисциплін та їх години)  Призначити викладачів  Погодити напрям навчання | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу)  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями **«ІТ-компані**я**»** і **«Кафедра»** та **«Факультет»** і **«Кафедра»**  асоціативний кратності один до одного. Між сутностями **«ІТ-компані**я**»** і **«Факультет»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **одна ІТ-компанія** взаємодіє з **одним факультетом**. **Один факультет** має **одну кафедру**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (ІТ-компанія, факультет і кафедра) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про ІТ-компанію, факультет, кафедру згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **OrganizationalStructure** (абстрактний базовий клас), **Faculty** і **ITCompany -** похідні класи від **OrganizationalStructure** . Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.4.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **OrganizationalStructure** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Faculty** та  **ITCompany** як **override**.  3.  Додати клас **Department** (Кафедра) з лаб. роб. №3, звязавши його асоціаціями з класами **Faculty** і **ITCompany**. Включити в клас **Department** атрибути і методи згідно з табл. 5.4.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Faculty,** **ITCompany** та **Department** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) якість підготовки на задану величину (відсоток), збільшити (зменшити) кількість кафедр на задану величину, порівняти два факультети за певними атрибутами, порівняти кафедри за кількістю викладачів (>, <), додати (видалити) кількість викладачів, порівняти кафедри за певними атрибутами (==, !=), збільшити (зменшити) загальне річне навантаження. У випадку перевантаження операцій == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Faculty,** **ITCompany** та **Department** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість викладачів на кафедрі, збільшити (зменшити) річне навантаження на факультеті, покращити (погіршити) якість підготовки в ІТ-компанії тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас SetOf**Departments**, атрибутом якого зробити масив (список) об’єктів класу **Department**, використавши індексатори для доступу до заданих за індексом факульттетів.  7. Додати в клас **Department** конструктор копії. За допомогою конструктора копії створити копію кафедри з тими самими атрибутами, що й оригінал, але без річного навантаження |
| **5** | **Опис предметної області**. Існує інституційно-адміністративна модель, в якій сутностями моделі є узагальнена сутність «**Академічна установа**». Сутності «**Університет**» і «**Факультет**» є нащадками узагальненої сутності «**Академічна установа**». Ці суб’єкти успадковують від узагальненої сутності «**Академічна установа**» спільні атрибути та операції. Комунікація сутностей «**Університет**» і «**Факультет**» здійснюється через сутність **«Стартап-Інкубатор»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.5.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.5.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Академічна установа | Базова сутність для всіх освітніх організаційних одиниць | Назва установи Кількість студентів Кількість підрозділів (кафедр/факультетів) Список спеціальностей або факультетів | Змінити назву Змінити кількість студентів Змінити кількість підрозділів Додати / видалити спеціальність або підрозділ | | Університет | Головна академічна установа, що координує факультети, формує рейтинг і бюджет | Назва університету (успадковано) К-сть факультетів (успадковано) К-сть студентів (успадковано)  К-сть наукових праць К-сть міжнародних проєктів Середній бал ЗНО вступників  Рейтинг (Osvita.ua)  Річний бюджет | Розрахувати рейтинг за версією Osvita.ua Розрахувати бюджет на основі рейтингу Додати / вилучити факультет | | Факультет | Академічна одиниця, що бере участь у підготовці фахівців спільно з ІТ-компаніями | Назва факультету (успадковано) Кількість студентів (успадковано)  Кількість кафедр Кількість спеціальностей Пов’язаний університет | Збільшити / зменшити кількість кафедр Додати / видалити спеціальність Змінити кількість студентів | | Стартап-Інкубатор | Інституція підтримки інноваційної активності студентів | Назва інкубатора,  Кількість менторів, Кількість активних проєктів,  Кількість залучених студентів,  Список стартапів  Обсяг інвестицій, які інкубатор отримає для впровадження своїх старапів | Додати студентський стартап  Призначити ментора  Оцінити результативність  Розрахувати рейтинг результативності  Вибрати найкращий стартап проєкт. | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу)  Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями **«Університет»** і **«Стартап-Інкубатор»** та **«Факультет»** і **«Стартап-Інкубатор»** асоціативний кратності один до одного. Між сутностями **«Університет»** і **«Факультет»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **один уніерситет** взаємодіє з **одним факультетом**. **Один університет** має **один стартап-інкубатор**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (ІТ-компанія, факультет і кафедра) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про університет, факультет, стартап-інкубатор згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **AcademicInstitution** (абстрактний базовий клас), **University** та **Faculty -** похідні класи від **AcademicInstitution**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.5.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **AcademicInstitution** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **University** та **Faculty** як **override**.  3.  Додати клас **StartupIncubator** (Стартап Інкубатор) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **University** та **Faculty**. Включити в клас **StartupIncubator** атрибути і методи згідно з табл. 5.5.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **University,** **Faculty** та **StartupIncubator** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) кількість факультетів в університеті на задану величину, збільшити (зменшити) кількість кафедр на факультетів на задану величину, порівняти два університети за рейтингом (Osvita.ua) (>,<), порівняти факультети за кількістю ствудентів (>, <), додати (видалити) кількість міжнародних проєктв в університеті, порівняти стартап інкубатори за кількістю проєктів (==, !=), збільшити (зменшити) обсяг івестицій в стартап проєкти. У випадку перевантаження операцій == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **University,** **Faculty** та **StartupIncubator** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) річний бюджет університету, збільшити (зменшити) кількість спеціальностей на факультеті, збільшити (зменшити) кількість залучених студентів у стартапи тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfUniversity**, атрибутом якого зробити масив (список) об’єктів класу **University**, використавши індексатори для доступу до заданих за індексом університетів.  7. Додати в клас **StartupIncubator** конструктор копії. За допомогою конструктора копії створити копію стртап інкубатора з тими самими атрибутами, що й оригінал, але без обсягу інвестицій і кількістю стартап проєктів |
| **6** | **Опис предметної області**. Існує освітнє середовище, в якому сутностями моделі є узагальнена сутність «**Функціональна структура**». Сутності «**Кафедра**» і **«ІТ-компанія»** є нащадками узагальненої сутності «**Функціональна структура**». Ці суб’єкти успадковують від узагальненої сутності «**Функціональна структура**» спільні атрибути та операції. Комунікація сутностей «**Кафедра**» і **«ІТ-компанія»** здійснюється через сутність **«ІТ акселератор»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.6.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.6.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Функціональна структура | Базова сутність, що виконує проєкти, взаємодіє з фахівцями, формує якість і вплив | Назва структури Кількість працівників Кількість реальних проєктів | Змінити назву Додати працівника Збільшити кількість проєктів | | Кафедра | Освітній підрозділ ЗВО | Назва (успадковано) Кількість викладачів як працівників (успадковано) Кількість дисциплін Кількість креативних студентів Якість освітньої програми | Покращити якість освітньої програми Залучити фахівців з компанії Збільшити кількість креативних студентів | | ІТ-компанія | Роботодавець, партнер кафедри, джерело практичного досвіду | Назва (успадковано) Кількість працівників (успадковано) Кількість реальних проєктів (успадковано)  Прибуток Кількість найнятих випускників | Найняти випускників Збільшити прибуток Збільшити проєктну активність | | ІТ акселератор | Сприяти розвитку стартапів, залучати студентів та випускників до підприємницької діяльності | Назва  Кількість проектів, участь в яких беруть викладачі і студенти;  Кількість студентів, які навчаються в акселераторі;  Обсяг інвестицій, які акселератор передає на кафедру.  Рівень інноваційності | Додати студентський стартап  Призначити ментора  Оцінити результативність стартапу  Розрахувати рейтинг студентів для конкурсу в акселератор;  Вибрати найкращий стартап проєкт. | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями **«Кафедра»** і **«ІТ акселератор»** та **«ІТ-компанія»** і **«ІТ акселератор»** асоціативний кратності один до одного. Між сутностями **«Кафедра»** і **«ІТ-компанія»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **одна кафедра** взаємодіє з **одною ІТ-компанєю**. **Одна компанія** має **один ІТ акселератор**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (ІТ-компанія, факультет і кафедра) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про кафедру, компанію, ІТ акселератор згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **FunctionalStructure** (абстрактний базовий клас), **Department** та **ITCompany -** похідні класи від **FunctionalStructure**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.6.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **FunctionalStructure** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Department** та **ITCompany** як **override**.  3.  Додати клас **ITAccelerator** (ІТ акселератор) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Department** та **ITCompany**. Включити в клас **ITAccelerator** атрибути і методи згідно з табл. 5.6.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Department,** **ITCompany** та **ITAccelerator** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) кількість викладачів на кафедрі на задану величину, збільшити (зменшити) кількість реальних проєктів в компанії на задану величину, порівняти дві кафедри за кількістю креативних студентіa (>,<), порівняти дві компанії за прибутком (>, <), додати (видалити) дисципліни на кафедрі, порівняти ІТ акселератори за обсягом інвестицій (==, !=), збільшити (зменшити) кількість студентів, що навчаються в акселераторі. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Department,** **ITCompany** та **ITAccelerator** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість дисциплін на кафедрі, збільшити (зменшити) кількість найнятих випускників в компанії, збільшити (зменшити) Кількість проектів в акселераторі тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfCompanies**, атрибутом якого зробити масив (список) об’єктів класу **ITCompany**, використавши індексатори для доступу до заданих за індексом компаній.  7. Додати в клас **ITAccelerator** конструктор копії. За допомогою конструктора копії створити копію ІТ акселераторів з тими самими атрибутами, що й оригінал, але без обсягу інвестицій і кількістю проєктів |
| **7** | **Опис предметної області**. Існує предметна область у вигляді освітнього середовища, в якому сутностями моделі є узагальнена сутність «**Освітній елемент**». Сутності «**Студент**» і «**Навчальна програма**» є нащадками узагальненої сутності «**Освітній елемент**». Ці суб’єкти успадковують від узагальненої сутності «**Освітній елемент**» спільні атрибути (Назва) та операції. Оскільки дві похідні сутності різнопланові — одна репрезентує *людину* , інша — *освітній продукт/структуру*, вводиться додатковий базовий атрибут «**Тип освітнього елемента**», значенням якого є «людина» або «структура». Комунікація сутностей «**Студент**» і «**Навчальна програма**» здійснюється через сутність **«Робота в ІТ-компанії»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.7.1 подані ролі, атрибути та операції сутностей освітньогго процесу.  Таблиця 5.7.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Освітній елемент | Базова сутність, що узагальнює освітні елементи | Назва освітнього елемента  Тип освітнього елемента (людина/структура) | Отримати назву  Змінити назву  Отримати тип Змінити тип | | Студент | Похідна сутність - учасник освітнього процесу, який навчається за програмою | ПІБ (успадковано як назва)  Тип (успадковано як людина)  Список оцінок (0–10),  Рейтинг (середній бал),  Розмір стипендії,  Назва навчальної програми | Розрахувати рейтинг  Розрахувати стипендію  Показати слабкі місця у навчанні за аналізом оцінок | | Навчальна програма | Похідна сутність - освітній маршрут студента | Назва програми (успадковано як назва),  Тип (успадковано як структура)  Кількість кредитів,  Список дисциплін,  Перелік результатів навчання,  Рівень (бакалавр, магістр),  Якість навчальної програми (оцінка) | Визначити якість навчальної програми  Визначити складність навчальної програми  Оцінити відповідність навчальної програми ринку праці | | Робота в ІТ-компанії | Професійна активність студента поза університетом | Назва компанії  Кількість проектів, які виконує студент;  Обсяг освітньої програми, яка збігається за змістом з роботою студента. | Розрахувати обсяг навчальної програми, який можна зарахувати студенту як результат його роботи в компанії в процесі виконання виробничих проектів;  Сформувати рекомендації щодо удосконалення індивідуальної навчального програми для наближення її до профілю роботи студента.  Оцінити вплив на професійну підготовку. | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Студент**» і **«Робота в ІТ-компанії» та** «**Навчальна програма**» і **«Робота в ІТ-компанії»** асоціативний кратності один до одного. Між сутностями **«Студент»** і **«Навчальна програма»** асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **один студент** навчається **по** **одній навчальній програмі**. **Один студент** працює **в** **одній компанії**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (ІТ-компанія, факультет і кафедра) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента, компанію, навчальну програму згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **EducationalElement** (абстрактний базовий клас), **Student** та **Curriculum -** похідні класи від **EducationalElement**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.6.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **EducationalElement** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Student** та **Curriculum** як **override**.  3.  Додати клас **WorkInCompany** (Робота в компанії) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Student** та **Curriculum**. Включити в клас **WorkInCompany** атрибути і методи згідно з табл. 5.7.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Student,** **Curriculum** та **WorkInCompany** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) рейтинг студента на задану величину, збільшити (зменшити) кількість дисциплін в навчальній програмі на задану величину, порівняти двох студентів за розміром стипендії (>,<), порівняти дві навчальні програми за кількістю кредитів (>, <), додати (видалити) оцінки студенту, порівняти роботу в ІТ-компанії за кількістю проєктів (==, !=), збільшити (зменшити) кількість проектів, які виконує студент. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Student,** **Curriculum** та **WorkInCompany** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість дисциплін навчальної програмиі, збільшити (зменшити) обсяг освітньої програми, яка збігається за змістом з роботою студента, збільшити (зменшити) рейтинг студента тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfCurriculum**, атрибутом якого зробити масив (список) об’єктів класу **Curriculum**, використавши індексатори для доступу до заданих за індексом наввчальних програм.  7. Додати в клас **Curriculum** конструктор копії. За допомогою конструктора копії створити копію навчальної програми з тими самими атрибутами, що й оригінал, але без списку дисциплін та переліку результатів навчання |
| **8** | **Опис предметної області**. Існує предметна область у вигляді банківської мережі, в якій сутностями моделі є узагальнена сутність «**Банківська мережа**». Сутності «**Клієнт**» і «**Банкомат**» є нащадками узагальненої сутності «**Банківська мережа**». Ці суб’єкти успадковують від узагальненої сутності «**Банківська мережа**» спільні атрибути та операції. Комунікація сутностей «**Клієнт**» і «**Банкомат**» здійснюється через сутність **«Банк»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам освітнього середовища операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.8.1 подані ролі, атрибути та операції сутностей банківської мережі.  Таблиця 5.8.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Банківська мережа | Узагальнена сутність, що представляє взаємодію об'єктів банківської системи | Ідентифікатор об’єкта в мережі,  Статус активності | Авторизувати,  Завершити сесію | | Клієнт | Власник грошей, користувач банкомата | Ідентифікатор клієнта (успадковано),  Статус активності (успадковано як «активний», «блокований», «неавторизований»)  Ім’я, Прізвище,  Номер картки,  PIN-код,  Баланс | Перевірити PIN-код,  Зняти кошти,  Поповнити рахунок,  Перевірити баланс | | Банкомат | Спеціальний пристрій, за допомогою якого можна керувати готівкою зі свого банківського рахунку | Ідентифікатор банкомата (успадковано),  Статус активності (успадковано як «працює», «вимкнений», «немає готівки»)  Локація,  Доступні кошти (сума готівки, яка є в банкоматі),  Стан (працює / вимкнений / немає готівки) | Прийняти карту клієнта,  Видати готівку,  Показати залишок,  Заблокувати банківську картку через неправильний пін-код  Прийняти готівку | | Банк | Установа, що володіє банкоматом та обслуговує клієнта | Назва банку,  Адреса  Ідентифікатори клієнта  Кількість грошей на рахунку клієнта  Рейтинг банку | Оновити баланс клієнта  Збільшити, зменшити кількість грошей на рахунку клієнта;  Переказ грошей з картки клієнта, зокрема на оплата комунальних платежів з картки клієнта. | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Клієнт**» і **«Банк»** та«**Банкомат**» і **«Банк»** асоціативний кратності один до одного. Між сутностями «**Клієнт**» і «**Банкомат**»асоціацій немає. Спрощена взаємодія між учасниками освітнього середовища: **один клієнт** користується **одним банкоматом**. **Один клієнт** користується **одним банком**. **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (клієнт, банкомат, банк) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта, банкомат і банк згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **BankingNetwork** (абстрактний базовий клас), **Client,** **ATM -** похідні класи від **BankingNetwork**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.8.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **BankingNetwork** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Client,** **ATM** як **override**.  3.  Додати клас **Bank** (Банк) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Client,** **ATM**. Включити в клас **Bank** атрибути і методи згідно з табл. 5.8.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Client,** **ATM** та **Bank** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, != , >, < ): збільшити (зменшити) баланс клієнта на задану величину, збільшити (зменшити) доступні кошти в банкоматі на задану величину, порівняти двох клієнтів за розміром баланса (>,<), порівняти два банкомати за статусом активності (>, <), додати (видалити) кошти у клієнта, порівняти банки за кількістю грошей на рахунку клієнта; (==, !=), збільшити (зменшити) рейтинг банку. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Student,** **Curriculum** та **WorkInCompany** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) баланс клієнта, збільшити (зменшити) обсяг готівки в банкоматі збільшити (зменшити) рейтинг банку тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **ATMNetwork**, атрибутом якого зробити масив (список) об’єктів класу **ATM**, використавши індексатори для доступу до заданих за індексом банкоматів.  7. Додати в клас **ATM** конструктор копії. За допомогою конструктора копії створити копію банкомата з тими самими атрибутами, що й оригінал, але без обсягу готівки. |
| **9** | **Опис предметної області**.  Існує предметна область у вигляді освітнього середовища, в якому сутностями моделі є узагальнена сутність «**Самоврядування**». Сутності «**Студент**» і «**Студентський парламент**» є нащадками узагальненої сутності «**Самоврядування**». Ці суб’єкти успадковують від узагальненої сутності «**Самоврядування**» спільні атрибути (назва, статус тощо) та операції.. Комунікація сутностей «**Студент**» і «**Студентський парламент**» здійснюється через сутність **«Студентська рада студмістечка»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам самоврядування операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.9.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.9.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Самоврядування | Узагальнена сутність як спільнота студентів, що організовують студентське життя | Назва або ПІБ,  Рівень участі,  Статус активності (бажаючий, кандидат, член парламенту) | Оновити статус, Подати/опрацювати ініціативу/пропозицію | | Студент | Учасник освітнього процесу, який може бути членом студ. парламенту | *Успадковано*:  ПІБ студента  Рівень участі – індивідуальний учасник  Статус активності .  *Власні атрибути:* Курс,  Рейтинг (середній бал),  Стипендія  Ідеї студента | Розрахувати рейтинг  Подати заявку на участь у парламенті  Взяти участь у голосуванні  Ініціювати пропозицію / ідею  Розрахувати розмір стипендії | | Студентський парламент | Представницький орган самоврядування | *Успадковано*:  Назва парламенту,  Рівень участі,  Статус активності,  *Власні атрибути*:  Склад (кількість членів) парламенту (назви департаментів),  Назва  проєкту  Кількість проєктів  Обсяг стипенд. фонду | Прийняти нового члена  Взяти участь у голосуванні  Оприлюднити проєкт / ініціативу  Формувати рекомендації щодо заохочення  студентів або їх відрахування  Призначити стипендії студентам.  Відхилити заявку  Оприлюднити результати голосування | | Студентська рада студмістечка | Орган студентського самоврядування, відповідальний за побут і проживання | Назва ради,  Склад,  Адреса гуртожитку,  Кількість мешканців,  Плани благоустрою,  Контактні особи | Розглядати звернення мешканців гуртожитку  Перевірити дотримання умов проживання студентів.  Організувати захід у гуртожитку  Формувати списки студентів на поселення у гуртожиток  Реагувати на порушення умов проживання  (виселення, догана тощо). | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Студент**» і **«Студентська рада студмістечка»** та«**Студентський парламент**» і **«Студентська рада студмістечка»** асоціативний кратності один до одного. Між сутностями «**Студент**» і «**Студентський парламент**»асоціацій немає. Спрощена взаємодія між учасниками самоврядування: **один студент** може бути членом **одного студентського парламенту**. **Один студент** може бути членом **одної студентської ради студмістечка**. **Сервіс** не є учасником процесу самоврядування, є лише інструментом, який використовують інші сутності (студент, студ. парламент) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класмістечкаів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента, студ. парламент, студ. раду згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **SelfGovernment** (абстрактний базовий клас), **Student,** **Student**P**arliament** **-** похідні класи від **SelfGovernment**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.9.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **SelfGovernment** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Student,** **Student**P**arliament** як **override**.  3.  Додати клас **StudCouncilCampus** (Студентська Рада студмістечка) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Student,** **Student**P**arliament**. Включити в клас **StudCouncilCampus** атрибути і методи згідно з табл. 5.9.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Student,** **Student**P**arliament** та **StudCouncilCampus** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) рейтинг студента на задану величину, збільшити (зменшити) кількість членів студ. парламенту на задану величину, порівняти двох студентів за розміром стипендії (>,<), порівняти дві студ. ради студмістечка за кількістю мешканціві (>, <), додати (видалити) стипендію студенту, порівняти студ. парламенти за обсягом стипенд. фонду; (==, !=), збільшити (зменшити) кількість проєктів в студ. парламенті. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Student,** **Student**P**arliament** та **StudCouncilCampus** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) стипендію студента, збільшити (зменшити) обсяг стипендіального фонду збільшити (зменшити) кількість мешканців в студ. містечку тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **StudCouncilCampusNetwork**, атрибутом якого зробити масив (список) об’єктів класу **StudCouncilCampus**, використавши індексатори для доступу до заданих за індексом студ. ради студмістечка.  7. Додати в клас **Student**P**arliament** конструктор копії. За допомогою конструктора копії створити копію студенського парламенту з тими самими атрибутами, що й оригінал, але без кількості проєктів та обсягу стипенд. фонду. |
| **10** | **Опис предметної області**. Існує предметна область у вигляді транспортної логістики, в якій сутностями моделі є узагальнена сутність «**Об’єкт транспортної системи**». Сутності «**Водій**» і «**Автомобіль**» є нащадками узагальненої сутності «**Об’єкт транспортної системи**». Ці суб’єкти успадковують від узагальненої сутності «**Об’єкт транспортної системи**» спільні атрибути (ID, Назва тощо) та операції. Комунікація сутностей «**Водій**» і «**Автомобіль**» здійснюється через сутність **«Двигун»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам транспортної системи операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.10.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.10.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Об’єкт транспортної системи | Узагальнена сутність яка забезпечує рух, керування, контроль або обслуговування. | ID, Назва (ідентифікатор)   * Тип об’єкта *(водій / транспортний засіб)* * Статус активності *(активний, не допущений, неактивний)* * Поточний стан *(нормальний / критичний / потребує уваги)* | Оновити статус Надіслати повідомлення про інцидент Зареєструвати об'єкт Ініціювати перевірку стану | | Водій | Керує автомобілем, відповідає за його стан і рух | *Успадковано:*  Ім’я, Прізвище,  Стан водія  *Власні атрибути:*  Номер посвідчення,  Категорія прав,  Стаж керування автомобілем,  Маршрут руху. | Вчинити порушення ПДР  Повідомити про ДТП або технічну несправність  Заявити про потребу заміни авто  Контроль фізичного стану водія | | Автомобіль | Об'єкт керування і обліку | *Успадковано:*  Модель,  Номер авто,  Технічний стан,  Статус,  *Власні атрибути:*  Рік випуску,  Ступінь пошкодження | Пройти технічний огляд  Пройти реєстрацію  Поламатися внаслідок ДТП  Розрахувати вартість ремонту  Бути заміненим | | Двигун | Центральний технічний компонент автомобіля | Тип,  Об’єм,  Потужність,  Стан,  Пробіг,  Температура,  Середні витрати пального | Розрахувати витрати пального на задану довжину маршруту, якість дороги, клімат.  Розрахувати тривалість маршруту з урахуванням заданої швидкості, запасу пального, параметрів двигуна.  Запустити двигун.  Перевірити температуру.  Вимкнути двигун. | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Водій**» і **«Двигун»** та«**Автомобіль**» і **«Двигун»** асоціативний кратності один до одного. Між сутностями «**Водій**» і «**Автомобіль**»асоціацій немає. Спрощена взаємодія між учасниками транспортної логістики: **один водій** керує **одним автомобілем**. **Один автомобіль** має **один двигун**. **Сервіс** не є учасником логістичного процесу, є лише інструментом, який використовують інші сутності (водій, автомобіль, діигун) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класмістечкаів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про водія, автомобіль, двигун згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **TransportSystemObject** (абстрактний базовий клас), **Driver,** **Car -** похідні класи від **TransportSystemObject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.10.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **TransportSystemObject** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Driver,** **Car** як **override**.  3.  Додати клас **Engine** (Двигун) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Driver,** **Car**. Включити в клас **Engine** атрибути і методи згідно з табл. 5.10.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  **4.** Реалізувати в класах **Driver,** **Car** та **Engine** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) стаж керування водієм автомобіля на задану величину, збільшити (зменшити) рік випуску автомобіля на задану величину, порівняти двох водіїв за стажем керуання автомобілем (>,<), порівняти два автомобіля за ступенем пошкодження (>, <), додати (видалити) витрати пального в класі двигуна, порівняти двигуни за величиною пробігу; (==, !=), збільшити (зменшити) потужність двигуна. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Driver,** **Car** та **Engine** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість ДТП, вчинених водієм, збільшити (зменшити) ступінь пошкодження автомобіля, збільшити (зменшити) об’єм двигуна тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfCar**, атрибутом якого зробити масив (список) об’єктів класу **Car**, використавши індексатори для доступу до заданих за індексом автомобілів.  7. Додати в клас **Engine** конструктор копії. За допомогою конструктора копії створити копію двигуна з тими самими атрибутами, що й оригінал, але без кількості величини пробігу та витрати пального |
| **11** | **Опис предметної області**. Існує предметна область у вигляді наукової роботи, в якій сутностями моделі є узагальнена сутність «**Об’єкт конференції**». Сутності «**Конференція**» і «**Автор доповіді**» є нащадками узагальненої сутності «**Об’єкт конференції**». Ці суб’єкти успадковують від узагальненої сутності «**Об’єкт конференції**» спільні атрибути (ID, Назва тощо) та операції. Комунікація сутностей «**Конференція**» і «**Автор доповіді**» здійснюється через сутність **«Стаття»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам конференції операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.11.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.11.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Об’єкт конференції | Узагальнена сутність яка забезпечує контроль і керування конференцією та авторами. | ID,  Назва,  Статус,  Дата події / подачі | Оновити статус,  Перевірити на відповідність критеріям,  Ініціювати комунікацію | | Конференція | Захід для представлення наукових доповідей | ID (успадковано)  Назва конференції (успадковано як назва),  Дата проведення (успадковано як Дата події),  Статус (активна/архів) (успадкована як Статус)  Місце,  Напрямки  конференції,  Гранична дата подачі | Реєструвати учасників  Отримати доповідь  Рецензувати доповідь  Відхилити доповідь  Прийняти доповідь  Друкувати доповідь  Перевірити  відповідність теми напрямку конференції | | Автор доповіді | Учасник конференції, який готує і подає наукову доповідь | ID (успадковано),  ПІБ (успадковано як Назва),  Дата подачі (успадковано як Дата події / подачі)  Статус (студент, аспірант викладач) (успадковано як Статус)  Науковий ступінь,  Тема доповіді,  Обсяг тексту, | Подати доповідь  Отримати рецензію  Внести правки до тексту доповіді  Розрахувати вартість участі в конференції | | Стаття | Об’єкт наукової подачі, що оцінюється та публікується | Назва статті  Кількість сторінок. слів  Кількість  посилань на релевантні джерела  Мова статті | Перевірити відповідність вимогам  Розрахувати вартість перекладу на англійську   Визначити рівень унікальності | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Конференція**» і **«Стаття»** та«**Автор доповіді**» і **«Стаття»** асоціативний кратності один до одного. Між сутностями «**Конференція**» і «**Автор доповіді**»асоціацій немає. Спрощена взаємодія між учасниками наукового середовища: **один автор доповіді** бере участь в **одній конференції.** **Один автор** подає **одну статтю**. **Сервіс** не є учасником наукового процесу, є лише інструментом, який використовують інші сутності (автор, стаття, конференція) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про конференцію, автора і статтю згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **ConferenceObject** (абстрактний базовий клас), **Conference** (Конференція), **Author** (Автор) **-** похідні класи від **ConferenceObject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.11.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **ConferenceObject** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Conference,** **Author** як **override**.  3.  Додати клас **Paper** (стаття) з лаб. роб. №3, звязавши його асоціаціями з похідними класами **Conference,** **Author**. Включити в клас **Paper** атрибути і методи згідно з табл. 5.11.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Conference,** **Author** та **Paper** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) кількіть учасників конференції на задану величину, збільшити (зменшити) вартість участі автора в конференції на задану величину, порівняти дві конференції за кількістю напрямків (>,<), порівняти двох авторів за датою подачі статей (>,<), порівняти дві статті за рівнем унікальності (>, <), додати (видалити) кількість сторінок в статті, порівняти статті за кількістю посилань на джерела (==, !=), збільшити (зменшити) вартість перекладу статті. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Conference,** **Author** та **Paper** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість напрямків конференції, збільшити (зменшити) кількість правок статей після рецензії, збільшити (зменшити) кількість статей, що прийняті (яким відмовлено) тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfAuthor**, атрибутом якого зробити масив (список) об’єктів класу **Author**, використавши індексатори для доступу до заданих за індексом авторів.  7. Додати в клас **Paper** конструктор копії. За допомогою конструктора копії створити копію об’єкту статті з тими самими атрибутами, що й оригінал, але без посилань на релевантні джерела та унікальності. |
| **12** | **Опис предметної області**. Існує предметна область практико-орієнтованого навчання, в якій сутностями моделі є узагальнена сутність «**Старт кар’єри**». Сутності «**Студент**» і «**Біржа практик**» є нащадками узагальненої сутності «**Старт кар’єри**». Ці суб’єкти успадковують від узагальненої сутності «**Старт кар’єри**» спільні атрибути та операції. Комунікація сутностей «**Студент**» і «**Біржа практик**» здійснюється через сутність **«ІТ-компанія»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам удосконалення практичного навчання операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.12.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.12.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Старт кар’єри | Узагальнена сутність для учасників і керуючих елементів системи | Назва об'єкта,  Тип (студент / компанія),  Статус активності (активний / неактивний, відхилений, виключений, очікування підтвердження, завершив участь ) | Отримати/змінити ідентифікатор  Отримати/змінити тип  Отримати/Змінити статус, | | Студент | Користувач біржі, який шукає та проходить практику | ПІБ (успадковано як назва)  Тип (успадковано як «студент»)  Статус активності (успадковано як: «активний / неактивний, відхилений, виключений, підтверджений, завершив участь»)  Факультет, Спеціальність,  Курс, Середній бал, Вид практики (виробнича, проєктна, переддипломна тощо) | Подати заявку на практику  Оцінити практику  Підтвердити завершення  Вибрати місце практики | | Біржа практик | Платформа для розміщення та призначення місць практики | Назва біржі (успадковано як назва),  Тип (успадковано як «компанія»)  Статус активності(успадковано як «активна, неактивна, очікує оновлення, завершена» )  Перелік компаній,  Перелік доступних практик,  Статус заявки,  Статус співпраці | Прийняти заявку студента  Призначити місце практики  Надіслати звітні документи  Показати місця практики (компаній) | | ІТ-компанія | Організація-партнер, що надає місця для проходження практики | Назва компанії,  Галузь,  Кількість вільних місць,  Статус співпраці (Активна / Призупинена / Завершена | Створити пропозицію  Прийняти студента  Надати характеристику студенту  Завершити практику | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Студент**» і **«ІТ-компанія»** та«**Біржа практик**» і **«ІТ-компанія»** асоціативний кратності один до одного. Між сутностями «**Студент**» і «**Біржа практик**»асоціацій немає. Спрощена взаємодія між учасниками практичного середовища: **один студент** працює в **одній ІТ-компанії**. **Один студент** звертається до **одної біржі практик**. **Сервіс** не є учасником практичного процесу, є лише інструментом, який використовують інші сутності (студент, біржа практик, компанія) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента, біржі практик, ІТ-компанії згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **CareerStart** (абстрактний базовий клас), **Student** (Студент), **PracticeExchange** (Біржа практик) **-** похідні класи від **CareerStart**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.12.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **CareerStart** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Student,** **PracticeExchange** як **override**.  3.  Додати клас **ІТCompany** (ІТ-компанія) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Student,** **PracticeExchange**. Включити в клас **ІТCompany** атрибути і методи згідно з табл. 5.12.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Student,** **PracticeExchange** та **ІТCompany** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) середній бал студента на задану величину, збільшити (зменшити) кількість компаній, з якими біржа практики має зв’язки, на задану величину, порівняти двох студентів за середнім балом (>,<), порівняти дві біржі практик за переліком доступних практик (>, <), додати (видалити) кількість вільних місць в компанії, порівняти компанії за галуззю (==, !=), збільшити (зменшити) стптус співпраці компанії з біржею. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Student,** **PracticeExchange** та **ІТCompany** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) курс навчання студента, збільшити (зменшити) кількість заявок студентів на практику, збільшити (зменшити) кількість вільних місць в компанії тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfCompanies**, атрибутом якого зробити масив (список) об’єктів класу **ІТCompany**, використавши індексатори для доступу до заданих за індексом компаній.  7. Додати в кла **Student** конструктор копії. За допомогою конструктора копії створити копію об'єктв студент з тими самими атрибутами, що й оригінал, але без середнього балу та курсу. |
| **13** | **Опис предметної області**.  Існує предметна область ріелторських послуг, в якій сутностями моделі є узагальнена сутність «**Сторони угоди**». Сутності «**Клієнт**» і «**Агенство нерухомості**» є нащадками узагальненої сутності «**Сторони угоди**». Ці суб’єкти успадковують від узагальненої сутності «**Сторони угоди**» спільні атрибути та операції. Комунікація сутностей «**Клієнт**» і «**Агенство нерухомості**» здійснюється через сутність **«Рієлтор»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам роботи з нерухомістю операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.13.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.13.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Сторони угоди про нерухомість | Узагальнена сутність для учасників угоди про нерухомість | Назва,  Тип сторони (агентство / клієнт), Статус участі (активна / тимчасова / завершена | Зареєструвати у системі (ініціалізувати атрибути), Активувати профіль (змінити статус участі),  Отримати / змінити атрибути, Надіслати запит / відповідь (визначити тип сторони угоди) | | Агентство нерухомості | Посередник, що підбирає, пропо Кількість укладених договорів нує та супроводжує операції з нерухомістю | Назва агентства (успадковано як назва),  Тип сторони (успадковано як «агенство»)  Статус участі (успадковано)  Список об’єктів нерухомості,  Ліцензія, | Прийняти запит клієнта  Шукати об’єкт відповідно до запиту клієнта  Оцінити вартість нерухомості  Підготувати договір  Закрити угоду | | Клієнт | Особа, яка звертається за послугами з оренди чи купівлі нерухомості | Ім’я, Прізвище (успадковано як назва)  Тип сторони (успадковано як «Клієнт»)  Статус участі (успадковано)  Тип запиту (купівля/оренда), Бюджет,  Бажаний район,  Статус запиту (новий, активний, завершений) | Шукати агенство нерухомості  Подати запит  Переглянути пропозиції  Укласти/відхилити угоду | | Рієлтор | Представник агентства, який супроводжує клієнта протягом угоди | Назва компанії,  Ім’я, Прізвище, Спеціалізація,  Кількість угод,  Статус доступності - Доступний / Зайнятий | Прийняти запит  Супроводжувати клієнта  Оформити угоду | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Клієнт**» і **«Рієлтор»** та«**Агенство нерухомості**» і **«Рієлтор»** асоціативний кратності один до одного. Між сутностями «**Клієнт**» і **«Агенство нерухомості**» асоціацій немає. Спрощена взаємодія між учасниками сфери нерухомості: **один клієнт** звертається в **одне агентсво нерухомості**. **Один рієлтор** обслуговує **одного клаєнта.** **Сервіс** не є учасником процесу роботи з нерухомістю, є лише інструментом, який використовують інші сутності (клієнт, агенство, рієлтор) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта, агенство, рієлтора згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **PartyToAgreement** (абстрактний базовий клас), **Client** (клієнт),  **RealEstateAgency** (Агентство нерухомості) **-** похідні класи від **PartyToAgreement**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.13.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **PartyToAgreement** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Client,** **RealEstateAgency** як **override**.  3.  Додати клас **Rieltor** (ріелтор) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Client,** **RealEstateAgency**. Включити в клас **Rieltor** атрибути і методи згідно з табл. 5.13.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **RealEstateAgency, Client** та **Rieltor** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) кількість об’єктів нерухомості на задану величину, збільшити (зменшити) бюджет клієнта на задану величину, порівняти два агенства за кількістю укладених угод (>,<), порівняти двох клієнтів за кількістю переглянутих пропозицій від агентсва (>, <), додати (видалити) кількість угод, укладених рієлтором, порівняти агентсва за списком об’єктів нерухомості (==, !=), порівняти рієлторів за спеціалізацією. У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **RealEstateAgency, Client** та **Rieltor** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість угод, збільшити (зменшити) кількість клієнтів, закріплених за рієлтором, збільшити (зменшити) список об’єктів нерухомості тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfClient**, атрибутом якого зробити масив (список) об’єктів класу **Client**, використавши індексатори для доступу до заданих за індексом клієнтів.  7. Додати в клас **Rieltor** конструктор копії. За допомогою конструктора копії створити копію об'єктів рієлтор з тими самими атрибутами, що й оригінал, але без кількісті угод |
| **14** | **Опис предметної області**.  Існує предметна область академічної мобільності, в якій сутностями моделі є узагальнена сутність «**Сторона програми обміну**». Сутності «**Студент**» і «**Відділ академічної мобільності**» є нащадками узагальненої сутності «**Сторона програми обміну**». Ці суб’єкти успадковують від узагальненої сутності «**Сторона програми обміну**» спільні атрибути та операції. Комунікація сутностей «**Студент**» і «**Відділ академічної мобільності**» здійснюється через сутність **«Міжнародна програма обміну»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам міжнародної мобільності операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.14.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.14.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Сторона програми обміну | Узагальнена сутність для  процесу академічної мобільності | Ідентифікатор (*код для однозначної ідентифікації сторін обміну*),  Тип учасника *(студент / відділ)*,  Статус взаємодії *(активний, завершений, скасований)* | Ініціювати участь у програмі (ініціалізувати значення атрибутів),  Отримати/змінити тип учасника  Отримати/змінити статус заявки про взаємодію | | Студент | Здобувач освіти, учасник академ. мобільності | Ідентифікатор (успадковано),  Тип учасника (успадковано як “студент”)  Статус заявки (успадковано як Статус взаємодії)  ПІБ,  Освітній рівень (бакалавр, магістр),  Середній бал,  Наукові здобутки | Розрахувати рейтинг студента,  Подати заявку,  Отримати результат розгляду заявки,  Підписати угоду | | Відділ академічної мобільності | Організатор і координатор програм обміну | Ідентифікатор (успадковано),  Тип учасника (успадковано як “відділ”)  Статус взаємодії (успадковано)  Назва програми,  Мінімальний середній бал Кількість учасників,  Список (масив) ЗВО-партнерів | Розглянути заявку,  Шукати університет  Схвалити заявку,  Відхилити заявку,  Оформити документи, | | Міжнародна програма | Представлення програми для академічної мобільності між університетами | Назва програми  Спеціальність  Країна  Тривалість програми (місяці)  Університет-партнер | Перевірити наявність місць у програмі  Провести конкурс серед претендентів на участь у міжнародних програмах | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Студент**» і **«Міжнародна програма обміну»** та«**Відділ академічної мобільності**» і **«Міжнародна програма обміну»** асоціативний кратності один до одного. Між сутностями «**Студент**» і «**Відділ академічної мобільності**» асоціацій немає. Спрощена взаємодія між учасниками мобільного середовища: **один студент** може бути участником **одної програми обміну**. **Один студент** може взаємодіяти з **одними відділом академ. мобільності.** **Сервіс** не є учасником освітнього процесу, є лише інструментом, який використовують інші сутності (студент, програма обміну, відділ академ. мобільності) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про студента, програму обміну, відділ академ. мобільності згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **ExchangeProgramSide** (абстрактний базовий клас), **Student** (Студент), **AcademicMobility** (відділ академічної мобільності) **-** похідні класи від **ExchangeProgramSide**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.14.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **ExchangeProgramSide** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Student,** **AcademicMobility** як **override**.  3.  Додати клас **InterProgram** (міжнародна програма) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Student,** **AcademicMobility**. Включити в клас **InterProgram** атрибути і методи згідно з табл. 5.14.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Student,** **AcademicMobility** та **InterProgram** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) середній бал студента на задану величину, збільшити (зменшити) кількість учасників відділу мобільності на задану величину, порівняти двох студентів за кількістю наукових здобутків (>,<), порівняти два відділи академ. мобільності за кількістю ЗВО-партнерів (>, <), додати (видалити) кількість місць на програму обміну, порівняти програми за тривалістю (==, !=), порівняти студентів за рейтингом (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Student,** **AcademicMobility** та **InterProgram** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість угод, збільшити (зменшити) кількість учасників, подавших заявки, збільшити (зменшити) рейтинг студента, збільшити (зменшити) тривалість програми тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfClient**, атрибутом якого зробити масив (список) об’єктів класу **InterProgram**, використавши індексатори для доступу до заданих за індексом програм обміну.  7. Додати в клас **Student** конструктор копії. За допомогою конструктора копії створити копію об'єктів студент з тими самими атрибутами, що й оригінал, але без значення рейтингу. |
| **15** | **Опис предметної області**. Існує предметна область Таксі-сервіс, в якій сутностями моделі є узагальнена сутність «**Сервіс таксі**». Сутності «**Клієнт**» і «**Таксі**» як перевізник, є нащадками узагальненої сутності «**Сервіс таксі**». Ці суб’єкти успадковують від узагальненої сутності «**Сервіс таксі**» спільні атрибути та операції. Комунікація сутностей «**Клієнт**» і «**Таксі**» здійснюється через сутність **«Автомобіль»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам таксі-сервісу операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.15.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.15.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Таксі сервіс | Узагальнена сутність для  процесу організації поїздки | Ідентифікатор замовлення (*код для однозначної ідентифікації поїздки*  Тип учасника *(клієнт / таксі)*,  Статус замовлення *(створене, активне, завершене, скасоване, немає машини)* | Ініціювати поїздку (ініціалізувати значення атрибутів),  Отримати/змінити тип учасника  Отримати/змінити статус замовлення | | Клієнт | Замовник послуги таксі | Ідентифікатор замовлення (успадковано),  Тип учасника (успадковано як  "клієнт")   Статус замовлення (успадковано)  Ім'я клієнта, Контактний номер,  Місце посадки,  Місце призначення,  Спосіб оплати | Замовити таксі (вказати місце посадки, місце призначення)  Оплатити поїздку (сума, спосіб оплати)  Скасувати замовлення | | Таксі-перевізник | Виконавець замовлення, перевізник | Ідентифікатор замовлення (успадковано)   Тип учасника (успадковано як "таксі")  Статус замовлення (успадковано)  Номер таксі,  Тариф за км, Тариф за хвилину  Вартість оплати, Відстань | Прийняти замовлення (місце посадки, місце призначення).  Розрахувати вартість поїздки (відстань, тариф)  Визначити оптимальний маршрут поїздки | | Автомобіль | Засіб пересування | Марка,  Модель,  Пробіг,  Витрата пального  Запас пального | Перевірити технічний стан, Визначити витрати пального,  Оцінити залишковий ресурс | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Клієнт**» і **«Автомобіль»** та«**Таксі**» і **«Автомобіль»** асоціативний кратності один до одного. Між сутностями «**Клієнт**» і «**Таксі**» асоціацій немає. Спрощена взаємодія між учасниками транспортного середовища: **один клієнт** може замовити **один автомобіль** в **одного таксі-перевізника**. **Сервіс** не є учасником транспортного процесу, є лише інструментом, який використовують інші сутності (клієнт, таксі, автомобіль) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта, таксі, автомобільі згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **TaxiService** (абстрактний базовий клас), **Client** (Клієнт) та **Taxi** (таксі-перевізник) **-** похідні класи від **TaxiService**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.15.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **TaxiService** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Client,** **Taxi** як **override**.  3.  Додати клас **Car** (міжнародна програма) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Client,** **Taxi**. Включити в клас **Car** атрибути і методи згідно з табл. 5.14.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Client,** **Taxi** та **Car** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) суму оплати клієнтом поїздки на задану величину, збільшити (зменшити) вартість оплати, нараховану таксі-перевізником на задану величину, порівняти два таксі за тарифом за км (>,<), порівняти два автомобіля за витратою пального (>, <), додати (видалити) кількість клієнтів таксі, автомобілі за пробігом (==, !=), порівняти клієнтів за способом оплати (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Client,** **Taxi** та **Car** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) вартість проїзду в таксі, збільшити (зменшити) запас пального в автомобілі, збільшити (зменшити) рейтинг студента, збільшити (зменшити) кількість замовлень одним клієнтом тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfCar**, атрибутом якого зробити масив (список) об’єктів класу **Car**, використавши індексатори для доступу до заданих за індексом автомобілів.  7. Додати в клас **Car** конструктор копії. За допомогою конструктора копії створити копію об’єкту автомобільт з тими самими атрибутами, що й оригінал, але без значення пробігу. |
| **16** | **Опис предметної області**. Існує предметна область фінансових послуг, в якій сутностями моделі є узагальнена сутність «**Фінансовий суб’єкт**». Сутності **«Клієнт»** і **«Банк»** є нащадками узагальненої сутності «**Фінансовий суб’єкт**». Ці суб’єкти успадковують від узагальненої сутності «**Фінансовий суб’єкт**» спільні атрибути та операції. Комунікація сутностей «**Клієнт**» і «**Банк**» здійснюється через сутність«**Сайт банку**»з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам фінансової взаємодії операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.16.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.16.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Фінансовий суб’єкт | Узагальнена роль для учасників фінансової взаємодії | Ідентифікатор,  Тип фінансового суб’єкта (Клієнт / Банк),   Статус взаємодії (активний / неактивний / завершений) | Завершити операцію  Ініціювати  фінансову операцію (ініціалізувати значення атрибутів),  Отримати/змінити тип фінансового суб’єкта  Отримати/змінити статус взаємодії | | Клієнт | Фізична або юридична особа, яка користується фінансовими послугами банку | Ідентифікатор (успадковано)  Тип суб’єкта (успадковано як "Клієнт")  Статус взаємодії (успадковано)  ПІБ,  Паспорт, ІПН,  Номер рахунку,  Баланс рахунку,  Кредитний рейтинг | Відкрити рахунок  Закрити рахунок  Поповнити рахунок  Зняти кошти  Отримати кредит | | Банк | Фінансова установа, що надає банківські послуги | Ідентифікатор (успадковано),  Тип суб’єкта (успадковано як "Банк"),  Статус взаємодії (успадковано)  Назва банку  Перелік послуг,  Процентні ставки,  Кредитні ліміти | Створити рахунок клієнта  Відкрити депозит  Надати кредит  Обробити платіж  Нарахувати відсотки по кредиту/депозиту | | Сайт банку | Онлайн-платформа для надання банківських послуг | URL  Сервіси  Кількість користувачів на день | Контроль залишків на поточних рахунках,  Оформлення депозиту,  Оплата комунальних послуг,  Переказ коштів на рахунки | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Клієнт**» і **«Сайт банку»** та«**Банк**» і **«Сайт банку»** асоціативний кратності один до одного. Між сутностями «**Клієнт**» і «**Банк**» асоціацій немає. Спрощена взаємодія між учасниками фінансового середовища: **один клієнт** може бути клієнтом **одного банку** і користуватися **одним сайтом банку**. **Сервіс** не є учасником фінансового процесу, є лише інструментом, який використовують інші сутності (клієнт, банк, сайт банку) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про клієнта, банк, сайт банку згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **FinancialSubject** (абстрактний базовий клас), **Client** (Клієнт) та **Bank** (Банк) **-** похідні класи від **FinancialSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.15.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **FinancialSubject** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Client,** **Bank** як **override**.  3.  Додати клас **Website** (сайт банку) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Client,** **Bank**. Включити в клас **Website** атрибути і методи згідно з табл. 5.16.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Client,** **Bank** та **Website** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) баланс клієнта на задану величину, збільшити (зменшити) процентні ставки в банку на задану величину, порівняти два банки за переліком послуг (>,<), порівняти два клієнта за кредитним рейтингом (>, <), додати (видалити) гроші у клієнта, порівняти URL двох сайтів банку (==, !=), порівняти клієнтів за кредитним рейтингом (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Client,** **Bank** та **Website** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кредитний рейтинг клієнта, збільшити (зменшити) кредитні ліміти банку, збільшити (зменшити) кількість користувачів сайту в день, збільшити (зменшити) кількість переказів коштів на рахунки через сайт банку тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfClient**, атрибутом якого зробити масив (список) об’єктів класу **Client**, використавши індексатори для доступу до заданих за індексом клієнтів.  7. Додати в клас **Website** конструктор копії. За допомогою конструктора копії створити копію об’єкту сайт банка з тими самими атрибутами, що й оригінал, але без значення кількості користувачів на день. |
| **17** | **Опис предметної області**. Існує предметна область медичних послуг, в якій сутностями моделі є узагальнена сутність **«Медичний суб’єкт**». Сутності **«Пацієнт»** і **«Лікар»** є нащадками узагальненої сутності **«Медичний суб’єкт**». Ці суб’єкти успадковують від узагальненої сутності **«Медичний суб’єкт**» спільні атрибути та операції. Комунікація сутностей **«Пацієнт»** і **«Лікар»** здійснюється через сутність«**Медична картка**»з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам медичної взаємодії операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.17.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.17.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Медичний суб’єкт | Узагальнена роль для осіб, що беруть участь у медичному процесі | Ідентифікатор,   ПІБ,  Тип медичного суб’єкта (Пацієнт /Лікар),   Статус взаємодії (активний / завершений / очікує) | Завершити операцію  Ініціювати  фінансову операцію (ініціалізувати значення атрибутів),  Отримати/змінити тип фінансового суб’єкта  Отримати/змінити статус взаємодії | | Пацієнт | Фізична особа, яка отримує медичну допомогу | Ідентифікатор (успадковано),  ПІБ (успадковано),  Тип суб’єкта (успадковано як “Пацієнт”,  Статус взаємодії (успадковано)  Дата народження,  Медична картка,  Історія хвороби,  Поточні скарги | Записатися на прийом,  Отримати діагноз, Отримати призначення лікування | | Лікар | Фізична особа, яка надає медичні послуги | Ідентифікатор (успадковано),  ПІБ (успадковано),  Тип суб’єкта (успадковано як “Лікар”,  Статус взаємодії (успадковано  Спеціалізація,  Стаж роботи,  Графік прийому,  Список пацієнтів,  Доступ до медичних карт | Провести прийом пацієнта,  Встановити діагноз, Призначити лікування | | Медична картка | Архів медичних даних пацієнта | Номер картки,  дані пацієнта, історія хвороб, призначення лікування | Оновити медичну історію  Додати результати аналізів  Переглянути історію хвороб | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Пацієнт**» і «**Медична картка**» та«**Лікар**» і «**Медична картка**»асоціативний кратності один до одного. Між сутностями «**Пацієнт**» і «**Лікар**» асоціацій немає. Спрощена взаємодія між учасниками медичного середовища: **один пацієнт** може бути клієнтом **одного лікаря** і мати **однну медичну картку**. **Один лікар** працює з **одною медичною карткою. Сервіс** не є учасником медичного процесу, є лише інструментом, який використовують інші сутності (пацієнт, лікар, медична картка) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про пацієнт, лікар, медична картка згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **MedicalSubject** (абстрактний базовий клас), **Patient** (Пацієнт) **Doctor** (Лікар) **-** похідні класи від **MedicalSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.15.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **MedicalSubject** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Patient,** **Doctor** як **override**.  3.  Додати клас **MedicalCard** (медична картка**)** лаб. роб. №3, звязавши його асоціаціями з похідними класами **Patient,** **Doctor**. Включити в клас **Website** атрибути і методи згідно з табл. 5.16.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Patient,** **Doctor** та **MedicalCard** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) стаж роботи лікаря на задану величину, збільшити (зменшити) кількість хророб пацієнта на задану величину, порівняти двох лікарів за кількістю пацієнтів (>,<), порівняти два пацієнта за кількістю хвороб (>, <), додати (видалити) список пацієнтів, порівняти дві медичні картки (==, !=), порівняти пацієнтів за історією хвороб (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Patient,** **Doctor** та **MedicalCard** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) поточні скарги пацієнта, збільшити (зменшити) графік прийому лікаря, збільшити (зменшити) кількість пацієнтів в день, збільшити (зменшити) кількість призначень, записаних в медичній картці тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfPatient**, атрибутом якого зробити масив (список) об’єктів класу **Patient**, використавши індексатори для доступу до заданих за індексом пацієнта.  7. Додати в клас **MedicalCard** конструктор копії. За допомогою конструктора копії створити копію об’єкту медична картка а з тими самими атрибутами, що й оригінал, але без призначення лікування |
| **18** | **Опис предметної області**. Існує предметна область мережі користувачів, в якій сутностями моделі є узагальнена сутність **«Суб’єкт мережі**». Сутності **«Соціальна мережа»** і **«Користувач»** є нащадками узагальненої сутності **«Суб’єкт мережі**». Ці суб’єкти успадковують від узагальненої сутності **«Суб’єкт мережі**» спільні атрибути та операції. Комунікація сутностей «**Соціальна мережа**» і «**Користувач**» здійснюється через сутність«**Друзі**»з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам соціальної взаємодії операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.18.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.18.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Суб’єкт мережі | Узагальнена роль для учасників взаємодії в мережі | Назва ,  Тип суб’єкта мережі (Соціальна мережа /Користувач),  Статус активності (активний / неактивний) | Ініціювати взаємодію  Отримати дані про взаємодію  Завершити взаємодію | | Соціальна мережа | Платформа для взаємодії користувачів | Тип суб’єкта мережі (успадковано як “Соціальна мережа”),  Статус активності (успадковано)  Назва мережі (успадковано як Назва))  Кількість користувачів  Спрямованість мережі (для науковців, для розваг тощо) | Додати користувача до мережі  Видалити користувача з мережі  Публікувати контент  Створити групи в мережі  Заблокувати користувача  Шукати користувачів | | Користувач | Фізична особа, яка комунікує через  мережу | Тип суб’єкта мережі (успадковано як “Користувач”),  Статус активності (успадковано)  Ім’я користувача (успадковано як Назва)  e-mail  Пароль  Кількість друзів | Зареєструвати користувача  Оновити профіль  Додати/видалити контент до мережі  Додати/видалити друзів  Ініціювати пошук користувачів | | Друзі | Користувачі, які можуть обмінюватися контентом | FirstName та LastName, e-mail друга  Статус дружби | Надіслати запит на дружбу  Підтвердити запит на дружбу  Відправити повідомлення другу  Заблокувати небажаного “друга” | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Соціальна мережа**» і **«Друзі»** та«**Користувач**» і **«Друзі»** асоціативний кратності один до одного. Між сутностями «**Соціальна мережа**» і «**Користувач**» асоціацій немає. Спрощена взаємодія між учасниками фінансового середовища: **один користувач** може бути клієнтом **одної соціальної мережі** і мати **одного друга**. **Сервіс** не є учасником соціальної взаємодії, є лише інструментом, який використовують інші сутності (користувач, соціальна мережа, друзі) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про користувача, соціальну мережу, друзів згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **NetworkSubject** (абстрактний базовий клас), **SocialNetwork** (Соціальна мережа), **User** (Користувач) **-** похідні класи від **NetworkSubject**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.18.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **NetworkSubject** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **SocialNetwork,** **User** як **override**.  3.  Додати клас **Friends** (друзі) лаб. роб. №3, звязавши його асоціаціями з похідними класами **SocialNetwork,** **User**. Включити в клас **Friends** атрибути і методи згідно з табл. 5.16.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **SocialNetwork,** **User** та **Friends** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) кількість користувачів соціальної мережі на задану величину, збільшити (зменшити) кількість друзів користувача на задану величину, порівняти дві соціальні мережі за кількістю користувачів (>,<), порівняти двох користувачів соціальної мережі за кількістю друзів (>, <), додати (видалити) контент у користувача, порівняти двох користувачів за e-mail (==, !=), порівняти двох друзів за FirstName та LastName (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **SocialNetwork,** **User** та **Friends** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) кількість користувачів соціальної мережі, збільшити (зменшити) кількість друзів користувача, збільшити (зменшити) обсяг контенту користувача тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfUser**, атрибутом якого зробити масив (список) об’єктів класу **User**, використавши індексатори для доступу до заданих за індексом користувачів.  7. Додати в клас **User** конструктор копії. За допомогою конструктора копії створити копію об’єкту скористувача з тими самими атрибутами, що й оригінал, але без значення паролю. |
| **19** | **Опис предметної області**. Існує предметна область інфраструктура міста, в якій сутностями моделі є узагальнена сутність **«Транспортна інфраструктура**». Сутності **«Місто»** і **«Транспорт»** є нащадками узагальненої сутності **«Транспортна інфраструктура**». Ці суб’єкти успадковують від узагальненої сутності **«Транспортна інфраструктура**» спільні атрибути та операції.. Комунікація сутностей «**Місто**» і «**Транспорт**» здійснюється через сутність«**Маршрутна мережа»** з атрибутами і методами класу «**Інфраструктура**» з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам інфраструктура міста операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.19.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.19.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Транспортна інфраструктура | Узагальнений суб’єкт міської транспортної взаємодії | Ідентифікатор  Назва суб’єкту  Статус активності (активний / неактивний)  Тип суб’.єкту (місто / транспорт) | Ініціювати взаємодію (встановити значення статусу “активний” )  Перевірити стан суб’єкту  Завершити дію (змінити статус на “неактивний”) | | Місто | Організує транспортну систему, контролює її ефективність | Назва міста (успадковано як Назва суб’єкту)  Статус (успадковано)  Тип (успадковано як "місто")  Кількість жителів  Площа міста  Кількість транспортних одиниць | Розрахувати щільність транспорту  Оцінити навантаження на інфраструктуру | | Транспорт | Забезпечує перевезення в межах міста | Ідентифікатор (успадковано)  Назва (успадковано як назва транспорту/маршруту),  Статус активності (успадковано),  Тип (успадковано як транспорт: автобус, трамвай, метро)  Максимальна пасажиромісткість  Кількість рейсів на день | Розрахувати перевізну здатність  Оцінити ефективність маршруту | | Маршрутна мережа | Структура маршрутів і їхня доступність для міських районів | Кількість маршрутів,  Середня довжина маршруту,  Рівень покриття міста - % міської території, яку обслуговує транспорт | Розрахувати щільність покриття  Оцінити ефективність мережі | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Місто**» і **«Маршрутна мережа»** та«**Транспорт**» і **«Маршрутна мережа»** асоціативний кратності один до одного. Між сутностями «**Місто**» і «**Транспорт**» асоціацій немає. Спрощена взаємодія між учасниками фінансового середовища: **одне місто** має **одну маршрутну мережу**, на якій курсує **один вид транспорту.** **Сервіс** не є учасником транспортної інфраструктури, є лише інструментом, який використовують інші сутності (місто, транспорт, маршрутна мережа) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про к місто, транспорт, маршрутну мережу згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **TransportInfrastructure** (абстрактний базовий клас), **City** (місто), **Тransport** (транспорт) **-** похідні класи від **TransportInfrastructure**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.19.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **TransportInfrastructure** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **City,** **Тransport** як **override**.  3.  Додати клас **RouteNetwork** (Маршрутна мережа) лаб. роб. №3 клас **Infrastructure**, звязавши його асоціаціями з похідними класами **City,** **Тransport**. Включити в клас **RouteNetwork** атрибути і методи згідно з табл. 5.19.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **City,** **Тransport** та **RouteNetwork** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) місто на задану кількість жителів, збільшити (зменшити) транспорт на задану максимальну пасажиромісткість, порівняти два міста за площею (>,<), порівняти два транспорта за кількістю рейсів на день (>, <), додати (видалити) кількість транспортних одиниць у місті, порівняти два міста за назвою (==, !=), порівняти два транспорта за типом (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **City,** **Тransport** та **RouteNetwork** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) маршрутну мережу, збільшити (зменшити) місто, збільшити (зменшити) транспорт тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfRoute**, атрибутом якого зробити масив (список) об’єктів класу **RouteNetwork**, використавши індексатори для доступу до заданих за індексом иаршрутів.  7. Додати в клас **RouteNetwork** конструктор копії. За допомогою конструктора копії створити копію об’єкту маршрутна мережа з тими самими атрибутами, що й оригінал, але без значення рівня покриття міста. |
| **20** | **Опис предметної області**. Існує предметна область цифрового простору, в якій сутностями моделі є узагальнена сутність **«Суб’єкт цифрового простору**». Сутності **«Молодь**» і **«Цифрова платформа»** є нащадками узагальненої сутності **«Суб’єкт цифрового простору**». Ці суб’єкти успадковують від узагальненої сутності **«Суб’єкт цифрового простору**» спільні атрибути та операції. Комунікація сутностей «**Молодь**» і «**Цифрова платформа**» здійснюється через сутність«**Проєкт»** з атрибутами і методами з лаб.роб. №3. Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» для реалізації не притаманних участникам цифрового простору операцій. У разі потреби використання меню (не обов’язково) для зручності користувача, доцільно створити окрему сутність «**Меню».** В таблиці 5.20.1 подані ролі, атрибути та операції сутностей самоврядування.  Таблиця 5.20.1. Основні сутності предметної області   |  |  |  |  | | --- | --- | --- | --- | | **Сутність** | **Роль сутності** | **Атрибути сутності** | **Операції (дії) сутності** | | Суб’єкт цифрового простору | Узагальнена сутність для суб’єктів цифрової взаємодії | Назва / Ім’я,  Статус активності (активний / неактивний)  Тип учасника (молодь / платформа) | Ініціювати взаємодію,  Перевірити статус,  Завершити дію | | Молодь | Учасник цифрового ринку, що створює кар’єру через платформу | Ім’я (успадковано як назва),  Статус активності (успадковано),  Тип учасника (успадковано як “ молодь”)  Вік,  Навички,  Рейтинг,  Кількість проєктів  Профіль | Додати навичку  Оцінити досвід (виконання проєкту)  Розрахувати рейтинг | | Цифрова платформа | Середовище, яке надає інструменти для навчання і роботи | Назва платформи (успадковано як назва),  Статус активності (успадковано),  Тип учасника (успадковано як “платформа”)  Кількість учасників,  Доступні проєкти,  Назва курсу | Призначити проєкт  Згенерувати звіт про активність  Алгоритм підбору проєкту  Рекомендувати курс | | Проєкт | Практичне завдання, яке виконує молодь, отримуючи досвід і оцінку | Назва,  Сфера,  Складність,  Статус виконання,  Оцінка | Призначити виконавця  Змінити статус  Оцінити результат | | Сервіс | Технічний посередник: забезпечує обмін даними,введення/виведення, збереження даних. | Шлях до файлу (ім’я файлу) Дані для обробки | Вивести дані на консоль Читати дані з консолі Записати дані у файл Читати дані з файлу |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Зв’язок між сутностями «**Молодь**» і **«Проєкт»** та«**Цифрова платформа**» і **«Проєкт»** асоціативний кратності один до одного. Між сутностями «**«Молодь**» і **«Цифрова платформа»** асоціацій немає. Спрощена взаємодія між учасниками цифрового прностору: **молодь** виконує **один проєкт** на **одній цифровій платормі.** **Сервіс** не є учасником цифрового простору, є лише інструментом, який використовують інші сутності (молодь, цифрова платформа, проєкт) для відображення результатів своїх дій.  На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. На основі діаграми класів здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# для обробки даних про молодь, цифрову платформу, проєкт згідно зі сценарієм роботи версії 1 лабораторної роботи №5.  **Версія 1 (пункти завдання 1,2,3).**  1.  Скопіювати проєкт версії 2 лабораторної роботи №4 і перейменувати його для версії 1 лабораторної роботи №5. Зберегти ієрархію успадкування класів версії 2 лабораторної роботи №4: **SubjectDigitalSpace** (абстрактний базовий клас), **Youth** (молодь), **DigitalPlatform** (цифрова платформа) **-** похідні класи від **SubjectDigitalSpace**. Клас **Service** бажано залишити такий самий, як в лаб. роботі №4. Не додавайте в клас **Service** меню команд. При необхідності використати меню слід зробити його як окремий клас. Для виведення команд меню використовуйте методи роботи з консоллю класу **Service.** Кожен клас має бути створений в окремому файлі*.* В класах зберігати поля і методи з лаб. роб. №4 (табл.5.19.1).  2.  Реалізувати динамічний поліморфізм методів, зробивши методи класу **SubjectDigitalSpace** (абстрактний базовий клас) віртуальними (**virtual**) та перевизначити їх в похідних класах **Youth,** **DigitalPlatform** як **override**.  3.  Додати клас **Project** (проєкт) лаб. роб. №3, звязавши його асоціаціями з похідними класами **Youth,** **DigitalPlatform**. Включити в клас **Project** атрибути і методи згідно з табл. 5.20.1 та лаб. роб. №3. *Якщо не модифікувати алгоритми методів лабораторних робіт №3 та №4, реалізація їх залишається без змін в лабораторній роботі* №5.  **Версія 2 (пункти завдання 4, 5).**  4. Реалізувати в класах **Youth,** **DigitalPlatform** та **Route Project Network** статичний поліморфізм операторів, додавши методи перевантаження **бінарних операторів** (+, -, ==, !=, >, < ): збільшити (зменшити) об’єкт молоді на задане значення рейтингу, збільшити (зменшити) цифрову платформу на задану кількість учасників, порівняти дві цифрові платформи за кількістю доступних проєктів (>,<), порівняти двох представників молоді за навичками (>, <), додати (видалити) два проєкти, порівняти дві цифрові платформи за назвою (==, !=), порівняти два проєкти за складнстю (==, !=). У випадку перевантаження операторів == (рівність), != (нерівність) потрібно додатково реалізувати методи **Equals()** і **GetHashCode().**  5.  Реалізувати в класах **Youth,** **DigitalPlatform** та **Project** статичний поліморфізм операторів, додавши методи перевантаження **унарних операторів** (++, -- ): збільшити (зменшити) молодь, збільшити (зменшити) цифрову платформу, збільшити (зменшити) проєкт тощо.  **Версія 3 (пункти завдання 6, 7).**  6.  Додати новий клас **SetOfProject**, атрибутом якого зробити масив (список) об’єктів класу **Project**, використавши індексатори для доступу до заданих за індексом проєктів.  7. Додати в клас **Project** конструктор копії. За допомогою конструктора копії створити копію об’єкту проєкт з тими самими атрибутами, що й оригінал, але без значення оцінки. |

### Література

1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#. КНУ ім. Тараса Шевченка
2. C# - творчість програмування. Том 1. Об’єктно-орієнтоване програмування: підручник// Під ред. Бичкова О.С. Волин. Обереги, 2024. – 292 с.
3. The C# Coding Standard. Access mode: <https://github.com/hassanhabib/CSharpCodingStandard>
4. C# Coding Standards and Best Practices. Access mode: <https://www.dofactory.com/csharp-coding-standards>
5. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І. В., Марущак П. О. – Тернопіль: ФОП Паляниця В. А., 2020 – 320 с. Режим доступу: <https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf>

## Лабораторна робота №6

**10 Варіантів редагован**і

## Взаємодії класів, делегати та обробка виняткових ситуацій

### Рейтинг лабораторної роботи №6

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п.п* | *Вид діяльності студента* | *Рейтинговий бал* | *Deadline* |
| 1 | Написання коду з 5 завдань | 0,5\*5=2,5 | 25 квітня |
| 2 | Захист роботи | 0,5 |
| 3 | Звіт з роботи | 0,5 |  |
| Разом за роботу | | 3,5 |  |

### Мета роботи:

1. Виконати лабораторну роботу, підсумувавши знання з усіх попередніх лабораторних робіт.
2. Реалізувати взаємодію класів у вигляді асоціації, агрегації, композиції.
3. Навчитись програмувати події та їх обробників.
4. Навчитись визначати та обробляти виняткові ситуації

### Методичні рекомендації до виконання лабораторної роботи

1. Уважно прочитайте лекції та методичні вказівки до лабораторної роботи.

2. Реалізуйте та проаналізуйте всі приклади з лекцій.

### Порядок виконання роботи

1. Створити директорію Lab6, в якій будуть розміщуватися проєкт цієї лабораторної роботи.

2. Виконати завдання свого варіанту у вигляді **одного** консольного проєкту.

3. Для кожного класу передбачити окремий файл.

4. Для ілюстрації роботи кожного пункту завдання використати меню команд, за створюватимуться об’єкти та викликатимуться їх методи.

### Приклади виконання завдань

#### 1. Приклад коду для обробки виключних ситуацій

При створенні програм із застосуванням структурованої обробки виключень передбачається використовувати такі чотири взаємозалежних елементи:

* тип класу, який надає детальну інформацію про виниклу виняткову ситуацію;
* член, який генерує, або надсилає (**throw**) екземпляр класу, що відповідає винятковій ситуації, стороні, що викликає;
* блок програмного коду, в якому був викликаний член, що генерує виняток;
* блок програмного коду сторони, що викликає, в якому виконується обробка, або захоплення (**catch**) даного виключення.

У мові програмування C# пропонуються чотири ключових слова (**try, catch, throw і finally**), за допомогою яких генеруються і обробляються виключення. Тип, що представляє відповідну проблему, є класом, похідним від **System.Exceptio**n (або його нащадком).

З урахуванням цього давайте з'ясуємо роль зазначеного базового класу. Створимо новий консольний додаток, в якому визначатимуться два типи класу *Car* (автомобіль) і *Radio* (радіо), пов'язані відношенням агрегації ("has-a"). Тип *Radio* визначає один метод, що включає і вимикає радіо.\

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

public class Radio

{

/// <summary>

/// метод, що вмикає та вимикає радіо

/// </summary>

/// <param name="on"></param>

public void TurnOn(bool on)

{

if (on)

Console.WriteLine("Радіозавади…");

else

Console.WriteLine("….І тиша…");

}

}

}

Клас *Car* (автомобіль) визначає таку поведінку. Якщо користувач об'єкта *Car* перевищить межу для швидкості (ця межа задається значенням члена-константи), то «двигун вибухає» і об'єкт *Car* стає непридатним для використання, що виражається змінами значення поля типу *bool* з ім'ям *carIsDead* (автомобіль зруйнований). Крім того, тип *Car* має кілька членів-змінних, що представляють поточну швидкість та ім'я, дане автомобілю користувачем, а також кілька конструкторів. Ось повне визначення цього типу (з відповідними коментарями).

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

public class Car

{

public const int maxSpeed = 100;// максимум швидкості.

private int currSpeed; // поточна швидкість автомобіля

private string carName; //назва автомобіля

private bool carIsDead;// Працює автомобіль?

private Radio theMusicBox = new Radio();// автомобіль має радіо

//відношення агрегації

/// <summary>

/// Конструктор////////////

/// </summary>

public Car()

{

currSpeed = 0;

carName = "";

}

/// <summary>

/// ////////

/// </summary>

/// <param name="name"></param>

/// <param name="currSp"></param>

public Car(string name, int currSp)

{

currSpeed = currSp;

carName = name;

}

/// <summary>

/// Запит щодо внутрішнього об’єкта/

/// </summary>

/// <param name="state"></param>

public void CrankTunes(bool state)

{

theMusicBox.TurnOn(state);

}

/// <summary>

/// Чи перегрівся автомобіль?/////////

/// </summary>

/// <param name="delta"></param>

public void Accelerate(int delta)

{

if (carIsDead)

Console.WriteLine("не працює "+ carName);

else

{

currSpeed += delta;

if (currSpeed >= maxSpeed)

{

carIsDead = true;

currSpeed = 0;

//створення об’єкта виключення

CarException ex = new CarException(string.Format("{0} перегрівся!", carName));

ex.Data.Add("Дата та час", string.Format("Автомобіль поломався {0}", DateTime.Now));

ex.Data.Add("Причина", " перевищення швидкості");

// використовується оператор throw для генерації виключення

throw ex;

}

else

Console.WriteLine("=› currSpeed = "+ currSpeed);

}

}

}

}

Припустимо, що потрібно створити для користувача виняток (з ім'ям *CarException*), що представляє помилку перевищення швидкості приреченого автомобіля. Насамперед тут має бути створення нового класу з *System.ApplicationExceptio*n (за згодою, класи виключень мають суфікс "*Exception*", що в перекладі означає "виключення"). Як і в разі будь-якого іншого класу, можна визначити для користувача члени, які будуть потім використовуватися в блоці *catch* в рамках програмної логіки викликів. Точно так само можна перевизначити будь-які віртуальні члени, визначені батьківськими класами. Наприклад, можна реалізувати *CarException*, перевизначивши віртуальну властивість *Messagе*.

Якщо ви хочете побудувати точний, призначений для користувача клас виключення, створений вами тип повинен відповідати кращим зразкам, які використовують виключення .NET. Зокрема, призначений для користувача виняток має підпорядковуватися наступним вимогам:

* бути похідним від *Exception* / *ApplicationException*;
* визначати конструктор, який використовується за умовчанням;
* визначати конструктор, який встановлює успадковане властивість *Message*;
* визначати конструктор, що обробляє "внутрішні виключення".

using System;

using System.Collections.Generic;

using System.Text;

namespace except\_example

{

/// <summary>

/// клас користувацького виключення

/// </summary>

public class CarException: ApplicationException

{

private string messageDetails;

public CarException()

{

messageDetails = "";

}

public CarException(string message):base(message)

{

}

/// <summary>

/// Перевизначення властивості Exception.Message

/// </summary>

public override string Message

{

get

{

return string.Format("Повiдомлення про помилку Car: {0}", messageDetails);

}

}

}

}

Метод *Accelerate*() може генерувати виключення, яке сторона, що викликає, повинна бути готова обробити такий виняток. При виклику методу, здатного генерувати виняток, ви повинні використовувати блок*try* **/** *catch*. Прийнявши виняток, ви можете викликати члени типу *System.Exception* і прочитати детальну інформацію про проблему. Що ви будете робити з отриманими даними, залежить, в основному, від вас. Ви можете помістити відповідну інформацію в файл звіту, записати її в журнал реєстрації подій Windows, відправити її електронною поштою адміністратору системи або показати повідомлення з описом проблеми кінцевому користувачеві. Тут ми просто виводимо інформацію у вікно консолі.

В рамках блоку *try / catch*можна також визначити необов'язковий блок *finally*. Завдання блоку *finally* – забезпечити безумовне виконання деякого набору операторів програмного коду, незалежно від наявності або відсутності виключення (будь-якого типу). Для прикладу припустимо, що ви хочете завжди вимикати радіо автомобіля перед виходом з *Main*(), незалежно від винятків.

using System;

namespace except\_example

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\* Створення та тестування автомобiля \*\*\*");

Car myCar = new Car("Zippy", 20);

myCar.CrankTunes(true);

try //спроба виконати дію, яка генерує виключення

{

for (int i = 0; i <= 10; i++)

myCar.Accelerate(10);

}

catch (CarException e) //перехоплення виключення та створення

//об’єкту виключення х відповідними вдастивостями

{

Console.WriteLine("\n\*\*\* Помилка1 \*\*\*");

Console.WriteLine("Метод: {0}", e.TargetSite);

Console.WriteLine("Повiдомлення : {0}", e.Message);

Console.WriteLine("Джерело помилки: {0}", e.Source);

}

catch (ArgumentOutOfRangeException ex) //перехоплення

//іншого виключення

{

Console.WriteLine("\n\*\*\* Помилка2 \*\*\*");

Console.WriteLine(ex.Message);

}

finally // виконується завжди, незалежно від виключень

{

myCar.CrankTunes(false); //вимикання радіо

}

// помилка оброблена, виконується насупнй оператор.

Console.WriteLine("\n\*\*\* Вихiд з обробника виключення \*\*\*");

Console.ReadLine();

}

}

}

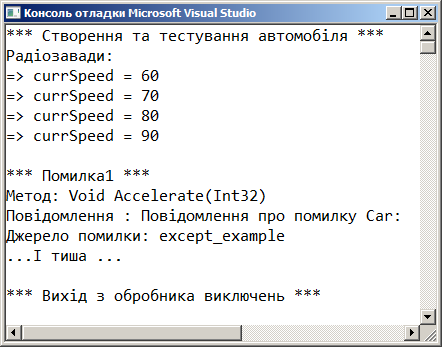


Рисунок 5 - Результат роботи програми обробки виключення

#### 2. Приклад коду застосування делегатів

Делегат – це тип, який представляє посилання на методи з певним списком параметрів і типом значення, що повертається, тобто делегат описує сигнатуру методу. При створенні екземпляра делегата цей екземпляр можна пов'язати з будь-яким методом з сумісною сигнатурою і типом значення, що повертається. Метод можна викликати (активувати) за допомогою екземпляра делегата.

Делегати використовуються для передачі методів в якості аргументів до інших методів. Обробники подій – це ніщо інше, як методи, що викликаються за допомогою делегатів. При створенні користувальницького методу клас (наприклад, елемент керування Windows) може викликати цей метод при появі певної події.

У наступному прикладі показано оголошення делегата:

public delegate тип\_значення\_що\_повертається ім’я\_делегата(список параметрів);

Делегату можна призначити будь-який метод з будь-якого доступного класу або структури, що відповідає типу делегата. Цей метод повинен бути **статичним методом** або методом екземпляра. Така гнучкість дозволяє програмно змінювати виклики методу, а також включати новий код в існуючі класи.

##### Властивості делегата

* Делегати подібні покажчикам на функції в C++, але є повністю об'єктно-орієнтованими і, на відміну від покажчиків C++ на функції-члени, інкапсулюють екземпляр об'єкта разом з методом.
* Делегати допускають передачу методів в якості параметрів.
* Делегати можна використовувати для визначення методів зворотного виклику.
* Делегати можна пов'язувати між собою; наприклад, при появі одної події можна викликати кілька методів.
* Точна відповідність методів типу делегата не потрібна.

##### Приклад коду з використанням делегата

Необхідно змінити рядок символів, видаливши пробіли або замінивши пробіли на інший символ. Операції модифікації рядка символів подати функціями, яким передається вхідний рядок для модифікації. Функції повертають рядок, що змінений відповідно до умови задачі. Для виклику функцій використати делегат, який повертає тип *string*, і приймає в якості параметра рядок.

using System;

using System.Collections.Generic;

using System.Text;

namespace examplDelegat

{

public class TestDelegat

{

//делегат strmodify для модифікації рядку

public delegate string strmodify(string str);

/// <summary>

/// функція, яку викликатиме делегат для заміни пробілів символом'-'

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string replaceSpace(string s)

{

return s.Replace(' ', '-');

}

/// <summary>

/// функція, яку викликатиме делегат для видалення пробілів

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string removeSpace(string s)

{

string temp = " ";

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

if (s[i] != ' ') temp += s[i];

return temp;

}

}

}

Код головної функції, яка використовуватиме делегат, працює так:

1. Створити об’єкт делегат відповідно до його оголошення. В якості параметра використати ім’я функції, яку викликатиме делегат. Використовуються складені імена:

**ім’я класу.ім’я делегата** для об’єкта делегата

**ім’я класу.ім’я функції** для параметра делегата

1. Викликати об’єкт делегата, передавши йому в якості аргументу вхідний рядок
2. Створити новий екземпляр делегата відповідно до його оголошення.
3. Викликати об’єкт делегата, передавши йому в якості аргументу новий рядок
4. Вивести на консоль результати виконання функцій, яким делегат передав управління.

using System;

namespace examplDelegat

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Using delegates");

// створення об'єкту objDelegate типу делегат

//в якості аргументу ім'я функції, яку викликатиме делегат

TestDelegat.strmodify objDelegate = new TestDelegat.strmodify(TestDelegat.replaceSpace);

string stroka = "t e s t 1"; //рядок, що модифікується

Console.WriteLine("вхiдний рядок до виклику делегату = " + stroka);

//виклик функції replaceSpace класу TestDelegat дл заміни пробілів

string stroka1 = objDelegate("t e s t 1");

Console.WriteLine("результат виклику делегата replaceSpace = " + stroka1);

// створення нового екземпляру objDelegate типу делегат

//в якості аргументу ім'я функції, яку викликатиме делегат

objDelegate = new TestDelegat.strmodify(TestDelegat.removeSpace);

//виклик функції removeSpace класу TestDelegat дл видалення пробілів

string stroka2 = objDelegate("t e s t 2");

Console.WriteLine("результат виклику делегата removeSpace = " + stroka2);

}

}

}

#### 3. Приклад коду використання подій і делегатів для їх підтримки

Подія являє собою автоматичне повідомлення про те, що відбулася деяка дія. Події діють за таким принципом: об'єкт, що виявляє інтерес до події, реєструє обробник цієї події. Коли ж подія відбувається, викликаються всі зареєстровані обробники цієї події. Обробники подій зазвичай представлені делегатами.

Події є членами класу і оголошуються за допомогою ключового слова **event**. Найчастіше для цієї мети використовується наступна форма:

**event делегат\_події ім’я\_події;**

Як і делегати, події підтримують групову адресацію. Це дає можливість декільком об'єктам реагувати на повідомлення про подію.

Методи екземпляра і статичні методи можуть бути використані в якості обробників подій, але між ними є одна істотна відмінність. Коли статичний метод використовується в якості обробника, повідомлення про подію поширюється на весь клас. А коли в якості обробника використовується метод екземпляра, то події адресуються конкретним екземплярів об'єктів. Отже, кожен об'єкт певного класу, якому потрібно отримати повідомлення про подію, повинен бути зареєстрований окремо. На практиці більшість обробників подій є методи екземпляра.

using System;

namespace evenApp1

{

class Program

{

static void Main(string[] args)

{

MyEvent evt = new MyEvent();

UserInfo user1 = new UserInfo(Name: "Alex", Family: "Erohin", Age: 18);

Console.WriteLine("Name:" + user1.Name+" Family="+ user1.Family +" Age="+ user1.Age);

// Додати обробник події

evt.UserEvent += user1.UserInfoHandler;

// Запустити подію

evt.OnUserEvent();

Console.ReadLine();

}

}

}

Клас події на базі делегата

using System;

using System.Collections.Generic;

using System.Text;

namespace evenApp1

{

delegate void UI();

class MyEvent

{

// оголошення події

public event UI UserEvent;

// використання методу для запуску події

public void OnUserEvent()

{

UserEvent();

}

}

}

Клас обробника події

using System;

using System.Collections.Generic;

using System.Text;

namespace evenApp1

{

class UserInfo

{

string name, family;

int age;

public UserInfo(string Name, string Family, int Age)

{

name = Name;

family = Family;

age = Age;

}

public string Name

{

set { name = value; }

get { return name; }

}

public string Family

{

set { family = value; }

get { return family; }

}

public int Age

{

set { age = value; }

get { return age; }

}

// Обробник події

public void UserInfoHandler()

{

Console.WriteLine("Подiя викликана!\n");

Name = " Serg";

Family = "Khmelnyk";

Console.WriteLine("Iм'я: {0}\nПрiзвище: {1}\nВiк: {2}", Name, Family, Age);

}

}

}

### Варіанти завдань для лабораторної роботи № 6. 9 Варіантів редаговані

#### Етапи виконання лабораторної роботи

Завдання не пов’язані з предметними областями лабораторних робіт №№3 − 4 і передбачають такі етапи.

**Етап 1** – об’єктно-орієнтований аналіз (OOA) предметної області лабораторної роботи №6 відповідно до її опису, поданому в варіантів завдання. Об'єктно-орієнтований аналіз дозволяє визначити усі сутності, які є складовими головного об’єкту, та їх дії. *На вході* – опис предметної області, інформаця про складові елементи головного об’єкту, отримана з відкритих джерел даних. *Результат* – модель предметної області у вигляді сукупності сутностей з атрибутами, операціями, що характеризують поведінку сутностей, відповідно до принципів SOLID, опису предметної області лабораторної роботи №6 та зв’язків між сутностями (асоціація, агрегація, композиція). Доцільно на цьому етапі розробити сценарій дій сутностей у вигляді Use Case діаграми.

**Етап 2** – об’єктно-орієнтоване проєктування (OOD). *На вході* – результат об’єктно-орієнтованої декомпозиції та аналізу (ООА), Use Case діаграма як сценарій дій сутностей. *Результат* – UML діаграма класів, яка містить класи і взаємозв'язки між ними. Методи класів мають корелюватися з прецедентами, визначеними в Use Case діаграмі.

**Етап 3** – об’єктно-орієнтоване програмування (OOP). *На вході* – UML діаграма класів і UML Use Case діаграма. *Результат* – код програми відповідно до вимог чистого коду (гарного стилю програмування) та принципів SOLID.

У цій лабораторній роботі слід дотримуватись **SOLID принципів єдиної відповідальності (SRP), відкритості – закритості (OCP), підстановки Лісков (LSP), поділу інтерфейсу** (**ISP**), **інверсії залежностей** (**DIP**)**.**

**Принцип єдиної відповідальності (SRP)** - кожен клас має відповідати тільки за одну область функціональності і мати тільки одну задачу або відповідальність.

**Принципу відкритості – закритості (OCP) -** класи повинні бути відкритими для розширення, але закритими для модифікації. Це означає, що поведінка класу може бути змінена без зміни його вихідного коду, шляхом додавання нового коду, а не модифікації існуючого.

**Принцип підстановки Лісков (LSP) -** класи-нащадки повинні мати можливість замінити батьківські класи без порушення функціональності**.**

**Принцип поділу інтерфейсу** (Interface Segregation Principle - **ISP**) вказує на те, що клієнти не повинні залежати від інтерфейсів, які вони **не використовують** повністю. Слід створювати більш специфічні інтерфейси, які містили б тільки необхідні клієнтам методи.

**Принцип інверсії залежностей** (Dependency Inversion Principle - **DIP**) говорить про те, що модулі верхнього рівня не повинні залежати від модулів нижнього рівня. Обидва рівні мають залежати від абстракцій. Це означає, що залежності між класами слід будувати на основі абстракцій, інтерфейсів або абстрактних класів, а не на конкретних реалізаціях.

Виконання лабораторної роботи передбачає три версії коду, кожна версія в окремому проєкті Visual Studio.В методі *Main*() класу *Program* продемонструвати виклик усіх методів усіх класів.

Усі значення, що розраховуються, записувати до *текстових файлів* методами класу **Service.** Також у файл записувати *протокол роботи програми*, тобто дані, що виводяться на консоль.

#### Рекомендований підхід до виконання

1. Завдання рекомендовано виконувати командою у складі **не більше трьох осіб**. Команди створюються студентами самостійно. Допускається виконання лабораторної роботи №6 **одноосібно** (без командної роботи).
2. Оцінювання кожного члена команди здійснюється **за особистим внеском** в загальний результат роботи.
3. Дозволяється **вільний вибір варіанту завдання**, який цікавий усім членам команди (студентом). Вибраний командою (студентом) варіант завдання **не повинен співпадати** з варіантами завдань інших команд в підгрупі.
4. Команда (або окремі її члени) мають право створювати додаткові класи, додавати поля та методи в класи завдань вибраного варіанту.
5. Допускається **ініціативне розширення** (звуження) функціональності програмної моделі, тобто розробляти «фантастичний» сценарій роботи розумних об’єктів, який демонструє уявні можливості, що можуть з’явитися завдяки розвитку ШІ, IoT та смарт-пристроїв, наприклад, у 2040 році.
6. За ініціативність, креативний підхід та якісну реалізацію таких розширень команда (студент) можуть отримати **додаткові бали**.

#### Реалізація зв’язків між класами

##### Реалізація асоціації класів

Асоціація — це найслабший тип зв’язку між класами, коли один клас використовує інший, але не володіє ним (не створює і не зберігає обов’язкову залежність на все життя об’єкта).

*Ключові ознаки реалізації асоціації:*

1. Незалежність життєвого циклу об'єктів

* клас-учасник асоціації не відповідає за створення або знищення іншого об’єкта;
* об’єкти, зв’язані асоціацією, можуть існувати окремо один від одного.

*Приклад***:** об’єкт *student* може існувати без об’єкта *teacher*.

2. Зовнішній об’єкт передається як параметр

* взаємодія часто реалізується через методи, що приймають інший об’єкт як параметр.

*Приклад***:** public void SendTask(Student student)

3. У межах асоціації один клас може зберігати посилання на інший об’єкт у своєму полі класу. але це не обов'язково і не означає, що клас володіє об’єктом, посилання на який він зберігає.

*Приклад***:** в класі Teacher є поле private Student student; // допустимо, але не обов’язково

4. Слабка зв’язаність (loose coupling)

* класи пов’язані лише через інтерфейс або методи, що знижує залежність між модулями;
* це сприяє гнучкості, масштабованості та тестованості системи.

5. Асоціація може бути однонаправленою або двосторонньою

* однонаправлена (одностороння) асоціація означає, що один клас має інформацію або посилання на інший клас і може з ним взаємодіяти, тоді як інший клас не має жодного уявлення про цей зв’язок;
* двонаправлена (двостороння) асоціація означає, що обидва класи зберігають посилання один на одного, тобто кожен із них може ініціювати взаємодію з іншим.

##### Реалізація агрегації класів

Агрегація — це тип зв’язку «ціле – частина», за якого:

* один клас використовує інші як складові, але не володіє ними повністю;
* частини можуть існувати незалежно від цілого (тобто вони створюються/знищуються не обов’язково разом із головним об’єктом).
* це слабший варіант композиції: структура частин ієрархічна, але життєвий цикл — роздільний.

*Ключові ознаки реалізації агрегації:*

* об'єкт одного класу передається іншому класу через конструктор, метод, або встановлюється через властивість (property);
* об'єкт створюється зовні, а не всередині класу – власника;
* об'єкт можна використовувати повторно в інших контекстах.

*Алгоритм реалізації агрегації:*

Крок 1. Створити агреговані класи.

Крок 2. Створити клас, що агрегує інші класи. До класу, що агрегує, додати як поля закриті агреговані об’єкти у вигляді *private ім’я\_агреговного\_класу ім’я\_агрегованого\_ об’єкту;*

Крок 3. Ініціалізувати агрегований об’єкт присвоївши йому значення відповідного об’єкту, який передається як параметр в конструктор класу, що агрегує.

Крок 4. Використати агрегований об’єкт. Такі об’єкти створюються окремо за допомогою оператора new і конструктора агрегованих класів з відповідними аргументами. Далі вони використовуються відповідно до сценарію роботи програми.

##### Реалізація композиції класів

Композиція — це жорсткий тип зв’язку «ціле – частина», за якого:

* один клас «володіє» іншим і повністю контролює його життєвий цикл.
* якщо об’єкт-композитор (контейнер) знищується, то разом з ним знищуються й об’єкти, які є його частинами.

*Ключові ознаки реалізації композиції:*

* об’єкт-власник створює або керує створенням частин всередині себе;
* частини не існують самостійно, лише в контексті контейнера (власника);
* коли контейнер знищується, знищуються й усі компоненти.

*Алгоритм реалізації композиції:*

Крок 1. Створити класи-компонентів.

Крок 2. Створити клас-контейнер, який компонує частини, тобто об’єкти-компоненти. Для цього в клас-контейнер включити як закриті поля об’єкти-компоненти у вигляді *private ім’я\_класу\_композиту ім’я\_об’єкту\_компоненту;*

Крок 3. Додати відкриті властивості (гетери, сетери) до закритих полів – компонентів.

Крок 4. Створити об’єкти-компоненти в конструкторі класу-контейнера, застосувавши оператор new та конструктори класів компонентів з відповцідними аргументами.

Крок 5. Використати об’єкт класу контейнера, в якому створені та існують об’єкти-компоненти. Безпосередній доступ до методів об’єктів-компонентів дозволений, якщо об’єкти-компоненти включені в клас-контейнер як відкриті (public) об’єкти або в клас –контейнер включені відкриті властивості (гетери, сетери).

|  |  |
| --- | --- |
| **№  варіанту** | **Зміст завдання** |
| **1**  **1** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного холодильника»**, вбудованими в нього електронними пристроями, програмним забезпеченням, сенсорними датчиками, агоритмами штучного інтелекту, а також логікою взаємодії з користувачем. Опис пристроїв звичайного холодильника поданий за посиланням <https://www.moyo.ua/news/kakie_byvayut_holodilniki_5_raznovidnosteyi_holodilnyh_ustanovok.html>.  Розумний холодильник — це не просто побутова техніка, це асистент зі здорового харчування, гарного настрою та емоційної підтримки.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумного холодильника». Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.1 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.1. Основні сутності предметної області розумного холодильника   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумний холодильник | Smart прилад з автоматичним керуванням | Складається з: ізотермічної шафи, електрообладнання, мікропроцесору, ШІ-пристроїв | Головна сутність (контейнер, в який вбудовані (композиція) або агреговані інші пристрої | Модель холодильника, Енергоспоживання,  Об’єм | Вмикати, Вимикати, Самодіагностувати, Визначити статус для різних сутностей,  Оновити ПЗ,  Аналізувати споживання продуктів,  Рекомендувати завантаження продуктів | | Ізотермічна шафа | Камера з теплоізоляцією | Має:  Корпус, Двері, Теплоізоляцію | Композиція | Матеріал,  Об’єм | Відкрити двері,  Закрити двері,  Виміряти температуру | | Корпус | Зовнішній каркас | Частина ізотермічного шкафа | Композиція | Матеріал,  Колір | Перевірити цілісність, Очистити поверхню | | Двері | Забезпечує доступ | Частина ізотермічного шкафа | Композиція | Тип відкривання, Тип ущільнення | Відкрити,  Закрити,  Контроль ущільнення | | Теплоізоляція | Захист від втрат температури | Частина ізотермічного шкафа | Композиція | Матеріал, Товщина | Оцінити ефективність | | Електричне обладнання | Забезпечує охолодження | Компресор, Двигун, Випарник, Конденсатор, Терморегулятор, Автоматика | Агрегація | Тип системи | Запустити систему, Зупинити систему,  Виявити несправність | | Компресор | Стискає холодоагент | Частина електрообладнання | Композиція | Потужність | Старт,  Стоп,  Моніторити параметри | | Електродвигун | Приводить компресор у дію | Частина електрообладнання | Композиція | Тип,  Потужність | Запустити,  Перевірити навантаження | | Холодоагент | Переносить тепло | Частина електрообладнання | Композиція | Тип,  Маса | Перевірити рівень,  Визначити тип | | Випарник | Випаровування холодоагенту | Частина електрообладнання | Композиція | Температура | Контролювати охолодження | | Конденсатор | Конденсація холодоагенту | Частина електрообладнання | Композиція | Тип охолодження | Відводити тепло, Діагностувати стан | | Терморегулятор | Контроль температури | Частина електрообладнання | Композиція | Діапазон температур | Встановити температуру, Зчитати температуру | | Прилади автоматики | Датчики, реле | Частина електрообладнання | Композиція | Тип | Реагувати на події, Оповістити про подію,  Контролювати навантаження | | Вбудований мікропроцесор | Центр керування | Частина холодильника | Композиція | Частота,  Пам'ять | Виконати алгоритм, Передати дані, Керувати обладнанням | | ШІ-пристрої | Пристрої та модулі, що реалізують ШІ-функціональність холодильника | Частина розумного холодильника | Композиція | Тип модуля, Виробник | Розпізнати продукти, Прогнозувати потреби, Аналіувати ззвички  користувача,  Рекомендувати продукти для закупівлі | | AI-модуль | Виконує машинне навчання або логіку знань | Частина ШІ-пристроїв | Композиція | Модель,  Тип алгоритму | Навчатись,  Генерувати рекомендації, Аналізувати дані споживання | | Камера (у складі ШІ) | Оптична система для розпізнавання продуктів | Частина ШІ-пристроїв | Композиція | Роздільна здатність | Зробити знімок, Ідентифікувати продукти | | Wi-Fi модуль | Забезпечує підключення до Інтернету | Частина холодильника | Агрегація | Протокол, Швидкість | Підключитись,  Передати дані,  Отримати оновлення | | Сенсори | Датчики контролю | Частина холодильника | Агрегація | Температура, Вага, Присутність, вологи | Збирати дані,  Сигналізувати відхилення | | Голосовий помічник | Керування холодильником через голосові команди | Автономний пристрій | Асоціація | Тип голосу (чоловічий, жіночий),  Мова | Будь-які дії користувача |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один холодильник має по одному** агрегованому чи композитному, чи асоційованому з перерахованих **пристроїв.** **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного холодильника» в текстовій формі.   * + - 1. Перевірка стану здоров’я користувача на основі даних із синхронізованих смарт-пристроїв для визначення раціону харчування.       2. Формування рекомендацій по здороому харчуванню відповідно до дієти, наявності продуктів в холодильнику, дотримання балансу вуглеводів, жирів та білків тощо.       3. Розробка персоналізованих рецептів на вимогу користувача, автоматичне попередження про відсутність потрібних продуктів, їх термін придатності, їх утилізацію тощо.       4. Аналіз емоційного стану людини, що відкриває холодильник, включення голосового помічника (музика, заспокійливі фрази) для покращення настрою людини, аромотерапія або кольорове освітлення для заспокоєння, рекомендація щодо продуктів для покращення емоційного стану, діалог з голосовим помічником.       5. Стимулювання мотивацій до здорового харчування, висновки про дотримання дієти тощо.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation, association (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний холодильник», відповідно до таблиці 6.1. Кожний пристрій «розумного холодильника» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.1. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.1. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом. алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумного холодильника властивості та операції. Клас «розумний холодильник» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів холодильників з однаковою поведінкою. Клас «розумний холодильник» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний холодильник», наприклад, «**Розумний холодильник з Internet**», який буде реалізовувати з’єднання з Internet, синхронізований з мобільним додатком, хмарними сервісами тощо. Атрибути та методи придумати самостійно.   **Версія 3 – Exceptions (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій (не менше 8 виняткових ситуацій), пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі холодильника:  * низька (надвелика) електрична напруга, що порушує нормальну роботу холодильника; * застаріле програмне забезпечення або збій в його роботі; * відсутність підключення до мережі Інтернет і неможливість оновити ПЗ; * перевищення температури всередині холодильника, що може призвести до псування продуктів; * механічні збої: дверей, вентилятору, сенсорів тощо. * вихід з ладу мікропроцесору; * помилки в роботі AI модулю, або голосового помічника; * неправильні команди, які віддіє користувач через голосовий помічник.   **Версія 4 – Delegate&Events (пункт завдання 6)**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно (не менше 5 подій), або реалізувати обробники таких подій:  * температура всередині холодильника змінюється (наприклад, через неправильну роботу сенсорів або зміни у налаштуваннях), як наслідок – псування продуктів; * споживання енергії холодильником перевищує норму, як наслідок – надлишкові грошові витрати; * відкриті двері холодильника, як наслідок – псування продуктів * зміна програмного забезпечення, як наслідок – покращення інтерфейсу користувача, покращення надійності і безпеки, покращення функціональності або додаткові збої в роботі холодильника тощо; * вихід з ладу будь-якого смарт пристрою, як наслідок – псування продуктів, погана робота голосового помічника, спотворений настрій користувача тощо. |
| **2**  **2** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного автомобіля»**. Опис конструктивних елементів звичайного легкового автомобіля поданий за посиланням <https://surl.li/jkrqys>. «**Розумний автомобіль»** — це трансформований інтелектуальний апарат, здатний не лише пересуватися сушею, водою чи повітрям, а й самонавчатися, адаптуватися до емоційного та фізичного стану пасажирів, вести з ними діалог, приймати рішення в критичних ситуаціях, передбачати потреби користувача до того, як він їх усвідомить, а також синхронізуватися з іншими пристроями та інфраструктурою у спільному цифровому середовищі. Такий автомобіль виконує роль персонального асистента, лікаря, охоронця, пілота й співрозмовника людини.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумного» автомобіля. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.2 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.2. Основні сутності предметної області розумного автомобіля   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Сутність | Роль | Опис зв’язків | Тип зв’язку | Атрибути | Операції | | Розумний автомобіль | Основна система, яка компонується з інших частин | Об'єднує всі основні пристрої та системи | Композиція | Ідентифікатор, Модель,  Тип трансформації, Кількість пасажирів | Трансформуватися, Виконати команду, Активувати безпеку, Регулювати клімат, Визначити стан водія | | Кузов | Зовнішній каркас | Частина автомобіля | Композиція | Матеріал,  Колір,  Герметичність | Відкрити двері, Змінити форму | | Двигун | Джерело руху | Компонент транспортної системи | Композиція | Тип (Електричний/Гібрид),  Потужність,  Стан | Запустити,  Зупинити,  Змінити режим | | Шасі | Несуча система | Складається з ходової частини, трансмісії, гальм, керма | Композиція | Тип підвіски,  Маса | Стабілізувати рух,  Змінити кліренс (автоматичне підняття кузова або адаптація висоти під час паркування або посадки пасажирів) | | Ходова  частина | Підсистема шасі | Компонент шасі | Композиція | Тип приводу,  Стан | Адаптувати під покриття | | Тормозна  система | Гальмування | Взаємодіє з системою безпеки | Композиція | Тип гальм, Ефективність | Активувати гальмування, Екстрене гальмування | | Рульове  управління | Керування напрямком | Входить до шасі | Композиція | Тип (Механічне/Електронне)  Чутливість | Змінити напрямок, Активувати автопілот | | Трансмісія | Передача моменту | Входить до шасі | Композиція | Тип,  Кількість передач | Перемикнути передачу | | Smart  система | Інтелектуальне керування | Пов’язана з усіма частинами, включаючи користувача | Агрегація | Рівень автонгомності, Здатність системи обмінюватися даними з іншими пристроями,  Прогр забезпечення | Розпізнати голос,  Визначити настрій, Взаємодіяти з користувачем,  Заблокувати доступ,  Зібрати аналітику | | Сенсор  стану  водія | Виявлення стану водія | Взаємодіє зі Smart Системою та безпекою | Агрегація | Типи сенсорів (Пульс, Тиск, Очі), Порогові значення | Перевірити стан,  Подати сигнал,  Заблокувати авто | | Голосова  система | Взаємодія з людиною | Комунікація з користувачем, SmartСистемою | Асоціація | Мова,  Стать Голосу,  Тип синтезу мови та звуків | Озвучити повідомлення, Розпізнати команду | | Модуль  трансформації | Перетворення транспортної форми | Входить до автомобіля, активується у спеціальних умовах | Композиція | Режими (Підводний, Надводний, Літальний),  Статус | Активувати режим, Перевірити готовність, Трансформувати | | Система  КліматКонтролю | Комфорт усередині салону | Пов’язана з сенсорами температури і вологості | Агрегація | Температура, Вологість,  Режим | Регулювати температуру, Підтримувати баланс клімату | | Система  безпеки | Захист користувачів | Взаємодіє з шасі, Smart Системою, сенсорами | Агрегація | Тип захисту, Активність | Активувати подушки безпеки,  Повідомити про загрозу, Заблокувати двигун | | Модуль штучного інтелекту | Інтегрована система прийняття рішень | Пов’язаний із сенсорами, SmartСистемою, інтерфейсами користувача, трансформаційними та безпековими модулями | Агрегація | Версія,  Режим роботи, Рівень автономності | Аналізувати дані,  Прийняти рішення, Адаптувати поводження, Оновити модель | | Модуль розпізнавання мовлення | Обробка голосових команд | Пов’язаний з голосовою системою | Асоціація | Мова,  Словник,  Точність | Розпізнати команду, Навчити новим фразам | | Модуль комп’ютерного зору | Аналіз навколишнього середовища | Отримує дані з камер, взаємодіє з модулем прийняття рішень | Агрегація | Кількість камер, Типи об'єктів, Алгоритми | Розпізнати об’єкт,  Оцінити відстань,  Відстежити рух | | Модуль діагностики стану водія | Визначення стану здоров’я водія | Взаємодіє з сенсорами стану, модулем безпеки | Агрегація | Порогові значення, Історія стану | Аналізувати стан, Викликати допомогу, Попередити про ризик | | Модуль рекомендацій | Надає поради та пропозиції користувачу | Взаємодіє з кліматом, навігацією, розкладом | Асоціація | Типи рекомендацій, Профіль користувача | Порадити маршрут, Запропонувати музику, Нагадати про зустріч | | Модуль самонавчання | Постійне вдосконалення поведінки | Пов’язаний із усіма іншими ШІ-модулями | Агрегація | Кількість епізодів, Алгоритм (наприклад, алгоритм машинного навчання Reinforcement Learning) | Оновити модель, Адаптувати рекомендації, Зберегти досвід | | Модуль прогнозування | Передбачення аварій, змін у середовищі | Взаємодіє з модулем зору, сенсорами, діагностикою | Агрегація | Типи подій,  Точність прогнозів | Спрогнозувати загрозу, Оцінити ймовірність аварії | | Модуль емоційної підтримки | Виявлення та реагування на настрій користувача | Взаємодіє з голосовим інтерфейсом, діагностикою та системою клімату | Асоціація | Емоційний стан, Рівень стресу | Запропонувати пораду, Увімкнути релакс музику, Активувати світлотерапію |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один автомобіль має по одному** агрегованому чи композитному, чи асоційованому з перерахованих **пристроїв.** **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного автомобіля» в текстовій формі.   * + - 1. Активація та медичний моніторинг при посадці користувача на основі даних, зчитаних сенсорами: пульс, тиск, вологість шкіри, пульсова хвиля, аналіз дихання та очей на ознаки сп’яніння або втоми.       2. Аналіз модулем ШІ ризиків (стрес, тахікардія, алкоголь, наркотики) та прогноз настання критичних станів.       3. Вибір режиму пересування (наземний, плавальний, аерорежим) та трансформація автомобіля в човен або літальний апарат в залежності від поточної ситуації.       4. Задіяти ШІ асистент для голосового управління, голографічного проєктора, налаштування мовної моделі на стиль спілкування користувача       5. Контроль мікроклімату, включити сенсори температури, вологості, СО₂, хімічних сполук, оптимізація мікроклімату, персоналізація середовиша для кожного пасажира.       6. Включити та налаштувати систему безпеки та самозахисту: * налаштувати систему комп’ютерного зору для аналізу пішоходів, тварин, небезпеки на дорогах; * прогноз ризику аварій, автогальмування; * включенння системи захисту (силове поле, автозамикання тощо).   + - 1. Автономна навігація та адаптація до маршрутів: * зміна маршруту, обхід заторів, вибір спокійного маршруту тощо   + - 1. Нічний режим       2. Самообслуговування (автономна діагностика, голосове повідомлення )   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний автомобіль», відповідно до таблиці 6.2. Кожний пристрій «розумного автомобіля» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.2. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.2. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 - Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумного автомобіля властивості та операції. Клас «розумний автом обіль» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів автомобілів з однаковою поведінкою. Клас «розумний автомобіль» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний автомобіль», наприклад, «**Машина час**у», яка буде реалізовувати переміщення в часі, «біо автомобіль», який здатен до самогенерації у випадку аварії, адаптації до користувача за фізичними ощнаками, емоціями, станом здоров’я тощо. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі розумного автомобіля (**не менше 8 виняткових ситуацій**):  * автомобіль не може інтерпретувати запит користувача через зміну контексту; * користувач намагається сісти за кермо, але дані не збігаються з профілем; * спроба перейти в режим човна, але глибини недостатньо; * водій у передінфарктному або в стані алкогольного чи наркотичного сп’яніння; * оновлення ПЗ спричинило конфлікт у системі навігації; * надмірна кількість паралельних команд від користувача; * помилки в роботі AI модулю, або голосового помічника * неправильні команди, які віддіє користувач через голосовий помічник.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * виявлено водія у стані сп’яніння. Як наслідок – блокування автомобіля; * наближення аварії (ризик зіткнення). Як наслідок – аварійне гальмування, запуск подушок безпеки, надсилання координат аварійної службі; * збій GPS або зв’язку. Як наслідок – перехід на автономну навігацію, пошук найближчої зони покриття; * зміна настрою користувача (стрес, злість, втома). Як наслідок – адаптація освітлення, музики, активування режиму «антистрес» тощо; * раптове погіршення здоров’я водія (високий тиск, непритомність). Як наслідок – автоматичне зупинення, виклик швидкої, активація медичного режиму (охолодження, вентиляція). |
| **3** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного годинника»**. Опис функціональних можливостей сучасних смарт годинників поданий за посиланням <https://surl.gd/psmmof>. «**Розумний годинник» майбутнього** — це персоналізований нейроінтерфейсно-керований мікрокомп'ютер, який у реальному часі синхронізується з фізіологічним, емоційним станом і когнітивними патернами користувача, забезпечуючи адаптивну взаємодію на рівнях, недосяжних для автономного ШІ. Смарт-годинник майбутнього виконуватиме роль персонального асистента, лікаря, охоронця, співрозмовника, когнітивного партнера, аналітика рішень, емоційного стабілізатора, контролера безпеки.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумного» годинника. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.3 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.3. Основні сутності предметної області розумного годинника   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Сутність | Роль | Опис зв’язків | Тип зв’язку | Атрибути | Операції | | Смарт-годинник | Головна система | Об’єднує всі модулі, координує роботу | Композиція | Id,  Серійний номер, Модель,  Статус живлення | Вмикнути,  Вимкнути,  Діагностувати,  Оновити ПЗ | | Біоінтерфейс | Взаємодія з тілом | Спілкується з сенсорами, передає дані в інші модулі | Агрегація | Тип,  Рівень точності, Активність | Зчитати параметри, Аналізувати дані, Стимулювати | | Нейроінтерфейс | Керування думкою | Зв'язок з мозком, керування командами | Асоціація | Калібровка, Чутливість, Адаптивність | Синхронізувати,  Виконати команду,  Вивчити модель | | Капсула життєзабезпечення (наношприц, біосенсори, адаптивне дозування з AI-керуванням) | Медичний вплив | Вбудований мікробіомедичний модуль, що підтримує життєві функції користувача на критичному рівні | Композиція | Рівень заряду,  Список медичних речовин (адреналін, глюкоза, седативні, вітаміни тощо) | Вколоти ін’єкцію,  **Відновити функціональний ресурс капсули,**  Оновити склад речовин | | Модуль візуалізації | Візуальний інтерфейс, який замінює традиційний екран | Виводить 3D зображення або HUD-інтерфейс (голограма, проєкція на лінзу окулярів, відображення на сітківці ока, вртуальна реальність) | Агрегація | Тип виводу, Роздільна здатність, Режим адаптації | Показати,  Змінити інтерфейс, Переключити режим | | Аналізатор настрою | Емоційна адаптація | Аналізує емоції, впливає на вигляд та функціонал | Асоціація | Рівень стресу, Емоційний фон, Історія настрою | Визначити настрій, Адаптувати інтерфейс, Повідомити про душевний, емоційний, фізичний стан користувача, рекомендації для нормалізації стану тощо | | Модуль телепортації | Переміщення | Використовує координати з нейроінтерфейсу | Композиція | Заряд,  Дальність, Рівень ризику | Ініціювати телепорт, Скасувати телепорт, Розрахувати маршрут | | Сенсори | Збір даних | Працюють у зв'язці з біо- та нейроінтерфейсом | Агрегація | Температура,  ЧСС, ЕКГ, GPS, Рівень глюкози | Зчитати дані,  Калібрувати,  Надіслати звіт | | Пам’ять / AI-ядро | Інтелектуальний центр | Зберігає, аналізує, навчається | Композиція | Обсяг пам'яті,  Рівень AI (ступінь складності, автономності, адаптивності AI),  Параметри навчання (Типи вхідних даних, Алгоритми навчання, Частота оновлення моделей тощо ) | Передбачити,  Навчитися,  Очистити пам’ять | | Комунікаційний модуль | Зв'язок | Працює з 5G/6G, хмарами, пристроями | Агрегація | Тип зв’язку, Пропускна здатність, Статус з’єднання | З’єднатись,  Надіслати,  Прийняти,  Зашифрувати | | Контролер безпеки | Охоронець для виявлення загроз, захист користувача та втручання в критичних ситуаціях | Взаємодіє з AI-ядром | Композиція | Рівень\_загрози, Поточна оцінка загрози (низький / середній / високий / критичний),  Режим охорони,  Геолокація | Сповістити,  Зафіксувати,  Активувати захист | | Живлення | Енергозабезпечення | Забезпечує живлення всіх модулів | Композиція | Рівень заряду, Джерело енергії, Режим живлення | Зарядити,  Переключити режим, Вимкнути модуль | | Календар і завдання | Нагадування, тайм-менеджмент, прогноз | Пов'язаний зі ШІ, модулем настрою,модулем візуалізації | Агрегація | Назва події,  Дата, Час, Місце, , Приоритет | Створити подію,  Видалити подію,  Перенести подію, Синхронізувати,  Нагадати,  Адаптувати план | | Повідомлення / месенджери | Push-повідомлення, | Взаємодіє з контактною книгою та календарем, пов'язаний зі ШІ, модулем настрою, модулем візуалізації | Асоціація | Тип повідомлення (текст / голос / медіа),  Відправник, Час, Емоційне забарвлення,  Пріоритет,  Статус | Отримати,  Створити,  Відправити |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один годинник має по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного годинника» в текстовій формі.   1. Активація та ранковий медичний моніторинг:  * аналізує якість сну, тривалість відпочинку, частоту серцевих скорочень, дихання, тривожні маркери; * зчитує мікросигнали з нейроінтерфейсу, порівнює з попередніми шаблонами; * пропонує оптимальний ритм дня, враховуючи заплановані зустрічі, емоційний баланс і розумове навантаження.  1. Адаптація в реальному часі протягом дня:  * відстежує стрес і втому через пульс, міміку, мову, темп ходьби; * автоматично змінює інтерфейс на спрощений, зменшує кількість вхідних повідомлень; * фіксує, коли виникає напруження або зниження концентрації, і пропонує (дихальну вправу, мікроперерву, аудіостимуляці.  1. Здійснення комунікації відподно до контексту:  * показує важливе повідомлення та фільтрує несуттєві; * пропонує шаблон відповіді, враховуючи стиль мови, емоційний фон контакту; * визначає, чи варто відповідати відразу, чи відкласти.  1. Реалізація безпеки і оцінка середовища:  * оцінює геолокацію, шум, атмосферний тиск, щільність натовпу; * активує пасивний захист або аварійний зв'язок у небезпечному середовищі; * ініціює маршрут до безпечної зони, екстрене сповіщення, телемедичну допомогу.  1. Інтеграція досвіду після трудового дня:  * підсумовує день: ментальне навантаження, зміни настрою, продуктивність; * пропонує висновки і, за згодою, оновлює поведінкові шаблони; * наслідує вечірній режим, фільтрує інформаційний потік, активує відновлення.  1. Фонове спостереження за користувачем в процесі сна:  * вимірює глибину і фазу сну; * модулює сенсорне середовище (світло, звук); * при критичних станах активує медичний протокол або зв’язується з контактом довіри.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний годинник», відповідно до таблиці 6.3. Кожний компонент «розумного годинника» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.3. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.3. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумного годинника властивості та операції. Клас «розумний годинник» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів годинників з однаковою поведінкою. Клас «розумний годинник» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний годинник», наприклад, «**Віртуальний самоадаптивний зовнішній шар свідомості**», який здатен до самонавчання, самозахисту та етичного коригування поведінки користувача, може розпізнавати і передбачати внутрішні конфлікти користувача і трансформувати інтерфейс, ритм повідомлення, темп взаємодії, щоб уникнути кризових станів. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі розумного годинника (**не менше 8 виняткових ситуацій**):  * ШІ прогнозує агресію користувача, проте людина залишається спокійною й логічною; * користувач щиро усміхається, але міміка не збігається з біосигналами: годинник фіксує тривогу, панічну атаку тощо; * ШІ виявляє конфлікт між очікуваною та фактичною емоційною поведінкою користувача, наприклад, очікувана поведінка – скорбота, фактична – радість; * годинник автоматично фільтрує повідомлення, які мають нестандартну форму, але фактично утворюють критично важливу інформацію; * користувач ігнорує рекомендації годинника залишити небезпечну зону або припинити фізичне навантаження, при очевидних ризиках для здоров'я; * модуль нейроінтерфейсу сприяє передачі сигналів із затримкою, через те, що ШІ приймає рішення на основі старих програм користувачів; * годинник починає надсилати фальшиві тривожні сигнали системи безпеки або контактам довіри через збій у сенсорах ЧСС та температури; * ШІ-ядро проводить цикл самонавчання без зовнішнього контролю, і намагається перебудовувати шаблони взаємодії без врахування зміни емоційного стану користувача; * проекційний модуль деформує яскраве зображення — показує розмітку або перевернуту голограму в HUD-інтерфейсі.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * зміна частоти серцевих скорочень (ЧСС), підвищення/зниження температури тіла, виявлення зміни частоти дихання. Дії годинника – активація медичного моніторингу; * отримано повідомлення з агресивним або травматичним змістом. Дії годинника – заспокійливе повідомлення або запит на оцінку важливості отриманого повідомлення; * втрата зв'язку з мережею / ізольованість у зоні без покриття. Дії годинника – активація автономного режиму безпеки, запуск резервного енергозбереження; * зміна настрою користувача (стрес, злість, втома). Дії годинника – активування режиму «антистрес» або режим ізоляції з відключенням повідомлень тощо; * підозрілий рух або навколишній шум (небезпечне середовище). Дії годинника – активація режиму прихованого моніторингу, надсилання попередження контакту безпеки. |
| **4** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного дзеркала»**. «**Розумне дзеркало» майбутнього** — це інтелектуальний пристрій, який взаємодіє з людиною, аналізує її стан, настрій, здоров'я, цілі, допомагає з вибором стилю одягу, навчанням, психоемоційною підтримкою, і персоналізовано комунікує. Смарт-дзеркало майбутнього виконуватиме роль персонального асистента, лікаря, стиліста, психолога, наставника, охоронця, співрозмовника, когнітивного партнера, аналітика рішень, емоційного стабілізатора, контролера безпеки.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумного» дзеркала. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.4 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.4. Основні сутності предметної області розумного дзеркала   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумне дзеркало | Центральна система, що керує всіма модулями | Містить усі внутрішні модулі та пов’язана з користувачем | Композиція | Id, Модель, Версія, Розташування,  Дата активації | Аналізувати користувача, Відображати інформацію, Взаємодіяти,  Оновити модулі | | Користувач | Людина, яка взаємодіє з дзеркалом | Має зв’язки з емоційним станом, здоров’ям, стилем, профілем | Асоціація | Id, Ім’я, Вік, Стать, Профіль здоров’я, Емоційний профіль, Смаки, Цілі, Розклад | Оновити профіль,  Отримати рекомендації,  Взаємодіяти з дзеркалом | | Віртуальна персона | Індивідуалізований цифровий співрозмовник | Пов’язаний з користувачем, інтерфейсом діалогу та емоційним аналізатором | Асоціація | Ім’я,  Тип особистості, Стиль спілкування, Рівень емпатії | Персоналізувати відповіді, Мотивувати користувача, Будувати довіру | | Фотоапарат або камера | Зйомка користувача та об’єктів | Частина дзеркала | Композиція | Роздільна Здатність, Режим Зйомки | Зробити фото,  Передати фото,  Визначити обличчя | | Модуль відеозв’язку | Здійснення аудіо- та відеодзвінків | Частина дзеркала | Композиція | Якість відео,  Канал передачі | Розпочати дзвінок, Прийняти дзвінок, Завершити дзвінок | | Wi-Fi / Bluetooth модуль | Комунікація з іншими пристроями | Додатковий зв'язаний компонент | Агрегація | Тип мережі, Протокол,  Швидкість | Підключитися до мережі, Передати/отримати дані | | Аналізатор емоцій | Визначає настрій користувача | Частина дзеркала, пов’язаний із співрозмовником і віртуальним помічником | Композиція | Настрій,  Вираз обличчя,  Тон голосу | Розпізнати емоції,  Порадити заспокоєння | | Детектор стану людини | Визначає фізіологічні й емоційні параметри | Інтегрований, працює з користувачем | Композиція | Температура,  Пульс,  Тиск,  Емоційний стан | Зчитати параметри, Визначити емоції,  Вивести попередження | | Монітор здоров’я | Аналіз фізіологічного стану | Агреґований модуль, працює з сенсорами | Агрегація | Пульс,  Температура шкіри, Рівень стресу,  Якість сну | Аналізувати показники, Вивести рекомендації, Виявити ризик | | Аналізатор тіла | Оцінює індекс маси тіла (ІМТ), склад тіла | Працює з відображенням людини, інтегрований | Композиція | Зріст, Вага,  Індекс маси тіла (ІМТ),  Відсоток жиру та м’язів | Виміряти ІМТ,  Побудувати графік, Порівняти зміни | | Радник стилю одягу | Рекомендації щодо зовнішнього вигляду | Взаємодіє з гардеробом, користувачем, погодним сервісом | Асоціація | База одягу,  Улюблені стилі одягу,  Актуальні тренди,  Погода | Згенерувати образ,  Показати тренди,  Поради по стилю | | Радник зачіски/зовнішності | Допомагає змоделювати образ | Працює з фото та відео | Композиція | База зачісок, Кольорів,  Стилів зовнішності | Застосувати стиль, Переглянути варіанти, Зберегти образ | | Модуль штучного інтелекту | Інтегрована система прийняття рішень | Працює з іншими модулями | Агрегація | Програмне забезпечення  Тип моделі нейронної мережі,  Рівень навчання моделі,  Джерело знань (Internet, БЗ)  Стан ПЗ  Час реакції на відовіді,  Мова взаємодії,  Режим (діалог, аналітика, рекомендації) | Генерація відповідей мовним синтезатором,  Адаптація рекомендацій до настрою та контексту. | | Голосовий помічник | Інтерфейс взаємодії голосом | Працює з іншими модулями | Агрегація | Мова,  Тип голосу,  Рівень розпізнавання | Відповісти на питання, Озвучити результат, Розпізнати команду | | Розкладник | Персональний асистент для організації активностей користувача та нагадування про заплановані події | Пов’язаний із календарем користувача та зовнішніми сервісами | Асоціація | Події,  Нагадування,  Цілі,  Важливідати | Додати подію, Синхронізувати календар, Показати розклад | | Навчальний  модуль | Когнітивний партнер для постійного навчання та саморозвитку | Пов’язаний з профілем користувача та його цілями | Асоціація | Навчальні цілі, Поточнийрівень, Рекомендовані теми, Прогрес | Рекомендувати контент, Відслідковувати прогрес, Перевірити знання | | Сенсор довкілля | Збір інформації про навколишнє середовище для формування порад щодо здоров’я, настрою, зовнішнього вигляду та активностей | Частина дзеркала, дані передаються в інші модулі | Композиція | Температура, Вологість, Освітлення,  Якість повітря | Аналізувати довкілля, Надати поради щодо здоров’я, настрою, зовнішнього вигляду та активностей |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **одне дзеркало має по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного дзеркала».   1. Ідентифікація та запуск сесії (задіяні фотоапарат, модуль ШІ для розпізнавання обличчя, Wi-Fi для роботи з БД) 2. Оцінка фізичного та емоційного стану (задіяні Детектор стану людини, Аналізатор емоцій, Wi-Fi для передачі даних ШІ, модуль ШІ для класифікації стану) 3. Інформація про порядок денний (задіяні Розкладник, Wi-Fi, модуль ШІ, Голосовий помічник) 4. Поради щодо зовнішності (задіяні Радник стилю одягу, Радник зачіски/зовнішності, Сенсор довкілля, модуль ШІ, Голосовий помічник) 5. Відеозв'язок (задіяні Модуль відеозв’язку, Wi-Fi для передачі даних ШІ, модуль ШІ) 6. Освітній сегмент (задіяні модуль ШІ, Голосовий помічник) 7. Емоційне відновлення (автоматична ініціалізація) (задіяні Аналізатор емоцій, модуль ШІ, Голосовий помічник) 8. Завершення сесії (задіяні Wi-Fi для передачі даних ШІ, модуль ШІ, Голосовий помічник )   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумне дзеркало», відповідно до таблиці 6.4. Кожний компонент «розумного дзеркала» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.4. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.4. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумного дзеркала властивості та операції. Клас «розумне дзеркало» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів дзеркал з однаковою поведінкою. Клас «розумне дзеркало» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумне дзеркало», наприклад, «**Віртуальне нейроадаптивне дзеркало**», яке взаємодіє з користувачем через нейроінтерфейс, зчитує сигнали мозкової активності, емоційний стан, когнітивні наміри та адаптує інформаційне і візуальне середовище в режимі реального часу. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі розумного дзеркала (**не менше 8 виняткових ситуацій**):  * втрата зображення користувача через несправність фотоапарата або камери; * невизначений емоційний стан через перешкоди для аналізу міміки або суперечливі сигнали; * немає підключення до мережі Wi-Fi через проблеми з мережею або маршрутизатором; * конфлікт когнітивного/емоційного стану через нестабільні нейросигнали; * помилка розкладу через пошкодження модуля з календарем; * збої в роботі модуля ШІ через перевантаження його запитами; * несинхронізований профіль користувача через йогог пощкодження; * збій модуля навчання через логічні або технічні помилки, як-от: перенавчання, втрата пам’яті, ресурсне обмеження тощо. * не працює голосовий помічник через відсутність активацій, зовнішній шум, збій або відсутність драйвера для аудіообладнання тощо.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * ідентифікація користувача. Дії дзеркала – активація алгоритмів розпізнавання облич; * визначення емоційного стану користувача. Дії дзеркала – розпізнавання емоцій: радість, сум, скорбота тощо; * втрата зображення користувача. Дії дзеркала – виведення повідомлення про помилку, активація альтернативних методів автентифікації; * невизначений запит до ШІ. Дії дзеркала – запит на уточнення або пропозиція альтернативних варіантів запитів, пошук схожих запитів в історії взаємодії; * неадекватна взаємодія з голосовим помічником. Дії дзеркала – реєстрація помилки з фіксацією в журналі помилок розпізнавання мовлення, активація альтернативного інтерфейсу тощо. |
| **5** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумних окулярів»**. «**Розумні окуляри» майбутнього** — це інтелектуальний пристрій, який взаємодіє з людиною, аналізує її стан, настрій, здоров'я, цілі, допомагає з вибором стилю одягу, навчанням, психоемоційною підтримкою, і персоналізовано комунікує. Смарт-окуляри майбутнього виконуватиме роль не просто гаджета, а слугуватимуть розширенням людських сенсорних, когнітивних та комунікативних можливостей, формуючи нову форму взаємодії з інформаційним і фізичним середовищем, зокрема мозково-комп’ютерний інтерфейс, цифровий провідник у навколишньому середовищі, особистий охоронець та аналітик безпеки, інтерпретатор реальності, голографічний комунікатор, контекстно-залежний радник, інструмент навчання  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумних» окулярів. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.5 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.5. Основні сутності предметної області розумних окулярів   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумні окуляри | Головний пристрій | Об'єднує всі інші модулі | Композиція | Id,  Модель,  Виробник, Енергоспоживання | Вмикнути,  Вимкнути,  Синхронізувати,  Оновити | | Когніто-інтерфейс | Зчитування нейросигналів | Вбудований модуль | Композиція | Тип сигналу,  Рівень інтенсивності | Зчитати сигнал, Калібрувати,  Передати команду | | Лінза-навігатор (AR) | Візуалізація даних у полі зору | Вбудована в оправу | Композиція | Прозорість, Роздільна здатність, Фокус | Відобразити,  Оновити картинку | | Лазер для проєкції на сітківку | Безпосередня проєкція зображення на сітківку | Пов’язаний із лінзою AR | Композиція | Довжина хвилі, Яскравість,  Фокус | Активувати лазер,  Вимкнути лазер | | Голографічний проєктор | Виведення 3D-об'єктів | Працює з AR-лінзою | Композиція | Тип об'єкта, Дальність виводу | Проєктувати,  Припинити проєкцію | | Відеокамера | Захоплення візуального потоку | Інтегрована в оправу | Композиція | Роздільна здатність,  FPS (кількість кадрів за секунду відеокамери),  Нічне бачення | Зняти,  Передати в потік,  Зберегти | | Мікрофон | Захоплення звуку | Взаємодіє з ІІ-модулем | Композиція | Чутливість,  Рівень шумозаглушення | Записати звук,  Передати звук | | Динаміки | Виведення звукової інформації | Частина інтерфейсу для слабозорих | Композиція | Гучність,  Формат виводу | Озвучити текст,  Передати інформацію | | Віртуальний дисплей | Альтернатива фізичному дисплею для зору | Працює з лазером або AR-лінзою | Композиція | Роздільна здатність,  Режим відображення | Відобразити UI,  Налаштувати інтерфейс | | Мікрокомп’ютер | Виконання обчислень, інсталяція ПЗ | Центральний обчислювальний блок | Композиція | CPU, RAM, Обсяг сховища,  OS | Запустити додаток,  Оновити,  Встановити ПЗ | | Компас | Орієнтація в просторі | Пов’язаний із сенсорами навігації | Композиція | Напрямок,  Кут відхилення | Визначити напрям,  Скинути орієнтацію | | Акселерометр | Визначення руху й положення | Взаємодіє з компасом і камерою | Композиція | Прискорення,  Нахил,  Координати | Виміряти рух,  Передати дані,  Аналізувати рух | | Транслятор реальності | Переклад мов у реальному часі | Співпрацює з мікрофоном і вірт. дисплеєм | Асоціація | Мова вхідна,  Мова вихідна, Точність | Перекласти,  Показати субтитри | | Модуль розпізнавання загроз | Виявлення небезпеки | Пов'язаний із відеокамерою, взаємодіє з зовнішнім середовищем | Агрегація | Тип загрози,  Рівень ризику, Точність розпізнавання | Аналізувати оточення,  Сповістити про загрозу | | Біосенсор зору | Контроль здоров’я очей | Зчитує дані безпосередньо з користувача | Агрегація | Вологість очей, Навантаження,  Втома | Попередити про напруження,  Відправити звіт | | Нейро-асистент | Інтелектуальний помічник | Взаємодіє з усіма пристроями через мікрокомп’ютер | Асоціація | Ім’я,  Рівень внтелекту,  Журнал рекомендацій | Дати пораду,  Вивчити звички,  Передбачити дію |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **одні окуляри мають по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумних окулярів».   1. Активація пристрою та ідентифікація користувча (задіяні акселерометр, біосенсор зору, когніто-інтерфейс для зв’язу із мозком, мікрокомп’ютер) 2. Орієнтація у просторі (задіяні лінза-навігатор (AR), лазер для проєкції на сітківку, голосовий інтерфейс). 3. Аналіз довкілля (задіяні відеокамера, транслятор реальності, модуль розпізнавання загроз) 4. інтелектуальна підтримка (задіяні нейро-асистент, когніто-інтерфейс, віртуальний дисплей) 5. Презентація та взаємодія (задіяні голографічний проєктор, динаміки для звукового супроводу, мікрофон та нейро-асистент) 6. Фізіологічний моніторинг (задіяні біосенсор зору виявляє втому, нейро-асистент ініціює 3-хвилинну вправу на фокусування) 7. Завершення (смарт-окуляри автоматично синхронізують дані на хмарний сервер, когніто-інтерфейс фіксує режим сну, мікрокомп’ютер ініціює фонове оновлення програмного забезпечення.)   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумні окуляри», відповідно до таблиці 6.5. Кожний компонент «розумних окулярів» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.5. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.5. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумних окулярів властивості та операції. Клас «розумні окуляри» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «розумні окуляри» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумних окулярів», наприклад, «**Нейрозір**», який взаємодіє з користувачем через нейроінтерфейс, призначений для нейро-когнітивної доповненої реальності, яка не лише відображає віртуальні елементи, а й прогнозує мисленнєві наміри користувача, моделює й візуалізує варіанти майбутніх дій користувача на основі поточного контексту та емоційного стану. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «розумних окулярів» (**не менше 8 виняткових ситуацій**):  * неможливо точно зчитати когнітивний намір користувача. Обробка винятку: користувач отримує запит на підтвердження дій голосом; * надмірний емоційний стрес викликає спотворення AR-середовища. Обробка винятку: активується режим стабілізації з нейтральним середовищем; * неможливо згенерувати однозначну симуляцію майбутніх сценаріїв. Обробка винятку: користувачеві надається декілька альтернативних моделей; * збій голографічного або AR-інтерфейсу. Обробка винятку: користувач переключається у мінімалістичний інтерфейс; * модуль розпізнавання загроз не відповідає. Обробка винятку: активується резервна система моніторингу на базі відеокамери та компаса; * втрата стабільного нейрозв’язку з користувачем. Обробка винятку: сигнал користувачу про потребу перерви; * біосенсор зору не повертає коректні дані. Обробка винятку: пропонується змінити положення окулярів * помилка навігаційного проєктора або компаса. Обробка винятку: увімкнення режиму ручного орієнтування; * виник цикл зворотного когнітивного підсилення. Обробка винятку: відновлення з базового стану   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного холодильника, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * активація нейро-дублювання особистості. Дії розумних окулярів – створення віртуального аватара користувача, який самостійно відповідає на повідомлення; * проникнення у віртуальну симуляцію альтернативної реальності. Дії розумних окулярів –активація режимів AR/VR-фільтрації та ініцація віртуального провідника для навігації; * втрата свідомості або досягнення критичних біоіндикаторах користувача. Дії розумних окулярів – активується GPS-маячок, надсилається SOS-сигнал, вмикається голосовий інтерфейс для виклику допомоги; * колективне підключення до свідомості інших користувачів. Дії розумних окулярів – відкривають безпечний канал обміну даними, фільтрують інформацію за рівнем довіри та інтегрують досвід інших користувачів у власні когнітивні моделі; * неадекватна взаємодія з голосовим помічником. Дії розумних окулярів – реєстрація помилки з фіксацією в журналі помилок розпізнавання мовлення. |
| **6** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного телевізору»**. Опис сучасного Смарт ТВ за посиланням <https://surl.li/gyfown>. «**Розумний телевізор» майбутнього** — це інтелектуальний пристрій, який взаємодіє з людиною, переходить на рівень адаптивних та прогностичних цифрових агентів, який реагує на події в реальному часі та попереджає користувача про потенційні загрози чи наслідки, використовуючи алгоритми ситуаційної обізнаності та прогностичної аналітики. «**Розумний телевізор» майбутнього –** це інтелектуальний медіацентр-когнітивний симулятор — персоналізована інформаційно-аналітична система для взаємодії з користувачем, що виконує адаптивні, прогностичні, навчальні, комунікативні та розважальні функції з елементами безпеки.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумних» окулярів. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.6 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.6. Основні сутності предметної області розумного телевізору   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумний телевізор | Головний пристрій | Об'єднує всі інші модулі | Композиція | Id,  Модель,  Виробник, Енергоспоживання | Вмикнути,  Вимкнути,  Оновити | | Пристрій відображення | Вивід візуальної інформації | Входить до складу телевізора | Композиція | Тип (OLED/Голографія), Розмір,  Роздільна здатність | Відображати, Масштабувати, Переключити режим | | Шасі | Електронна основа системи | Містить електронні компоненти | Композиція | Матеріал,  Розміри, Енергоспоживання | Керувати сигналами, Підтримувати | | Корпус | Фізична оболонка пристрою | Містить усі внутрішні блоки | Композиція | Колір,  Матеріал,  Кнопки, Роз’єми | Захистити,  Вентилювати | | Аудіосистема | Вивід звуку | Агрегується з акустичними пристроями | Агрегація | Потужність, Кількість каналів, Режим | Вивести звук,  Змінитиг учність | | Процесор | Обробка інформації | Пов’язаний з усіма модулями телевізора | Композиція | Частота,  Кількість ядер, Енергоспоживання | Обробити дані,  Виконати завдання | | Пам’ять | Зберігання даних | Входить до системної архітектури | Композиція | ROM,  RAM,  Кеш | Зберегти,  Зчитати | | Мережеві інтерфейси | Зв’язок із мережею | Асоціюється з Wi-Fi, Bluetooth | Асоціація | Тип (Wi-Fi/Bluetooth), Швидкість | Підключитися, Передатидані | | Тренувальний симулятор | Симуляція дій для тренування | Використовується з голографічним дисплеєм | Агрегація | Тип тренування, Модель об'єкта | Симулювати,  Аналізувати результат | | Віртуальна лабораторія | Візуалізація та дослідження 3D-моделей | Працює з аналітичними модулями | Агрегація | Тип Об'єкта, Модель, Сценарій Впливу | Побудувати3D,  Моелювати зміну | | Біометричний сканер | Ідентифікація користувача | Взаємодіє з системою безпеки | Асоціація | Тип сканування, Точність | Сканувати обличчя, Авторизувати | | Прогностичний модуль (SmartPredictor) | Прогноз подій на основі аналізу контенту, даних, ситуації | Асоціюється з відеоаналізатором сенсорами, зовнішніми джерелами даних | Асоціація | Тип алгоритму, Рівень довіри, Джерела даних | Аналізувати контент, Виявити загрозу, Попередити користувача, Побудувати прогноз, Сценарій моделювання | | Симулятор бойових сценаріїв | Імітація бойових умов та тренувань | Співпрацює з AR/VR, голографічним проектором | Агрегація | Тип сценарію,  Рівень складності, Реалістичність | Запустити сценарій, Змінити оточення | | AR-тренажер водія/пілота/оператора | Віртуальне керування транспортом | Взаємодіє з інтерфейсом реальності та сенсорами руху | Композиція | Тип тренажера, Режим моделювання | Активувати режим,  Оцінити реакцію | | Модуль моделювання 3D-об'єктів | Побудова та редагування моделей | Взаємодіє з мікрокомп’ютером і віртуальним дисплеєм | Агрегація | Формат моделі, Рівень деталізації | Створити модель, Редагувати модель, Зберегти модель |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один телевізор має по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного телевізору».   1. Завантаження системи. Задіяні пристрої: Шасі, Корпус, Процесор, Пам’ять, Мережеві інтерфейси, Біометричний сканер. 2. Режим «Інформаційна панель дня». Задіяні пристрої: Пристрій відображення Аудіосистема Прогностичний модуль Пам’ять Мережеві інтерфейси. 3. Тренажерний режим. Задіяні пристрої: AR-тренажер водія/пілота/оператора, Тренувальний симулятор, Пристрій відображення (3D-графіка, AR-інтерфейс), Процесор, Аудіосистема, Пам’ять, Мережеві інтерфейси. 4. Бойовий сценарій. Задіяні пристрої: Симулятор бойових сценаріїв, Модуль моделювання 3D-об'єктів, Візуалізатор внутрішньої структури, Процесор, Пам’ять. 5. Віртуальна лабораторія. Задіяні пристрої: Віртуальна лабораторія, Модуль моделювання 3D-об'єктів, Візуалізатор внутрішньої структури, Пристрій відображення, Аудіосистема, Пам’ять. 6. Прогностичний режим. Задіяні пристрої: Прогностичний модуль, Процесор, Пам’ять, Пристрій відображення, Аудіосистема, Мережеві інтерфейси. 7. Завершення сесії. Задіяні пристрої: Біометричний сканер, Прогностичний модуль, Процесор, Аудіосистема, Пристрій відображення, Мережеві інтерфейси.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний телевізор», відповідно до таблиці 6.6. Кожний компонент «розумного телевізору» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.6. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.6. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який містить загальні для всіх smart-пристроїв у системі розумних окулярів властивості та операції. Клас «розумний телевізор» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «розумний телевізор» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний телевізор», наприклад, «**Нейроадаптивний бойовий телевізор**», який взаємодіє з користувачем через нейроінтерфейс, призначений для прогнозування бойових подій, керування роботизованими військами, симуляції майбутніх сценаріїв та ментального тренування особового складу в умовах доповненої/віртуальної реальності. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «розумного телевізору» (**не менше 8 виняткових ситуацій**):  * збій в прогностичному модулі. Обробка винятку: скасовуються останні передбачення, перемикається на резервну модель поведінки; * перегрів процесора при 3D-візуалізації. Обробка винятку: знижується частота процесора, призупиняються фонові задачі, увімкнення активного охолодження; * не працює візуалізатор структури. Обробка винятку: перевантажується модуль, очищається кеш сцен, використовується базова бібліотека об'єктів; * збій аудіосистеми. Обробка винятку: виправляють фазовий зсув, тимчасово вимикають AR-звук або синхронізують його із фізичним звуком; * втрата доступу до мережі. Обробка винятку: скидають VPN-тунель, використовують альтернативний інтерфейс (супутниковий/лазерний); * помилка в тренувальному симуляторі. Обробка винятку: аналізують причину збою, перезавантажують ядро симулятора з попередніми параметрами; * невідповідність біометричних даних користувача. Обробка винятку: повторна ідентифікація, вмикає режим гостьового доступу; * помилка AR-тренажера. Обробка винятку: корекціяє просторової прив’язки, переініціалізує навколишнє середовище; * віртуальна лабораторія перестає відповідати. Обробка винятку: скасована остання сесія, очищає пам’ять, перезапускає збережений проєкт у безпечному режимі.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного телевізора, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * підключення до Wi-Fi або мережі. Дії розумного телевізора – ініціалізує з'єднання з Інтернетом, підключає телевізор до потокових сервісів; * використання голосових команд. Дії розумного телевізора – аналізує команду користувача виконує необхідну дію інформує користувача; * активація екологічного режиму. Дії розумного телевізора – знижує яскравість екрану, вимикає функції, що споживають багато енергії; * зупинка або пауза контенту. Дії розумного телевізора – зупиняє відтворення відео або аудіо виводить на екран кнопку або повідомлення для поновлення відтворення; * неадекватна взаємодія з голосовим помічником. Дії розумного телевізора – реєстрація помилки з фіксацією в журналі помилок розпізнавання мовлення. |
| **7** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«інтелектуального адаптивного підручника»**. «**Інтелектуальний адаптивний підручник»**— це інтелектуальний пристрій, який взаємодіє з людиною, переходить на рівень адаптивних та прогностичних цифрових агентів, який реагує на події в реальному часі та попереджає користувача про потенційні загрози чи наслідки, використовуючи алгоритми ситуаційної обізнаності та прогностичної аналітики. «**Інтелектуальний адаптивний підручник» майбутнього –** це особистий інтелектуальний простір, де навчання – це захоплююча пригода, а не рутина, це особистий цифровий наставник, з яким студент взаємодіє і розвивається разом із ним.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумних» окулярів. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.7 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.7. Основні сутності предметної області інтелектуального адаптивного підручника   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Підручник | Базовий навчальний цифровий ресурс | Зв'язаний з усіма модулями, пристроями, користувачами | Композиція | ID, Назва, Тематика, Версія, Автор, Технології | Активувати,  Оновити,  Синхронізувати | | Користувач (Студент) | Основний суб’єкт навчання | Має зв’язок з модулями, гейміфікацією, рекомендаціями, нейроінтерфейсом | Один до багатьох | ID, Ім’я,  Рівень знань,  Стиль навчання, Інтереси,  Біодані,  Прогрес | Пройти курси,  Надати зворотний зв’язок, Аналізувати активність | | Розумні окуляри | AR-пристрій користувача | Взаємодіє з користувачем і CoreAI | Асоціація | AR-проекція, камера, мікрофон, Wi-Fi, Bluetooth, нейроінтерфейс | Виведення AR,  Зчитування уваги, емоцій, Голосове введення | | Віртуальний шолом | VR-пристрій | Інтегрується з CoreAI і тактильними пристроями | Асоціація | Роздільна здатність, кут огляду, сенсори, навушники | Імерсивне (VR, AR) навчання,  Моделювання | | Тактильні рукавички | Взаємодія у VR | Працює спільно з VR-шолом | Асоціація | Датчики, вібромотори, форма | Сенсорна взаємодія з віртуальними об’єктами | | Нейроінтерфейс | Зчитування мозкової активності | Дані для CoreAI і окулярів | Асоціація | Сенсори ЕЕГ, калібрування | Аналіз концентрації, Контроль думками | | Біометричні сенсори | Аналіз фізіології користувача | Інтегровані в окуляри/дисплей, дають дані CoreAI | Агрегація | Пульс, температура, погляд, емоції | Виявлення втоми, Визначення рівня залучення | | Комунікаційний модуль | Передача даних | Вбудований у систему, зв’язок з іншими пристроями | Агрегація | Wi-Fi, Bluetooth, 6G, супутник | Синхронізація,  Доступ до хмари,  Мережевий обмін  Перехід у режим паузи | | Проектор у повітря (AirProj) | Проєкція голограм | Альтернативний дисплей, підключений до CoreAI | Асоціація | Лазер, ультразвук, голографічна лінза | Голографічна візуалізація контенту | | Енергомодуль | Живлення пристрою | Частина ядра-пристрою | Композиція | Сонячна панель, індукція, тепло, акумулятор | Подача енергії, автономність | | Модуль трансляції (SmartCast) | Передача зображення на інші екрани | Пов’язаний із зовнішніми пристроями | Асоціація | HDMI, Wi-Fi Direct, AR-Link | Виведення зображення на телевізор, проектор тощо | | AR/VR-пристрій | Пристрій взаємодії з доповненою/віртуальною реальністю | Пов'язаний із модулем, лабораторіями, підручником | Асоціація | ID, Тип, Модель, Сумісність | Сканування,  Запуск віртуальних сцен,  Взаємодія з об’єктами | | Голографічний наставник | Персоналізований віртуальний викладач | Пов'язаний із користувачем і контентом | Асоціація | ID, Ім’я, Манера пояснення, Голос, Мова | Віртуальне пояснення, Емоційна адаптація, Запитання та відповіді | | Гнучкий голографічний дисплей | Носій для виведення контенту | Пов’язаний із підручником | Асоціація | ID, Тип, Розмір, Режими | Відображення 3D моделей, Перемикання сцен, Портативна проєкція | | Біометричний моніторинг | Відстеження фізіологічного стану учня | Пов’язаний із користувачем і контентом | Асоціація | ID, Тип (пульс, зір, стрес), Інтерпретація | Сканування,  Оповіщення про втому, Рекомендації з перерв | | Сервіс взаємодії з викладачем | Зв’язок учня з реальним викладачем | Пов’язаний із користувачем і контентом | Асоціація | ID, Канал зв’язку, Тип взаємодії | Відправка питань, Отримання пояснень, Спільне обговорення | | Модуль контенту | Окремий навчальний блок | Пов'язаний із завданнями, оглядами, AR/VR-сценами | Агрегація | ID, Назва, Тип, Складність, Тривалість, Передумови | Адаптація,  Перемикання формату, Генерація підказок | | Адаптивний механізм | Інтелектуальна система персоналізації | Пов’язаний із користувачем, модулем, контролем прогресу | Композиція | ID, Назва, Алгоритм, Вхідні дані, Ваги параметрів | Генерація маршрутів,  Підбір пояснень,  Аналіз відповідей | | Система рекомендацій | Пропозиція персоналізованого контенту | Спирається на адаптивний механізм і дані користувача | Композиція | ID, Тип, Джерела, Параметри | Визначення слабких місць, Підбір тем,  Пропозиції контенту | | Контрольне завдання | Перевірка знань | Пов'язане з модулями, користувачами | Агрегація з модулем | ID, Тип, Складність, Оцінка, Відповіді, Пояснення | Створення,  Оцінювання,  Адаптація | | Лабораторне завдання (VR) | Інтерактивне застосування знань | Зв'язок із модулями, AR/VR-пристроями | Агрегація | ID, Назва, Мета, Опис, Необхідні ресурси | Запуск симуляції,  Перевірка результатів, Спільне виконання |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один підручник** має **по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «інтелектуального адаптивного підручника».   * + - 1. Активація та ідентифікація користувача**.** Задіяні сутності:Біометричні сенсори, Нейроінтерфейс, Комунікаційний модуль, Енергомодуль.       2. Ініціалізація навчальної сесії. Задіяні сутності: Система рекомендацій, Адаптивний механізм, Модуль контенту, Біометричний моніторинг.       3. Виведення контенту. Задіяні сутності: Гнучкий голографічний дисплей, Проектор у повітря (AirProj), AR/VR-пристрій, Голографічний наставник, Модуль контенту, Енергомодуль.       4. Взаємодія з матеріалом. Задіяні сутності: Нейроінтерфейс, Біометричний моніторинг, Голографічний наставник, Адаптивний механізм.       5. Контрольні та лабораторні завдання. Задіяні сутності: Контрольне завдання, Нейроінтерфейс, Біометричний моніторинг, Голографічний наставник, Адаптивний механізм, Лабораторне завдання (VR), AR/VR-пристрій, Система рекомендацій.       6. Передача результатів викладачу. Задіяні сутності: Сервіс взаємодії з викладачем, Комунікаційний модуль, Біометричний моніторинг.       7. Трансляція навчання на зовнішні пристрої. Задіяні сутності: Модуль трансляції (SmartCast), Комунікаційний модуль, Енергомодуль.       8. Завершення сесії та збереження прогресу. Задіяні сутності: Адаптивний механізм, Система рекомендацій, Комунікаційний модуль.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «інтелектуальнийо адаптивний підручник», відповідно до таблиці 6.7. Кожний компонент «інтелектуального адаптивного підручника» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.7. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.7. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який мйістить загальні для всіх smart-пристроїв у системі інтелектуального адаптивного підручника властивості та операції. Клас «інтелектуального адаптивного підручника» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «інтелектуальний адаптивний підручник» має реалізувати інтрефейс. * розробити клас, похідний від класу «інтелектуальний адаптивний підручник», наприклад, «**Нейроадаптивний ментальний навчальний супутник**», який повністю інтегрується з нейронною системою людини через синтетичні нейроканали. Він є персоналізованим віртуальним наставником, що постійно супроводжує власника, формує когнітивні шляхи навчання, моделює ментальні симуляції і може проєктувати знання безпосередньо в короткотривалу та довготривалу пам’ять мозку. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «інтелектуального адаптивного підручника» (**не менше 8 виняткових ситуацій**):  * невідповідність рівня складності контенту когнітивним можливостям користувача. Обробка винятку: адаптивний механізм знижує складність пояснення, переформульовує терміни, активує голографічного наставника; * біометричні сенсори виявили перевтому або стрес. Обробка винятку: система призупиняє сесію, рекомендує перерву, активує легку гру в AR; * збій у нейроінтерфейсі або нестабільна синхронізація з мозковою активністю. Обробка винятку: тимчасово переходить на альтернативний режим взаємодії: голосове або тактильне управління, надсилає діагностичні дані до сервісного модуля; * збій аудіосистеми. Обробка винятку: виправляють фазовий зсув, тимчасово вимикають AR-звук або синхронізують його із фізичним звуком; * неможливо адаптувати тему через відсутність релевантного контенту. Обробка винятку: система рекомендацій формує запит до зовнішнього наукового сховища, аналізує альтернативні теми або пропонує викладачеві вручну втрутитись; * АR/VR-пристрій не може створити просторову симуляцію лабораторного завдання Обробка винятку: активується режим симуляції з текстово-графічною підтримкою, пропонується альтернативна візуалізація через гнучкий голографічний дисплей; * конфлікт між індивідуальним навчальним шляхом і поточним освітнім стандартом. Обробка винятку: система ініціює обговорення з викладачем через інтегрований сервіс, надає звіт з аргументами щодо адаптації траєкторії.; * виявлено небажану або застарілу інформацію в контенті. Обробка винятку: модуль контенту надсилає запит на верифікацію до навчального хмарного репозиторію, тимчасово приховує сумнівну інформацію;   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи «інтелектуального адаптивного підручника», придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * студент довго не може зрозуміти пояснення теми. Дії інтелектуального підручника – ініціює активацію голографічного наставника*;* * виявлено високий рівень емоційної залученості. Дії інтелектуального підручника – активує інтенсивний режим навчання, збільшуючи складність завдань для розвитку навички критичного мислення; * у мозковій активності студента фіксуються хвилі розчарування, напруги або втрати контролю над ситуацією. Дії інтелектуального підручника – знижує складність поточного завдання, переключає студента на гру, яка допомагає краще зрозуміти тему; * AR-пристрій зафіксував, що студент дивиться вбік від дисплея більше 20 секунд. Дії інтелектуального підручника – автоматично призупиняє подачу нового матеріалу, показує короткий візуальний огляд пройденого та запитує, чи потрібно повторити; * неадекватна взаємодія з голосовим помічником. Дії інтелектуального підручника – реєстрація помилки з фіксацією в журналі помилок розпізнавання мовлення. |
| **8** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного нейрообруча»**. «Р**озумний нейрообруч» майбутнього** — це інтелектуальний пристрій, який може автоматично налаштуватися під потреби людини, бажання та емоційний стан, дозволити користувачам інтегрувати свої думки в реальний світ, у стресових ситуаціях коригувати когнітивні навантаження людини, дозволити користувачам змінювати свій стиль навчання чи роботи, залежно від потреби, допомагати знаходити внутрішню рівновагу, зменшити тривожність і підвищити стійкість, стати потужним інструментом саморозвитку. «Р**озумний нейрообруч»–** це інноваційний інструмент, що трансформує спосіб взаємодії з навколишнім світом, навчанням і саморозвитком, забезпечуючи новий рівень комфорту, персоналізації та емоційної підтримки.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумних» окулярів. Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.8 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.8. Основні сутності предметної області розумного нейрообруча   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Нейрообруч | Головна система, що забезпечує зчитування і передачу мозкових сигналів | Компонує всі внутрішні модулі та датчики; асоціюється з користувачем і розумними пристроями | Композиція | ID,  Серійний номер, Стан живлення, Прошивка | Увімкнути,  Вимкнути,  Ініціалізувати,  Калібрувати,  Передати команду,  Зчитати сигнал | | Датчики чола | Збір мозкових імпульсів із лобової долі | Жорстко пов’язані з Нейрообручем | Композиція | Тип,  Чутливість,  ID | Зчитати сигнал,  Передати імпульс | | Датчики вух | Збір імпульсів з бокових ділянок мозку | Жорстко пов’язані з нейрообручем | Композиція | Тип,  Чутливість,  ID | Зчитати активність, Передати | | Еталонні датчики | Контроль фонових сигналів | Жорстко інтегровані з основним обручем | Композиція | Тип,  Калібрування | Налаштувати еталон, Порівняти сигнали | | Зарядний порт | Забезпечує живлення пристрою | Частина фізичної структури нейрообруча | Композиція | Тип роз’єму,  Напруга,  Струм | Зарядити,  Діагностувати живлення | | LED-ліхтар | Візуальний індикатор стану/функцій | Безпосередньо вбудований в обруч | Композиція | Колір, Інтенсивність, Режим | Миготіти,  Сигналізувати,  Змінювати колір | | Мікропроцесор (SoC) | Центр обробки сигналів, комунікація | Керує роботою всіх модулів, обмінюється даними з іншими пристроями | Композиція | Частота,  Обсяг пам’яті, Кількість ядер | Обробити сигнал, Шифрувати,  Керувати,  Навчатися | | Модуль зв’язку з комп’ютером | Передача даних у комп’ютерні системи | Взаємодіє з ПК через Bluetooth, Wi-Fi, нейроінтерфейс | Агрегація | Тип протоколу, Швидкість передачі | Встановити з’єднання, Передати дані,  Отримати команду | | Користувач | Людина, яка носить обруч | Взаємодіє з нейрообручем через мозкову активність | Асоціація | Ім’я, Рівень Зосередженості, ID Користувача, Мозкові патерни | Сконцентруватися, Надіслати думку,  Керувати | | Розумний пристрій (SmartDevice) | Зовнішній пристрій, яким керує Нейрообруч | Асоціюється через командні інтерфейси | Асоціація | Тип пристрою, Статус,  IP | Увімкнути/Вимкнути, Змінити стан | | Система обміну думками | Канал зв’язку між двома або більше нейрообручами | Пов’язує кілька нейрообручів у єдину мережу | Асоціація | Протокол,  ID зв'язку, Криптографія | Передати думку,  Отримати думку, Синхронізувати | | Система антистресу | Механізм зниження напруги та стресу | Працює спільно з біодатчиками та мікропроцесором | Агрегація | Рівень тривожності, Дихання, ЧСС (частота серцевих скорочень) | Активувати дихальну практику,  Показати голограму, Змінити освітлення | | Підсистема керування увагою | Виявлення та підтримка концентрації | Отримує сигнали з мозку, аналізує вектор уваги | Агрегація | Інтенсивність, Тривалість Фокусування | Збільшити стимуляцію, Переналаштувати контент, Викликати звукову/графічну підказку | | Модуль ментального переносу | Тимчасове переміщення свідомості в симульоване середовище | Взаємодіє з VR-пристроєм, нейрополюсним сенсором | Композиція | Ступінь занурення, Час сесії, Віртуальний аватар | Активація переносу, Повернення,  Синхронізація | | Нейрохмарний буфер | Віддалене збереження та обробка ментальних станів | Працює з голографічною пам’яттю та комунікаційним модулем | Композиція | Обсяг даних,  Стан підключення, Історія синхронізації | Завантаження,  Збереження,  Відновлення | | Модуль нейроемпатії | Визначення співпереживання та емпатичних реакцій | Взаємодіє з голографічним наставником та інтерфейсом емоцій | Композиція | Емпатичний спектр, Резонанс,  Когнітивна віддача | Аналіз,  Відгук,  Посилення мотивації | | Інтерфейс гіперфокусу | Посилення концентрації через нейростимулюючий вплив | Співпрацює з сенсором гіперзосередженості | Композиція | Режим індукції, Рівень імпульсу, Дозволи | Ініціація фокусування, Підтримка уваги | | Нейроперекладач думок | Перетворення думок у текст або візуальні образи | Взаємодіє з просторово-мисленнєвим модулем | Композиція | Мова виводу, Точність, Контекстність | Рендеринг думки, Форматування,  Передача | | Імпульсний нейропередавач | Безконтактна передача думки іншому користувачу | Працює з комунікаційним модулем | Композиція | Користувач-одержувач,  Статус каналу, Розмір пакета | Надсилання,  Отримання,  Шифрування | | Нейроінтуїтивний модуль | Аналіз інтуїтивних рішень користувача | Співпрацює з Адаптивним механізмом | Композиція | Імовірність реакції, Рівень спонтанності, Глибина інтуїції | Передбачення рішень, Навчання, Корекція рекомендацій |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один нейрообруч** має **по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «нейрообруча» в умовах війни, нестабільного доступу до інтернету, психологічного навантаження на студентів і викладачів, у стресових обставинах (повітряні тривоги, евакуація тощо).   1. Активація пристрою, ідентифікація користувача**.** Задіяні сутності:Нейрообруч, Користувач, LED-ліхтар (світлова індикація стану), Зарядний порт (перевірка заряду) 2. Початок навчального сеансу. Задіяні сутності: Нейрообруч, Датчики чола, Датчики вух, Еталонні датчики, Мікропроцесор (SoC), Користувач, Підсистема керування увагою, Інтерфейс гіперфокусу. 3. Оцінка емоційного стану студента. Задіяні сутності: Нейрообруч, Система антистресу. Модуль ментального переносу 4. Поглиблення уваги під час пояснення матеріалу. Задіяні сутності: Інтерфейс гіперфокусу, Нейроінтуїтивний модуль, Нейрообруч, Користувач. 5. Початок повітряної тривоги – стресова активація. Задіяні сутності: Система антистресу, Датчики чола, Еталонні датчики, Мікропроцесор (SoC), Нейрообруч.    * + 1. Евакуація до укриття – підтримка комунікації. Задіяні сутності: Система обміну думками, Нейроперекладач думок, Імпульсний нейропередавач, Модуль зв’язку з комп’ютером, Нейрообруч.        2. Психоемоційна стабілізація в укритті. Задіяні сутності: Підсистема керування увагою, Інтерфейс гіперфокусу, Нейроінтуїтивний модуль, Користувач.        3. Завершення тривоги – поступове повернення до навчання. Задіяні сутності: Підсистема керування увагою, Інтерфейс гіперфокусу, Нейроінтуїтивний модуль, Користувач.        4. Завершення заняття – підбиття нейроаналітик. Задіяні сутності: Нейрохмарний буфер, Розумний пристрій, Модуль зв’язку з комп’ютером, Нейрообруч.        5. Режим відновлення. Задіяні сутності: Система антистресу, Модуль ментального переносу, Нейроінтуїтивний модуль, Нейрообруч.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «нейрообруч», відповідно до таблиці 6.9. Кожний компонент «нейрообруча» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.9. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.9. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який мйістить загальні для всіх smart-пристроїв у системі інтелектуального адаптивного підручника властивості та операції. Клас «нейрообруча» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «нейрообруч» має реалізувати інтрефейс. * розробити клас, похідний від класу «інтелектуальний адаптивний підручник», наприклад, «**Нейросфера**», який є повністю безконтактним, автономним пристроєм квантово-нейронного зв’язку, що формує навколо голови користувача сферичне голографічне поле, зчитує думки на рівні намірів, а не сигналів, і дозволяє управляти цифровими або біологічними системами однією думкою. Нейрообруч розширюється до енергоінформаційної оболонки свідомості, де технології не лише читають мозок, а й стають його природним продовженням. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «нейрообруча» (**не менше 8 виняткових ситуацій**):  * перебої в квантовому каналі або втрачена авторизація. Обробка винятку: автоматичне перемикання в автономний режим з тимчасовим збереженням даних у внутрішньому буфері; сповіщення користувача через LED-індикатор; * когнітивний конфлікт у системі обміну думками. Обробка винятку: тимчасова ізоляція сеансу з найвищим пріоритетом, запуск протоколу емоційної нейтралізації через модуль нейроемпатії; * втрата даних у модулі ментального переносу. Обробка винятку: відкат до останньої стабільної ментальної конфігурації, повторна ініціалізація, попередження про часткову втрату прогресу; * аномальні сигнали від еталонних сенсорів. Обробка винятку: активація системи антистресу, виклик екстреного протоколу зв’язку з викладачем або родичами через SmartDevice; * невідповідність між даними сенсорів вух і чола. Обробка винятку: автоматичне калібрування, вивід підказки щодо правильного надягання пристрою, перезапуск ініціалізації; * збій у нейроперекладі думок. Обробка винятку: тимчасове відключення модуля, заміна на текстовий інтерфейс, відправлення звіту розробнику; * збій у системі фокусування уваги (гіперфокус). Обробка винятку: рекомендація перерви, активація антистресової програми, зниження інтенсивності стимулювання; * конфлікт інтерфейсу з іншими пристроями. Обробка винятку: перехід на резервний канал, тимчасове блокування конфліктного пристрою, логування інциденту.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного телевізора, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * користувач хоче поділитися думкою без слів. Дії нейрообруча – активація нейроперекладача думок, надсилання її через систему обміну думками до іншого нейрообруча, збереження сеансу в нейроінтуїтивному модулі*;* * сигнал про потенційну небезпеку (ракета/дрон). Дії нейрообруча – автоматичне формування ментального резюме знань останньої сесії. Завантаження даних у нейрохмарний буфер.LED-індикація червоного кольору для фізичного сигналу тривоги; * автоматичне резервне копіювання думок під час сну. Дії нейрообруча – активація модуля ментального переносу, завантаження ментальних концептів (абстрактних уявлень) у нейрохмарний буфер, очищення оперативного когнітивного простору користувача; * запит на підсилення креативності. Дії нейрообруча – Аактивація нейроінтуїтивного модуля, подача нейроімпульсів, які стимулюють нестандартне мислення, створення асоціативних патернів у свідомості користувача для генерації ідей; * ментальне перевантаження під час інтенсивного навчання. Дії нейрообруча – активація модуля нейроемпатії для виведення користувача з перевантаження, стабілізація стану через систему антистресу. |
| **9** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«розумного стола з вбудованим штучним інтелектом»**. Можливості сучасного розумного столу описані <https://www.smartboard.com.ua/articles/98/>. «**Розумний стіл**», програмну модель якого пропонується розробити в рамках лабораторного завдання, є інтелектуальним пристроєм, який взаємодіє з людиною як інтегрований помічник, підвищуючи ефективність роботи, комфорт і організацію часу за допомогою інтелектуальних функцій та персоналізації. «**Розумний стіл» майбутнього** – цене лише зручне робоче місце, а й інтелектуальний помічник, персоналізований асистент продуктивності, розумний організатор навчального і робочого простору, технічний коуч з організації часу, цифровий наставник, розумний партнер для досліджень.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «розумного столу». Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.9 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.9. Основні сутності предметної області розумного столу   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумний стіл | Базова сутність | Має зв'язки з усіма іншими пристроями та функціями. Створює інтегровану систему для комфорту користувача. | Композиція | Тип столу,  Розмір,  Матеріал,  Колір,  Дата виготовлення | Регулювати висоту, Моніторити час роботи, Керувати підключеними пристроями,  Активувати функції органайзера | | Електродвигун | Механічний компонент для регулювання висоти | Підключений до розумного стола для зміни висоти. | Агрегація | Потужність,  Швидкість регулювання, Максимальна висота | Регулювати висоту столу | | Клавіатура | Інтерфейс введення даних | Підключена до комп'ютера вбудованого в стільницю для введення текстових даних та управління столом. | Асоціація | Тип клавіатури, Підключення (проводове/бездротове) | Вводити дані,  Активувати функції столу | | Мультисенсорний дисплей | Інтерфейс виведення інформації | Виводить дані, графіку та мультимедійний контент, забезпечує взаємодію з користувачем. | Асоціація | Тип дисплею (сенсорний/мультитач), Роздільна здатність, Розмір | Вивести інформацію, Взаємодіяти з користувачем | | Штучний інтелект | Логіка та інтелектуальні функції | Контролює роботу столу, нагадує про завдання, покращує зосередженість та організує робочий процес. | Агрегація | Алгоритми, налаштування інтерфейсу,  Рівень адаптивності | Рекомендувати перерви, Налаштовувати підказки, Моніторити продуктивність | | Нагрівач для напоїв | Компонент для підігріву напоїв | Підключений до стола для підігріву напоїв (кава, чай) з можливістю автоматичного включення за запитом. | Асоціація | Тип нагрівача, Температура, Налаштування часу | Підігрівати напої | | Док-станція | Зарядка пристроїв | Бездротова док-станція для зарядки смартфонів та інших пристроїв. | Агрегація | Тип док-станції,  Типи підтримуваних пристроїв (USB, бездротове) | Заряджати пристрої | | USB-концентратор | Підключення пристроїв | Забезпечує розширення можливостей підключення до столу зовнішніх пристроїв через USB порти. | Агрегація | Кількість портів,  Типи підключення (USB 2.0, 3.0), Сумісність з пристроями | Підключати пристрої через USB | | Bluetooth динамік | Виведення звуку | Підключається до стола для відтворення звуків через Bluetooth. | Агрегація | Потужність,  Частотний діапазон,  Тип підключення | Відтворювати звук | | Розумний асистент | Нагадування та допомога | Взаємодіє з користувачем для нагадувань, покращення зосередженості та моніторингу завдань. | Асоціація | Алгоритми рекомендацій,  Голосове розпізнавання, Синтез мови | Нагадувати про завдання, Організовувати робочий процес,  Давати поради | | Таймер роботи | Моніторинг часу | Відслідковує час роботи користувача за столом, нагадує про необхідність перерви. | Асоціація | Максимальний час роботи,  Час перерви, Залишковий час | Відстежувати час, Сповіщати про необхідність перерви | | Модуль керування командами | Підтримка командної роботи | Організовує зручний простір для командної роботи, допомагаючи з інтерфейсом для спільних завдань та ігор. | Агрегація | Режими комунікації, Типи інтерактивних завдань,  Механізм взаємодії | Організовувати командні завдання,  Взаємодіяти з іншими користувачами | | Інтерактивні ігри | Розваги та відпочинок | Розумний стіл може запускати інтелектуальні ігри для відпочинку, змагань або розвитку командної роботи. | Агрегація | Типи ігор,  Механіка гри,  Інтерфейс взаємодії | Запускати інтерактивні ігри,  Організовувати змагання | | Інтерфейс для органайзера | Моніторинг завдань та організація роботи | Програмне забезпечення, яке допомагає користувачеві слідкувати за завданнями, контролювати виконання. | Асоціація | Завдання,  Пріоритети,  Статус,  Часові рамки | Нагадувати про завдання, фільтрувати за пріоритетами,  Створювати списки задач | | Проєкція емоцій | Елемент для зміни настрою | Створює емоційні інтерфейси або кольори для підтримки настрою користувача, адаптує навколишнє середовище. | Агрегація | Тип проєкції,  Колір,  Інтенсивність | Проєктувати кольори для заспокоєння, мотивації, концентрації | | Нейроінтерфейс | Контроль емоційного стану користувача | Збирає дані з мозкової активності для адаптації роботи столу та забезпечення комфортного навчання/роботи. | Агрегація | Рівень стресу,  Рівень концентрації, Фізіологічні показники | Моніторити мозкову активність, Адаптувати стол для оптимального стану користувача | | Інтелектуальна підсвітка | Підвищення концентрації користувача | Регулює інтенсивність підсвітки столу для максимізації зосередженості користувача на важливих завданнях. | Агрегація | Інтенсивність світла, Колір,  Налаштування часу | Адаптувати підсвітку в залежності від навколишнього середовища і потреб користувача | | Модуль сенсорної взаємодії | Збір інформації про параметри навколишнього середовища | Отримує дані від сенсорів, що встановлені в столі або приміщенні | Агрегація | Тип сенсора,  Значення параметра (температура, вологість, освітлення), Одиниці виміру,  Час зчитування | Зчитати значення, Передати дані в інші модулі,  Оновити показники,  Визначити зміни умов |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один стідл** має **по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «розумного столу».   1. Початок сесії та виявлення присутності користувача. Задіяні сутності: Модуль сенсорної взаємодії, Штучний інтелект, Інтелектуальна підсвітка. 2. Регулювання висоти стільниці відповідно до уподобань користувача. Задіяні сутності: Електродвигун, Штучний інтелект 3. Підігрів напоїв (чай, кава), заряджання пристроїв увімкнено. Задіяні сутності: Нагрівач для напоїв, Док-станція, USB-концентратор 4. Вітання, підготовка до роботи, визначення завдань на день. Задіяні сутності: Bluetooth динамік, Розумний асистент, Інтерфейс для органайзера 5. Виконання завдань користувачем, з використанням екрану і клавіатури. Задіяні сутності: Мультисенсорний дисплей, Клавіатура, Штучний інтелект 6. Керування столом за допомогою голосу та жестів. Задіяні сутності: Модуль керування командами, Розумний асистент 7. Автоматичне планування сесій роботи та відпочинку. Задіяні сутності: Таймер роботи, Штучний інтелект 8. Голосове збереження плану дня. Задіяні сутності: Розумний асистент, Модуль керування командами 9. Рекомендації зробити перерву. Запуск інтерактивної гри з емоційною релаксацією у перерві. Задіяні сутності: Таймер роботи, Штучний інтелект, Інтерактивні ігри, Проєкція емоцій, Мультисенсорний дисплей 10. Зчитування концентрації користувача під час гри. Задіяні сутності: Нейроінтерфейс, Штучний інтелект, Проєкція емоцій. Модуль сенсорної взаємодії 11. Аналіз зовнішніх умов і надання рекомендацій щодо мікроклімату. Задіяні сутності: Модуль сенсорної взаємодії, Штучний інтелект 12. Планування нових завдань користувачем у цифровому органайзері. Задіяні сутності: Інтерфейс для органайзера, Мультисенсорний дисплей, Клавіатура. 13. Аналіз активності користувача та рекомендація завершити роботу. Задіяні сутності: Штучний інтелект, Розумний асистент, Таймер роботи 14. Завершення сесії, переведення столу в сплячий режим та відключення непотрібних компонентів. Задіяні сутності: Електродвигун, Інтелектуальна підсвітка, Нагрівач для напоїв, Bluetooth динамік, Штучний інтелект   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний стіл», відповідно до таблиці 6.9. Кожний компонент «розумного столу» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.9. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.9. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який мйістить загальні для всіх smart-пристроїв у системі інтелектуального адаптивного підручника властивості та операції. Клас «розумного столу» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «розумного столу» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний стіл», наприклад, «**Телепортаційний розумний стіл**», який є інноваційним робочим місцем майбутнього, здатне миттєво переміщати користувача та його робоче середовище у віртуальні або фізичні простори. Стіл синхронізується з нейроінтерфейсом, модулем розширеної реальності та квантовим модулем телепортації. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «розумного столу» (**не менше 8 виняткових ситуацій**):  * відмова електродвигуна підйому стільниці. Обробка винятку: повідомити користувача через дисплей і голосовий інтерфейс, ручне регулювання або технічна діагностика; * відсутність відповіді від розумного асистента. Обробка винятку: перехід до резервного текстового інтерфейсу, запис в журналі збоїв для подальшої діагностики; * невірні дані сенсорів навколишнього середовища. ігнорування даних із помилками, перехід до стандартних налаштувань комфорту, повідомлення користувача про некоректність показників; * нейроінтерфейс не зчитує сигнали. Обробка винятку: змінити положення датчика або перейти на ручне керування, зберегти журнал збоїв для аналізу; * конфлікт команд користувача. Обробка винятку: аналіз пріоритету команд (наприклад, голос проти сенсорного вводу), запит уточнення від користувача, логування конфлікту для машинного навчання; * несправність інтелектуальної підсвітки. Обробка винятку: автоматичне перемикання на статичне освітлення, повідомлення: "Режим підсвітки обмежено"; * нерозпізнана голосова команда. Обробка винятку: запуск режиму навчання команд, логування для покращення ШІ; * помилка завантаження інтерактивної гри. Обробка винятку: Пропозиція обрати іншу гру або режим релаксації, вивід повідомлення на дисплей, логування технічної помилки гри; * замороження інтерфейсу органайзера. Обробка винятку: автоматичне оновлення інтерфейсу, вивід останніх збережених даних, попередження користувача про відновлення попередньої сесії.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного телевізора, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * користувач сідає за стіл. Дії розумного столу – сканування положення тіла, адаптація висоти столу, запуск нейроінтерфейсу, перевірка налаштувань комфорту, регулювання температури і світла; * користувач просить показати список справ. Дії розумного столу – обробка голосової команди, запуск органайзера, виведення завдань на мультисенсорному дисплеї, активація голосового керування завданнями; * користувач вмикає режим концентрації. Дії розумного столу – приглушення освітлення, вимкнення звуків, блокування сповіщень, запуск таймера роботи, виведення фокус-інтерфейсу органайзера; * увімкнення інтерактивної гри. Дії розумного столу – перехід до ігрового режиму, адаптація підсвітки та звуку, запуск гри на дисплеї, вимкнення зовнішніх сповіщень; * завершення дня / вимкнення столу. Дії розумного столу – збереження активностей дня, вимкнення підсвітки, опускання стільниці до мінімуму, вимкнення нагрівача, дисплею, динаміків, прощальна емоція на проєкції. |
| **10**  **10** | **Опис предметної області**. Існує предметна область, що охоплює об’єкти та процеси, пов’язані з конструктивними елементами **«інтелектуальної кухні з вбудованим штучним інтелектом»**. Можливості сучасної інтелектуальної кухні описані <https://www.smartboard.com.ua/articles/98/>. «**інтелектуальної кухні**», програмну модель якого пропонується розробити в рамках лабораторного завдання, є інтелектуальним пристроєм, який взаємодіє з людиною як інтегрований помічник, підвищуючи ефективність роботи, комфорт і організацію часу за допомогою інтелектуальних функцій та персоналізації. «**Інтелектуальна кухня» у вигляді розумного кухонного столу** – цене лише зручне робоче місце, а й інтелектуальний помічник, персоналізований асистент продуктивності, технічний коуч з організації часу, цифровий наставник, розумний партнер для досліджень.  Застосування SOLID принципу єдиної відповідальності вимагає включення в систему сутності «**Сервіс**» як технічної, що не входить у предметну область, а використовується для реалізації не притаманних об’єктам предметної області операцій. У разі потреби використання меню (не обов’язково) для зручності користувача доцільно створити окрему сутність «**Меню».** В сутності «**Сервіс**» і «**Меню»** не можна вбудовувати об’єкти предметної області, які входять у склад «**інтелектуальної кухні**». Вбудовування сутностей предметної області в сутності «**Сервіс**» і «**Меню»** означає, що допоміжні технічні сутності використовують об’єкти предметної області, що суперечить реальності і є неможливом. Навпаки, сутності предметної області мають задіяти технічні сутності «**Сервіс**» і «**Меню»** для завдання даних з консолі, виведення повідомлень на консоль, відображення меню команд та виклику методів згідно із сценарієм роботи програми. В таблиці 6.10 подані ролі, атрибути, операції сутностей та типи зв’язків між ними.  Таблиця 6.10. Основні сутності предметної області інтелектуальної кухні   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Сутність** | **Роль** | **Опис зв’язків** | **Тип зв’язку** | **Атрибути** | **Операції** | | Розумний кухонний стіл | Базова система | Поєднує всі модулі та пристрої в єдину інтерактивну платформу | Композиція | ID, модель, режим роботи, користувацький профіль | Ініціалізація, самодіагностика, перехід між режимами, персоналізація | | Камера гіперспектрального бачення | Аналіз вмісту продуктів | Підключена до комп’ютера для аналізу складу їжі та виявлення шкідливих речовин | Агрегація | Спектральні діапазони, роздільна здатність | Виявлення нітратів, визначення свіжості, відстеження алергенів | | Вбудована плита | Приготування їжі | Керується через ШІ та рецептурний модуль | Агрегація | Температурні зони, таймер, тип нагріву | Автоприготування, контроль температури, аварійне вимкнення | | Мультисенсорний дисплей | Виведення інформації, інтерфейс | Показує рецепти, підказки, етапи приготування | Агрегація | Роздільна здатність, сенсорність, режим енергозбереження | Відображення покрокових інструкцій, відео, сканування жестів | | Програмне ядро з ШІ | Обробка інформації, ухвалення рішень | Аналізує дані з усіх сенсорів та камер, пропонує рецепти | Композиція | Нейромережа, база знань, історія дій | Генерація рецептів, оптимізація раціону, розпізнавання звичок користувача | | Сканер харчової структури | Миттєве визначення мікроструктури їжі | Працює в парі з камерою, впливає на якість рекомендацій | Асоціація | Тип продукту, глибина сканування | Визначення фальсифікації, хімічного оброблення, ступеня дозрівання | | Генератор рецептур | Створення нових комбінацій страв із урахуванням здоров’я користувача | Взаємодіє з ШІ, враховує стан здоров’я, алергії, сезонні продукти | Асоціація | Профіль користувача, історія харчування, список доступних продуктів | Автоматичне створення нових рецептів, дієтичних або інноваційних страв | | Проєктор доповненої реальності | Навігація при готуванні | Спроектований на стільницю, вказує де різати, що додати, таймінги | Агрегація | Яскравість, точність позиціонування | Показ покрокових вказівок, інтерактивний помічник у доповненій реальності | | Інтелектуальні ваги | Зважування інгредієнтів | Вбудовані в поверхню столу | Композиція | Точність, одиниці виміру | Автоматичне зважування, передача даних в рецептурний модуль | | Система визначення якості продуктів | Аналіз якості та свіжості | Отримує дані від камер, вагів, сканерів | Композиція | Свіжість, час зберігання, рівень обробки | Попередження про зіпсовані продукти, поради щодо першочергового використання | | Бездротова док-станція | Зарядка смартфона | Додатковий зручний функціонал | Агрегація | Потужність, тип сумісності | Зарядка пристроїв, обмін даними | | Модуль кулінарного прогнозування | Передбачення смаку, настрою після їжі | Зв’язується з профілем користувача, ШІ, сканером | Асоціація | Настрій користувача, втома, минулі вподобання | Прогноз ймовірної реакції організму, зміна рецепту в реальному часі | | Нейрокулінарний інтерфейс | Зчитування кулінарних бажань напряму з мозку користувача | Зв’язок з Голографічним шеф-кухарем, Аналізатором здоров’я користувача, Сенсором настрою користувача | Асоціація | Рівень концентрації, емоційний стан, кулінарні патерни | Генерація страв на основі думок, корекція рецепту за настроєм | | Голографічний шеф-кухар | Віртуальний аватар кулінарного наставника у голограмі | Зв’язок із Нейрокулінарним інтерфейсом, Аналізатором здоров’я,  Сенсором настрою | Агрегація | Ім’я, стиль, голос, візуалізація | Жестова взаємодія,  персоналізовані поради,  кулінарне навчання | | Аналізатор здоров’я користувача (з ДНК) | Зчитує дані з носимого пристрою/ДНК-профілю, коригує страви під конкретну людину | Зв’язок із Нейрокулінарним інтерфейсом, Голографічним шеф-кухарем, Сенсором настрою | Агрегація | Калорійність, гени, алергени, дієтичні рекомендації | Персоналізація їжі під генетичний профіль,  прогноз реакцій | | Сенсор настрою користувача | Визначає емоційний стан і підбирає відповідну їжу або музику | Зв’язок із Нейрокулінарним інтерфейсом, Аналізатором здоров’я, Голографічним шеф-кухарем, | Асоціація | Настрій, втома, тривожність | Пропозиція "їжі-комфорту",  медитативне освітлення,  релакс- музика |   Потрібно здійснити **об’єктно-орієнтований аналіз (OOA)** і визначити класи, поля і методи. Спрощена взаємодія між учасниками предметної області: **один кухонний стіл** має **по одному** агрегованому чи композитному, чи асоційованому з перерахованих **елементів.** Сутності **Сервіс** і **Меню** не є сутностями предметної області, є лише інструментом, який **використовують** інші сутності для реалізації своїх дій. Для визначення сценарію роботи програми потрібно побудувати Use Case діаграму.  Можливий варіант сценарію роботи «інтелектуальної кухні (розумного кухонного столу)».   1. Початок сесії та виявлення думок, настрою та бажань користувача щодо приготування їжі. Задіяні сутності: Нейрокулінарний інтерфейс, Сенсор настрою користувача, Аналізатор здоров’я користувача. 2. Генерація персоналізованих ідей, аналіз ДНК, настрою та стану здоров’я. Задіяні сутності: Програмне ядро з ШІ, Генератор рецептур, Модуль кулінарного прогнозування, Голографічний шеф-кухар. 3. Формування дієтичних рекомендацій. Задіяні сутності: Аналізатор здоров’я користувача (з ДНК), Програмне ядро з ШІ, Генератор рецептур, Модуль кулінарного прогнозування, Сенсор настрою користувача. 4. Аналіз наявних інгредієнтів. Задіяні сутності: Камера гіперспектрального бачення, Сканер харчової структури, Система визначення якості продуктів. 5. Вибір рецепта і підготовка до готування. Задіяні сутності: Модуль кулінарного прогнозування, Генератор рецептур, Програмне ядро з ШІ, Мультисенсорний дисплей. 6. Візуалізація процесу приготування. Задіяні сутності: Проєктор доповненої реальності, Голографічний шеф-кухар, Програмне ядро з ШІ. 7. Контроль дотримання дієти. Задіяні сутності: Інтелектуальні ваги, Програмне ядро з ШІ, Мультисенсорний дисплей 8. Зважування інгредієнтів та приготування. Задіяні сутності: Інтелектуальні ваги, Вбудована плита, Мультисенсорний дисплей, Програмне ядро з ШІ. 9. Моніторинг готування. Задіяні сутності: Сканер харчової структури, Програмне ядро з ШІ, Камера гіперспектрального бачення. 10. Подача їжі та зворотний зв’язок. Задіяні сутності: Голографічний шеф-кухар, Сенсор настрою користувача, Нейрокулінарний інтерфейс. 11. Завершення сесії. Задіяні сутності: Бездротова док-станція, Мультисенсорний дисплей.   На основі результатів ООА здійснити **об’єктно-орієнтоване проєктування (OOD)** і побудувати діаграму класів. Методи класів повинні корелюватися (співпадати) з прецедентами Use Case діаграми. На основі діаграми класів слід здійснити **об’єктно-орієнтоване програмування,** розробивши **код класів** мовою C# та реалізувавши сценарій роботи програми.  **Версія 1 – Composition, aggregation (пункти завдання 1,2,3).**   1. Створити новий проєкт. Додати в проєкт класи, що описують пристрої, з яких складається «розумний стіл», відповідно до таблиці 6.10. Кожний компонент «інтелектуальної кухні» описати окремим класом в окремому файлі. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють роботу пристроїв відповідно до операцій, вказаних в табл. 6.10. 2. Реалізувати зв’язки між класами (асоціація, агрегація, композиція) згідно з типами зв’язків, указаними в табл. 6.10. Для реалізації зв’язків типів асоціація, агрегація, композиція скористатися алгоритмами, описаними в підрозділі «Реалізація зв’язків між класами», використавши конструктори класів із створенням нових об’єктів (композиція), з передачею об’єктів як параметри (агрегація та асоціація). 3. Реалізувати методи згідно із сценарієм, описаним в Use Case діаграмі та визначеними студентом алгоритмами.   **Версія 2 – Inheritanсe (пункти завдання 4)** (не обов’язково, для додаткового балу)   1. Пропонується на вибір команди чи студенту такі альтернативи:  * розробити абстрактний базовий клас, який мйістить загальні для всіх smart-пристроїв у системі інтелектуального адаптивного підручника властивості та операції. Клас «інтелектуальної кухні» зробити похідним від абстрактного базового класу. * розробити інтерфейс, який узагальнює основні функції керування, діагностики та оновлення для реалізації різних типів окулярів з однаковою поведінкою. Клас «розумного столу» має реалізувати інтрефейс. * розробити клас, похідний від класу «розумний стіл», наприклад, «**Кулінарний БіоСинтезатор**», який є інноваційним робочим місцем майбутнього, він не лише готує їжу, а й синтезує її з молекул, адаптуючи склад до генетичного профілю, медичних показників та настрою користувача в реальному часі. БіоСинтезатор здатен створювати повністю персоналізовані страви, яких не існує у класичній кулінарії. Атрибути та методи придумати самостійно.   **Версія 3 (пункт завдання 5).**   1. Створити **власний клас виняткових ситуацій** для визначення та обробки виключних ситуацій, пов’язаних із семантикою предметної області у вигляді моделювання обробки збоїв в роботі «інтелектуальної кухні» (**не менше 8 виняткових ситуацій**):  * конфлікт даних між генетичним аналізом і вподобаннями користувача. Обробка винятку: запит на підтвердження; * виявлено зіпсований продукт (зараження, токсини, паразити). Обробка винятку: блокування приготування, аудіо- та візуальне попередження, пропозиція замінників, звіт про безпеку; * збої в роботі сенсора настрою (невідповідні сигнали). Обробка винятку: повторний зчитування, запит у користувача, аналіз настрою або пропуск аналізу, релакс-музика; * перевантаження нейрокулінарного інтерфейсу. Обробка винятку: тимчасове відключення нейрозв’язку, повернення до класичних каналів керування (дотик, голос), діагностика; * збої в оновленні бази рецептів із хмари. Обробка винятку: перехід у режим офлайн-кухаря, використання локального кешу рецептів, запис помилки для повторного оновлення; * помилка ваги — інгредієнт не відповідає очікуваному складу/масі. Обробка винятку: повторне зважування, повідомлення користувача, автоматичне перерахування рецепту або відміна дії; * користувач не реагує на інструкції системи (відволікання, сон, відсутність). Обробка винятку: призупинення процесу, звуковий сигнал, перевірка наявності, при тривалій неактивності — збереження страви/очищення. * неможливо ідентифікувати продукт камерою/сканером. Обробка винятку: повторне сканування, пропозиція вручну вибрати продукт, запит на додавання до бази, якщо невідомий інгредієнт.   **Версія 4 - Delegate (пункт завдання 6).**   1. Визначити **події** та реалізувати обробники подій через **делегати**. Події, що відбуваються в процесі роботи розумного телевізора, придумати самостійно (**не менше 5 подій**), або реалізувати обробники таких подій:  * початок сканування інгредієнтів. Дії інтелектуальної кухні: активується камера гіперспектрального бачення, ініціалізується сканер харчової структури, виводиться результати аналізу; * виявлено несвіжий або шкідливий продукт. Дії інтелектуальної кухні: виводиться попередження, пропонується замінник або ігнорування, оновлення рецепту відповідно до доступних продуктів; * отримано результати аналізу ДНК користувача. Дії інтелектуальної кухні: визначається алергенність, дієтичні обмеження, коригується список дозволених інгредієнтів, оновлюється персональний план; * визначено поживна ціннвссть обраних інгредієнтів. Дії інтелектуальної кухні: система формує діаграму білки-жири-вуглеводи, показує відповідність страви добовим нормам, пропонує баланс; * активовано нейрокулінарний інтерфейс (керування думками). Дії інтелектуальної кухні: система зчитує бажання користувача, адаптує рецепт або запускає процес, зберігає переваги для подальшого навчання ШІ. |
| **11** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «р*озумних дверей у квартиру»*, які дозволяють розпізнати особу і сигналізувати у випадку сторонньої людини, сканувати людину у повний зріз, відкриватися за голосовою командою або за думкою розпізнаної людини, пропускати тільки розпізнану людину (мешканців квартири), подавати сигнал тривоги у поліцію у випадку спроби злому дверей або несанкційного проникнення у квартиру, записувати та вести облік людей, що відвідують квартиру, розпізнавати емоції та настрій людини, вітати та проводжати людини голосовим супроводом тощо. «*Розумні двері*» складаються з комп’ютера, візуалізатора, проєктора, сканера обличчя та тіла людини, інтерактивного сенсорного дисплею, вбудованого програмного забезпечення, датчиків, камери, звукового генератора, мобільного зв’язку тощо. Кожний пристрій «Розумних дверей» описати окремими класами в окремих файлах та відповідними зв’язками між класами. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «Розумних дверей*»*. Придумати методи, які описують сценарій роботи «Розумних дверей*»*, відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумних дверей*» клас «*Розумний охоронець*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний охоронець*» (<https://blueset.ru/product/gsm-storozh-smart>) має включити сирену, здійснювати дозвон і розсилку SMS повідомлень за списком номерів, записаному в пам'ять приладу, виконувати інші за фантазією студента дії. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки сбоїв або відхиленн від стандартного сценарію в роботі «*Розумних дверей*» (наприклад, почався несанціонований сигнал сирени). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного автомобіля, придумати самостійно. |
| **12** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «р*озумний сканер тіла людини»*, який дозволяє розпізнати особу і сигналізувати у випадку сторонньої людини, сканувати людину у повний зріз, визначати склад тіла − процентний вміст кісток, жиру, води і м'язів в людському тілі (<https://tanita-ua.com/page/info_text_1>), будувати 3D модель людини для конструювання індивідуальних протезів, зняття точних вимірювань і відстеження змін форми тіла в часі, для тривимірної візуалізації результатів майбутньої пластичної операції, для моделювання фасону одягу тощо. «*Розумний сканер*» складається з комп’ютера, візуалізатора, проєктора, аналізатора тіла (ваги, пульт управління), інтерактивного сенсорного дисплею, вбудоване програмне забезпечення тощо. Кожний пристрій «Розумного сканера» описати окремими класами в окремих файлах з відповідними зв’язками між класами. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумного сканера»*. Придумати методи, які описують сценарій роботи «*Розумного сканера»*, відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумного сканера*» клас «*Розумний 3D принтер*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*3D принтер*» має друкувати штучну копію людини з матеріалу, який вибирає користувач. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки збоїв або відхилень від стандартного сценарію в роботі «Розумного *сканера*» (наприклад, сканер почав генерувати несанкціоновані команди). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи розумного *сканера*, придумати самостійно. |
| **13** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «*Розумної фітнес-футболки*», яка дозволяє розпізнати володаря футболки, записати кількість кроків, спалені калорії, пройдену відстань, ЕКГ, пульс, частоту дихання, здійснює навігацію до потрібногопункту призначення, формує рекомендації щодо обсягу фізичного навантаження людини, має антимікробні властивості. Склад розумної футболки: Bluetooth, програмне забезпечення, датчики, акумулятор, мікроконтролер, бездротові електроди, спеціальні сенсори тощо (<https://itsell.ua/news/umnaya-futbolka-predupregdaet-o-serdechnom-pristupe>). Кожний пристрій «*Розумної фітнес-футболки*» описати окремими класами в окремих файлах з відповідними зв’язками між класами. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумної фітнес-футболки»* відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумної фітнес-футболки*» клас «*Розумний діагност*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний діаг*ност» попереджає про серцевий напад або гіпертонічний криз за кілька днів. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних, запису та читання файлів, виходу масиву за його межі, із семантикою предметної області у вигляді обробки збоїв або відхилень від стандартного сценарію в роботі «*Розумної фітнес-футболки*» (наприклад, навігатор почав неправильно визначати пункти призначення, неправильний прогноз фізичного стану людини). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи *Розумної фітнес-футболки*, придумати самостійно. |
| **14** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «*Розумних ваг*», які дозволяють розпізнати володаря ваг, контролювати масу тіла, дотримання дієт і якості силових навантажень, визначає індекси маси тіла, розраховує частку жиру, м'язів, кісток та води в організмі, швидкість обміну речовин, тип статури та біологічний вік. Дані аналізуються та структуруються у мобільному додатку. Склад «*Розумних ваг*» (<https://deep-review.com/reviews/huawei-scale-3/>): бездротові електроди для подачі електросигналів, батарея (акумулятор), тензометричні датчики, Bluetooth, програмне забезпечення, датчики, акумулятор, мікроконтролер, спеціальні сенсори, підтримка Wi-Fi, для синхронізації даних зі смартфоном, екран ваг тощо 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумних ваг»* відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумних ваг*» клас «*Розумний діагност*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний діаг*ност» попереджає про проблеми з вагою людини, зневоднення організму тощо. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних володаря ваг, передачі даних на смартфон, із семантикою предметної області у вигляді обробки збоїв або відхилень від стандартного сценарію в роботі «*Розумних ваг*» (наприклад, похибки в показниках, неправильний прогноз фізичного стану людини). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи *Розумник ваг*, придумати самостійно. |
| **15** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «*Розумної парасольки*», яка дозволяє розпізнати володаря парасольки, подає сигнал, якщо володар забув парасольку, наприклад, в транспорті, прогнозує опади, існує синхронізація зі смартфоном для повідомлень, якщо володар парасольки віддалився від неї на кілька метрів. Склад «*Розумної парасольки*» (<https://habr.com/ru/company/mvideo/blog/598895/>): ручка, спиці, світлодіодний індикатори майбутніх опадів, сили вітру, вологості, Bluetooth, підтримка Wi-Fi для синхронізації даних зі смартфоном,, батарейки (акумулятор), звуковий датчик тощо. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумної парасольки»* відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумної парасольки*» клас «*Розумний гаджет*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний гаджет*» налаштовується під настрій володаря і виконує відповідно до бажань володаря функції. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних володаря розумної парасольки, передачі даних на смартфон, із семантикою предметної області у вигляді обробки збоїв або відхилень від стандартного сценарію в роботі «*Розумної парасольки*» (наприклад, похибки в прогнозуванні опадів). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи *Розумної парасольки*, придумати самостійно. |
| **16.** | Створити проєкт для консольного застосунку мовою C# для роботи з класами, що зв’язані *асоціацією, агрегацією та композицією*.   1. Включити в проєкт класи для моделювання роботи «*Розумної маски*», яка дозволяє розпізнати володаря маски, подає сигнал, якщо володар забув надягнути маску або не закрив маскою ніс, визначити рівень забруднення повітря навколо людини в масці, подає сигнал, якщо приміщення, в якому знаходиться людина в масці, не відповідає санітарним нормам щодо насиченості вірусами, здійснює експрес-аналіз зараженості людини Ковід-19, здійснює синхронізацію зі смартфоном через Wi-Fi, можна передати дані в смартфон. Склад «*Розумної маски*»: сама маски, датчики-аналізатори повітря, Bluetooth, підтримка Wi-Fi, відповідний застосунок на смартфоні, світлодіодні та звукові індикатори та сигналізатори, хімічні реагенти для експрес-аналізу тощо. 2. Включити в класи закриті поля, відкриті властивості (аксесори), відкриті методи, які моделюють принцип роботи «*Розумної маски»* відповідно до п.1 завдання. (мобілізувати студентську фантазію). 3. Розробити похідний від «*Розумної масик*» клас «*Розумний гаджет*» або інший пристрій (відповідно до фантазії студента) та промоделювати його роботу. «*Розумний гаджет*» налаштовується під настрій володаря і виконує відповідно до бажань володаря функції. 4. Реалізувати визначення та обробку виключних ситуацій, пов’язаних із помилками введення даних володаря розумної парасольки, передачі даних на смартфон, із семантикою предметної області у вигляді обробки збоїв або відхилень від стандартного сценарію в роботі «*Розумної маски*» (наприклад, похибки в прогнозуванні опадів). 5. Визначити події та реалізувати обробники подій через делегати. Подія – це факт виконання якоїсь дії. Події, що відбуваються в процесі роботи *Розумної маски*, придумати самостійно. |

### Література

1. О.С. Бичков, Є.В. Іванов Об’єктно-орієнтоване програмування мовою C#/

2.C# 2005 и платформа .Net 3.0 для профессионалов. Нейгел К., Ивьен Б. и др. –М.: ООО “И.Д. Вильямс", 2008. –1376 с.

## Підсумкові рейтингові бали за лабораторні роботи, МКР та проєкт з дисципліни «Вступ до ООП»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№ п.п* | | *Номер лабораторної роботи* | *Рейтинговий бал* | *Deadline* |
| 1 | | ЛР1 (у формі практичних занять) | 2,5 | 15 лютий |
| 2 | | ЛР2 (у формі практичних занять) | 4,5 | 25 лютий |
| 3 | | ЛР3 | 5,5 | 15 березня |
| 4 | | ЛР4 | 5,5 | 25 березня |
| 5 | | ЛР5 | 4,5 | 10 квітня |
| 6 | | ЛР6 | 3,5 | 25 квітня |
| Разом за лабораторні роботи | | | 26 | |
| 7 | МКР1 | | 5 |  |
| 8 | МКР2 | | 5 |  |
| Разом за МКР | | | 10 | |
| **Разом мінімально допустимий рейтинг** | | | **36** | |
| 9 | Проеєкт | | 24 | 30 травня |
| Разом за усі роботи | | | 60 | |

## Проєктні завдання з дисципліни «Вступ до ООП»

#### Сума балів за проєкт 12

Студенти вибирають теми за власним бажанням. Неприпустимо записуватися на одну тему більше ніж одному студенту.

Пропонується застосовувати технологію *EduScum*, розробленої для реалізації проєктного підходу в освіті на базі методології гнучкої розробки програмного забезпечення *Agile Software Development with SCRUM*.

За технологією *EduScum* проєкт передбачає 6 версій (releases), кожна з яких оцінюєтся у 4 бали. Кожна версія (release) проєкту – це готовий до використання застосунок з обмеженою функціональністю відповідно до пріоритетів задач бізнес-процесів, що узгоджуються з викладачем (product owner).

**За готовий проєкт, тобто за 4 версії, можна отримати 12 балів (3бали\*4версій=12 балів)**. Студент самостійно визначає, на якій версії (релізу) він закінчить свою розробку. Відповідно до кількості релізів визначається сума балів за проєкт.

Студент може об’єднати деякі версії проєкту, погодисши з викладачем сценарій роботи версій програми. Для отримання максимальні 12 балів сумарна кількість версій проєкту не може бути менше чотирьох. В такому випадку кількість балів, що нараховується за кожну версію проєкту, складе 3 балів (3 бали\*4 версії=12 баліяв).

### Порядок виконання проєкту – версії програм (етапи або releases):

#### Версія 1. «Об'єктно-орієнована декомпозиція предметної області завдання, побудова сценарію та розробка каркасу програми» 3 бали

* 1. Однайомлення з предметною областю та постановкою задачі.
  2. Аналіз рекомендованих джерел інформації
  3. Пошук та аналіз додаткових джерел інформації за потребою.
  4. Розробка сценарію роботи програми
  5. Об'єктно-орієнована декомпозиція предметної області завдання (ідентифікаця об'єктів та їх взаємозв'язків)
  6. Розробка діаграми класів та їх взаємозв'язків. Мають бути враховані такі типи взаємозв'язків:
     1. асоціації;.
     2. агрегації;
     3. композиції.

Для першої версії зв'язок типу «успадкування» не використовувати.

* 1. Написання коду класів або генерація коду класів по розробленій діаграмі класів. Кожний клас має бути описаний в окремому файлі. В класах оголосити поля та методи відповідно до сценарію версії 1 програми. Код методів можна реалізувати в подальших версіях програми
  2. Сторення класу Program та функції Main. Код функції Main потрібний як точка входу в програму для її компіляції та запуску.
  3. Збірка програми, компіляція, запуск на виконання.
  4. Результат роботи програми:

ПІБ студента, курс, група

варіант завдання

версія 1

Старт імітації

Фініш імітації

======================================================================

#### Версія 2. «Конструктори та аксесори класів» ‒ 3 бали

1. Скопіювати проект програми версії 1. Перейменувати його у версію 2.
2. Додати в кожний файл класу код, що реалізує конструктори класів.
3. Додати до кожного класу код, що реалізує аксесори класів.
4. Відповідно до C# документації <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/constructors> розробити для кожного класу такі типи конструкторів:
   1. Конструктори без параметрів
   2. Конструктори з параметрами (конструктори ініціалізації полів класу)
   3. Статичні конструктори
   4. Закриті конструктори
   5. Конструктори копій
   6. Конструктори, що викликають інші конструктори
5. В конструкторах класів врахувати взаємозв'язки класів:
   1. асоціації
   2. агрегації
   3. композиції
6. Передбачити такі різновиди методів доступу (аксесорів) https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/using-properties
   1. властивості, що автоматично реалізуютьсях
   2. закриті та відкриті властивості
   3. властивості читання та запису
7. У функцію Main додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
8. Вивести на екран протокол роботи конструкторів, аксесорів у вигляді значень атрибутів об’єктів класів

====================================================================

#### Версія 3. «Реалізація предикатних функцій та деяких методів відповідно до постановки завдання» − 3 бали

1. Скопіювати проект програми версії 2. Перейменувати його у версію 3.
2. Додати в кожний файл класу код, що реалізує предикатні функції, які повертають значення true/false і служать для визначення стану об'єктів.
3. Узгодити з Product owner (викладачем) пріоритетні задачі, які слід реалізувати в третій версії проєкту.
4. Розробити код методів класів відповідно до узгодженого з Product owner (викладачем) сценарію роботи версії 3 програми.
5. У функцію Main додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
6. Здійснити виклики методів, що узгодження з Product owner (викладачем) відповідно до сценарію роботи прогами.
7. Додатково до протоколу роботи програми версії 2 вивести результати роботи доданих до класів методів.

=====================================================================

#### Версія 4. «Перевантаження операторів та реалізація методів другого пріоритету відповідно до завдання» − 3 бали

1. Скопіювати проект програми версії 3. Перейменувати його у версію 4.
2. Додати в класи код, що реалізує перевантаження операторів згідно з документацією: https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/operator-overloading

https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/expressions#operator-overloading

* 1. бінарні (арифметичні) оператори + - \* /
  2. унарні(арифметичні) оператори + - ++,--, !, true, false
  3. оператори порівняння ==. !=, <,>, <=, >=

1. Додати в класи код методів, що реалізують сценарій роботи програми і є задачами другого пріоритету, узгодженими з Product owner (викладачем).
2. У функцію Main додати код, що створює об'єкти класів. Врахувати взаємозв'язки класів.
3. Здійснити виклики методів, що перевантажують унірні та бінарні оператори.
4. Здійснити виклики методів, що узгодження з Product owner (викладачем) та відповідають сценарію роботи програми.
5. Додатково до протоколу роботи програми версії 3 вивести результати роботи доданих до класів методів.

====================================================================

#### Версія 5. «Реалізація успадкування класів, інтерфейсів та абстрактних класів» ‒ 4 бали

1. Скопіювати проект програми версії 4. Перейменувати його у версію 5.
2. Модернізувати діяграму класів, додавши додаткові базові, похідні, абстрактні класи та інтерфейси.
3. Додати в класи код, що реалізує успадкування класів (базові, абстрактні, інтерфейси) відповідно до діаграми класів.
4. Модернізувати конструктори класів, врахувавши конструктори базових класів під час розробки конструкторів похідних класів

<https://docs.microsoft.com/en-en/dotnet/csharp/programming-guide/classes-and-structs/inheritance>

1. Додати в класи код методів, що реалізують сценарій роботи програми і є задачами третього пріоритету, узгодженими з Product owner (викладачем).
2. У функцію Main додати код, що створює об'єкти базових та похідних класів. Врахувати взаємозв'язки класів.
3. Здійснити виклики методів, що узгодження з Product owner (викладачем) та відповідають сценарію роботи програми.
4. Додатково до протоколу роботи програми версії 4 вивести результати роботи доданих до класів методів.

===================================================================

#### Версія 6. «Реалізація обробників виключних ситуацій та повного сценарію роботи програми» - 4 бали

1. Скопіювати проект програми версії 5. Перейменувати його у версію 6
2. Проаналізувати опис предметної області та виявити критичні (ризикові) ситуації, які можуть привести до негативних наслідків за сценарєм роботи програми.
3. Реалізувати обробку виняткових (критичних) ситуацій засобами обробки виключень C#

https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/try-catch-finally

1. В якості виняткових ситуацій розглянути ситуації, які обробляються винятками стандартних класів C#, врахувавши:
   1. некоректність введення даних,
   2. вихід за межі масивів,
   3. неможливість відкрити файл
   4. ділення на нуль тощо
2. Створити користувацькі класи винятків. В якості методів користувацьких класів винятків, враховувати винятки згідно зі сценарієм роботи програми за вибраним проєктом. Скмантику винятків узгодити з Product owner (викладачем).
3. У функцію Main додати код, що створює об'єкти похідних та виняткових класів. Врахувати взаємозв'язки класів.
4. Здійснити виклики методів, що узгодження з Product owner (викладачем) та відповідають сценарію роботи програми.
5. Додатково до протоколу роботи версії 5 програми вивести результати роботи доданих до класів методів з критичними ситуаціями та результати їх обробки.

==================================================================

### Звіт з виконання проєкту

Для формування звіту з проєкту рекомендується використати програми автоматизації створення документації коду, зокрема **Doxygen** (<https://www.doxygen.nl/index.html>).

Скачати безкоштовно Doxygen 1.9.1 можно із сайту: <https://besplatnye-programmy.com/raznoe-dlya-razrabotchikov/2479-doxygen.html>.

Для включення у звіт HIPO діаграм та діаграм класів додатково до Doxygen потрібно встановити програму **Graphviz (**https://graphviz.org/download/)

Формування звіту з використанням Doxygen здійснюєится протягом 1-2 хвилину і являє собою локальний сайт із більше ніж 30 html сторінок.

Інструкція з користування Doxygen 1.9.1 для студентів знаходиться за посиланням: <https://github.com/tkovalyuk/introduction-to-OOP>, файли DoxygenInstructionDoc.docx, DoxygenДОКУМЕНТИРОВАНИЕ ПРОЕКТОВ НА C#.pdf

#### Приклади екранних копій звіту, створеного за допомогою Doxygen 1.9.1

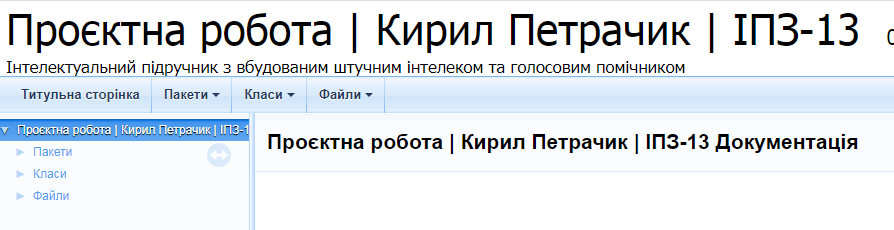


Рисунок 6 − Стартова сторінка звіту, згенерованого програмою **Doxygen**

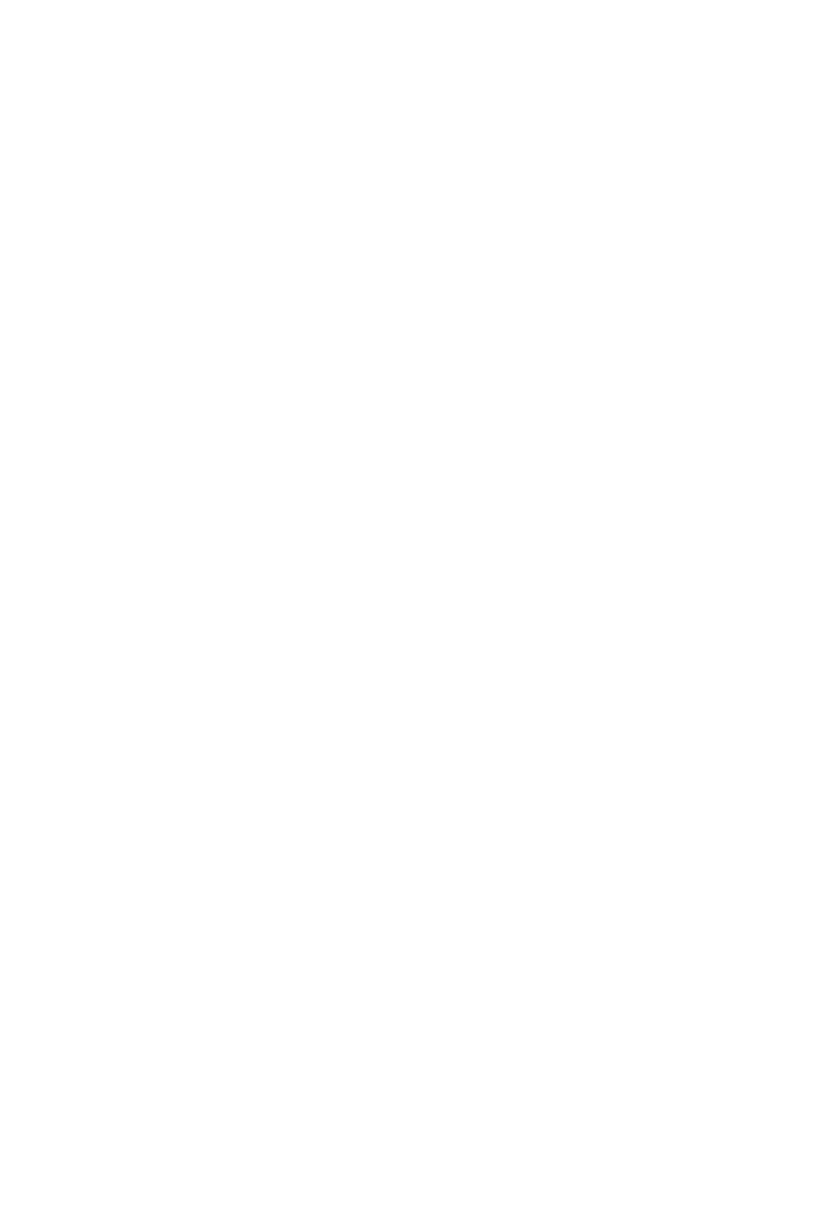
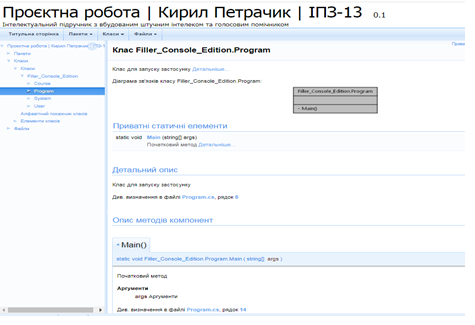
****

Рисунок 7 ‒ Сторінка документації класу.

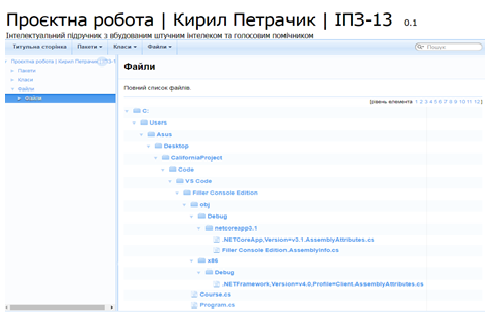


Рисунок 8 ‒ Сторінка документації файлів

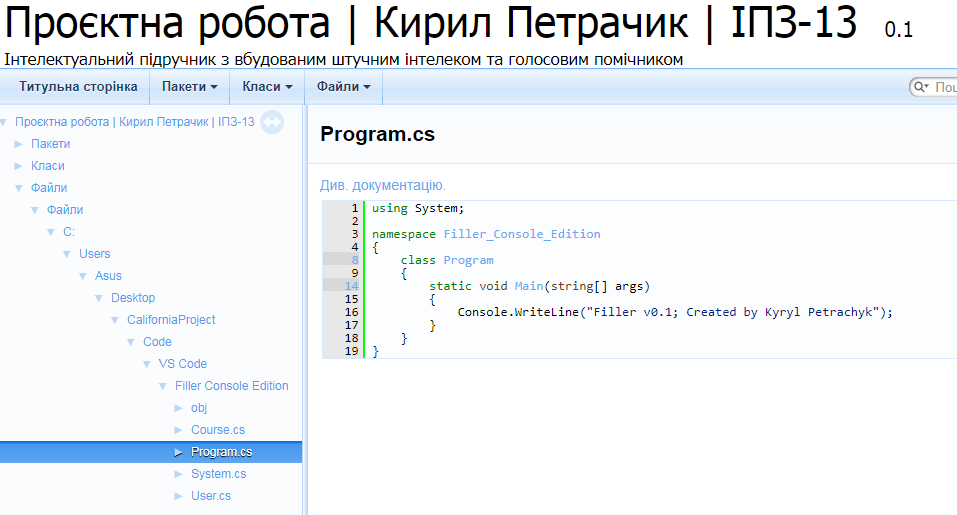


Рисунок 9 ‒ Сторінка документації коду методу

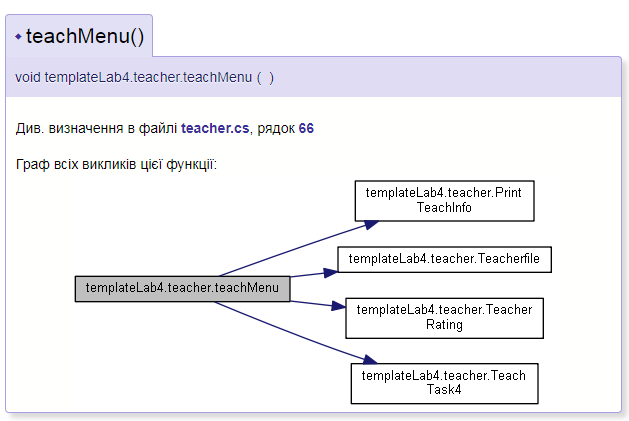


Рисунок10 ‒ Сторінка документації графу викликів функції

### Теми проєктів з дисципліни «Вступ до ООП»

| **№ п.п.** | **Назва предметної області** | **Складність** | **Опис предметної області (бізнес-кейс)** | **Джерела (реальні прототипи та аналоги)** |
| --- | --- | --- | --- | --- |
|  | Моделювання бізнес-процесів в системі Smart-city | 100 | **Мета**: на основі моделювання бізнес-Smart-city показати шляхи покращення та спрощення управління містом, благоустрій міського середовища, забезпечення безпеки та підвищення якості життя жителів міста  **Задачі моделювання**:   1. Громадська безпека: ситуаційний центр для управління камерами спостереження на вулицях міста, формування карти криміногенної ситуації в місті, прогнозування місць можливих злочинів, робота call- центра з реєстрації заяв громадян, зворотній зв’язок з жителями міста 2. Комунальний транспорт: електронний розклад і контроль за його виконанням, GPS навігація для контролю за місцем знаходження транспорту, електронний квиток, контроль стану працездатності транспорту, облік та контроль пасажирів тощо 3. Житловий фонд міста: облік наявності житла, контроль за його станом, комунальні платежі через електронні сервіси, тощо. 4. Електронні комунікації жителів з владою: контактний центру 1551, система інформування киян, електронні петиції тощо 5. І.т.д.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач Smart-city | <https://mind.ua/openmind/20204322-stolicya-z-rozumom-sogodni-i-zavtra-sistemi-kyiv-smart-city>  <https://www.phoenixcontact.com/online/portal/ua?1dmy&urile=wcm%3Apath%3A/uauk/web/main/solutions/subcategory_pages/Smart_cities/0db6c7f7-d5ba-4959-ba59-dd009317cb85> |
|  | Модклювання процесів створення та використанняСмарт-контрактів | 100 | Смарт-контракт - це угода, що виконується автоматично у разі настання заздалегідь визначених сторонами умов. Використання смарт-контрактів дозволить оптимізувати бізнес-процеси, пов'язані із взаємодією між контрагентами, а також мінімізувати час та витрати під час угоди. Зацікавлені сторони прописують у смарт-контрактах умови угод, санкції за їх невиконання та ставлять цифрові підписи. Розумний контракт визначає самостійно, чи дотримані всі умови, чи все виконано, і приймає рішення: успішно завершити угоду та видати потрібне (гроші, нерухомість, акції) або накласти на учасників штрафні санкції та закрити доступ до активів. Мета роботи – на основі моделювання бізнес-процесів при створенні та використання смарт-контрактів створити навчальний інструмент, що демонструє переваги смарт-контрактів зі звичайними, «паперовими» договорами **Об'єкти смарт-контрактів, що моделюються**:   1. Сторони угоди, що мають цифровий підпис, які погоджуються або відмовляються від відповідності товару або послуги з висунутих раніше вимогах. 2. Предмет договору ‒ товар або послуги, які будуть відправлені в обмін на грошові кошти. 3. Умови, при дотриманні яких буде проведений автоматичний обмін благами, наприклад, відповідність поставленого товару стандартам якості. Повинні мати повний математичний опис. 4. Децентралізована платформа, в якій написаний алгоритм (програмний код) самого смарт-контракту.  Задачі моделювання:  1. Облік та передача прав власності. 2. Операції з цінними паперами. 3. Проведення міжнародних розрахунків, наприклад, з використанням акредитиву. 4. Ідентифікація особистості. 5. Фінансові звітність. 6. Обробка платежів за кредитам. 7. Складання та передача активів за заповітом 8. Перевірка на відповідність поставлених товарів встановленим стандартам. 9. Передача інших цифрових активів тощо  Робота смарт–контракта: Смарт-контракт записується в блокчейн, де його код розміщується в програмний контейнер – блок. У цьому блоці об'єднуються всі повідомлення, що стосуються смарт-контракту. Повідомлення – це входи та виходи смарт-контракту (умови, які «чекає» смарт-контракт, та дії, які він виконує). За допомогою повідомлень контракт пов'язується з цифровим або реальним світом поза блокчейном.  **Обов'язкові атрибути смарт–контракта:**   * + - 1. електронний підпис на основі публічних та приватних ключів, що є у двох та більше сторін контракту;       2. приватне децентралізоване середовище, в яке записуються смарт-контракти і яке підтримує входи та виходи для оракулів.       3. оракули – це спеціальні програми, що виконують роль мостів між блокчейном та зовнішнім світом;       4. сам предмет договору та всі необхідні для нього інструменти (розрахункові рахунки, програми-оракули тощо);       5. точно визначені умови виконання договору, які учасники договору підтверджують своїми підписами, і навіть підтверджують достовірність джерела цифрових даних.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при роботі смарт-контрактів або яка пропонує бізнес-кейси, на яких можна відпрацювати технологію використання сматр-контрактів. | <https://www.h-x.technology/ru/services/smart-contract-development-ru>  <https://btc-up.com/shho-take-smart-kontrakty/>  <https://cyberleninka.ru/article/n/smart-kontrakty-funktsii-i-primenenie>  <https://ethereum.org/ru/>  <https://www.popmech.ru/technologies/397902-ethereum-platforma-dlya-blokcheyn-sistem-i-eyo-sozdatel-vitalik-buterin/> |
|  | Проектування індивідуальних освітніх траєкторій відповідно до мотивацій та здібностей студента | 100 | **Мета**: на основі моделювання бізнес-процесів при проектуванні індивідуальних освітніх траєкторій показати шляхи реалізації студентоцентрованого навчання через створення індивідуальних навчальних планів за критеріями задоволення мотиваційних вимог студентів.  **Об’єкти моделювання:** студенти, мотивації, навчальний план, дисципліни.  **Задачі моделювання**:   1. Реалізація алгоритму латентно-семантичного аналізу мотиваційних листів студентів; 2. Визначення професійної орієнтації студента 3. Визначення областей знань для вибраної студентом професії 4. Визначення компетентностей (знань та умінь) для опанування вибраної професії 5. Визначення дисциплін, які формуватимуть потрібні знання та уміння 6. Побудова індивідуального навчального плану   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач побудови індивідуального навчального плану | [https://cyberleninka.ru/article/v/proektirovanie-individualnoy-obrazovatelnoy-traektorii-i-marshruta-studenta-vuza-buduschego-bakalavra](https://cyberleninka.ru/article/v/proektirovanieindividualnoyobrazovatelnoytraektoriiimarshrutastudentavuzabuduschegobakalavra)  <https://habr.com/ru/company/Voximplant/blog/446738/>  https://habr.com/ru/company/Voximplant/blog/446738/  https://habr.com/ru/company/Voximplant/blog/446738/ |
|  | Моделювання бізнес-процесів в тренінговій системі формування Soft skills | 100 | **Мета**: на основі моделювання бізнес-процесів в тренінговій системі формування Soft skillsпоказати шляхи формування особистісних якостей, зокрема, креативність, лідерські, комунікаційні, виконавські, вольові та інші навички.  **Об’єкти моделювання:** людина, що тренується; система, що тренує людину; види тренінгів; .  **Задачі моделювання**:   1. Проходження психологічних тренінгів (форма групової роботи) онлайн і визначення здатності для розвитку:    1. психологічної стійкості, міцності нервів;    2. комунікабельності, навичок спілкування, міжособистісної та суспільної взаємодії;    3. швидкого прийняття рішень, чітких дій в складних життєвих і професійних ситуаціях.    4. адаптивності в колективі. 2. Проходження бізнес-тренінгів і визначення здатності для розвитку навичок:    1. управління персоналом;    2. структурування власних цілей, розробки стратегії реалізації ідей і проектів;    3. гнучкого реагування на мінливу ситуацію та швидкого прийняття рішень;    4. управління власним часом та використання власних робочих ресурсів. 3. Проходження кейсів (описів практичних ситуацій) для аналізу та прийняття рішень в ситуаціях, що описують кейси. 4. Побудова індивідуального плану розвитку особистості 5. Профорієнтація 6. Написання резюме 7. тощо   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач управління розвитком Soft skills. | <https://www.mental-skills.ru/training/>  <https://t1.ua/porady/4217-psykholohichni-treninhy-dlya-choho-vony-potribni.html> |
|  | Моделювання бізнес-процесів діяльності проектно-орієнтованого університету за моделлю потрійної спіралі | 100 | **Мета**: на основі моделювання бізнес-процесів в діяльності проектно-орієнтованого університету показати шляхи підвищення ефективності навчального процесу та якості підготовки студентів відповідно до вимог ринку праці.  Необхідно показати форму взаємодії університетів, індустрії та влади, яка стимулює інноваційні процеси в університеті, перетворює його на підприємницький, розвиває у студентів навики підприємництва а проектної діяльності, та інш  **Об’єкти моделювання:** університет, викладачі, студенти, компанії, працівники компаній, проекти,  **Задачі моделювання**:   1. Моделювання співпраці університетів і компаній:    * + 1. Розробка освітніх програм. навчальних планів та навчальних програм дисциплін відповідно до вимог індустрії;        2. Фінансова, методична та організаційна підтримка університетів з боку компаній        3. Виконання спільних проектів;        4. Реалізації дульної освіти студентів (20 годин на тиждень – робота в компанії, 20 годин – навчання в університеті);        5. Стажування студентів та викладачів в компаніях;        6. Викладання працівниками компаній в університетах; 2. Моделювання співпраці підприємців з державою:   a) Податкові пільги у разі взаємодії компаній з університетами;   1. Державні замовлення на проекти та товарів 2. Створення ринків збуту товарів, що вироблені компаніями 3. Моделювання співпраці держави і університету    1. Державне фінансування університетів;    2. Державне замовлення щодо набору студентів та випуску фахівців    3. Автономія університетів (невтручання влади в академічні, фінансові, кадрові питання) 4. Моделювання розвитку бізнесу внаслідок розвитку проектно-орієнтованих університетів 5. Моделювання розвитку университетув внаслідок партнерства з компаніями 6. Моделювання управління розвитком компетентностей студентів внаслідок навчання їх на реальних проектах   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моделювання діяльності проектно-орієнтованого університету за моделлю потрійної спіралі | <http://www.kbuapa.kharkov.ua/e-book/apdu/2017-2/doc/7/02.pdf>  <http://dspace.nbuv.gov.ua/bitstream/handle/123456789/28113/08-Galan.pdf?sequence=1>  <http://www.economy.nayka.com.ua/?op=1&z=4798>  <http://ief.org.ua/wp-content/uploads/2017/01/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BE%D1%82%D1%80%D1%96%D0%B9%D0%BD%D0%BE%D1%97-%D1%81%D0%BF%D1%96%D1%80%D0%B0%D0%BB%D1%96.pdf>  <https://pidruchniki.com/84904/ekonomika/nova_rol_universitetiv_innovatsiynomu_pidpriyemnitstvi>  <https://cyberleninka.ru/article/n/model-troynoy-spirali-kak-mehanizm-innovatsionnogo-razvitiya-regiona> |
|  | Моделювання бізнес-процесів функціонування освітньо-виробничих ІТ-кластерів | 100 | **Мета**: на основі моделювання бізнес-процесів кластерної організації ІТ-освіти показати шляхи покращення якості вищої освіти через взаємодію ІТ-освіти та ІТ-індустрії.  **Об’єкти моделювання:** університет, ІТ компанії, студенти, викладачі, ментори з ІТ-компаній освітні програми.  **Задачі моделювання**:   1. Удосконалення інфраструктури університету; 2. Фінансова підтримка університетів (поставка обладнання, создание лабораторій, проведення майстер-класів, фінансування наукових проектів); 3. Оновлених технічних знань, мережі створення і використання знань, доступ до прикладних знань з використанням в академічних дослідженнях і навчальному процесі; 4. Запровадження нових технологій і процесів, що дозволяє досягти конкурентних переваг; 5. Набір кваліфікованого персоналу і підвищення кваліфікації працівників; 6. Інтеграція з ринком праці та пропозиції щодо роботи для випускників; 7. Стажування і стипендії для викладачів; 8. Оплата роялті при передачі технологій. 9. Наявність ефективної системи захисту прав інтелектуальної власності в університетах; 10. Наявність податкового кредиту при співробітництві з університетами; 11. Доступ до фізичних ресурсів університету та атестації своїх співробітників; 12. Доступ до дослідницьких і консалтингових послуг університету; 13. Поліпшення громадського іміджу компанії в суспільстві; 14. Збільшення продажів і прибутковості; 15. Створення інноваційної культури в своїх установах; 16. Набір випускників для виконання проектів..   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в процесі функціонування освітньо-виробничих ІТ-кластерів | <http://danmarksvaekstraad.dk/file/14334/Opl%C3%A6gafChristianKetels.pdf>  <https://inecon.org/docs/2018/Smorodinskaya_Skolkovo_20180615.pdf>  <https://vbudushee.ru/upload/iblock/f47/f47425d3a3eeae0b4d37ce157f622aea.pdf>  <https://www.researchgate.net/publication/332148597_Educational-industrial_cluster_for_development_of_youth_entrepreneurship>  <https://cyberleninka.ru/article/n/model-troynoy-spirali-kak-mehanizm-innovatsionnogo-razvitiya-regiona>  <https://cyberleninka.ru/article/n/ispolzovanie-klasternogo-podhoda-v-tselyah-povysheniya-konkurentosposobnosti-sistemy-obrazovaniya-sravnitelnyy-analiz-2> |
|  | Моделювання електронного підручника із вбудованим штучним інтелектом | 100 | **Мета**: на основі моделювання бізнес-процесів електронного підручника із вбудованим штучним інтелектом показати шляхи покращення якості підготовки студентів.  **Навчальний підручник** перестає бути чим-то статичним. На першому етапі - це живий підручник із постійно оновленим і адекватним реальному часу змістом, на другому - «алмазний буквар», тобто підручник із вбудованим штучним інтелектом, який є особистим тьютером, що відповідає індивідуальним вимогам студентів.  **Об’єкти моделювання:** студент, підручник, штучний інтелект, знання та уміння студента  **Задачі моделювання**:  1. Ініціалізація користувача.  2. Аналіз вимог користувача, його вподобань до навчальних предметів та графік дедлайну для тестування.  3. Збирання інформації по навчальним дисциплінам.  4. Планування графіку для навчання користувача.  5. Видача користувачу навчального матеріалу.  6. Тестування користувача по пройденій темі.  7. Аналіз успішності користувача.  8. Оптимізація навчального процесу.  9. Модифікація контенту підручника відповідно до вимог в індустрії  **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання електронного підручника зі штучним інтелектом. | <https://cyberleninka.ru/article/n/osnovy-razrabotki-elektronnyh-uchebnikov-po-napravleniyu-shemotehnika-elektronnyh-sistem>  <https://www.livejournal.com/media/43629.html>  <https://ntinews.ru/blog/publications/iskusstvennyy-intellekt-v-obrazovanii-sotsialno-filosofskie-aspekty.html>  <https://www.slideshare.net/liketaurus/ss-2898496> |
|  | Моделювання бізнес-процесів системи управління знаннями організації | 100 | **Мета**: на основі моделювання бізнес-процесів системи управління знаннями організації показати шляхи поліпшення використання існуючих або створення нових індивідуальних чи колективних ресурсів знань з метою підвищення конкурентоспроможності організації.  **Об’єкти моделювання**: організація; інформація; документи, фахівці, система управління знаннями.  **Задачі моделювання:**   1. Ідентифікація та отримання знань    1. Web-ресурси    2. Документи    3. Фахівці (експерти, викладачі, науковці, 2. Створення (нових) знань 3. Зберігання знань    1. Бази даних    2. Онтології 4. Обмін знаннями та розповсюдження знань    1. Пошукові системи    2. Системи навігації    3. Системи комунікації    4. E-learning    5. Інтелектуальний аналіз 5. Застосування знань    1. Стратегічне планування знань    2. Аналіз областей знань (карти знань, аналітичні висновки та огляди, структурні схеми тощо)    3. Розробка цільових профілів знань    4. Розробка кадрової концепції    5. Розробка вимог до інформаційних та комунікаційних систем 6. Архітектура системи менеджменту знань   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності системи управління знаннями . | <https://www.redcross-irkutsk.org/upload/catalog/files/products/891.pdf>  <http://window.edu.ru/resource/402/26402/files/_5.pdf>  <https://www.gapm.ru/wordpress/wp-content/uploads/aniskina_quality_2018.pdf>  <https://pdf.standartgost.ru/catalog/Data2/1/4293779/4293779674.pdf>  <https://books.google.com.ua/books?id=z0i4DwAAQBAJ&pg=PA141&lpg=PA141&dq=CWA+14924-5&source=bl&ots=Re37rQjAQq&sig=ACfU3U29uwQeIc9xsD7-sLE0ZiFPJTT5yA&hl=uk&sa=X&ved=2ahUKEwiYsOi4icfoAhXqkIsKHXB6DxIQ6AEwA3oECGEQLA#v=onepage&q=CWA%2014924-5&f=false> |
|  | Моделювання бізнес-процесів на платформі Smart університет | 95 | **Мета**: на основі моделювання бізнес-процесів програмно-інформаційної платформи Smart-університет показати шляхи покращення управління архітектурою та організаційною структурою університету, поліпшення системи електронної мережевої взаємодії викладача та студента, підвищення якості підготовки студентів за рахунок створення смарт-середовища університету.  **Задачі моделювання**:   1. Децентралізація управління університетом: центри управління переходять до повноважень факультетів і кафедр, диктат ректорату зменшується, ректорат вирішує тільки стратегічні задачі. 2. Студентоцентрований підхід: імітація навчального процесу за гнучкими персональними індивідуальними освітніми траєкторіями, тобто за індивідуальними планами, які будуються за бажаннями студентів, їхніми професійними здібностями, мотиваціями, кар’єрними баченням тощо. 3. Смарт-середовище університету:    1. програмна імітація реалізації принципів смарт-освіти:       * модернізація освітніх програм і навчальних планів спеціальностей, робочих навчальних програм дисциплін в режимі real time;       * самостійна дослідницька проектна діяльність студентів за реальними темами і проектами компаній;       * розподілене навчальне середовище (дистанційне навчання, мобільні технології навчання, практика в компаніях, навчання в професійному середовищі, навчання протягом всього життя тощо;       * взаємодія викладачів і студентів з професійним співтовариством (проф. компаніями);    2. програмна імітація застосування смарт-технологій:       * створення сучасного навчального контенту з використанням активних методів навчання (бізнес-кейси, ділові та рольові ігри, хакатони, доповнена реальність, проектний підхід тощо);       * електронний доступ студентів до навчального контенту (навчальні сайти викладачів з конспектами лекцій, завданнями до лаб.роб, бізнес-кейсами тощо;       * використання ресурсів MOOC – масові відкриті освітні Інтернет-курси і зарахування результатів їх проходження в рейтингах студентів;    3. програмна імітація ?????   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної платформи «Smart- університет». | <https://www.kyivsmartcity.com/school/>  <http://www.euroosvita.net/index.php/?category=1&id=6361>  <http://wiki.kubg.edu.ua/%D0%A1%D0%A2%D0%92%D0%9E%D0%A0%D0%95%D0%9D%D0%9D%D0%AF_%D0%A1%D0%9C%D0%90%D0%A0%D0%A2-%D0%A3%D0%9D%D0%86%D0%92%D0%95%D0%A0%D0%A1%D0%98%D0%A2%D0%95%D0%A2%D0%A3_%E2%80%94_%D0%9D%D0%90%D0%A8%D0%95_%D0%A1%D0%A2%D0%A0%D0%90%D0%A2%D0%95%D0%93%D0%86%D0%A7%D0%9D%D0%95_%D0%97%D0%90%D0%92%D0%94%D0%90%D0%9D%D0%9D%D0%AF>  <http://www.univ.kiev.ua/pdfs/official/Development-strategic-plan.pdf>  <https://www.digiblitz.com/wp-content/uploads/2017/08/smartLehren_Brochure.pdf> |
|  | Моделювання бізнес-процесів в Smart системі моніторингу дорожнього руху | 95 | **Мета**: на основі моделювання бізнес-процесів в інтелектуальній системі моніторингу дорожнього руху (ІСМДР) показати шляхи оперативного контролю транспортних потоків на дорогах, покращити облік зафіксованих системою дорожніх подій та ідентифікації транспортних засобів, що порушили ПДР та скоїли ДТП.  **Об’єкти моделювання:** транспортний засіб, номерний знак, дорога та смуги,засобиGPS-моніторингу, датчики виявлення автомобілів, програмне та технічне забезпечення ІСМДР.  **Задачі моделювання**:   1. Відео моніторинг інтенсивності дорожнього руху:  * отримання та обробка відео зображення з різних джерел відеоспостереження; * підрахунок кількості автомобілів за визначений період часу на смузі; * визначення середньої швидкості руху за визначений період часу на смузі; * визначення середньої зайнятості смуги за визначений період часу на смузі. * класифікація типів транспортних засобів; * видача сигналів про інциденти та про зупинку транспортного засобу. * видача сигналу за поганої видимості. * фіксація транспортних засобів, що порушують правила дорожнього руху.  1. Розпізнавання державних номерних знаків транспортних засобів:   виведення прийнятого зображення на екран;   * пошук на зображенні транспортного засобу номерних знаків, їх детекція та розпізнавання з подальшим занесенням в загальну базу розпізнаних номерів; * порівняння розпізнаного номера з номерами, наявними в базі для розшуку транспортного засобу; * зберігання кольорового зображення транспортного засобу з розпізнаним номером.  1. GPS-моніторинг та керування транспортним засобом:  * відслідковування будь-якої кількості мобільних об’єктів; * отримання інформації про місце розташування об’єктів з точною локалізацією на карті напрямку і швидкості руху; * отримання статистики пересувань об’єктів за будь-який заданий проміжок часу з відображенням на карті всіх переміщень; * отримання відомості про пройдений об’єктами шлях у кілометрах; * зберігання інформації в базах даних про кожний об’єкт, включаючи маршрути, обмін повідомленнями й ін.; * підтримання зв’язку з водіями в процесі моніторингу дорожнього руху для інформування про погодні умови, стан дороги, дорожні аварії тощо.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моніторингу дорожнього руху. | <https://nau.edu.ua/ua/menu/science/naukovi-rozrobki/intelektualna-sistema-monitoringu-dorozhnogo-ruxu.html>  <https://apluss.pro/upravlinnya-transportnoyu-systemoyu/intelektualni-transportni-sistemi-its/> |
|  | Моделювання бізнес-процесів в системі автоматичного контролю пасажирських перевезень | 95 | **Мета**: на основі моделювання бізнес-процесів в системі контролю пасажиропотоків в громадському транспорті показати шляхи забезпечення безперебійної роботи громадського транспорту; покращення обліку та регулювання кількості пасажирів в громадському транспорті та оперативному контролю руху транспорту в режимі реального часу.  **Об’єкти моделювання:** транспортний засіб, номерний знак, дорога та зупинки,засобиGPS-моніторингу, датчики підрахунку кількості пасажирів.  **Задачі моделювання**:   1. Відслідковування пасажирами даних про пересування громадського транспорту за допомогою GPS:  * Пошук місцезнаходження автобусів та електротранспорту за номерами маршрутів та зупинками; * Визначення часу прибуття транспорту на певну зупинку; * Відображення графіку руху транспорту. * Відображення карти руху транспорту  1. Перевірка Державними органами кількості транспортних засобів на маршрутах:  * пошук на зображенні транспортного засобу номерних знаків, їх детекція та розпізнавання з подальшим занесенням в загальну базу розпізнаних номерів; * порівняння розпізнаного номера з номерами, наявними в базі для розшуку транспортного засобу; * зберігання кольорового зображення транспортного засобу з розпізнаним номером.  1. Автоматичний підрахунку кількості пасажирів в транспорті:  * відслідковування будь-якої кількості мобільних об’єктів; * отримання інформації про місце розташування об’єктів з точною локалізацією на карті напрямку і швидкості руху; * отримання статистики пересувань об’єктів за будь-який заданий проміжок часу з відображенням на карті всіх переміщень; * отримання відомості про пройдений об’єктами шлях у кілометрах; * зберігання інформації в базах даних про кожний об’єкт, включаючи маршрути, обмін повідомленнями й ін.; * підтримання зв’язку з водіями в процесі моніторингу дорожнього руху для інформування про погодні умови, стан дороги, дорожні аварії тощо.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моніторингу пасажиропотоку. | <https://gps-ukraine.com/ua/kontrol-pasazhyropotoku/>  <http://uapatents.com/4-103090-sposib-avtomatichnogo-pidrakhunku-ta-obliku-pasazhiriv.html>  <http://tvoemisto.tv/news/mobilnyy_dodatok_dlya_vidslidkovuvannya_transportu_u_lvovi_de_zavantazhyty_76429.html>  <https://stud.com.ua/120693/informatika/sistemi_identifikatsiyi_pasazhiriv> |
|  | Моделювання бізнес-процесів в інформаційній системі «Smart Region» | 95 | **Мета**: на основі моделювання бізнес-процесів в інформаційній системі «Smart Region» покращити доступ користувачів до інформації щодо проектів, які плануються або реалізуються в регіоні для підтримки або відхилення їх.  **Об’єкти моделювання:** користувачі, проекти, регіональна влада, .  **Задачі моделювання**:   1. Реєстрація, авторизація та вхід користувача до системи. 2. Перелік проектних платформ: назва проекту, опис завдань проекту, вартість проекту, очікувані результати впровадження проекту, головування користувачів за підтримку проектів або їх відхилення. 3. Платформа громадського фінансування: допомога постраждалим від стихійних лих, підтримка з боку вболівальників чи фанатів, підтримка політичних кампаній, фінансування стартап-компаній та [малого бізнесу](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BB%D0%B8%D0%B9_%D0%B1%D1%96%D0%B7%D0%BD%D0%B5%D1%81), створення вільного програмного забезпечення тощо. 4. Бюджет регіону: джерела та обсяг доходів та податків, статті та обсяг витрат, які планує регіональна влада.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної системи «Smart Region» | <https://poltava.to/news/52894/>  <https://i1.poltava.to/uploads/2019/10/2019-10-10/tz-smart-region.pdf> (п. 3.2, стор 15 - 30 ) |
|  | Моделювання бізнес-процесів функціонування інноваційних ІТ-кластерів | 95 | **Мета**: на основі моделювання бізнес-процесів ІТ кластерів показати шляхи підвищення продуктивності і гарантування конкурентоздатності регіону та держави на міжнародному рівні, а також покращення якості підготовки ІТ фахівців через взаємодія ІТ-освіти та ІТ-індустрії.  **Об’єкти моделювання:** ІТ компанії, ІТ-фахівці, мережа ІТ компаній та фахівців, університети проекти, події  **Задачі моделювання**:   1. ІТ—освіта:    1. Модернізація навчальних програм у ЗВО, формування та коригування навчальних планів;    2. Шкільні та студентькі олімпіади з математики, програмування, ІТ ;    3. ІТ конференції (за участю представників найбільших ІТ-компаній, інвестиційних фондів, бізнесу та уряду, дизайнерів, топ-менеджерів, підприємців, бізнес аналітиків та стартаперів.)    4. Фінансова допомога ЗВО    5. Навчальні курси для студентів та стажування викладачів    6. І.т.д. 2. ІТ-право:    1. Юридична підтримка та захист прав учасників кластера    2. Правове консультуання та лобіювання законів, що стосуються розвитку ІТ індустрії    3. Навчальні семінари з правової тематики діяльності ІТ компаній 3. Міжнародна діяльність    1. Співробітництво з посольствами для формування позитивному іміджу ІТ індустрії регіону    2. пошук інвесторів, партнерів,    3. вихід на нові ринки впровадження стартапів.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в процесі функціонування інноваційних ІТ-кластерів | <https://appau.org.ua/info/hajtek-klastery-v-ukrayini-potochnyj-status-kvo-ta-vyklyky-rozvytku/>  <https://www.beratergruppe-ukraine.de/wordpress/wp-content/uploads/2016/01/PP_04_2015_ukr.pdf>  <https://dou.ua/lenta/articles/it-cluster-review/>  <https://itcluster.lviv.ua/>  <https://nv.nltu.edu.ua/Archive/2018/28_9/16.pdf>  <https://www.researchgate.net/publication/329523872_Perspektivi_rozvitku_IT-klasteriv_v_Ukraini_na_osnovi_vencurnogo_investuvanna/link/5c0db83c299bf139c74d5280/download>  <http://dspace.puet.edu.ua/bitstream/123456789/7046/1/Dissertation%20Onipko.pdf> |
|  | Моделювання бізнес-процесів впровадження стандарту ISTE (International Society for Technology in Education) | 95 | **Мета**: на основі моделювання бізнес-процесів при впровадженні стандарту ISTE у ЗВО показати шляхи удосконалення освітнього процесу та покращення якості підготовки студентів.  **Об’єкти моделювання:** студенти, викладачі, завідувач кафедри (гарант освітньої програми), цифрові технології, стандарти ISTE  **Задачі моделювання:**   1. Реалізація шляхів для формування таких компетентностей у студента: 2. Внутрішня мотивація 3. Громадянин цифрового суспільства 4. Критичне пізнання дійсності 5. Відкритість новому знанню 6. Розвинуте алгоритмічне та інженерне мислення 7. Творча взаємодія з іншими людьми 8. Здатність до співпраці і командній роботі 9. Реалізація шляхів для формування таких компетентностей у викладача ЗВО: 10. Аналітик 11. Спеціаліст 12. Лідер 13. Громадяни цифрового суспільства 14. Організатор навчання у співпраці 15. Спеціаліст по педагогічному дизайну 16. Фасилітатор (що забезпечує успішну групову комунікацію) 17. Реалізація шляхів для формування таких компетентностей у викладача інфоматики: 18. компетентність 19. Ефективні стратегії викладання і навчання 20. Ефективна освітнє середовище 21. Ефективне використання предметних навичок 22. Реалізація шляхів для формування таких компетентностей в освітнього технолога: 23. Підтримка і супровід колег 24. Допомога в викладанні і оцінюванні 25. Створення цифрової освітнього середовища 26. Розробка і оцінка навчально-методичних матеріалів 27. Популяризація цифрового громадянства 28. Фахівець в галузі професійного розвитку 29. Реалізація шляхів для формування таких компетентностей у керівника освітньої організації 30. стратегічне керівництво 31. Створення освітньої культури цифрового століття 32. Професійне впровадження інновацій 33. Системні перетворення 34. Популяризація цифрового громадянства   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання впровадження стандартів ISTE у ЗВО | <https://hope.edu/academics/education/resources/Refreshed%20ISTE%20Standards%20for%20Educators%20-Permitted%20Educational%20Use.pdf>  <https://cdn.iste.org/www-root/Libraries/Documents%20%26%20Files/PDFs/ISTE%20Standards%202017%20RUS%20web%20version.pdf?_ga=2.61439659.667422961.1585251488-889930646.1584814270>  <https://www.iste.org/standards/standards-in-action/global-reach>  <https://www.iste.org/standards>  <https://www.iste.org/standards/for-students>  <https://en.wikipedia.org/wiki/International_Society_for_Technology_in_Education>  <https://id.iste.org/my-profile/standards-download> |
|  | Моделювання бізнес-процесів впровадження стандарту CDIO (Conceive - Design - Implement - Operate) для модернізації інженерної освіти | 95 | **Мета**: на основі моделювання бізнес-процесів при впровадженні стандарту CDIO у ЗВО показати шляхи удосконалення освітнього процесу, покращення якості підготовки студентів та визначити відповідність діяльності ЗВО стандартам CDIO.  **Об’єкти моделювання:** студенти, викладачі, освітня програма, стандарти CDIO  **Задачі моделювання:**   1. Моделювання шляхів формування у студента здатностей Планувати, Проектувати, Виробляти і Застосовувати інженерні продукти та оцінка відповідності освітнього процесу стандарту CDIO (standard 1). 2. Моделювання шляхів придбання студентами результатів навчання для розвитку особистісних умінь і навичок створення продуктів, процесів і систем, а також дисциплінарних знань відповідно до цілей освітньої програми та інтересів роботодавців ( standard 2 та CDIO Syllabus). 3. Моделювання шляхів розробки навчального плану, що передбачає таку організацію навчального процесу, яка веде до придбання особистісних і міжособистісних навичок, а також навичок створення продуктів, процесів і систем (standard 3). 4. Моделювання шляхів ведення проектно-впроваджувальної діяльності студентів (standard 5): 5. отримання проектно-впроваджувального досвіду в рамках освітнього процесу (дослідження в процесі виконання лабораторних робіт, курсового та дипломного проектування і т.д.) 6. отримання проектно-впроваджувального реального досвіду в процесі дуальної освіти, хакатонів, конкурсних робіт тощо. 7. Моделювання шляхів інтегрованого навчання для отримання предметних знань поряд з особистісними навичками (наприклад, під час виконання технічного проекту розглядати соціальні питання) (standard 7). 8. Моделювання шляхів оцінювання особистісних і міжособистісних навичок, навичок створення продуктів, процесів і систем, а також дисциплінарних знань, які студенти набувають в процесі навчання (standard 11)   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання впровадження стандартів CDIO у ЗВО | <http://iie.tpu.ru/files/CDIO_Standard.pdf>  <http://www.cdio.org/implementing-cdio/standards/12-cdio-standards#standard1>  <http://iie.tpu.ru/files/CDIO%20Syllabus_rus.pdf>  <http://www.cdio.org/knowledge-library/project-based-learning>  <http://edu.sfu-kras.ru/sites/edu.sfu-kras.ru/files/Mezhdunarodnye_standarty_CDIO_v_obrazovatelnom_standarte_TPU.pdf> |
|  | Моделювання застосовування серйозних ігор в навчальному процесі (гейміфікація освіти) | 95 | **Мета**: на основі моделювання бізнес-процесів при застосовування серйозних ігор в освітньому процесі ЗВО показати шляхи удосконалення освітнього процесу, покращення зацікавленість та вмотивованість студентів до набуття знань та умінь.  **Об’єкти моделювання:** студенти, викладачі, серйозна гра (платформа, завдання), результати навчання, дисципліна,  **Задачі моделювання:**   * + 1. Моделювання динаміки гри:  1. Розвиток мотивацій студентів, 2. Підтримка інтересу до вивчення дисципліни    * 1. Реалізація механіки гри:         1. Сценарій, правила та стратегія гри         2. Взаємодія з гравцем         3. Інформування      2. Моделювання елементів гри         1. Бали         2. Рівні         3. Квести         4. Нагороди         5. Досягнення         6. Прогрес   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті застосовування серйозних ігор в навчальному процесі. | <https://osvitanova.com.ua/posts/2596-heimifikatsiia-v-osviti>  <https://ain.ua/2017/10/02/yak-igri-zminyuyut-osvitu/>  <https://www.ar25.org/article/top-10-prykladiv-geymifikaciyi-peretvorennya-u-gru-v-osviti-yaki-zminyat-nashe-maybutnye>  <http://blog.gioschool.com/gamification>  <https://www.slideshare.net/samoyira/ss-53922372>  <https://nus.org.ua/view/yak-navchannya-na-osnovi-igor-rozvyva-navychky-21-go-stolittya/> |
|  | Моделювання процесу управління розвитком компетентностей студентів (випускників) | 95 | **Мета**: на основі моделювання бізнес-процесів управління розвитком компетентностей студентів показати шляхи покращення рівня освіти (гнучка система освіти), рівня компетентностей студентів, зменшення ризиків невдач, що можуть виникнути при самоосвіті.  **Об’єкти моделювання:** компанії, роботодавці, вимоги роботодавців (компетенції, що вимагаються), система управління розвитком компетентностей студентів, студент, викладач, компетентності  **Задачі моделювання:**   1. Моделювання процесу визначення вимог роботодавців до компетентностей студентів/випускників:    1. Аналіз ситуації на ринку праці    2. Визначення роботодавцем компетенцій (вимог до вакансій)    3. Аналіз викладачем компетенцій роботодавця    4. Визначення вимог до компетентностей студентів 2. Формування компетентностей у студента відповідно до обраного профілю (професії)    1. Ознайомлення з вимогами до компетентностей студентів    2. Створення навчального плану    3. Виконання навчального плану    4. Сертифікація |  |
|  | Моделювання бізнес-процесів боротьби з коронавірусом | 95 | **Мета**: на основі моделювання бізнес-процесів для боротьби з коронавірусом показати шляхи підвищення її ефективності, зменшення ризиків для населення тощо.  **Об’єкти моделювання:** хворий, лікар, лікарня, ліки та захисні засоби, соціальне середовище (міста, країна), фінанси, бізнес.  **Задачі моделювання:**   1. Створення штабу по боротьбі з епідемією 2. Мобілізація фінансових та медичних ресурсів 3. Закриття кордонів 4. Закупівля або термінове виробництво засобів індивідуального захисту та апаратів ШВЛ 5. Запровадження карантину та обмеження пересування та скупчення громадян 6. Тотальне тестування населення, що знаходиться в зоні ризику 7. Обсервація прибулих із країн із високим рівнем зараження 8. Госпіталізація хворих 9. Контроль карантинних заходів, відстеження переміщення тих, хто заразився корона вірусом 10. Створення карти розповсюдження захворювання та об’єктів ураження 11. Розробка мобільного застосування, який показує присутність можливого носія інфекції поблизу власника мобільного телефону з цим додатком. 12. Реєстрація та маркування населення 13. Заходи для підтримки бізнесу   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації | <https://nv.ua/ukr/world/countries/borotba-z-koronavirusom-v-izrajili-priklad-dlya-vsogo-svitu-novini-50078040.html> |
| 1. п | Моделювання бізнес-процесів функціонування цифрового університету | 90 | **Мета**: на основі моделювання бізнес-процесів програмно-інформаційної платформи цифрового-університету показати шляхи покращення управління архітектурою та організаційною структурою університету, поліпшення системи електронної мережевої взаємодії викладача та студента та електронного документообігу.  **Задачі моделювання**:   1. Електронний кабінет студента. 2. Електронний кабінет викладача. 3. Електронний деканат. 4. Електронний аудиторний, бібліотечний фонд. 5. Онлайн звітність. 6. Електронний документообіг.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної платформи цифрового університету | <http://www.euroosvita.net/index.php/?category=1&id=6361> |
|  | Моделювання бізнес-процесів в системі електронних ресурсів університету | 90 | **Мета**: на основі моделювання бізнес-процесів інформаційної системи «Електронні ресурси університету» показати шляхи покращення якості навчання через використання електронного навчального контенту та е-співпраці між студентами, студентами та викладачами, студентами та адміністрацією.  **Задачі та об’єкти моделювання**:   1. [Бібліотека Університету](http://wiki.kubg.edu.ua/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D0%A3%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82%D1%83): електронні інформаційні ресурси, інформаційні послуги (віртуальна довідка, робота з ресурсами відкритого доступу та консультування по роботі з ними, міжбібліотечний абонемент та електронна доставка документів, виконання різних тематичних запитів на замовлення, перевірка документів на антиплагіат різними сервісами,   консультування з питань використання наукометричних баз даних, зокрема SCOPUS та WEB OF SCIENCE, інформування про нові надходження на електронну пошту та електронна форма подовження терміну користування книгою).   1. [Електронний репозиторій](http://wiki.kubg.edu.ua/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D1%96%D0%B9) університету: моделювання доступу до електронного ресурсу бібліотеки з навчальними, довідковими, методичними матеріалами. 2. [Електронні журнали](http://wiki.kubg.edu.ua/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%96_%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB%D0%B8): моделювання роботи сайтів електронних журналів (реєстрація читачів та авторів статей, подача статей авторами, перегляд і читання журналів та статей, тощо). 3. [Вікі-портал Університету](http://wiki.kubg.edu.ua/%D0%92%D1%96%D0%BA%D1%96-%D0%BF%D0%BE%D1%80%D1%82%D0%B0%D0%BB_%D0%A3%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82%D1%83): моделювання використання вікі-технології для оперативної публікації освітніх матеріалів, написання рефератів та есе, проведення електронного обговорення та дискусій за викладеними матеріалами тощо 4. [Корпоративна електронна пошта](http://wiki.kubg.edu.ua/%D0%9A%D0%BE%D1%80%D0%BF%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BF%D0%BE%D1%88%D1%82%D0%B0): моделювання комунікації студентів і співробітників Університету, автоматичне створення корпоративної електронної скриньки за запитом, моделювання використання текстових, голосових та відео чатів. 5. [Електронне навчання (Е-навчання)](http://wiki.kubg.edu.ua/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%95-%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F)): моделювання застосування змішаного навчання на базі комбінації очного та дистанційного навчання; виконання лабораторних і практичних дистанційних робіт; моделювання автоматичної перевірки виконання лабораторних і практичних завдань, моделювання проведення електронних МКР, іспитів, заліків; використання електронного контенту у вигляді конспектів та презентацій лекцій, навчальних посібників тощо. 6. [Електронне портфоліо викладача](http://wiki.kubg.edu.ua/%D0%95-%D0%BF%D0%BE%D1%80%D1%82%D1%84%D0%BE%D0%BB%D1%96%D0%BE_%D0%B2%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%B0%D1%87%D0%B0) та студента: імітація створення та використання візитних карток або персональних веб-сайтів викладачів та студентів або їх груп, яке автоматично генерується на вікі-порталі університету.   І т.д.  **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної системи «Електронні ресурси університету». | <http://wiki.kubg.edu.ua/%D0%95%D0%9B%D0%95%D0%9A%D0%A2%D0%A0%D0%9E%D0%9D%D0%9D%D0%95_%D0%A1%D0%95%D0%A0%D0%95%D0%94%D0%9E%D0%92%D0%98%D0%A9%D0%95_%D0%A3%D0%9D%D0%86%D0%92%D0%95%D0%A0%D0%A1%D0%98%D0%A2%D0%95%D0%A2%D0%A3> |
|  | Моделювання бізнес-процесів в системі Kyiv Smart City School | 90 | **Мета**: на основі моделювання бізнес-процесів в системі комунікаційна-освітньої платформи Kyiv Smart City School показати шляхи покращення освітнього процесу для школярів, посилити мотивацію школярів до опанування інформаційних та інших смарт технологій, популяризувати ІТ освіту.  **Задачі моделювання**:   1. Навчальний процес (тренінги) програмна імітація навчання дітей:    1. основам програмування у вигляді демонстрації завдань на розвиток логіки, мислення, алгоритмізації та їх розв’язків;    2. основам смарт технологій для користування ресурсами системи Smart City; 2. Заходи, що виховують необхідну для Smart City свідомість (екологічну, інтелектуальну, технічну, технологічну і.т.д.) 3. Заходи з формування soft skills (здатність навчатися, креативність, адаптивність, самокритичність, відповідальність, вміння комунікувати та працювати в команді) 4. Електронна комунікація учнів, батьків і вчителів з освітніх та інших питань. 5. Електронний навчальний контент   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач комунікаційно-освітньої платформи Kyiv Smart City School | <https://www.kyivsmartcity.com/school/>  <https://khreschatyk.news/bezkoshtovna-it-osvita-dlya-ditej-v-kyyevi-vidkryly-shist-filij-kyiv-smart-city-school/>  <https://gb.kyivcity.gov.ua/projects/archive/10/show/1058> |
|  | Моделювання бізнес-процесів в інформаційній системі управління ЖКГ міста | 90 | **Мета**: на основі моделювання бізнес-процесів в інформаційній системі управління житлово-комунальним господарством міста показати шляхи покращення стану ЖКГ, оперативно надавати інформацію щодо послуг у сфері житлово-комунального господарства  **Об’єкти моделювання**:   1. Населення 2. Постачальники послуг 3. Банки для прийому платежів 4. Житлово-експлуатаційні контори (ЖЕКи) 5. Розрахунково-касовий центр 6. Установи влади   **Задачі моделювання:**   1. Населення: укладання договорів про взаємодію населення з постачальниками послуг та оплата населенням вартості послуг. У випадку неякісних послуг або їх відсутності оформлення населенням скарг. Перегляд населенням даних про сплату коштів або заборгованості в Інтернет-ресурсі за кодом споживача послуг та ПІБ. 2. Постачальники послуг: розрахунок вартості послуг (ціна\*кількість або обсяг послуг); облік обсягу послуг (обсяг спожитої електроенергії, води, тепла тощо); перерахунки вартості послуг у випадку їх відсутності, аварійної ситуації, збоїв в роботі постачальника послуг тощо; облік пільг; облік субсидій; розрахунок пені за прострочені платежі; формування звітів за поточний період (місяць, квартал, рік) 3. Банки: прийом платежів відповідно до даних, що є в рахунках населення; друк квитанцій про сплату рахунків; формування щоденного звіту про прийом платежів; передача даних про сплату до БД. 4. ЖЕК: реєстрація населення з визначенням адреси прописки, власника квартири та технічних характеристик квартири (кількість кімнат, метраж, висота стелі, тощо); формування всіх видів довідок; формування звітної документації, інформація про постачальників послуг. 5. Паспортний стіл: прописка та виписка мешканців будинку(листок прибуття, листок вибуття); формування списку мешканців і виборців (діти до 18 років не враховуються у списку виборців); довідка про склад сім'ї, про реєстрацію іноземного громадянина, про вільну житлоплощі і ін довідки, 6. Розрахунково-касовий центр: прийом інформації про нарахування за послуги від постачальників послуг; видача інформації в банки про стан особових рахунків громадян; реєстрація банківських виписок по платіжках населення; прийом готівкових платежів та зарахування їх на особові рахунки громадян; надання інформації громадянам про величину, характер нарахувань за надані послуги та поточному сальдо на їх особових рахунках через спеціалізовані термінали; зв'язок з відділами субсидій, управлінням соціального захисту населення та іншими органами державного контролю; інформація про рух квартиронаймачів.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач управління ЖКГ міста. | <https://i1.poltava.to/uploads/2019/10/2019-10-10/tz-smart-region.pdf>  <http://search.ligazakon.ua/l_doc2.nsf/link1/JH82S00A.html>  <http://ua-referat.com/%D0%92%D0%BF%D1%80%D0%BE%D0%B2%D0%B0%D0%B4%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%85_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC_%D1%83_%D1%81%D1%84%D0%B5%D1%80%D1%96_%D0%B6%D0%B8%D1%82%D0%BB%D0%BE%D0%B2%D0%BE-%D0%BA%D0%BE%D0%BC%D1%83%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B3%D0%BE%D1%81%D0%BF%D0%BE%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B0> |
|  | Моделювання бізнес-процесів в інформаційні системі управління дорожнім рухом | 90 | **Мета**: на основі моделювання бізнес-процесів в системі управління дорожнім рухом показати шляхи зниження рівня завантаженості вулиць і доріг, Підвищення безпеки дорожнього руху.  **Об’єкти моделювання:** транспортний засіб, номерний знак, детектори руху транспорту, камери відеонагляду, GPS-датчики таксі та громадського транспорту, карта міста, оперативні дані щодо ДТП, ремонтів та перекриття вулиць і доріг.  **Задачі моделювання**:   1. Підрахунок інтенсивності трафіку на вузлах та перегонах; 2. Прогнозування інтенсивності на автомобільних дорогах; 3. Проектування «Зеленої хвилі» з оптимізацією алгоритмів керування світлофорними об’єктами; 4. Оптимізація схем організації дорожнього руху: 5. Автоматична фіксація порушень Правил дорожнього руху; 6. Надання пріоритету руху громадського транспорту; 7. Моніторинг умов руху в режимі реального часу; 8. Інформування учасників руху про дорожні умови; 9. Моделювання паркінгів та автомобільних стоянок, інформування щодо наявності вільних паркувальних місць; 10. Інформування щодо графіків руху громадського транспорту; 11. Інформування щодо прокату велосипедів тощо.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач управління дорожнім рухом | <https://apluss.pro/upravlinnya-transportnoyu-systemoyu/intelektualni-transportni-sistemi-its/> |
|  | Моделювання бізнес-процесів в інформаційній системі комунікації влади з громадянами «Smart-портал» | 90 | **Мета**: на основі моделювання бізнес-процесів в інформаційній системі комунікації з громадянами «Smart-портал» показати шляхи створення єдиного інформаційного простору для забезпечення оперативного доступу мешканців міста до інформації про діяльність керівництва ради, її виконавчого комітету, депутатів для вирішення практичних питань громадян.  **Об’єкти моделювання:** громадяни, рада, депутати, інформаційний простір (інформація).  **Задачі моделювання**:   1. Електронний кабінет мешканця міста:    1. Створення персонального онлайн-кабінету мешканця міста;    2. Підключення зовнішніх ресурсів надавачів послуг (сайти, портали, бази даних, електронні карти та ін.);    3. Організація прямого інформаційного обміну між «кабінетами» мешканця міста та надавачів послуг;    4. Розміщення шаблонів та зразків документів, необхідних для отримання послуги;    5. Отримання інформації щодо отримання послуги    6. формування статистичних та аналітичних звітів. 2. Робота контакт-центру (call-center):    1. Фіксація звернення громадян, формування картки звернення (ПІБ, адреса, телефон, дата, час звернення, зміст звернення)    2. Автоматичне визначення виконавця та повідомлення автору звернення;    3. Електронна маршрутизація звернення від замовника до виконавця з повідомленням громадянам про передачу замовлення виконавцю;    4. Контроль за графіком виконанням звернення;    5. Формування звіту щодо виконання звернень;    6. Кабінет замовника звернення зі звітністю про виконання замовлення. 3. Виконання адміністративних послуг: 4. Електронні петиції: 5. Електронні черги:    1. В дитячий садок, до школи    2. В державну службу зайнятості    3. Пенсійний фонд    4. Митниця 6. Онлайн контроль виконання передвиборчих програм з розміщенням на ГІС-карті інформації про діяльність керівництва ради та депутатів: 7. оприлюднення проектів та прийнятих рішень; 8. оприлюднення підсумків поіменного голосування депутатів за різними питаннями; 9. інформація щодо депутатів облради: 10. оприлюднення нормативної документації (тексти законів, указів, розпоряджень тощо); 11. виконання онлайн-трансляції засідань ради;   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач «Smart-порталу» | <https://bissoft.org/smart-portal>  <https://bissoft.org/smart-city> |
|  | Моделювання бізнес-процесів в інформаційній системі «Smart Manager» | 90 | **Мета**: на основі моделювання бізнес-процесів в інформаційній системі «Smart Region» покращити доступ користувачів до інформації щодо проектів, які плануються або реалізуються в регіоні для підтримки або відхилення їх.  **Об’єкти моделювання:** менеджер, проекти, робочі завдання, бізнес-процеси завдань, документи.  **Задачі моделювання**:  Smart Manager являє собою персональний робочий стіл, де співробітник бачить всі поставлені перед ним робочі завдання, їх строки виконання, ступінь важливості та може відслідковувати бізнес-процеси, в яких він бере участь. Це може бути домовленість/затвердження документів, делегування доручень, домовленість про оплату, аналіз заборгованості і т. д.  **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної системи «Smart Manager» | <https://www.it.ua/news/kievgorstroj-ispolzuet-novuju-razrabotku-erp-sistemy-it-enterprise-smart-manager> |
|  | Інформаційна платформа для реалізації стартап-проектів | 90 | **Мета**: на основі моделювання бізнес-процесів в інформаційній платформі для реалізації стартап-проектів показати шляхи покращення процесу пошуку інвесторів для реалізації ідей і проектів.  **Об’єкти моделювання:** автор, ідея або проект, бізнес-модель Canvas, команда проекту.  **Задачі моделювання**:   1. Створення команди. 2. Генерування та аналіз доцільності ідеї проекту. 3. Побудова бізнес-моделі стартап-проекту. 4. Розробка бізнес-плану стартап-проекту 5. Презентація ідеї та проекту   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної платформи для реалізації стартап-проектів. | <https://moy-gorod.org/>  <https://ela.kpi.ua/bitstream/123456789/29447/1/Rozrobka_startup-proektiv_Konsp.lekts.pdf>  <http://gentledevs.com/startup-solutions/> |
|  | Моделювання бізнес-процесів розвитку кар'єри випускника в ІТ індустрії | 90 | **Мета**: на основі моделювання бізнес-процесів розвитку кар'єри випускника показати шляхи гарантування конкурентоздатності ІТ випускників на ринку праці через управління розвитком їх компетентностей.  **Об’єкти моделювання:** ІТ компанія, ІТ-фахівці, компетентності випускника, ІТ професія, рівень кваліфікації, службові сходи кар’єри (професійна роль, статус і види діяльності)  **Задачі моделювання**:   1. Розвиток ІТ-освіти та формування базових компетентностей випускників відповідно до освітніх програм та вимог ІТ індустрії (рівень junior); 2. Працевлаштування в ІТ компанію (складання і розсилка резюме, проходження співбесід, тестування, випробувальний період робота); 3. Виконання ІТ-проектів в компанії; 4. Оцінка ефективності, оцінка знань, зовнішня сертифікація; 5. Підвищення на посаді   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в процесі моделювання ІТ кар’єри. | <https://www.softserveinc.com/uk-ua/careers/professional-development>  [CWA 16458-1:2018](ftp://ftp.cencenelec.eu/CEN/WhatWeDo/Fields/ICT/eEducation/WS/eSkills/ICTSkills/CWA%2016458-1_2018.pdf) European ICT Professional Role Profiles – Part 1: 30 ICT Profiles - <https://itprofessionalism.org/about-it-professionalism/competences/where-to-buy-the-e-cf-standard/>  <https://www.cepisecompetencebenchmark.org/> |
|  | Моделювання бізнес-процесів створення бібліотек освітнього контенту з  персональними треками для ІТ-спеціальностей | 90 | **Мета**: на основі моделювання бізнес-процесів створення бібліотек освітнього контенту показати шляхи покращення якості підготовки студентів.  **Об’єкти моделювання:** студент, викладач, індивідуальний навчальний план, робоча програма дисципліни, навчально-методичні матеріали  **Задачі моделювання**:   1. Визначення платформи для освітнього контенту 2. Визначення мотивацій студентів та їх індивідуальних навчальних планів 3. Створення інформаційних ресурсів за індивідуальним навчальним планом    1. Теоретичний курс    2. Практичні та лабораторні заняття    3. Тести    4. Бізнес-кейси 4. Агрегація ресурсів за потребами студентів 5. Електронний каталог 6. Онлайн -магазин   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в процесі моделювання освітнього контенту. | <http://www.ng.sibstrin.ru/brest_novosibirsk/2017/prez/%D0%90%D0%BD%D0%B4%D1%80%D1%8E%D1%88%D0%B8%D0%BD%D0%B0.pdf>  <https://texterra.ru/blog/obrazovatelnyy-kontent-mekhanizm-sozdaniya-loyalnogo-polzovatelya.html>  <http://files.runet-id.com/2015/csf/presentations/12feb.csf15-1-3--alekseeva.pdf>  <http://www.curator.ru/e-learning/publications/publication29.html> |
|  | Моделювання бізнес-процесів організації освітнього процесу в університеті | 90 | **Мета**: на основі моделювання бізнес-процесів в організації освітнього процесу показати шляхи покращення якості підготовки студентів.  **Об’єкти моделювання:** студенти, викладачі, освітній процес, види навчальних занять, результати навчання  **Задачі моделювання**   1. Форми організації освітнього процесу, 2. Види навчальних занять і контролю 3. Планування освітнього процесу 4. Науково-методичне забезпечення освітнього процесу 5. Оцінювання результатів навчання 6. Присвоєння кваліфікації випускникам 7. Здобувачі освіти 8. Науково-педагогічні та педагогічні працівники 9. Академічна мобільність 10. Забезпечення якості освітнього процесу   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання освітнього процесу | <http://www.univ.kiev.ua/pdfs/official/Organization-of-the-educational-process.pdf>  <http://mobility.univ.kiev.ua/?page_id=804&lang=uk>  <http://nmc.univ.kiev.ua/docs/Polojennya%20QAS%202019.pdf>  <https://mobility.univ.kiev.ua/?page_id=798&lang=uk> |
|  | Моделювання бізнес-процесів інформаційної системи для Центру сприяння працевлаштуванню випускників | 90 | **Мета**: на основі моделювання бізнес-процесів Центру сприяння працевлаштуванню випускників показати шляхи удосконалення працевлаштування студентів в ІТ-компанії.  **Об’єкти моделювання:** кадрове агенство, студенти, ІТ-компанії, договори, вимоги ІТ-компаній (вакансії)  **Задачі моделювання**   1. Організація проведення презентацій роботодавців і ярмарків вакансій 2. Ведення бази даних потенційних роботодавців та вакансій на об'єктах ринку праці 3. Ведення бази даних випускників та здобувачів на працевлаштування, облік їх можливостей 4. Пошук і підбір претендентів на вакантні місця роботи 5. Укладення договорів з підприємствами на підбір і розподіл молодих фахівців 6. Аналіз вимог роботодавців до випускників вузу 7. Аналіз попиту фахівців на ринку праці 8. Аналіз конкурентоспроможності випускників ВНЗ 9. Ведення обліку працевлаштування молодих фахівців вузу 10. Формування різних статистичних звітів по випуску та працевлаштування молодих фахівців 11. Моніторинг працевлавштування та кар'єри претендента   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання роботи Центру сприяння працевлаштуванню випускників | http://www.work.ua/, http://job.ukr.net/, <http://www.trud.ua/>, <http://www.jobs.ua/>, <http://rabota.ua/>  <https://tusur.ru/ru/o-tusure/struktura-i-organy-upravleniya/departament-obrazovaniya/tsentr-sodeystviya-trudoustroystvu-vypusknikov>  <https://aist.tusur.ru/srcstv> |
|  | Моделювання бізнес-процесів системи дистанційної освіти (СДО) в університеті | 90 | **Мета**: на основі моделювання бізнес-процесів СДО забезпечити досту до освітніх ресурсів шляхом використання сучасних інформаційних технологій та телекомунікаційних мереж і формування єдиного освітнього простору в рамках усього світового співтовариства.  **Об’єкти моделювання**, СДО, студенти, освітні ресурси, дистанційний курс, тьютор  **Задачі моделювання:**   1. Основні види навчальних занять за дистанційною формою навчання є: лекція, семінар, урок, практичні заняття, лабораторні заняття, консультації та інші; 2. Веб-ресурси навчальних дисциплін (програм), що необхідні для забезпечення дистанційного навчання, можуть містити: 3. методичні рекомендації щодо їх використання, послідовності виконання завдань, особливостей контролю тощо; 4. документи планування навчального процесу (навчальні програми, навчально-тематичні плани, розклади занять); 5. відео- та аудіозаписи лекцій, семінарів тощо; 6. мультимедійні лекційні матеріали; 7. термінологічні словники; 8. практичні завдання із методичними рекомендаціями щодо їх виконання; 9. віртуальні лабораторні роботи із методичними рекомендаціями щодо їх виконання; 10. віртуальні тренажери із методичними рекомендаціями щодо їх використання; 11. пакети тестових завдань для проведення контрольних заходів, тестування із автоматизованою перевіркою результатів, тестування із перевіркою викладачем; 12. ділові ігри із методичними рекомендаціями щодо їх використання; 13. електронні бібліотеки чи посилання на них; 14. бібліографії; 15. дистанційний курс, що об’єднує зазначені вище веб-ресурси навчальної дисципліни (програми) єдиним педагогічним сценарієм; 16. інші ресурси навчального призначення. 17. Отримання навчальних матеріалів, спілкування між суб’єктами дистанційного навчання під час навчальних занять, що проводяться дистанційно, забезпечується передачею відео-, аудіо-, графічної та текстової інформації у синхронному або асинхронному режимі. 18. Практичне заняття, яке передбачає виконання практичних (контрольних) робіт, відбувається дистанційно в асинхронному режимі. 19. Лабораторне заняття проводиться дистанційно з використанням відповідних віртуальних тренажерів і лабораторій. 20. Ділові ігри, виконання проектів у групах тощо проводяться дистанційно у синхронному або асинхронному режимі. 21. Контрольні заходи з навчальної дисципліни при здійсненні підготовки фахівців за дистанційною формою навчання включають проміжний (тематичний, модульний), підсумковий та інші контролі знань, умінь та навичок, набутих студентом у процесі навчання. 22. Усі контрольні заходи у можуть здійснюватись у вигляді відеоконференц-зв’язку за умови забезпечення аутентифікації того, хто навчається,   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті моделювання роботи СДО | <https://zakon.rada.gov.ua/laws/show/z0703-13>  <http://vnz.org.ua/dystantsijna-osvita/pro>  <https://www.kname.edu.ua/images/Files/Normativny_Dokumenty/polozhennja_pro_organisaciyu_distanc_navchannya.pdf> |
|  | Моделювання процесу розробки серйозної гри для навчання мовам програмування | 90 | **Мета**: на основі моделювання бізнес-процесів при застосовування серйозних ігор в освітньому процесі ЗВО показати шляхи удосконалення освітнього процесу, покращення зацікавленість та вмотивованість студентів до набуття знань та умінь.  **Об’єкти моделювання:** студенти, викладачі, серйозна гра (платформа, завдання), результати навчання, дисципліна,  **Задачі моделювання**   * + 1. Вступна частина (мета ігрового заняття).     2. Підготовча частина (правила гри).     3. Гра (постановка завдань, сáме гра).     4. Аналіз результатів, підведення підсумків.     5. Моделювання сценаріїв та інструментів в іграх Kodable, [Code Monkey](https://pedpresa.ua/goto/https:/www.playcodemonkey.com/), [Vim Adventures](https://pedpresa.ua/goto/https:/vim-adventures.com/), [Lightbot](https://pedpresa.ua/goto/http:/lightbot.com/) пр.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті застосовування серйозних ігор при вивченні мов програмування | <https://mistosite.org.ua/ru/articles/yak-stvoryty-hru-ohliad-5-etapiv-na-prykladi-hry-dlia-uchasti-munitsypalitetu-helsynky>  <https://pedpresa.ua/196111-top-10-igor-z-yakymy-dytyna-navchytsya-programuvaty.html>  приклади ігор  <https://code.org/>  <https://vim-adventures.com/>  <https://lightbot.com/>  <https://scratch.mit.edu/>  <https://codecombat.com/>  <http://www.ceebot.com/ceebot/index-e.php> |
|  | Прогнозування ринку праці  в ІТ галузі | 90 | **Мета**: на основі моделювання бізнес-процесів при прогнозуванні потреби ринку праці  в ІТ фахівцях показати шляхи розвитку ІТ-галузі  **Об’єкти моделювання:** ринок праці, компанії, фахівці, вакансії, професії  **Задачі моделювання (**див. методику формування середньострокового прогнозу потреби у фахівцях та робітничих кадрах на ринку праці):   1. Розрахунок продуктивності праці 2. Розрахунок прогнозного індексу продуктивності праці за видами економічної діяльності 3. Розрахунок прогнозного індексу значення ВДВ (валова додана вартість) 4. Розрахунок прогнозного індексу кількості зайнятих 5. Розрахунок часток зайнятих за професіями 6. Розрахунок прогнозної додаткової потреби у фахівцях 7. Зіставлення отриманих значень прогнозної додаткової потреби у фахівцях та робітничих кадрах на ринку праці j-го виду економічної діяльності за i-ю професією з пропозиціями центральних та місцевих органів виконавчої влади   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті Прогнозування потреби в ІТ фахівцях на ринку праці | <https://zakon.rada.gov.ua/rada/show/v0305731-13/sp:max100>  <https://www.sworld.com.ua/simpoz6/63.pdf>  <http://kiis.com.ua/materials/pr/20151230_PROON/Report_Employment_KIIS_final_ukr.pdf>  <http://fsb.net.ua/files/book_46682_01.pdf>  <https://phm.cuspu.edu.ua/ojs/index.php/SNYS/article/download/1562/pdf>  http://www.economy.nayka.com.ua/?op=1&z=679 |
|  | Визначення обсягу  підготовки фахівців з вищою освітою | 90 | **Мета**: на основі моделювання бізнес-процесів при прогнозуванні обсягу підготовки фахівців з вищою освітою показати шляхи оптимізації державного замовлення на фахівців  **Об’єкти моделювання:** міністерство економіки , міністерство освіти, університети, центр моніторингу і прогнозування кадрів, компетентності випускників  **Задачі моделювання:**   1. Моделювання тенденцій розвитку економіки регіону 2. Прогноз ВДВ (валова додана вартість) та чисельності зайнятого населення на 3-5 років та на цій основі здійснення розрахунку та прогнозу потреби в кваліфікованих кадрах. 3. Визначення потреби в кваліфікованих кадрах за рівнями професійної освіти методом вибіркового обстеження керівників підприємств. 4. Розрахунок коефіцієнтів насиченості кваліфікованими кадрами за рівнями освіти, професійними групами за видами економічної діяльності. 5. Визначення загальної додаткової потреби у кваліфікованих кадрах за рівнями освіти, професійними групами та видами економічної діяльності.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті прогнозування обсягу підготовки фахівців з вищою освітою. | <http://www.economy.nayka.com.ua/?op=1&z=679>  <http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Npndfi_2008_4_15.pdf>  <https://cyberleninka.ru/article/n/metodika-prognozirovaniya-sprosa-na-spetsialistov-v-sisteme-vysshego-professionalnogo-obrazovaniya-regiona>  <http://iubpe.sfu-kras.ru/assets/content/files/1453360144_Kadrovaya_potrebnost_2011.pdf> !!! |
|  | Моделювання процесу працевлаштування випускників в ІТ-компанію | 90 | **Мета**: на основі моделювання бізнес-процесів працевлаштування випускників показати шляхи удосконалення проходження студентами співбесід, врахування наявних у студента знань та навичок  **Об’єкти моделювання:** університети, компанії, вакансії (вимоги роботодавців), випускники, компетентності випускників,  **Задачі моделювання:**   1. проаналізувати процес отримання і засвоєння знань студентами; 2. розглянути процес формування тестових завдань на підприємствах, проаналізувати їх; 3. сформувати на основі опитування студентів параметри привабливості підприємства, виходячи з його параметрів; 4. імітація проведення співбесід;   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації працевлаштування випускників. | <http://elib.hduht.edu.ua/bitstream/123456789/1157/1/-2015%20%D0%94%D0%B0%D0%B2%D1%8B%D0%B4%D0%BE%D0%B2%D0%B0%20%D0%BF%D0%BE%D1%81%D0%B8%D0%B1%D0%BD%D0%B8%D0%BA%20%D1%87%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA.pdf> |
|  | Smart поліція | 90 | **Мета:** на основі моделювання бізнес-процесів роботи Smart поліції показати шляхи удосконалення її діяльності, системи підтримки операційної діяльності поліції в smart city/  **Об’єкти моделювання**  **Задачі моделювання:**   * + - * 1. Підтримка операційної діяльності поліції:   системи електронних досьє на злочинців і злочини, включаючи єдині каталоги і системи класифікації;  системи розпізнавання обличчя злочинців, номерів машин  системи реєстрації, класифікації і розпізнавання крадених речей   * + - * 1. Інформаційно-телекомунікаційна система для збору, консолідації, обробки, зберігання, диспетчеризації даних, що потрібні:   патрульним службам;  службам розслідування злочинів і кримінологічним експертам;  Оперативним поліцейським  Поліцейським аналітикам тощо   * + - * 1. Соціальна динаміка:   Визначати незвичайні скупчення людей в заданих районах або автомобілів за даними мобільних операторів або Wi-Fi;  моніторинг записів у соціальних мережах з гео-тегами (check-in),  аналіз змісту записів для визначення потрібного контенту і прив'язка його до географічного місця   * + - * 1. Data mining   виявлення зв'язків у кримінальному середовищі на основі аналізу поліцейських досьє  Виявлення кібер-злочинів  **Результат:** : ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності Smart поліції. | <https://cyberleninka.ru/article/n/umnaya-politsiya-v-umnom-gorode>  <https://library.ku.ac.ke/wp-content/downloads/2011/08/Bookboon/IT,Programming%20and%20Web/policing-cyber-crime.pdf> |
|  | Моделювання бізнес-процесу автоматизації роботи scrum master (проектного менеджера) | 90 | **Мета** розробки проекту є підвищення ефективності та швидкості роботи проектного менеджера, шляхом надання інструментів управління та слідкування за проектом.  **Об’єкти моделювання:** менеджер проекту, команда проекту, проект, задачі проекту замовник, програмна система управління проектом, sprint проекту  **Задачі моделювання:**   1. Провести ініціалізацію    1. Отримати завдання та терміни від замовника    2. Провести роботу з вимогами    3. Скласти характеристику об'єкту автоматизації 2. Виконати планування    1. Скласти беклог проекту    2. Провести набір команд    3. Провести аналіз часових і трудових ресурсів    4. Призначити часові терміни    5. Провести аналіз і розбити великі завдання на менші    6. Призначити складності і пріоритетності завданням 3. Проводити виконання    1. Виконати постановку і планування ітерації    2. Скласти беклог ітерації    3. Призначити команди на ітерацію    4. Розподілити завдання між членами команд 4. Проводити моніторинг і контроль    1. Провести контроль виконання і аналіз ризиків    2. Провести огляд якості    3. Провести огляд та аналіз закінчених ітерацій    4. Слідкувати за навантаженнями    5. Перерозподіляти завдання    6. Призначати збори    7. Вести звітність    8. Тримати зв'язок з замовником    9. Провести демонстрацію звітності та готових рішень замовнику    10. Провести роботу з вимогами 5. Виконати завершення    1. Провести закриття проекту    2. Передати документацію замовнику    3. Передати готовий продукт замовнику   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності менеджера проекту |  |
|  | Моделювання бізнес-процесів в системі Smart-education в ІТ галузі | 85 | **Мета**: на основі моделювання бізнес-процесів Smart-education показати шляхи підвищення якості освіти, покращення рівня звань, умінь, навичок та досвіду, забезпечення вимог ІТ індустрії шляхом  **Задачі моделювання**:   1. Створення освітнього контенту, що модернізується в реальному часі і знаходиться у вільному доступі для студентів в будь-який час 2. Активні методи навчання 3. Індивідуальні освітні траєкторії навчання з урахуванням мотивацій, здатностей, бажань, професійної орієнтації студента 4. Використання розумних пристроїв (смартфони, розумні дошки, розумні комп’ютери тощо) 5. Розподілене освітнє середовище, академічна мобільність, організація самостійної пізнавальної, дослідницької, проектної діяльності студентів. 6. Взаємодія студентів і викладачів з ІТ компаніями 7. Електронний контроль знань студентів та їх відвідуваності занять   **Результат:** ОО програма, яка в текстовому (консольному) режимі інформує про процеси, що відбуватимуться при вирішенні задач Smart- education | <https://www.slideshare.net/PROelearning/smart-education>  <http://smartmesi.blogspot.com/2012/02/smart-education_16.html> |
|  | Моделювання бізнес-процесів в системі Smart-house | 85 | **Мета**: на основі моделювання бізнес-процесів Smart house показати шляхи підвищення якості управління домашньою автоматикою, зменшення оплати комунальних платежів за рахунок економії ресурсів (вода, опалення, електрика тощо)  **Задачі моделювання**:   1. Захист від витоку води і газу; 2. Протипожежна сигналізація; 3. Повідомлення про забутих відкритих дверях і вікнах 4. Контроль температури повітря, теплої підлоги, системами вентиляції і кондиціонування; 5. Управління освітленням; 6. Розподіл аудіо- і відеосигналу в будь-якій частині будинку; 7. Відео-трансляція з камер спостереження або домофона   **Результат:** ОО програма, яка в текстовому (консольному) режимі інформує про процеси, що відбуватимуться при вирішенні задач Smart- house | <https://www.everest.ua/ru/ai-platform-2/smart-city/tekhnologiya-umnogo-doma-kak-ai-sozdayet-prostranstvo-komfortnoye-dlya-zhizni/>  <https://lvivska-ploscha.com.ua/ru/news1/tekhnolohyia-riel-smart-home-v-zhylom-komplekse-l-vovskaia-ploshchad>  <https://iotconf.ru/ru/article/ustroystva-dlya-smart-house-v-2019-godu-bolee-umnie-bolee-ekonomnie-97174> |
|  | Моделювання бізнес-процесів в системі Smart-медицина | 85 | **Мета**: на основі моделювання бізнес-процесів інформаційної системи Smart-медицина показати шляхи покращення управління медичними закладами, зменшити витрати хворого на лікування, покращити якість та ефективність лікування  **Задачі моделювання**:  1.Ведення електронної карти хворого  2. Електронний запис на прийом (вибір лікаря за профілем хвороби, запис на прийом, автоматичне нагадування про візит до лікаря)  3. Аналітична система уточнення та постановки діагнозу  4. Доступ до історії хвороби пацієнта  5. Контроль призначення ліків, вибір оптимального лікування за критеріями ціна-якість  6. Електронні протоколи огляду хворих, рекомендаційна система визначення протоколу лікування  І т.д.  **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач інформаційної системи Smart-медицина. | <https://www.livemedical.ru/tool/3110/>  <https://24tv.ua/elektronnaya_meditsina_v_rezhime_smart_city_n1103761>  <https://profit.kz/news/42725/V-Astane-realizuetsya-proekt-Smart-medicina/> |
|  | Моделювання бізнес-процесів в системі автоматичної оплати проїзду в комунальному транспорті | 85 | **Мета**: на основі моделювання бізнес-процесів в системі автоматичної оплати проїзду автоматичного підрахунку та обліку пасажирів в громадському транспорті показати шляхи покращення контролю пасажиропотоку, обліку та регулювання кількості пасажирів в громадському транспорті та оплати проїзду.  **Об’єкти моделювання:** транспортний засіб, номерний знак, дорога та зупинки,засобиGPS-моніторингу, датчики підрахунку кількості пасажирів, валідатори контролю оплати, електронний квиток,.  **Задачі моделювання**:   1. Проведення розрахунку за транспортні послуги у салоні будь-якого транспортного засобу (автобус, тролейбус тощо. ); 2. Друк одноразового проїзного документа, вигляд якого може налаштовуватись відповідно до правил та норм чинного законодавства; 3. Фіксацію наданих транспортних послуг у пам’яті валідатора та передачу цих даних у БД; 4. Друк звітності під час контролю на лінії та при закритті зміни, з виведенням інформації про маршрути, кількість наданих транспортних послуг різного типу та суму готівкової виручки тощо; 5. Реєстрацію усіх дій (відкриття/закриття зміни, зміна маршруту, контроль на лінії тощо.) за допомогою службових карток (кондукторів, касирів, служби контролю тощо);   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач автоматичної оплати проїзду | <https://gpsm.ua/ua/blog/smart-city-umnij-gorod/asop-avtomaticheskaja-sistema-oplati-proezda/>  <http://szvk.te.ua/modules/payment-system/>  <https://khm.gov.ua/uk/content/pro-vprovadzhennya-avtomatizovanoyi-sistemi-obliku-oplati-proyizdu-v-miskomu-pasazhirskomu>  <https://www.kremen.gov.ua/assets/uploads/files/6c572a65ad6e9d4a8594e7db246cf9351fd92c33poryadok_funktsionuvannya_ta_vymogy_do_avtomatyzovanoi_systemy_obliku_oplaty_proizdu.doc> |
|  | Моделювання бізнес-процесу академічної мобільності студентів університету | 85 | **Мета**: на основі моделювання бізнес-процесів академічної мобільності студентів показати шляхи підвищення мотивації студентів до навчання.  **Об’єкти моделювання:** студент, програма академічної мобільності в університеті-партнера, індивідуальний навчальний план у базовому університеті, мотиваційні фактори, договір на академічну мобільність, відділ академічної мобільності  **Задачі моделювання:**   1. Підписання договорів ЗВО з університетами -партнерами 2. Відбір учасників академічної мобільності 3. Підготовка документів на конкурс академічної мобільності 4. Моделювання проходження академічної мобільності:    1. ступенева мобільність — навчання у вищому навчальному закладі – партнера (подвійний диплом)    2. кредитна мобільність —- навчання у вищому навчальному закладі –партнері з метою здобуття кредитів 5. Визнання результатів академічної мобільності 6. Звіт за результатами програми академічної мобільності   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті академічної мобільності студентів | <https://mobility.univ.kiev.ua/?page_id=798&lang=uk>  <http://mobility.univ.kiev.ua/?page_id=804&lang=uk> |
|  | Моделювання процесів функціонування розумного автомобіля | 80 | **Мета**: на основі моделювання бізнес-процесів розумного автомобіля показати шляхи покращення та спрощення управління авто, зменшити витрати на його утримання, зекономити витрати пального, покращити безпеку водія та пасажирів тощо  **Задачі моделювання**:  1.Розуміння емоцій водія під час керування автомобілем  2. Виконання голосових команд управління автомобілем (безключовий доступ, дистанційний запуск двигуна, система Start / Stop, клімат-контроль з іонізатором, круїз-контроль, камера заднього виду, включення та пошук улюбленої музики тощо)  3. Керування автомобілем в режимі автопілоту (контроль маршруту, напрямку, відстані, наближення до інших рухомих та нерухомих об’єктів, збільшення та зменшення швидкості тощо)  4. Пасивні та активні системи безпеки і допомоги водію (ремені та подушки безпеки, які автоматично спрацьовують у разі аварій, адаптивна система попередження аварій з автоматичним управління рулем та іншими пристроями)  5. Автоматичні помічники водія (утримання автомобіля на полосі, безпечне паркування, моніторинг сліпих зон, інформаційне попередження водія тощо)  6 GPS контроль, захист авто під час угону тощо  **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач розумного автомобіля | <https://hi-news.ru/auto/avtomobili-nauchat-ponimat-emocii-voditelej.html>  <https://kit-e.ru/articles/sensor/2012_1_7.php>  <https://rg.ru/2019/07/31/eshche-odin-intellektualnyj-krossover-dlia-rossii-deshevle-milliona-rublej.html> |
|  | Моделювання бізнес-процесів при автоматизації роботи диспетчерської служби таксопарку. | 80 | **Мета**: на основі моделювання бізнес-процесів в системі диспетчерської служби таксопарку показати шляхи покращення обслуговування пасажирів, рівномірного розподілу навантаження на автомобілі і водіїв, зменшення холостого пробігу автомобілів.  **Об’єкти моделювання:** диспетчер, водій, автомобіль, клієнт, АТП.  **Задачі моделювання**:   1. Облік та виконання замовлень клієнтів: 2. Швидкий прийом замовлень:    * за номером телефону (звичайному або мобільному);    * за адресою проживання;    * за місцем перебування пасажира; 3. Постійний контроль диспетчером всіх етапів виконання замовлень:    * прийом виклику,    * подача машини,    * посадка, "з пасажиром",    * повернення машини на стоянку; 4. Облік кількості замовлень зроблених:    * з підприємства;    * конкретним пасажиром (в тому числі з мобільного телефону); 5. Ведення обліку VIP і "ненадійних" пасажирів (білий і чорний списки); 6. Контроль коректності адреси і номера телефону по актуальній базі міста (яка може оновлюватися і поповнюватися за допомогою Internet або вручну); 7. Формування звітів, пов'язаних з урахуванням замовлень. 8. Облік роботи водіїв: 9. контроль місця перебування машини в робочий час (на базі, з пасажиром, стоянка / черга, обід, за містом та інш.); 10. облік кількості поїздок кожним водієм за зміну і за довільний період; 11. ведення бази власників машин, водіїв, бригад, графіків їх роботи. 12. Облік роботи диспетчерів: 13. диспетчер завжди бачить навантаження водія на зміні (кількість виконаних замовлень), і тривалість зміни персонально для кожного водія; 14. Розподіл робіт в бригаді диспетчерів:     * по етапах замовлення - один приймає замовлення, інший працює з водіями;     * по замовленнями - диспетчер виконує замовлення від початку і до кінця; 15. Автоматично ведеться черга на стоянках; 16. Автоматично ведеться швидкий пошук адреси при її "ручному" введенні при відсутності номера телефону у пасажира або в базі; 17. Автоматично ведеться журнал робіт виконаних диспетчером за зміну або довільний період з деталізацією по всіх елементах замовлення: виклик, подача, посадка, повернення на стоянку; 18. Ведення бази диспетчерів, бригад, графіків їх роботи.   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач автоматизації роботи диспетчерської служби таксі. | <https://ssb.com.ua/index.php?option=com_content&task=view&id=23&Itemid=37> |
|  | Моделювання бізнес-процесів сайту Інтернет-магазину | 80 | **Мета**: на основі моделювання бізнес-процесів  сайту Інтернет-магазину показати можливість покращити його роботу та отримати додатковий прибуток  **Об’єкти моделювання:** Інтернет-магазин, товар, корзина, клієнт, замовлення, менеджер (адміністратор)  **Задачі моделювання:**   * + 1. Дії клієнта:        1. Зареєструватися на сайті        2. Вибрати товар в корзину        3. Визначити вартість корзини        4. Оформити замовлення на доставку        5. Оплатити товар     2. Дії магазину:        1. Забезпечити товар        2. Зареєструвати замовлення        3. Скомпонувати замовлення        4. Здійснити доставку     3. Дії адміністратора (менеджера)        1. Забезпечити роботу сайту        2. Відкривати та закривати вікно замовлення (у разі пікового навантаження)        3. Узгоджувати заміну товару з клієнтом у разі відсутності товару        4. Вести облік оплати        5. Вести облік замовлень та їх виконання   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті роботи сайту Інтернет-магазину | <https://rozetka.com.ua/?gclid=Cj0KCQjw1Iv0BRDaARIsAGTWD1u28tIpXGkRnkksN3n_7NOFKhtufFieGt-is_PbafpOAg7fqHri1jwaAq9YEALw_wcB>  <https://megamarket.zakaz.ua/ru/> |
|  | Комп’ютерний моніторинг і прогноз успішності студентів та контроль відвідування занять | 75 | **Мета**: на основі моделювання бізнес-процесів в системі моніторингу успішності та відвідуваності занять показати шляхи покращення оперативності обліку успішності та відвідуваності занять студентів.  **Об’єкти моделювання:** студенти, викладачі, дисципліни, журнал обліку успішності студентів.  **Задачі моделювання**:   1. Облік присутності студентів на лекціях та лабораторних заняттях 2. Облік виконаних лабораторних, практичних та домашніх завдань 3. Нарахування балів за виконані роботи та активність на лекціях 4. Нарахування штрафів за порушення Deadline 5. Нарахування балів за індивідуальні творчі, ініціативні роботи, участь у конференціях, конкурсах, олімпіадах тощо 6. Підсумкові відомості перед іспитом 7. Прогноз результатів іспиту за дисципліною 8. Підсумкові відомості після іспиту   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моніторингу успішності та обліку відвідування занять студентами | <http://usu.kz/uchet_poseschaemosti_studentov.php>  <https://intelligencloud.com/ru/baseCapabilities-attendanceControl>  <https://cyberleninka.ru/article/n/monitoring-uspevaemosti-studentov-kak-odno-iz-napravleniy-otsenki-kachestva-sistemy-upravleniya-v-vuze>  <https://mgimo.ru/upload/iblock/e09/avtomatizaciya-provedeniya-monitoringa-uspevaemosti-studentov-v-vuzah.pdf>  <http://elar.uspu.ru/bitstream/uspu/5634/1/03Strelkova2.pdf> |
|  | Моделювання процесу підготовки та захисту  бакалаврського дипломного проекту | 75 | **Мета**: на основі моделювання бізнес-процесів при підготовці та захисту дипломного проекту (ДП) показати шляхи удосконалення процесу дипломного проектування та покращення результатів захисту дипломних проектів студентів.  **Об’єкти моделювання:** студенти, керівники ДП, проект, екзаменаційна комісія, стейкхолдери.  **Задачі моделювання**:   1. Розподіл студентів за керівниками ДП 2. Вибір студентом теми ДП 3. Календарний графік та контроль виконання ДП 4. Моделювання процесу консультацій студентів за темою ДП 5. Моделювання процесу рецензування ДП 6. Моделювання процесу перевірки ДП на плагіат 7. Моделювання захисту ДП (презентація, виступ, дискусії тощо) 8. Формування протоколу захисту ДП   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться при вирішенні задач моніторингу підготовки та захисту ДП | <https://drive.google.com/drive/my-drive>/Метод\_вказівки\_кваліф\_бакалавр.pdf |
|  | Моделювання діяльності мобільного (пересувного) готелю з доставкою до  заданого гостем місця перебування | 75 | **Мета**: на основі моделювання бізнес-процесів мобільного готелю з доставкою до  заданого гостем місця показати шляхи удосконалення відпочинку відповідно до фантазій клієнта, надавши їм можливість вибирати самим місце, де буде знаходитись готель.  **Об’єкти моделювання:** клієнт, мобільний (пересувний) готель (трейлер), адміністратор готелю, засоби зв’язку, засоби пересування готелю  **Задачі моделювання:**   1. Реєстрація клієнта на сайті пересувного готелю 2. Бронювання та оплата готелю (трейлеру) 3. Доставка готелю (трейлеру) до заданого клієнтом місця 4. Моделювання подорожі разом з мобільним готелем 5. Фантазії студента щодо діяльності пересувного готелю   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності мобільного (пересувного) готелю . | <https://hoteltv.com.ua/uk/mobilnyj-otel-s-dostavkoj-do-zhelaemogo-mesta/>  <https://www.segodnya.ua/ua/world/europe/v-finlyandii-poyavilsya-otel-na-sanyah-1124427.html>  <https://his.ua/article/revoljutsiya-v-mire-puteshestviy-mobilnyy-otel-na-avtopilote_2018-09-05> |
|  | Моделювання електронного банкінгу | 75 | **Мета**: на основі моделювання бізнес-процесів м електронного банкінгу показати шляхи удосконалення користування банківськими послугами.  **Об’єкти моделювання:** банк, сайт банку, клієнт, менеджер банку, банківська картка клієнта, транзакції (послуги банку)  **Задачі моделювання:**   1. Оплата комунальних послуг (електроенергія, газ, телефон і тд,) 2. Оплата рахунків за підключення (IP-телефонія, сотове та пейджингове підключення, Інтернет) та будь-яких інших послуг 3. Переказ грошей зі свого рахунку (в тому числі валютного) на будь-який рахунок в будь-якому банку 4. Переказ коштів в оплату рахунків за товари, в тому числі куплених в Інтернет-магазинах 5. Продаж и купівля зарубіжної валюти 6. Поповнення/зняття коштів з дебітних и кредитних пластикових карт 7. Отримання виписки про стан рахунку на даний момент у різних формах   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності електронного банкінгу |  |
|  | Моделювання бізнес-процесів виходу України з кризи після карантину | 100+ | **Задачі моделювання**   1. інструменти допомоги державі та містам; 2. поради бізнесу; 3. інструменти для університетів і шкіл; 4. проекти для зупинки та/або передбачення розповсюдження коронавірусу; 5. ідеї корисних флешмобів; 6. ідеї для зменшення паніки населення. | <https://mail.google.com/mail/u/0/?tab=wm&ogbl#inbox/FMfcgxwHMPmVzgHmCLgrqKLPhXlbcZDs> |
|  | Моделювання бізнес-процесів роботи автоматизованого сервісного центру з надання поліцейських послуг | 75 | **Мета**: на основі моделювання бізнес-процесів роботи Smart поліції показати шляхи удосконалення її діяльності, покращення запобігання злочинності, підвищення оперативності реагування на правопорушення тощо.  **Об’єкти моделювання:** центр реєстрації та надання поліцейських послуг, поліцейський, клієнт, сервісна послуга, кримінальна подія, адміністративне правопорушення, заявка клієнта, технічні пристрої та технології для фіксації та розкриття злочинів  **Задачі моделювання:**   1. Дорожньо-транспортні пригоди:   сплата штрафу за трафік  довідка про стан дорожнього руху  запит на перевірку транспортного засобу  сплата штрафу за неправильне паркування  реєстрація та розмитнення транспортного засобу тощо   1. Туристична безпека   Туристичні довідки  Охорона туристів  консультації та допомога відпочивальникам,  реєстрація скарг громадян щодо ушкодження і зникнення майна,  аналіз оперативної ситуації в місцях скупчення туристів  Втрачені та знайдені предмети   1. Кримінальні злочини: 2. Реєстрація повідомлення громадян про злочин 3. Подання кримінальної скарги 4. Запит на відвідування затриманого 5. Моніторинг статусу скарги тощо   **Результат**: ОО програма, яка в текстовому консольному режимі інформує про процеси, що відбуватимуться в результаті імітації діяльності роботи автоматизованого поліцейського сервісного центру. | <https://gulfnews.com/uae/government/fully-automated-police-station-opens-at-city-walk-1.2091514>  <https://delo.ua/business/kak-tehnologii-pomogajut-kievskoj-policii-raskry-347476/>  <https://www.bayut.com/mybayut/smart-police-stations-dubai/> |