**Черкаський національний університет імені Богдана Хмельницького**

**Кафедра інтелектуальних систем прийняття рішень**

**КУРСОВА РОБОТА**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**НА ТЕМУ «Програма обліку пацієнтів ветеринарної клініки»**

Студентки 2 курсу, групи КН-21\_\_\_

Галузі знань 12 «Інформаційні технології»

Спеціальності 122 «Комп’ютерні науки»

Погорілої К. С.\_\_\_\_\_\_\_\_\_\_\_

Керівник: ст. викл. кафедри ІСПР \_\_\_\_

Любченко К. М.\_\_\_\_\_\_\_\_\_\_\_

Національна шкала: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_ Оцінка: ECIS\_\_\_\_\_\_\_

Члени комісії:

\_\_\_\_\_\_\_ Любченко К. М.\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_ Стабецька Т. А.\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_ Царик Т. Ю.\_\_\_\_\_

(підпис) (прізвище та ініціали)

м. Черкаси – 2023 рік

Зміст

**Вступ1**

**Розділ 1. Основні поняття ветеринарії2**

**1.1 Найпоширеніші терміни2**

**1.1.1 Пацієнт 2**

**1.1.2 Ветеринарна медицина 2**

**1.1.3 Ветеринар 3**

**1.2 Огляд та аналіз існуючих програм3**

**1.2.1 VetBlue 4**

**1.2.2 ClinicPro 4**

**1.3 Висновок до першого розділу5**

**Розділ 2. Проектування програми обліку пацієнтів ветеринарної клініки6**

**2.1 Огляд основних принципів ООП6**

**2.2 Блок-схема7**

**2.3 Діаграма класів програми обліку пацієнтів11**

**2.4 Висновок до другого розділу12**

**Розділ 3. Розробка програми обліку пацієнтів ветеринарної клініки13**

**3.1 Розгляд компонентів програми обліку13**

**3.2 Реалізація компонентів програми обліку пацієнтів18**

**3.3 Тестування програми обліку пацієнтів24**

**3.4 Висновок до третього розділу30**

**Висновки31**

**Використані джерела32**

**Вступ**

У наш час є дуже актуальним ефективне використання інформації, особливо для медичних організацій. Необхідність використовувати великі обсяги інформації під час вирішення різних діагностичних, терапевтичних та інших завдань зумовило створити системи для медичних установ з метою використання та зберігання всієї потрібної інформації.

**Актуальність теми** полягає вдопомозі працівникам ветеринарної клініки здійснювати ефективне управління пацієнтами.

**Метою курсової роботи є** розробка програми обліку пацієнтів ветеринарної клініки із застосуванням принципів об’єктно-орієнтованого програмування.

Для досягнення зазначеної мети необхідно вирішити наступні **завдання**:

1. аналіз алгоритму для обліку даних;
2. опрацювання теоретичної інформації;
3. побудова блок-схеми;
4. реалізація програмного продукту;
5. формування висновків.

**Розділ 1. Основні поняття ветеринарії**

**1.1 Найпоширеніші терміни**

Спочатку трохи історії. Ветеринарія, як наука та професія, виникла доволі давно. Вона існувала у різних культурах. Наприклад, у давньому Римі були ветеринари, вони лікували, насамперед, військових коней, також й інших тварин. У давньому Єгипті також були люди, які славились тим, що лікували різну худобу.

У середньовіччі ветеринари були дуже часто пов’язані з зоологією та ботанікою, також вони допомагали у вивченні різних видів тварин та природи. У ранньому середньовіччі в Індії людей, які лікували велику різноманітність тварин, починаючи від слонів й закінчуючи мавпами, називали «пашуваки».

З часом ветеринарія стала більш професійною дисципліною. У 18-19 століттях, були створені перші ветеринарні коледжі в Європі, а саме коледж ветеринарної медицини в Ліоні, Франція (у 1761 році) та коледж ветеринарної медицини в Лондоні, Англія (у 1791 році), також в США у 1852 році було засновано перший ветеринарний коледж в Бостоні.

На сьогодні ветеринари – це професіонали, які займаються лікуванням та профілактикою захворювань у тварин.

**1.1.1 Пацієнт**

Пацієнт (лат. patiens – терплячий, страждаючий) – фізична особа, яка отримує медичну допомогу (профілактичну, діагностичну, лікувальну), або піддається медико-біологічним дослідженням (клінічним випробуванням), або є донором крові чи певного органа при трансплантації тощо [2].

У фармації найбільш поширеними є такі терміни: хворий, відвідувач.

У ветеринарній медицині пацієнтами є тварини, які потребують медичної допомоги.

**1.1.2 Ветеринарна медицина**

Ветеринарія (або ветеринарна медицина) – напрям наукових знань і практичної діяльності, спрямованих на боротьбу з хворобами тварин, охорону людей від зооантропонозів (інфекцій, загальних для тварин і людини), випуск доброякісної в санітарному плані продукції та вирішення ветеринарно-санітарних проблем захисту навколишнього середовища [2].

Ветеринарія здійснює контроль за тим, щоб тварини у сільському та міському господарствах були здорові.

Перелік наук, які тісно пов’язані з ветеринарією:

* анатомія;
* гістологія;
* фізіологія;
* хірургія;
* фармакологія;
* паразитологія.

Головною метою ветеринарної медицини є попередження різних захворювань та їх контроль. Це може бути отримано за допомогою виконання профілактичних дії, наприклад, щеплення, правильне харчування, дотримання норм гігієни. Також, ветеринарна медицина має вагоме значення в якості продуктів тваринного походження (м’ясо, яйця, молоко).

**1.1.3 Ветеринар**

Ветеринар – спеціаліст, що займається ветеринарією [3]. Він постійно відновлює та підтримує здоров’я тварин, за допомогою профілактики, діагностики та лікування різних травм або захворювань.

Ветеринари використовують різні методи для лікування тварин. Наприклад, хірургічні втручання, фізіотерапія, медичні препарати та інше.

Основні знання про хвороби тварин дають такі науки, як: вірусологія, мікробіологія, патофізіологія та паразитологія. Ветеринар повинен знати фізіологію, анатомію і біологічну хімію тварин. Є перелік тварин, які є обов’язковими для вивчення – велика та дрібна рогата худоба, собаки, коти, свійський птах, коні.

**1.2 Огляд та аналіз існуючих програм**

Розглянемо дві вже існуючі програми для обліку пацієнтів ветеринарної клініки.

**1.2.1 VetBlue**

VetBlue – це програма для обліку пацієнтів ветеринарної клініки, яка дає змогу легко та швидко створювати електронні медичні картки пацієнтів. Також вести облік лікування та препаратів, планувати прийоми [7].

Програма надає можливість створювати та заповнювати звіти, робити аналіз статистики, що не аби як допомагає клініці планувати свою роботу, діяльність, а також розвиватися.

Перевагами цієї програми є:

* простий інтерфейс;
* широкий функціонал;
* можливість роботи онлайн або офлайн.

Недоліками даної програми є:

* вартість (для невеликої клініки може бути доволі висока);
* потребує початкового корегування (налаштування) перед використанням.

**1.2.2 ClinicPro**

ClinicPro – це програма для обліку пацієнтів медичних закладів, що дозволяє створювати електронні медичні картки, контролювати процес лікування, планувати прийоми, а також вести облік різних лікарських засобів.

Дана програма, як і VetBlue, дає змогу створювати звіти й аналізувати роботу клініки, для подальшого її функціонування й розвитку.

Плюсами даної програми є:

* широкий функціонал;
* легка інтеграція з іншими програмними засобами, задля більш ефективної роботи;
* доволі зрозумілий та зручний інтерфейс.

Програма має деякі недоліки:

* вартість (не всі клініки можуть собі дозволити);
* обладнання (наявність потужного комп’ютера або ноутбука, також швидкий інтернету);
* складність в налаштування (можуть знадобитись деякі кваліфіковані навички для налаштування та використання).

**1.3 Висновок до першого розділу**

В даному розділі було більш детально розглянуто історію розвитку такої науки, як ветеринарія (ветеринарна медицина), а також її основні терміни, з якими вона тісно пов’язана. Також розглянуто дві існуючі програми для обліку пацієнтів ветеринарної клініки, їх переваги та недоліки.

**Розділ 2. Проектування програми обліку пацієнтів ветеринарної клініки**

**2.1 Огляд основних принципів ООП**

На сьогодні ООП є однією з основних парадигм програмування й більшість мов програмування його підтримують.

ООП дає змогу програмістам створювати більш складні та масштабовані програми, збільшуючи їх ефективність та простоту обслуговування.

Спочатку розглянемо основні поняття й принципи ООП.

ООП (Об’єктно-Орієнтоване Програмування) – це набір правил, згідно яких створюється код. Основна ідея ООП полягає в створенні класу, який відіграє роль шаблону, що містить різні методи, функції, змінні, замість написання програми. Складові такого класу взаємодіють між собою [4].

ООП засновано на трьох основних принципах:

* інкапсуляція;
* наслідування;
* поліморфізм;
* абстракція.

Інкапсуляція – це процес розділення елементів абстракції, що визначає її дані й методи. Вона призначена для ізоляції інтерфейсів від їх реалізації. Тобто, клас повинен складатись з двох частин – інтерфейс та реалізація. В самій реалізації забезпечується механізм приховування (принцип, який розмежовує доступ до внутрішніх об’єктів для різних частин програми) [8].

У ООП інкапсуляція тісно пов’язана з ще одним принципом – абстракції даних.

Абстракція – надання об’єкту різних характеристик, за допомогою яких він відрізняється від інших об’єктів.

Наслідування – це механізм системи, який дозволяє одним класам наслідувати властивості, поведінку інших класів [4].

Базовий клас – це клас, від якого наслідуються інші класи. Базовим класом також може бути абстрактний клас або інтерфейс.

Похідний клас – клас, який наслідується від базового.

Поліморфізм – властивість функції, яка дозволяє редагувати різні дані різних типів.

Є дві форми поліморфізму:

* класичний поліморфізм (перевизначення virtual / override, приведення типів);
* аd-hoc поліморфізм.

**2.2 Блок-схема**

Дана програма працює завдяки класам, в яких є методи для опрацювання інформації та її зберігання в файл. Блок-схема методу, який зчитує інформацію з файлу, а також додає дані в список (масив) розташована на рис. 2.1.

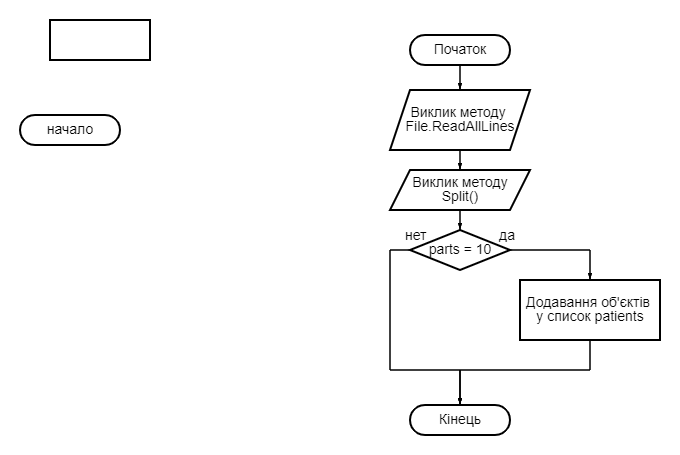
****

Рис. 2.1. Блок-схема методу «LoadPatFile»

Блок-схема для зберігання інформації або її змін у файл (рис. 2.2).

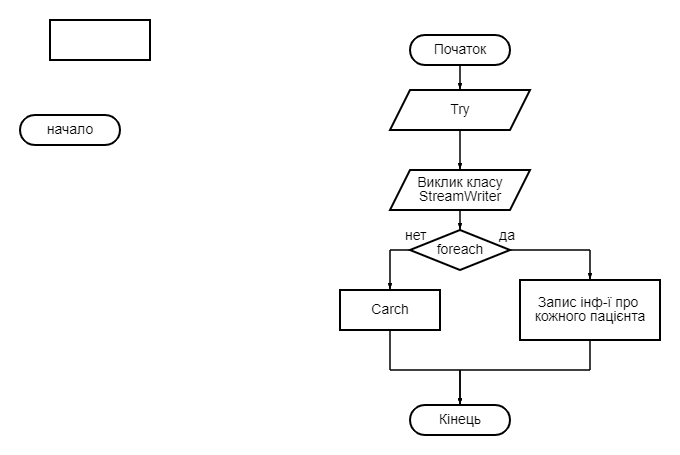


Рис. 2.2. Блок-схема методу «SaveFile»

Блок-схема методу, який додає всі дані про пацієнта та зберігає їх у файл (рис. 2.3).

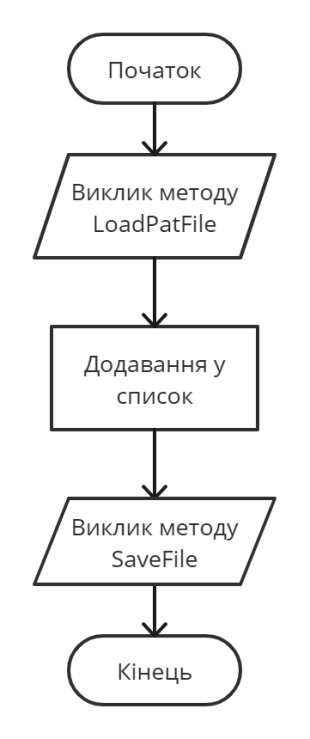
****

Рис. 2.3. Блок-схема методу «AddPatient»

Блок-схема методу, який видаляє дані про пацієнта, відповідно його номеру в списку (рис. 2.4). Якщо не вірно вказаний номер пацієнта – з’являється повідомлення, що він відсутній у списку.

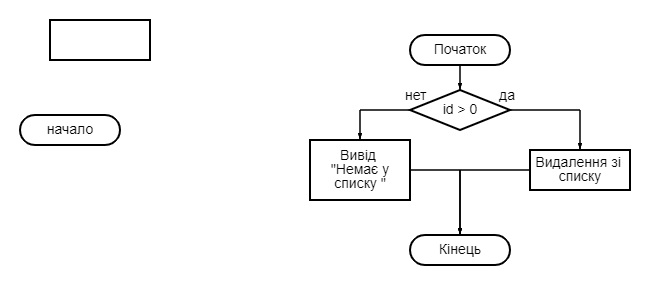


Рис. 2.4. Блок-схема методу «RemovePatient»

Блок-схема методу, який змінює конкретну інформацію про пацієнта (рис. 2.5). Якщо номер пацієнта введено не правильно або введене число відсутнє у списку, то з’являється «Немає у списку».

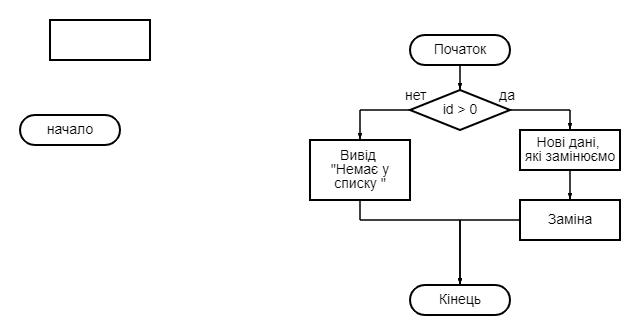


Рис. 2.5. Блок-схема методу «UpdatePatient»

Блок-схема методу, який виводить інформацію про пацієнтів, відповідно до вказаної дати (рис. 2.6).

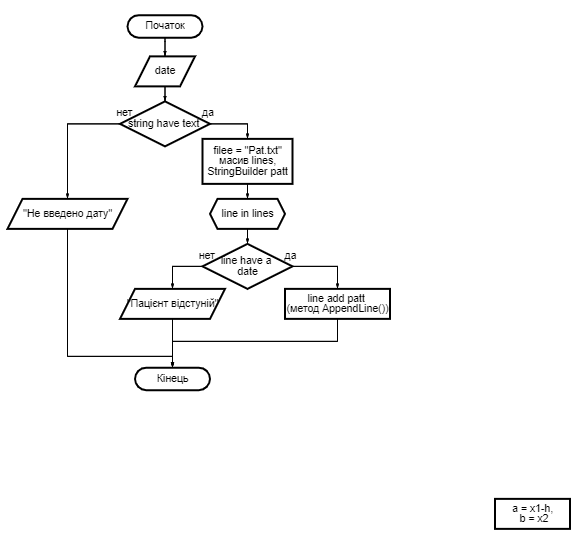


Рис. 2.6. Блок-схема методу «PrintData»

Блок-схема методу, який виводить інформацію про пацієнтів, відповідно до введеного лікаря (рис. 2.7).

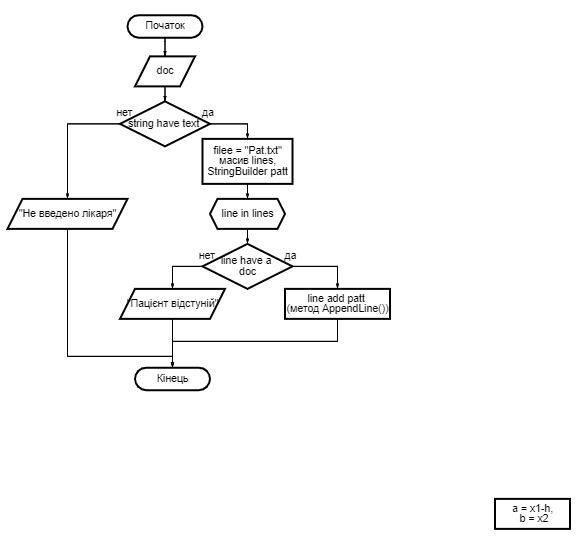


Рис. 2.7. Блок-схема методу «PrintDoc»

Блок-схема методу, який виводить всі дані про пацієнтів, які є у файлі (рис. 2.8).

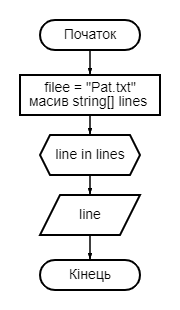


Рис. 2.8. Блок-схема методу «Print»

**2.3 Діаграма класів програми обліку пацієнтів**

Програма складається з трьох класів, які з’єднані між собою наслідуванням (рис. 2.9).

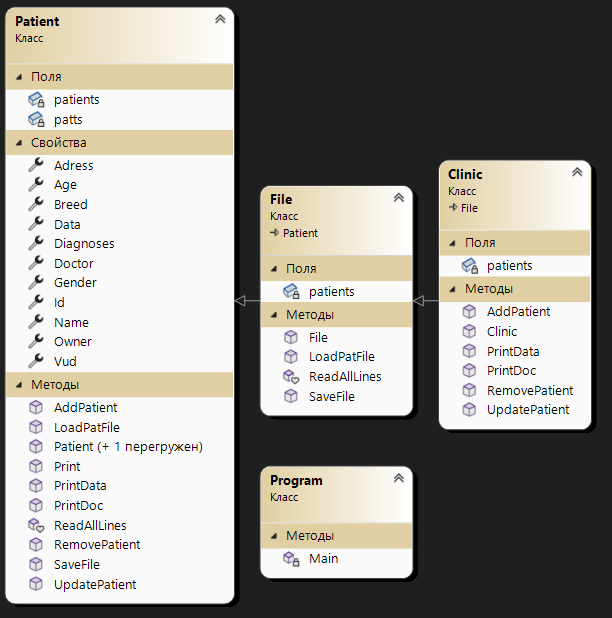


Рис. 2.9. Діаграма класів

Клас Patient являється базовим класом - від нього наслідується клас File. У тілі цього класу знаходяться властивості, які потрібні для роботи з даними, методи, які відображають роботу з даною інформацією та текстовим файлом.

У тілі класу File знаходяться тільки ті методи, які стосуються виключно роботи з файлом, а саме – LoadPatFile та SaveFile. Один метод зчитує інформацію з файлу, інший – зберігає її.

У тілі класу Clinic знаходяться методи для роботи з даними. За допомогою них інформацію можна редагувати, видаляти, додавати або за допомогою відповідних критеріїв відображати інформацію про конкретних пацієнтів. Також даний клас наслідується від класу File

У тілі класу Program є метод Main, який викликає головну форму.

**2.4 Висновок до другого розділу**

У даному розділі були розглянуті блок-схеми всіх методів програми, на основі яких відбувається робота з даними. Показана діаграма класів програмного продукту.

Також було описано, з чого складається кожен клас даної програми.

**Розділ 3. Розробка програми обліку пацієнтів ветеринарної клініки**

**3.1 Розгляд компонентів програми обліку**

Спочатку потрібно підключити усі потрібні бібліотеки, які знадобляться для написання програми обліку пацієнтів ветеринарної клініки й подальшої роботи в програмі Visual Studio.

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

Безпосередньо, перед тим, як створювати список й методи для роботи з ним, потрібно написати клас, який буде зберігати в собі всі властивості, методи, які необхідні для роботи з даними та текстовим файлом.

class Patient

{

public string Name { get; set; }

public string Owner { get; set; }

public string Breed { get; set; }

public string Vud { get; set; }

public string Gender { get; set; }

public int Age { get; set; }

public int Id { get; set; }

public string Doctor { get; set; }

public string Adress { get; set; }

public string Diagnoses { get; set; }

public string Data { get; set; }

}

Після створення класу Patient й записування в нього всіх властивості, які будуть використовувати при роботі зі списком, потрібно написати конструктор. Конструктор буде ініціалізувати всі властивості даного класу. Також необхідно створити методи для роботи з даними у файлі. Методи для зчитування інформації з файлу, а потім її збереження в нього – LoadPatFile() SaveFile(). Також методи для додавання, видалення, редагування даних, відображення інформації, відповідно до критеріїв вводу й просто відображення інформації про всіх пацієнтів, які є у файлі – AddPatient(), RemovePatient(), UpdatePatient(), PrintData(), PrintDoc() та Print().

public virtual void LoadPatFile()

{

string[] lines = File.ReadAllLines("Pat.txt");

foreach (string line in lines)

{

string[] parts = line.Split('\n');

int id = int.Parse(parts[0]);

string name = parts[1];

string owner = parts[2];

string vud = parts[3];

string breed = parts[4];

string gender = parts[5];

int age = int.Parse(parts[6]);

string adress = parts[7];

string doctor = parts[8];

string diagnoses = parts[9];

string data = parts[10];

Patient pt = new Patient()

{ Id = id, Name = name, Owner = owner, Vud = vud, Breed = breed, Gender = gender, Age = age, Adress = adress, Diagnoses = diagnoses, Doctor = doctor; Data = data};

patients.Add(pt);

}

}

public virtual void AddPatient(Patient patient) { patients.Add(patient); }

public virtual void RemovePatient(int id)

{

Patient pt = patients.Find(p => p.Id == id);

patients.Remove(pt);

}

public virtual void UpdatePatient()

{

Console.WriteLine("Id:");

int idUp = int.Parse(Console.ReadLine()) - 1;

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

Console.WriteLine($"Введiть новий дiагноз, вiк, адресу та нового лiкаря для пацiєнта: {idUp} {idUp} {idUp} {idUp} {idUp} {idUp}:");

string newDiagnoses = Console.ReadLine();

patients[idUp].Diagnoses = newDiagnoses;

int newAge = Convert.ToInt32(Console.ReadLine());

patients[idUp].Age = newAge;

string newAdress = (Console.ReadLine());

patients[idUp].Adress = newAdress;

string newDoctor = (Console.ReadLine());

patients[idUp].Doctor = newDoctor;

string newData = (Console.ReadLine());

patients[idUp].Data = newData;

}

public virtual void SaveFile()

{

try

{

using (StreamWriter wr = new StreamWriter("Pat.txt"))

{

foreach (Patient pt in patients)

{string line = "" + pt.Id + "" + pt.Name + "" + pt.Age + "" + pt.Gender + "" + pt.Owner + "" + pt.Adress + "" + pt.Diagnoses + "" + pt.Doctor + ""+ pt.Data;

wr.WriteLine(line); }

}

}

catch (Exception ex) {Console.WriteLine("Failed to save data: " + ex.Message); }

}

public virtual void PrintDoc()

{

Console.WriteLine("Enter doc: ");

string doc = Console.ReadLine();

if (!string.IsNullOrWhiteSpace(doc))

{

string filee = "Pat.txt";

string[] lines = File.ReadAllLines(filee);

StringBuilder patt = new StringBuilder();

foreach (string line in lines)

{line.Contains("Лікар: " + doc)) { patt.AppendLine(line); }

}

}

public virtual void PrintData()

{

Console.WriteLine("Enter data: ");

string date = Console.ReadLine();

if (!string.IsNullOrWhiteSpace(date))

{

string filee = "Pat.txt";

string[] lines = File.ReadAllLines(filee);

StringBuilder patt = new StringBuilder();

foreach (string line in lines)

{line.Contains("Дата прийому: " + date)) { patt.AppendLine(line); }

}

public void Print()

{

string filePath = "Pat.txt";

string[] lines = File.ReadAllLines(filePath, Encoding.UTF8);

foreach (string line in lines)

{ Console.WriteLine(line); }

}

Наступним кроком необхідно створити клас, який буде наслідуваним від класу Patient. У ньому також потрібно буде створити масив, для зберігання всіх пацієнтів, конструктор від базового класу й реалізувати необхідні методи для роботи з текстовим файлом.

class File : Patient

{

public File(string filePat) : base(0, "", "", "", "", "", 1, "", "", "", "")

{

patients = new List<Patient>();

LoadPatFile();

}

internal static string[] ReadAllLines(string v)

{

throw new NotImplementedException();

}

private List<Patient> patients = new List<Patient>();

Для роботи з файлом необхідно перевизначити метод LoadPatFile, який зчитує інформацію з вказаного файлу й додає дані в список.

public override void LoadPatFile()

{

try

{

string[] lines = File.ReadAllLines("Pat.txt");

foreach (string line in lines)

{

string[] parts = line.Split('\n');

if (parts.Length == 10)

{ int id = int.Parse(parts[0]);

string name = parts[1];

string owner = parts[2];

string vud = parts[3];

string breed = parts[4];

string gender = parts[5];

int age = int.Parse(parts[6]);

string adress = parts[7];

string doctor = parts[8];

string diagnoses = parts[9];

string data = parts[10];

Patient pt = new Patient()

{ Id = id, Name = name, Owner = owner, Vud = vud, Breed = breed, Gender = gender, Age = age, Adress = adress, Doctor = doctor, Diagnoses = diagnoses, Data = data };

patients.Add(pt); }

}

}

catch (Exception ex) {Console.WriteLine("Failed to load data: " + ex.Message); }

}

Для того, щоб була можливість зберігати інформацію про пацієнтів у файл потрібно метод SaveFile, який створено в базовому класі, перевизначити. Також у ньому викликається спеціальний клас StreamWriter для роботи з файлами.

private void SaveFile()

{

try

{

using (StreamWriter wr = new StreamWriter("Pat.txt"))

{

foreach(Patient pt in patients)

{ string line = "Номер: " + pt.Id + ";" + "Ім'я: " + pt.Name + ";" + "Власник (ПІБ): " + pt.Owner + ";" + "Вид тварини: " + pt.Vud + ";" + "Порода: " + pt.Breed + ";" + "Стать ч/ж: " + pt.Gender + ";" + "Вік: " + pt.Age + ";" + "Домашня адреса: " + pt.Adress + ";" + "Діагноз: " + pt.Diagnoses + ";" + "Лікар: " + pt.Doctor + ";" + pt.Data + ".";

wr.WriteLine(line); }

}

}

catch(Exception ex)

{Console.WriteLine("Failed to save data: " + ex.Message); }

}

**3.2 Реалізація компонентів програми обліку пацієнтів**

Після того, як було перевизначено в похідному класі File методи для зчитування та зберігання інформації у файл, потрібно створити ще один похідний клас від класу File, в якому будуть також перевизначені методи, які необхідні для обробки або відображення даних про пацієнтів ветеринарної клініки.

У методі AddPatient викликаються методи LoadPatFile та SaveFile й потім відбувається процес додавання нового пацієнта безпосередньо у файл.

public override void AddPatient(Patient patient)

{

LoadPatFile();

patients.Add(patient);

SaveFile();

}

Також потрібно перевизначити метод, який видаляє конкретного пацієнта зі списку, згідно його номеру. У ньому викликається метод SaveFile, щоб зберегти всі зміни у файл. Також додається перевірка чи пацієнт з вказаним номер наявний у файлі.

public override void RemovePatient (int id)

{

Patient pt = patients.Find (p => p.Id == id);

if (pt != null){ patients.Remove(pt); SaveFile();}

else {Console.WriteLine("Немає у списку");}

}

Для редагування інформації про пацієнта необхідно перевизначити метод UpdatePatient. У ньому також за допомогою номера пацієнта можна змінювати конкретну інформацію про нього.

public override void UpdatePatient()

{

Console.WriteLine("Id:");

int idUp = int.Parse(Console.ReadLine()) - 1;

if(idUp >= 0 && idUp < patients.Count)

{Console.WriteLine("Введiть 1 щоб змiнити дiагноз, 2 - вiк, 3 - адресу, 4 – лiкаря, 5 – дату прийому");

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

if (chacha == 1)

{Console.WriteLine($"Введiть новий дiагноз для пацiєнта {idUp}:");

string newDiagnoses = Console.ReadLine();

patients[idUp].Diagnoses = newDiagnoses;

}

else if (chacha == 2)

{Console.WriteLine($"Введiть новий вiк для пацiєнта {idUp}:");

int newAge = Convert.ToInt32(Console.ReadLine());

patients[idUp].Age = newAge; }

else if (chacha == 3)

{Console.WriteLine($"Введiть нову адресу для пацiєнта {idUp}:");

string newAdress = (Console.ReadLine());

patients[idUp].Adress = newAdress; }

else if (chacha == 4)

{Console.WriteLine($"Введiть нового лiкаря для пацiєнта {idUp}:");

string newDoctor = (Console.ReadLine());

patients[idUp].Doctor = newDoctor; }

else if (chacha == 5)

{Console.WriteLine($"Введiть нову дату прийому для пацiєнта {idUp}:");

string newData = (Console.ReadLine());

patients[idUp].Data = newData; }

SaveFile();

Console.WriteLine("Iнформацiя про пацiєнта успiшно змiнена");

}

else {Console.WriteLine($"Пацiєнт з даним номером - не знайднений");}

}

}

Для того, щоб знати, які саме пацієнти були на прийомі у конкретного лікаря потрібно перевизначити метод PrintDoc й додати перевірку.

public override void PrintDoc()

{

Console.WriteLine("Enter data: ");

string doc = Console.ReadLine();

if (!string.IsNullOrWhiteSpace(doc))

{string filee = "Pat.txt";

string[] lines = File.ReadAllLines(filee);

StringBuilder patt = new StringBuilder();

foreach (string line in lines)

{if (line.Contains("Лікар: " + doc)) { patt.AppendLine(line); }}

if (patt.Length > 0) {Console.WriteLine(patt.ToString());}

else {Console.WriteLine("Пацієнт(-и) відстній(-і) з вказаним лікарем");}}

else {Console.WriteLine("Не введено лікаря");}}

Для того, щоб мати змогу дізнатись, які саме пацієнти відвідували клініку в конкретний день потрібно перевизначити метод PrintData й додати перевірку.

public override void PrintData()

{ Console.WriteLine("Enter data: ");

string date = Console.ReadLine();

if (!string.IsNullOrWhiteSpace(date))

{ string filee = "Pat.txt";

string[] lines = File.ReadAllLines(filee);

StringBuilder patt = new StringBuilder();

foreach (string line in lines)

{if (line.Contains("Дата прийому: " + date)) { patt.AppendLine(line); }}

if (patt.Length > 0) {Console.WriteLine(patt.ToString());}

else {Console.WriteLine("Пацієнт(-и) відстній(-і) з вказаною датою"); }

}

else {Console.WriteLine("Не введено дату");}

}

Тепер, коли є всі необхідні методи, можна створити Windows Form.

Спочатку створюється головна форма. На ній розташовані чотири кнопки й текст. Перша кнопка має назву «Додати», друга – «Змінити», третя – «Видалити», четверта – «Список пацієнтів» (рис. 3.1).

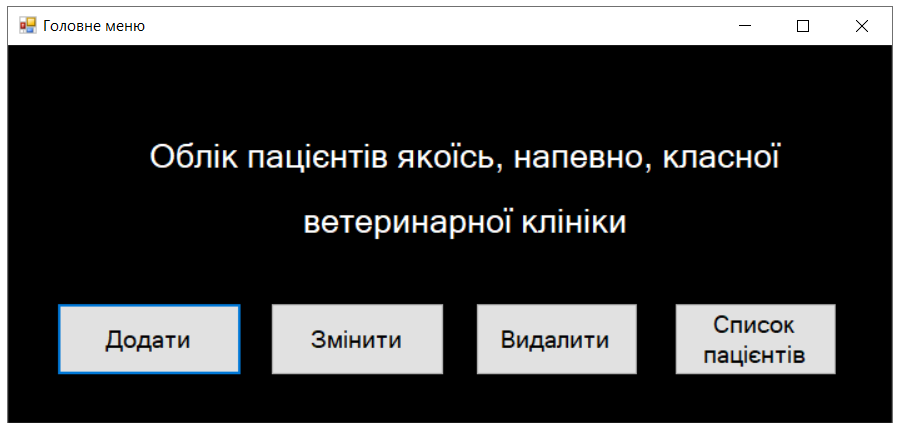


Рис. 3.1. Головна форма

При виборі кнопки «Додати» з’являється нова форма, на якій розташовані поля для вводу тексту й кнопка для збереження введених даних у файл (рис. 3.2).

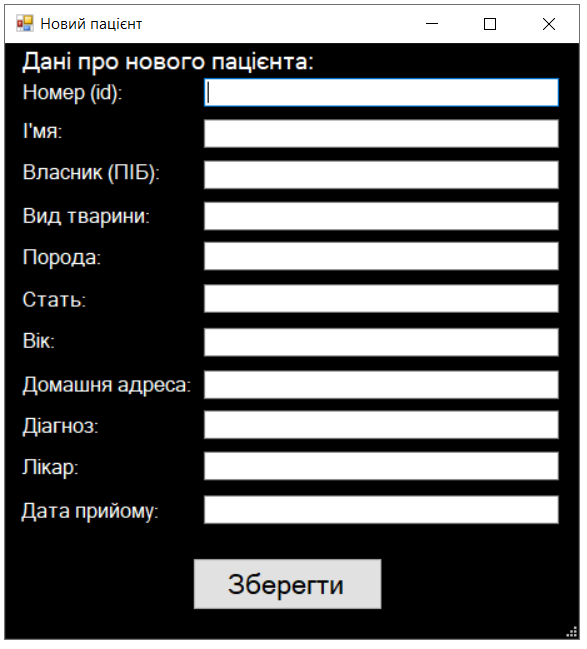


Рис. 3.2. Форма для додавання нового пацієнта

Якщо обрати кнопку «Видалити», то з’явиться нова форма, на якій буде розташоване поле, в яке потрібно вписати номер пацієнта, інформацію про якого необхідно видалити, а також ще одна кнопка «Видалити», яка зберігає зміни у файл (рис. 3.3).

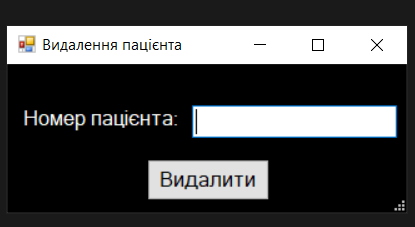


Рис. 3.3. Форма для видалення пацієнта

При виборі кнопки «Змінити» також з’являється нова форма, на якій розташовані текстові поля для вводу номера пацієнта та нової інформації, а також кнопка «Зберегти». При натисканні на кнопку зміни про конкретного пацієнта, номер якого було введено, зберігаться у файл (рис. 3.4).

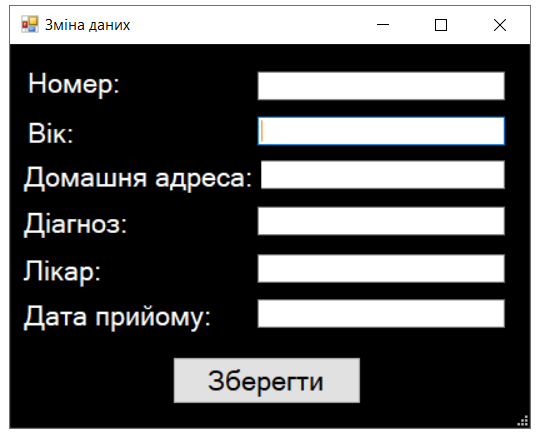


Рис. 3.4. Форма для зміни даних про пацієнта

Якщо натиснути на кнопку «Список пацієнтів», то з’явиться нова форма, на якій знаходяться три кнопки – «Дати прийому», «Лікаря» та «Весь список» (рис. 3.5).

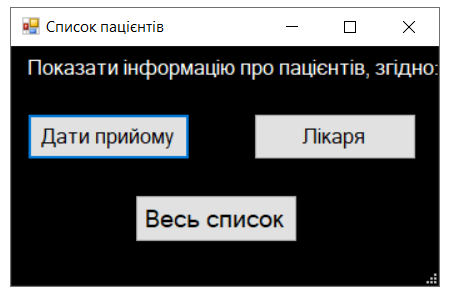


Рис. 3.5. Форма відображення списку пацієнтів

При виборі кнопки «Дати прийому» з’являється нова форма, на якій розміщене текстове поле, в яке вводиться дата прийому (рис. 3.6).

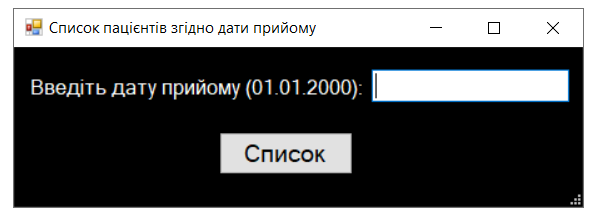


Рис. 3.6. Форма відображення пацієнтів за датою прийому

При виборі кнопки «Лікаря» також з’являється нова форма, в поле якої вводиться ПІБ лікаря (рис. 3.7).

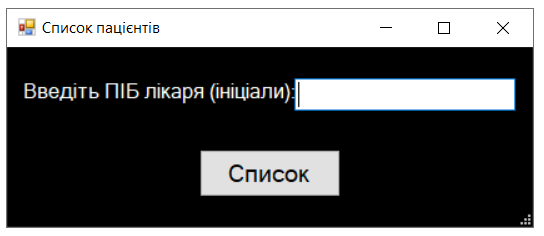


Рис. 3.7. Форма відображення пацієнтів згідно їх лікаря

При виборі кнопки «Весь список» з’явиться нове вікно, в якому буде інформація про всіх пацієнтів.

**3.3 Тестування програми обліку пацієнтів**

При виборі першої кнопки «Додати» на головній формі, з’являється нова форма й вводяться всі дані про нового пацієнта (рис. 3.8).

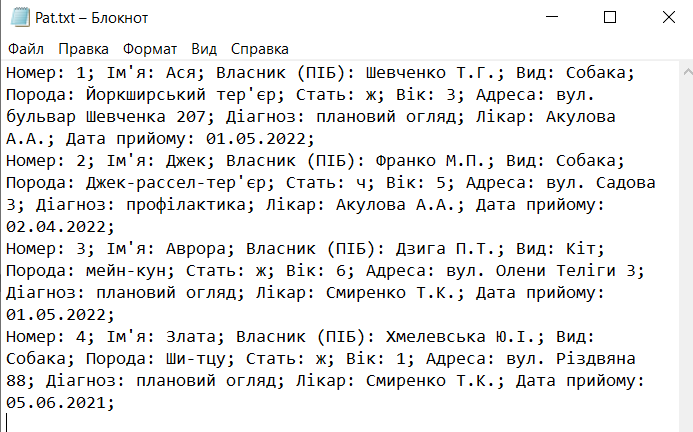


Рис. 3.8. Результат роботи методу AddPatient

Якщо всі дані введено правильно, то з’являється повідомлення, що все збережено (рис. 3.9).

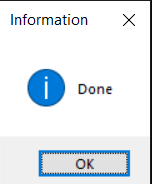


Рис. 3.9. Повідомлення про успішне додавання нового пацієнта

Якщо якесь поле залишилось порожнім, то з’являється відповідне повідомлення (рис. 3.10).

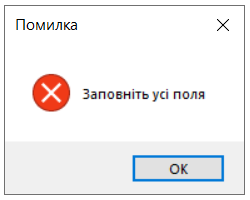


Рис. 3.10. Повідомлення про пропущені поля

При виборі другої кнопки «Змінити» відкривається нова форма, на якій розміщені поля, куди вводяться нові дані про пацієнта. При натисканні на кнопку «Зберегти» зміни зберігаються у файл (рис. 3.11).

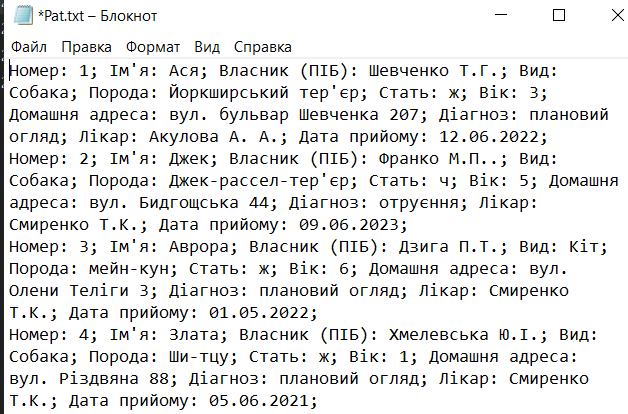


Рис. 3.11. Реалізація методу UpdatePatient

Якщо всі дані введено коректно, то з’являється повідомлення про успішне збереження змін (рис. 3.12).

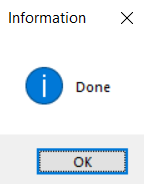


Рис. 3.12. Повідомлення про успішне збереження змін

Якщо номер пацієнта було введено не правильно або він відсутній у файлі, то з’являється нове повідомлення про помилку (рис. 3.13).

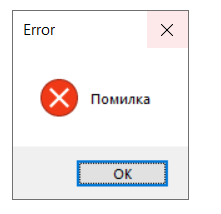


Рис. 3.13. Помилка при введені номера пацієнта

При виборі третьої кнопки «Видалити» відкривається нова форма, на якій є поле, куди вводиться номер пацієнта й кнопка «Видалити». При натисканні на кнопку інформація про пацієнта, номер якого було введено видаляється й зміни зберігаються у файл (рис. 3.14).

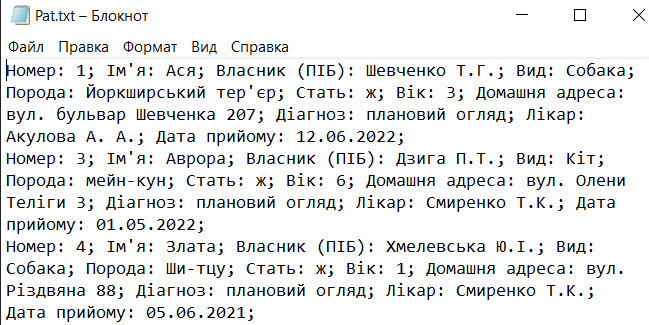


Рис. 3.14. Реалізація методу RemovePatient

Якщо введено не коректний номер пацієнта або він відсутній у списку, то з’являється повідомлення, що пацієнта не знайдено (рис. 3.15).

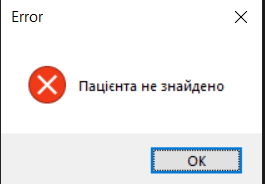


Рис. 3.15. Повідомлення, що пацієнта не знайдено

Якщо пацієнт з введеним номер є у списку, дані про нього видаляються й з’являється відповідне повідомлення (рис. 3.16).

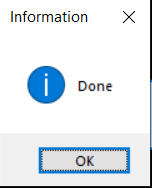


Рис. 3.16. Повідомлення про успішне видалення пацієнта

При виборі кнопки «Дати прийому», що знаходиться на формі, яка відкрилась після натискання на четверту кнопку «Список пацієнтів», з’являється нова форма, в яку вводиться дата відвідування. Потім після натискання на кнопку «Список», яка розміщена на цій формі, з’являється нове вікно, яке відображає пацієнтів, які були на прийомі згідно введеної дати (рис. 3.17).

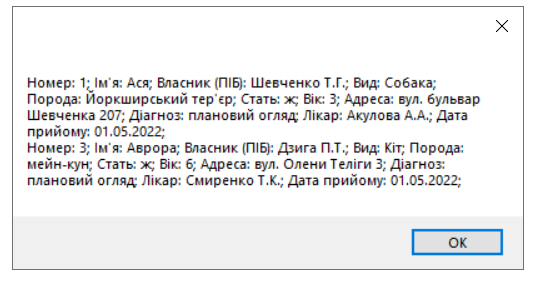


Рис. 3.17. Реалізація методу PrintData

Якщо формат дати введено не так, як вказано на формі, то з’являється відповідне повідомлення (рис. 3.18).

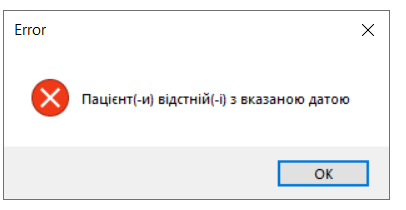


Рис. 3.18. Повідомлення про неправильно введену дату прийому

При виборі кнопки «Лікаря», що знаходиться на тій же формі, що й кнопка «Дати прийому», з’являється нова форма, в яку вводиться ПІБ лікаря. Потім після натискання на кнопку «Список», яка розміщена на цій формі, з’являється нове вікно, яке відображає пацієнтів, які були на прийомі у введеного лікаря (рис. 3.19).

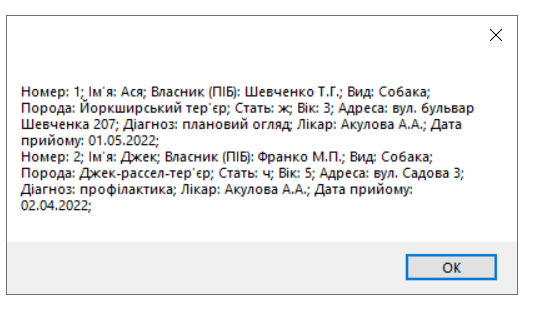


Рис. 3.19. Реалізація методу PrintDoc

Якщо було некоректно введено ПІБ лікаря, то з’являється повідомлення про помилку (рис. 3.20).

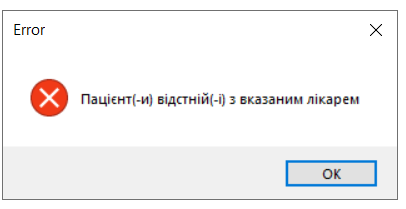


Рис. 3.20. Повідомлення про невірну форму вводу лікаря

Якщо натиснути на кнопку «Весь список» з’явиться нове вікно, в якому буде відображена інформація про всіх пацієнтів (рис. 3.21).

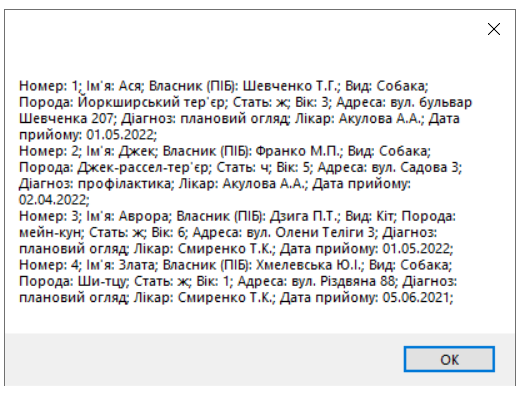


Рис. 3.21. Реалізація методу Print

**3.4 Висновок до третього розділу**

У даному розділі було розписано про методи, які потрібні для роботи даної програми обліку. Також описано, які використовуються форми для реалізації програми обліку пацієнтів ветеринарної клініки за допомогою Windows Form.

Також було показано, як саме відбувається робота програми безпосередньо в самому файлі й як відбувається робота кожного методу.

**Висновки**

Під час виконання даної курсової роботи було опрацьовано велику кількість нової інформації про основні терміни, які стосуються ветеринарної медицини, існуючі програми обліку, а також про різноманітні принципи ООП.

В документації було проаналізовано та сформульовано основні задачі даної роботи, її мету. Розроблено план виконання роботи.

Також було розглянуто декілька програм для обліку пацієнтів, які користуються попитом. Зроблено їх аналіз та вказано про переваги і недоліки кожної з них.

Також було розглянуто поетапне створення коду програмного продукту. Описано з чого він складається, з яких методів, для чого ці методи, як вони працюють. Дана програма обліку складається з трьох основних класів – Patient, File та Clinic.

У другому розділі було створено блок-схеми, які показують процес роботи кожного методу, відповідно до коду програми. Також діаграму класів й сказано, які саме класи були створені й коротко описано їх роботу.

У третьому розділі було більш детально розписано про складові коду. А саме те, що спочатку створюється клас Patient, який зберігає в собі інформацію про всі властивості, необхідні для безпосередньої роботи з даними пацієнтів, деякі методи. Потім створюється клас File, який є похідним від базового. Він перевизначає деякі методи з базового класу, завдяки яким відбуваються основні функції – зберігання інформації в файл та зміна даних у файлу. Потім створено клас Clinic, який є наслідуваним від класу File й реалізує всі методи двох інших класів. Також було описано методи, які використовувалися для редагування інформації в файлі або її відображення відповідно до критеріїв, а також виконано тестування програми й показано результат роботи.

Отже, документація до курсової роботи містить опис етапів створення програми обліку пацієнтів ветеринарної клініки та реалізацію даної програми.

**Використані джерела**

1. Ю.Є. Гребенович, О.О. Супруненко – «МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ТА ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ», 2013 - С.32.
2. «Фармацевтична енциклопедія» [Електронний документ]. Режим доступу: <https://www.pharmencyclopedia.com.ua/alphabet>. Перевірено: 10.06.20223.
3. «Вікіпедія. Ветеринар» [Електронний документ]. Режим доступу: <https://uk.wikipedia.org/wiki/Ветеринар>. Перевірено: 22.05.2023.
4. «Habr» [Електронний документ]. Режим доступу: <https://habr.com/ru/articles/463125/>. Перевірено: 15.05.2023.
5. «Вікіпедія. Ветеринарія» [Електронний документ]. Режим доступу: <https://uk.wikipedia.org/wiki/Ветеринарія>. Перевірено: 22.05.2023.
6. «Вікіпедія. ООП» [Електронний документ]. Режим доступу: <https://uk.wikipedia.org/wiki/ООП>. Перевірено: 15.05.2023.
7. «VetBlue» [Електронний документ]. Режим доступу: <https://www.eveterinarysoftware.com/>. Перевірено: 20.05.2023.
8. «Вікіпедія. Інкапсуляція» [Електронний документ]. Режим доступу: <https://uk.wikipedia.org/wiki/Інкапсуляція>. Перевірено: 22.05.2023.