

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ ЗАКЛАД
«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»**

**Навчально - науковий інститут
математики та інформаційних технологій**

Кафедра математики та інформатики

Маніта Олександр Олександрович

**ДОСЛІДЖЕННЯ ТА РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ
ЗАКЛАДУ ВЕТЕРИНАРНОЇ МЕДИЦИНИ**

кваліфікаційна робота

здобувача вищої освіти другого (магістерського) рівня

за спеціальністю 122 «Комп'ютерні науки»

Особистий підпис _____ Олександр МАНІТА

Науковий керівник _____ Юрій КОЗУБ,
доктор технічних наук,
професор кафедри
математики та інформатики

Завідувач кафедри _____ Юрій КОЗУБ,
доктор технічних наук,
професор кафедри
математики та інформатики

АНОТАЦІЯ

Маніта О.О.

Тема: Дослідження та розробка системи управління закладу ветеринарної медицини

Спеціальність: 122 «Комп'ютерні науки»

Установа: ДЗ ЛНУ імені Тараса Шевченка, 2024р.

Кваліфікаційна робота містить: 69 стор., 28 рис., 35 джерела, 2 додатки.

Об'єкт дослідження – діяльність клініки з надання ветеринарних послуг населенню.

Предмет дослідження – автоматизовані інформаційні системи керування.

Мета роботи – розробка програмного застосування для обліку інформації про тварин та управління ветеринарною клінікою «Авіцена».

Результати роботи. Досліджено існуючі автоматизовані інформаційні системи закладів з ветеринарії та діяльність об'єкту дослідження ветеринарної клініки «Авіцена». Проаналізовані альтернативні варіанти автоматизації і обґрунтовано рішення про розробку власної інформаційної системи, обрано середовище розробки, визначено склад сутностей і атрибутів, сформульовані функціональні можливості інформаційної системи.

Виконано програмну реалізацію АІС для управління ветеринарною клінікою «Авіцена» засобами Java, Swing, Eclipse.

Ключові слова: Eclipse, Java, Swing, АІС, програмне застосування

ABSTRACT

Manita O.O.

Theme: Research and development of a veterinary medicine institution management system

Specialty: 122 «Computer Science»

Institution: Taras Shevchenko National University of Luhansk, 2024

Qualification work contains: 69 pages, 28 figures, 35 sources, 2 appendices.

Object of research activities of the clinic to provide veterinary services to the population.

Subject of research - automated control information systems

Purpose of the study development of software application for accounting of information about animals and management of veterinary clinic «Avicenna»

Results of research. The existing automated information systems of veterinary institutions and the activity of the object of research veterinary clinic «Avicenna» were studied. Alternative options of automation are analyzed and the decision on development of own information system is substantiated, the development environment is chosen, the composition of entities and attributes is determined, the functional capabilities of the information system are formulated. Software implementation of AIS for management of veterinary clinic «Avicenna» by using Java, Swing, Eclipse is developed.

Keywords: Eclipse, Java, Swing, AIS, software application

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	5
ВСТУП	6
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.....	10
1.1. Дослідження існуючих програм та інформаційних систем.....	11
1.2. Загальні критерії автоматизації ветеринарної клініки «Авіцена»	15
1.3. Висновки до розділу 1	17
РОЗДІЛ 2. АНАЛІЗ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ	18
2.1. Вибір мови програмування.....	18
2.2. Вибір середовища розробки.....	26
2.3. Вибір бібліотеки графічного інтерфейсу.....	33
2.4. Висновки до розділу 2	39
РОЗДІЛ 3. ОПИС РЕАЛІЗАЦІЇ ТА ФУНКЦІОНАЛУ ВЕТЕРИНАРНОЇ СИСТЕМИ «АВІЦЕНА»	41
3.1. Опис реалізації та застосування ООП підходу	41
3.1.1. Абстракція	41
3.1.2. Інкапсуляція	42
3.1.3. Успадкування	44
3.1.4. Поліморфізм	45
3.2. Опис функціоналу системи.....	46
3.2.1. Сторінка входу	46
3.2.2. Сторінка адміністратора	46
3.2.3. Сторінка персоналу готелю	48
3.2.4. Сторінка власника.....	49
3.3. Висновки до розділу 3	50
ВИСНОВКИ.....	52
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	54
ДОДАТКИ.....	57

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	- application programming interface;
AWT	- інструментарій абстрактних вікон;
ERP	- Enterprise Resource Planning;
GUI	- графічний інтерфейс користувача;
IFC	- Internet Foundation Classes;
JDK	- Java Development Kit;
JFC	- Java Foundation Classes;
JVM	- Java Virtual Machine;
AIC	- автоматизована інформаційна система;
АСК	- автоматизована система керування;
ІКТ	- інформаційно-комунікаційні технології;
ІС	- інформаційні системи;
МОЗ	- міністерство охорони здоров'я;
ООП	- об'єктно-орієнтовані мови програмування;
ПЗ	- програмне забезпечення.

ВСТУП

У наш час програмне забезпечення значно полегшує життя людей. Майже немає галузі, яка б не використовувала програмне забезпечення в якійсь формі для полегшення діяльності в ньому. ІТ-компанії використовують різні програмні засоби для створення нового програмного забезпечення. Світ розвивається в цьому напрямку, і ми все більше і більше покладемося на інформаційні технології та програмне забезпечення, щоб людство могло еволюціонувати та досягти нових вершин.

На сьогоднішній день практично кожна організація має програмне забезпечення (ПЗ), що допомагає організації, упорядкуванні та керуванні процесами діяльності, у тому числі і ветеринарні клініки. В даний момент на території Білгород-Дністровського працюють три ветеринарні клініки: «Авіцена», «Ветеринарна Допомога» та Ветеринарна Клініка ФОП «Pavlo Pavlovich Shkvar». Всю інформацію про послуги, ціни і різне клієнт ветеринарної клініки може отримати тільки перебуваючи безпосередній в ній самій або у телефонному режимі. Тому і постало питання про створення програмного застосування автоматизованої системи керування ветеринарної клініки «Авіцена».

Інформаційні процеси (використання, зберігання, захист, пошук, збирання, передавання, опрацювання інформації) присутні у всіх областях медицини і галузі охорони здоров'я. Важливою складовою інформаційних процесів є інформаційні потоки. Від їх впорядкованості залежить чіткість функціонування галузі в цілому і ефективність управління нею. Потоки починаються в місцях виникнення інформації і забезпечують її доставку до місць прийняття рішень. Вони складаються з окремих повідомлень, відображених у сигналах і документах, і рухаються в просторі і часі від джерела інформації до одержувача. Для роботи з інформаційними потоками призначені інформаційні системи (ІС). Крім того, з початку спалаху COVID-19 і з впровадженням карантину та обмежувальних заходів

пов'язаних із поширенням коронавірусної хвороби актуальність дослідження автоматизації системи керування ветеринарної клініки «Авіцена» зростає.

На сьогодні в Україні сформована стійка система закладів охорони здоров'я домашніх тварин, що надають ветеринарну допомогу. У приватних установах продуктивність праці і ефективність наданої ветеринарної допомоги значно вище, ніж в більшості державних медичних установ. Як наслідок, кількість відвідувань спеціалістів ветеринарного профілю державних закладів за останні три роки в Україні зменшилась, а кількість відвідувань приватних ветеринарних закладів збільшилась. Також зберігається тенденція скорочення кількості державних ветеринарних закладів. Все вище наведене підкреслює актуальність обраної теми дослідження.

Мета роботи – дослідження існуючих автоматизованих інформаційних систем (AIC) закладів ветеринарних клініки, що надають медичну допомогу тваринам. Розробка простого у користуванні програмного застосування для обліку інформації про тварин та управління ветеринарною клінікою «Авіцена», що забезпечує основні функціональні можливості, необхідні ветеринару.

Для досягнення мети магістерського дослідження необхідно вирішити такі **завдання**:

1. Провести аналіз існуючих AIC ветеринарних закладів;
2. Провести аналітичний огляд базових технологій розробки AIC;
3. Розробити AIC для управління ветеринарною клінікою «Авіцена».

Наукова новизна: розроблено AIC для управління ветеринарною клінікою «Авіцена» засобами Java, Swing, Eclipse.

Практичне значення. Результати дослідження можуть бути запроваджені для будь-якої ветеринарної установи, що передбачає прийом пацієнтів за записом. Впровадження такої системи допоможе: підвищити ефективність управління ветеринарною клінікою; підвищити якість та

оперативність прийняття рішень у процесі лікування тварин; підвищити якість та зменшити тривалість обслуговування пацієнтів; підвищити ефективність праці медичного персоналу; зменшити терміни й спростити процедури підготовки звітних матеріалів за результатами роботи ветеринара.

Зв'язок роботи з державними програмами, планами науково-дослідних робіт. Результати дипломної роботи можуть бути використані як державними ветеринарними клініками, так і приватними клініками (кабінетами).

Методи дослідження:

Теоретичні методи: аналіз науково-технічних джерел з проблем дослідження.

Емпіричні методи: оптимізації розробки і функціонування програмних додатків і програмного забезпечення.

Методологічну основу дослідження склали положення системного аналізу, у якості методу дослідження ефективності вибирається порівняльний аналіз досліджуваних технологій на відповідність критеріям.

Особистий внесок магістра складається в:

- аналіз існуючих рішень;
- виборі методів досліджень і технологій реалізації;
- створення інформаційної системи обслуговування ветеринарної практики.

Структура дипломної роботи визначена метою і завданнями дослідження та складається зі вступу, трьох розділів, висновків, списку використаних джерел, який складається з 35 бібліографічних посилань, 2 додатків. Обсяг магістерської роботи – 71 сторінка.

Перший розділ присвячено аналізу існуючих рішень, досліджується актуальність проблеми, проводиться огляд та аналіз аналогічних програм для ветеринарних клінік з метою уникнення помилок та усунення недоліків у програмній розробці.

Другий розділ роботи приділено дослідженню засобів проектування. Здійснено аналіз сучасним мовам програмування, середовищам розробки, бібліотекам графічного інтерфейсу та обрано для розробки програмного забезпечення засоби Java, Swing, Eclipse.

У третьому розділі розглянуто створення програмного продукту, функціонал системи та тестування найбільш важливих процедур програми із зазначенням всіх використаних компонентів.

Додатки містять таблиці атрибутів ветеринарної клініки «Авіцена», фрагменти лістинга коду файлів.

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Призначення ветеринарної клініки - амбулаторне лікування тварин, здійснення профілактичних ветеринарно-санітарних заходів, організація по упередженню і з ліквідації заразних і незаразних хвороб тварин, а також проведення діагностичних досліджень.

Спеціалісти ветеринарної клініки забезпечують застосування лікарських засобів і методів, вилучаючих від'ємний вплив на тварин при діагностиці, лікуванні та профілактиці, високоефективних ветеринарних препаратів і методів ветеринарного впливу; гарантують безпеку ветеринарних заходів для здоров'я тварин при дотриманні наступних умов:

- споживач надає тварину для огляду, повідомляє про випадки, пов'язані з раптовим падіжом або одночасним масовим захворюванням тварин, або про їх незвичну поведінку;

- забезпечує відповідне дотримання та годування тварин згідно зоогігієнним вимогам, а так само дотримання дієти на вимогу ветеринарного лікаря, а також проведення обов'язково лікувально-профілактичних заходів у необхідні терміни (вакцинація, дегельмінтизація).

Ветеринарний лікар попереджає споживача про можливі ускладнення, а також про інші не залежні від виконавця обставин, що загрожують якості наданої споживачеві ветеринарної послуги (виконуваної роботи) [16].

Як і будь-яка інша організація, ветеринарна клініка вимагає уваги до свого обліку. Щоб врахувати всі аспекти ветеринарії, потрібна комплексна автоматизація діяльності. Можна використовувати кілька різних програм, але ефективніше використовувати одну програму, яка буде відразу об'єднувати всі моменти, а також дозволить проводити аналіз про виконану роботу. Програма для ветеринарії для управління і обліку ветеринарії

здатна автоматизувати діяльність будь-якої клініки з лікування тварин, а також зможе охопити всі сторони в роботі організації [17].

1.1. Дослідження існуючих програм та інформаційних систем

Проведемо аналіз існуючих сучасних програм для ветеринарних клінік, які бувають платні та безкоштовні, хмарні та ті, що інсталюються.

CosmoZoo

Онлайн сервіс для ветеринарних клінік і їх клієнтів CosmoZoo (рис.1.1) – покликаний вирішити проблему комунікації ветеринарних клінік з клієнтами [29].

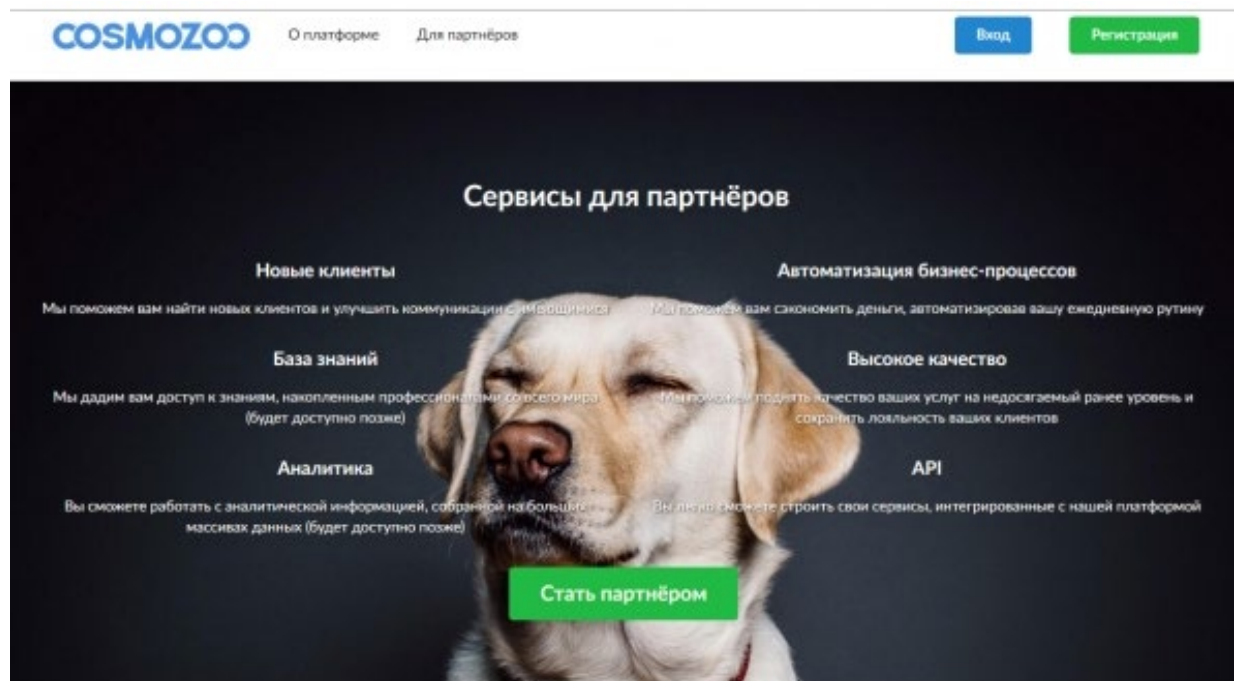


Рис. 1.1 – Головна сторінка онлайн сервісу CosmoZoo

Господарі домашніх тварин можуть зберігати в сервісі медичні карти вихованців, а також реєструватися на прийом в ветеринарні клініки. Можливості для ветеринарів та адміністраторів клінік включають обробку вхідних заявок в клініку (електронна черга), управління розкладом лікарів, реєстрацію користувачів і їх тварин, управління медичними картами (записи про процедурах і вакцинації).

VetDesk

Ексклюзивне авторське програмне забезпечення VetDesk для ветеринарних клінік (рис.1.2).

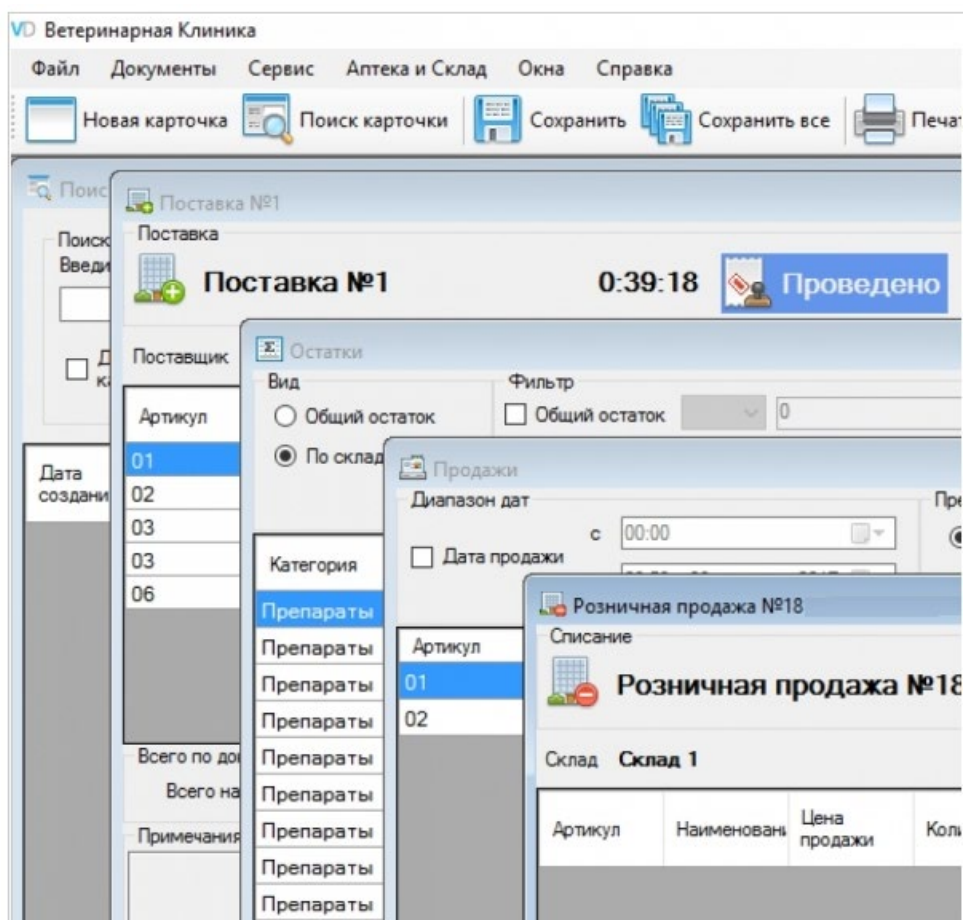


Рисунок 1.2 – Модуль розширення «VetDesk Аптека і Склад»

VetDesk вдає із себе настільний додаток, повністю автоматизує документообіг ветеринарної клініки [30].

Програмний продукт, що розробляється ветеринарами для ветеринарів. Єдиний програмний продукт, створюваний високо кваліфікованою командою розробників, під чуйним керівництвом самих користувачів - ветеринарних лікарів, «з нуля» і «для себе» у 2012 році. Програма дозволяє повністю відмовитися від паперової картотеки клієнтів і їх вихованців, так само як і від паперових бланків результатів аналізів, рекомендацій і призначень. Завдяки швидкому пошуку відкриття картки вихованця або знаходження потрібного прийому, аналізу або рахунку

займає лічені секунди навіть в тому випадку, якщо база даних накопичила суттєвий обсяг. Клієнти залишаються задоволені моментально одержуваними на руки документами, які можна не тільки надрукувати, а й надіслати електронною поштою. Автоматичні нагадування, ведення рахунків і статистика по ним допоможе керівнику підприємства бути завжди в курсі фінансових показників. Довідники, які можна заповнювати «на льоту» допоможуть швидко і правильно заповнити потрібні форми і не забути нічого істотного.

Система дозволяє вести облік власників тварин, надходження та витрачання ветеринарних препаратів, автоматично перевіряє і коригує залишки матеріалів перед додаванням в рахунку прийомів або перед продажом або списанням. Реалізована повна підтримка обліку товарів з термінами придатності, в тому числі і облік партій одного товару з різними термінами придатності. Повна підтримка сканерів штрих-кодів. Звіт про залишки дозволяє швидко і своєчасно знаходити товари, запас яких потрібно поповнити або списати. Звіт з продажу покаже результати роботи за будь-який часовий період.

VetDesk включає всі форми дослідження: «УЗД серця», підтримка зберігання і одночасного використання безлічі наборів норм аналізів крові (в залежності не тільки від виду тварини, а й від використовуваного обладнання), додані редаговані довідники для обладнання ультразвукових досліджень, редагований довідник типів подій для нагадувань. Програма постійно оновлюється, створюються нові модулі взаємодії.

Єдиним недолік цієї програми – всі модулі адаптовані під Російське законодавство.

Sofie

IBM спільно з компанією LifeLearn представили онлайн сервіс Sofie, побудований на платформі штучного інтелекту IBM Watson. У базі даних Sofie інформація про 1500 хвороб і сотні тисяч кейсів з історіями хвороби. На підставі цієї інформації Sofie видає рекомендацію по новому пацієнтові

[31]. Причому, вона працює як голосовий помічник. Ветеринар просто питає її (говорить в свій смартфон) і вона відповідає, або видає необхідну інформацію (рис. 1.3).

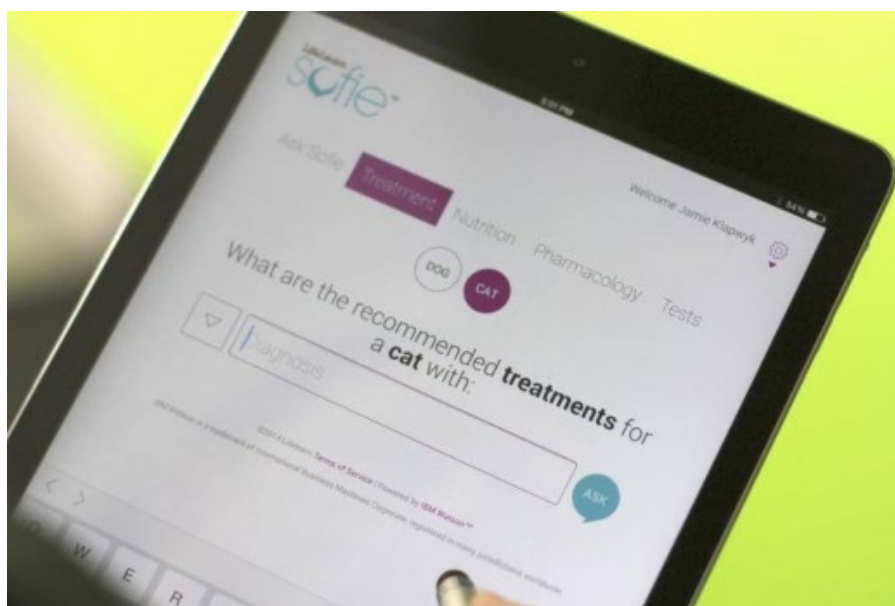


Рисунок 1.3 – Мобільний сервіс Sofie

Ветменеджер

Онлайн сервіс «Ветменеджер» для ветеринарних клінік, дозволяє вести медичні карти тварин-пацієнтів (в які можна заносити записи, результати опитувань, рентген знімки, УЗД, фотографії та відео, скановані документи), виставляти рахунки клієнтам за виконані послуги і придбані товари, планувати прийоми ветеринарів, відправляти SMS і E-mail розсилки клієнтам, вести облік товарів і матеріалів на складі (рис. 1.4).



Рисунок 1.4 – Медична карта пацієнта у сервісі «Ветменеджер»

Є можливість роботи з програмою на мобільних пристроях Android і iOS. Вартість сервісу починається від 120 грн. в місяць за користувача. Список інтеграцій постійно поповнюється. Є відкрите API і можна додати інтеграцію з іншими сервісами [28].

1.2. Загальні критерії автоматизації ветеринарної клініки «Авіцена»

З урахуванням аналізу існуючих програм автоматизації інформаційних систем ветеринарних клінік, враховуючи всі аспекти ветеринарії, створимо комплексну автоматизацію діяльності ветеринарної клініки «Авіцена», яка розташована на вулиці Маяковського 8, міста Білгород-Дністровський, Одеської області (зовнішній вигляд клініки рис. 1.5).



Рисунок 1.5 – Ветеринарна клініка «Авіцена»

Можна використовувати кілька різних програм, але ефективніше використовувати одну програму, яка буде відразу об'єднувати всі моменти, а також дозволить проводити аналіз про виконану роботу. Розроблена програма для ветеринарії, для управління і обліку ветеринарії здатна автоматизувати діяльність будь-якої клініки з лікування тварин, а також зможе охопити всі сторони в роботі організації.

«Авіцена» пропонує населенню такі послуги: вакцинації; хірургії; терапії; обстеження ультразвуком. Так само при клініці знаходиться аптека, яка здійснює продаж лікарських препаратів і кормів для тварин. Ветеринарна клініка «Авіцена» існує в нашому місті вже 7 років, за цей час організація отримала багато позитивних відгуків, з'явилися постійні клієнти, і її послуги залишаються затребуваними у населення.

У ветеринарній клініці «Авіцена» облік і контроль діяльності взаємодії з клієнтами організації не автоматизовано. Працівники та клієнти організації не мають можливості доступу до актуальної інформації про клініку, послуги, що надаються, можливості отримання консультації ветеринарного лікаря, наявності ліків і т. ін.

Для розвитку даної сфери послуг населенню, підвищення якості та оперативного обслуговування, необхідна автоматизація такої діяльності. Кожного, хто буде звертатися в клініку, можна зафіксувати в програмі, що здійснює управлінський автоматизований облік в ветеринарії. Для цього необхідно створити спеціальну базу клієнтів, де вказується кличка тварини, порода, ім'я господаря, контакти, а також будь-яка додаткова інформація, яка потрібна. У базі програми ветеринарії повинен здійснюватися швидкий пошук потрібного клієнта по перших символах. Такий пошук відбувається по кличці тварини, імені господаря, телефону, а також будь-якого іншого полю програми ветеринарії.

У програмі, яка веде автоматизований управлінський облік в ветеринарній клініці, існує книга запису до фахівців ветеринарії. У відведеному для цього модулі у вигляді списку відображається час роботи кожного фахівця, зайняті на прийом години, а також вільний для запису час. При запису вибирається клієнт з бази, або відразу додається новий. Далі вказується послуга ветеринарії, яка буде проводитися, або набір послуг. При записі до фахівця в програмі автоматизації та обліку «Управління ветеринарною клінікою» також є можливість показати, чи прийшов клієнт на прийом сам або його направив хтось із співробітників

або будь-яка стороння організація. Це робиться для того, щоб можна було призначити ставки винагороди за напрямок і відстежити за період найактивнішого направляючого, а також хто і скільки заробив.

Софт для ветеринарії та автоматизованим управлінням обліками становить список послуг, які надає клініка, послуги ветеринарії для зручності поділяються на категорії і підкатегорії. Кожна послуга може входити в той чи інший прайс-лист і мати різну вартість. На кожную послугу ветеринарії при бажанні складається собівартість, вказується, чого і скільки потрібно, підраховується сума. Програма для ветеринарних клінік з автоматизованими обліками містить можливість налаштування ставки за надання послуг фахівцями. Ставки можуть задаватися з різними величинами на різні категорії, підкатегорії або конкретні найменування послуг, а також можуть бути у вигляді відсотка від вартості послуги або просто мати фіксовану суму. Надалі в програмі ветеринарії за певний період можна за допомогою спеціального звіту побачити, хто і скільки заробив.

1.3. Висновки до розділу 1

В результаті проведеного аналізу порівняння функцій АІС існуючих ветеринарних організацій, з урахуванням висунутих вимог і побажань наявності необхідних для замовника модулів побудовано нову концепцію розробки програми для ветеринарної клініки «Авіцена» з автоматизованими обліками і управлінням, яка буде мати можливості ведення обліку з будь-яких матеріалів або препаратів, матиме набір різних звітів: аналітичних, фінансових, управлінських, складських. Кожен звіт програми ветеринарії матиме свої параметри для вибірки даних та майже всі звіти формуватимуться за будь-який обраний період. У замовника була необхідна умова створення настільної програми не залежної від інтернет - з'єднання (територіально клініка розташована в місцевості, де часто відключають електрику).

РОЗДІЛ 2

АНАЛІЗ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

2.1 Вибір мови програмування

Після проведеного моделювання системи, можна почати вибір засобів розробки програмної системи.

Почнемо з вибору мови програмування. Розглянемо найпопулярніші об'єктно-орієнтовані мови програмування (ООП), в яких парадигма ООП є основною концепцією поняття об'єктів і класів.

Java - це універсальна мова програмування, яка є одночасною, орієнтованою на класи, об'єктно-орієнтованою і спеціально розробленою з метою мінімізації залежностей від конкретної реалізації. Її основна мета - забезпечити розробникам можливість "писати один раз, запускати де завгодно" (WORA), тобто скомпільований код Java може працювати на будь-яких платформах, що підтримують Java, без необхідності повторної компіляції.

Наприклад, ви можете створити та скомпілювати програму Java на UNIX і запустити її на платформах Microsoft Windows, Macintosh або UNIX без будь-яких змін у вихідному коді. Цю універсальність досягається завдяки компіляції програми Java в проміжну мову, відому як байт-код. Формат байт-коду не залежить від конкретної платформи. Для виконання байт-коду на будь-якій платформі використовується віртуальна машина, відома як Java Virtual Machine (JVM).

Java була спроектована Джеймсом Гослінгом в компанії Sun Microsystems (яка пізніше була придбана корпорацією Oracle) і вперше випущена в 1995 році як ключовий компонент платформи Sun Microsystems Java. Мова має багато спільного з синтаксисом C та C++, але водночас обмежена меншими можливостями низького рівня порівняно з цими мовами.

Корпорація Oracle володіє офіційною реалізацією платформи Java SE з моменту придбання Sun Microsystems 27 січня 2010 року. Ця реалізація ґрунтується на початковій версії Java від Sun. Продукт Oracle доступний для операційних систем Microsoft Windows, Mac OS X, Linux та Solaris.

Реалізація Oracle має дві відокремлені версії:

- Середовище виконання Java (JRE), яке містить необхідні елементи платформи Java SE для запуску Java-програм для кінцевих користувачів.
- Java Development Kit (JDK), спрямований на розробників програмного забезпечення і включає в себе інструменти розробки, такі як компілятор Java, Javadoc, Jar та відладчик.

Мова програмування Java використовує автоматичний збирач сміття для управління пам'яттю в життєвому циклі об'єктів. Програміст визначає, коли створюються об'єкти, а виконавча система Java відповідає за відновлення пам'яті, коли об'єкти більше не використовуються. Автоматичне вивільнення недосяжної пам'яті відбувається сміттєзбірником.

Хоча виток пам'яті є малоймовірним, виникнення подібної ситуації можливе, якщо код програміста містить посилання на об'єкт, який вже не потрібен, особливо, коли об'єкти, які більше не використовуються, зберігаються в контейнерах, які все ще використовуються. У разі виклику методів для неіснуючого об'єкта виникає виняток "NullPointerException".

Збір сміття може відбутися в будь-який час. В ідеалі це відбудеться, коли програма не працює. Він гарантовано спрацює, якщо на купі недостатньо вільної пам'яті для виділення нового об'єкта; це може спричинити миттєву зупинку програми. Явне управління пам'яттю неможливо на Java (логотип середовища на рис. 2.1) [2].



Рис. 2.1 – Логотип середовища Java

C# – це сучасна об'єктно-орієнтована мова програмування, розроблена в 2000 році Андерсом Хейлсбергом в Microsoft як суперниця Java (на яку досить схожа). Вона була створена тому, що Sun (куплений пізніше Oracle) не хотів, щоб Microsoft вносила зміни в Java, тому Microsoft вирішила створити свою мову замість цієї. C# швидко зростала за популярністю з моменту її створення, завдяки широкій підтримці корпорації Майкрософт, яка допомогла їй отримати велику кількість платформ та підтримки; зараз це одна з найпопулярніших мов програмування у світі.

C# представляє собою універсальну мову програмування, розроблену для створення програм на платформі Microsoft, що вимагає наявності системи .NET у середовищі Windows. Зазвичай C# розглядається як гібрид, який використовує найкращі аспекти мов C та C++, створюючи справжньо-сучасну мову програмування. Навіть при тому, що фреймворк .NET підтримує різні мови програмування, C# швидко завоювала статус однієї з найпопулярніших.

C# може бути використана для створення різноманітних застосунків, але її основне використання спостерігається у розробці настільних додатків і ігор для операційної системи Windows. Зараз C# також набуває популярності в розробці веб-додатків і зростає попит на неї в сфері мобільних застосунків. Інструменти, такі як Xamarin, дозволяють додаткам, написаним на C#, працювати практично на будь-якому мобільному пристрої.

Широке використання C# також спостерігається в ігровій індустрії, зокрема, завдяки ігровому рушію Unity 3D. Unity став найпопулярнішим ігровим рушієм на сучасному ринку, і понад третина найкращих ігор створена саме на ньому. За допомогою Unity розроблено близько 770 мільйонів активних користувачів ігор. Цей рушій також використовується у віртуальній реальності, з 90% усіх ігор для Samsung Gear і 53% ігор для Oculus Rift VR, розроблених за допомогою Unity.

C# – дуже популярний інструмент для створення цих додатків, і тому є чудовим вибором для будь-якого програміста, який сподівається проникнути в індустрію розвитку ігор, або для всіх, хто цікавиться віртуальною реальністю.

C# пропонує ряд функцій, які сприяють легкості вивчення. Це високорівнева мова, відносно проста для читання, і багато складних завдань абстрагуються, щоб зменшити навантаження на програміста. Наприклад, управління пам'яттю відокремлене від користувача і автоматизоване за допомогою схеми збору сміття .NET.

Також важливо відзначити, що C# є статично набраною мовою, тому код перевіряється перед трансформацією в додаток. Це спрощує виявлення помилок, що може бути особливо корисним для новачків-програмістів.

Навіть якщо синтаксис C# є більш послідовним і логічним порівняно з C++, відзначається, що в навчанні все ж є свої виклики. C# визнається як складна мова, і освоєння її може займати більше часу, ніж в рази простіші мови, такі як Python. Це означає, що користувачам необхідно вивчати значну кількість коду для створення розширених програм, що може бути викликом для деяких початківців.

Завдяки своїй потужності, гнучкості та ефективній підтримці, C# швидко завоювала позицію однією з найбільш популярних мов програмування. На сьогоднішній день вона займає четверте місце серед найпопулярніших мов програмування, регулярно використовується близько 31% усіх розробників. Крім того, вона обіймає третю позицію за кількістю

учасників у спільноті StackOverflow (яка сама була розроблена з використанням C#), налічуючи понад 1,1 мільйона тем.

Ця висока популярність перекладається на активний ринок праці. Щомісяця по всьому світу розміщується понад 17 000 вакансій, пов'язаних із C#, із середньою річною зарплатою понад 72 000 доларів.

Якщо звужуватися лише до США, щомісяця розміщується понад 6 000 робочих місць із щорічною зарплатою в 92 000 доларів (логотип середовища C# на рис. 2.2) [32].



Рис. 2.2 – Логотип середовища C#

C++ – мова програмування, яка має імперативні та об'єктно-орієнтовані функції. Її також називають мовою програмування середнього рівня. Розроблена Bjarne Stroustrup в лабораторіях Bell з 1979 року. Вперше вона з'явилася у 1985 році. Вона складна, загального призначення, статичного типу, чутлива до регістру та вільної мови програмування. Підтримує процедурне, об'єктно-орієнтоване та загальне програмування, а також має наявність багатої стандартної бібліотеки з великим набором функцій, що управляють файлами та методами, які маніпулюють структурами даних тощо.

C++ є широко використовуваною серед програмістів та розробників, особливо у сфері розробки додатків. Вона включає важливі деталі, такі як ядро мови, що надає весь необхідний будівельний матеріал, такий як змінні, типи даних, літерали і т. д. Мова підтримує об'єктно-орієнтоване програмування, включаючи його ключові концепції, такі як успадкування, поліморфізм, інкапсуляція і абстракція. Ці концепції роблять мову C++

унікальною і в основному дозволяють легко та концептуально розробляти програми.

Існує кілька переваг використання C++ для розробки додатків, а завдяки її особливостям багато додатків, розроблених цією мовою, є безпечними у використанні. Нижче наводиться приклад найкращих застосувань на основі продукту C++.

Програми, засновані на графічному інтерфейсі, які використовуються користувачами, такі додатки як Adobe Photoshop та інші. Багато додатків систем Adobe розроблені в C++: Illustrator, Adobe та готові зображення, а розробники Adobe вважаються активними у спільноті C++.

– Ігри: Ця мова також використовується для розробки ігор. Вона перекриває складність 3D-ігор та підтримує багатокористувальницький варіант з мережею. Це допомагає оптимізувати ресурси. Використання мови програмування C++ дозволяє проводити процедурне програмування для інтенсивних функцій процесора та забезпечує контроль над обладнанням. Завдяки своїй високій швидкодії, C++ широко використовується у розробці різноманітних ігор або в ігрових двигунах. Основний акцент на використанні C++ спрямований на розробку інструментів для ігор.

Наприклад, у галузі анімації використовуються програми, розроблені за допомогою C++. Програмне забезпечення для 3D-анімації, моделювання та візуалізації визначається як потужний інструментарій. Це знаходить застосування в розробці у режимі реального часу, обробці зображень, мобільних сенсорних додатках та візуальних ефектах, де моделювання переважно здійснюється за допомогою C++. Мова також використовується для анімації, оточення, графіки руху, віртуальної реальності та створення персонажів. Віртуальні реальні пристрої стали найпопулярнішими в сучасному світі розваг.

– Веб-браузер: Мова використовується і для розробки браузера. C++ використовується при створенні Google Chrome та Інтернет-браузера Mozilla Firefox. Деякі програми браузера Chrome написані на C++, такі як файлова система, карта зменшення обробки даних великих кластерів. У Mozilla також є програма, написана на C++, це клієнт електронної пошти Mozilla Thunderbird. C++ є механізмом візуалізації проектів Google та Mozilla з відкритим кодом.

Робота з базами даних: Цю мову програмування також використовують для розробки програмного забезпечення, яке взаємодіє з базами даних або має відкритий вихідний код. Наприклад, MySQL є однією з найбільш популярних систем управління базами даних і широко використовується в організаціях та серед розробників. Це сприяє економії часу, коштів та облікових записів бізнес-систем, а також полегшує розповсюдження програмного забезпечення. Інші програми, які базуються на роботі з програмним забезпеченням баз даних, включають в себе такі популярні сервіси, як Wikipedia, Yahoo, YouTube тощо. Ще одним прикладом є RDBMS Bloomberg, який дозволяє інвесторам отримувати фінансову інформацію в режимі реального часу. Основною мовою розробки для нього є C++, завдяки чому доступ до баз даних забезпечується швидко та точно для постачання інформації з бізнесу та фінансів, а також новин з усього світу.

Робота з медіа: Мову програмування C++ також використовують для створення медіаплеєрів, управління відео- та аудіофайлами. Наприклад, відомий медіапрогравач Winamp Media, розроблений на C++, дозволяє насолоджуватися музикою, отримувати доступ та обмінюватися відео та аудіофайлами. Програвач має ряд функцій, таких як підтримка обкладинок, потокове відтворення аудіо та відео, а також забезпечує можливість прослуховування інтернет-радіостанцій.

– Компілятори: Більшість компіляторів написані виключно мовою C++. Компілятори, які використовуються для трансляції коду інших мов, таких як C#, Java та інші, переважно також реалізовані з використанням C++. Ця мова також широко застосовується при створенні самих зазначених мов програмування і володіє платформонезалежністю, що дозволяє створювати різноманітне програмне забезпечення.

– Операційні системи: C++ також використовується для розробки багатьох операційних систем, зокрема для платформ Microsoft та деяких складових операційної системи Apple. Мова використовується при розробці операційних систем, таких як Microsoft Windows 95, 98, 2000, XP, а також для створення продуктів, таких як Office, Internet Explorer та Visual Studio, а також мобільних операційних систем.

– Сканування: Програми для сканування фільмів чи камер, також часто розробляються мовою C++. Ця мова була використана при розробці технології PDF для друку, обміну, архівації та публікації документації.

– Інші області використання: C++ застосовується в медичних та інженерних програмах, а також в системах автоматизованого проектування. Програми, що схожі на програми для МРТ-сканування та САМ-системи, в основному використовуються в лікарнях, галузевому, державному та національному уряді, а також в інших галузях, таких як будівництво та видобуток. Розробники вважають C++ першочерговою мовою для роботи над різноманітними розробками.

C++ – це мова програмування, яка застосовується в різних галузях, але основними напрямками її використання є системне програмування та вбудовані системи. У вигляді системного програмування, C++ використовується для розробки операційних систем та драйверів, що взаємодіють з апаратним забезпеченням. В області вбудованих систем, ця мова застосовується в розробці вбудованих систем, таких як автомобілі, робототехніка та технічні пристрої.

C++ славиться своєю великою та активною спільнотою розробників, що робить його привабливим для новачків та забезпечує доступ до онлайн-рішень. Ця мова також визначається як одна з найбезпечніших завдяки своїм особливостям безпеки. Для багатьох розробників, особливо для початківців, C++ є вибором для вивчення основ програмування, оскільки вона пропонує концептуальний підхід, має простий синтаксис і дозволяє легко виправляти помилки. Багато програмістів вважають за краще спочатку вивчити C++, а потім переходити до інших мов програмування.

Але більшість розробників намагаються дотримуватися C++ лише через широке різноманіття використання та сумісність з декількома платформами та програмним забезпеченням (логотип середовища C++ на рис. 2.3) [33].



Рис. 2.3 – Логотип середовища C++

Після аналізу наведених вище мов програмування, для розробки програмного застосування було обрано мову програмування Java через такі її переваги:

- мультиплатформенність, що є вимогою до ПЗ;
- велика кількість бібліотек, що дозволить з легкістю розширювати функціонал додатку;
- велика популярність, що дозволить швидко і легко вирішити проблеми під час написання коду в більшості випадків.

2.2 Вибір середовища розробки

Eclipse, давно визнаний як найпопулярніше інтегроване середовище розробки (IDE) для Java, є безкоштовним із відкритим кодом. Це

середовище було розроблено переважно на Java, але його архітектура плагінів дозволяє розширювати його за допомогою інших мов програмування. Започаткований у 2001 році як проект компанії IBM з метою заміни родини IDE, заснованих на Smalltalk, таких як IBM Visual Age ID, Eclipse перейшов на платформу Java.

Портативність Java гарантує кросплатформенність Eclipse: ця IDE працює на Linux, Mac OS X, Solaris і Windows. Зовнішній вигляд Eclipse визначається, принаймні частково, інструментарієм Java Standard Widget (SWT). Ефективність Eclipse також залежить від використання віртуальної машини Java (JVM). Хоча раніше у Eclipse була репутація сповільненої роботи, пов'язаної зі старими апаратними ресурсами та JVM, сучасні версії все ще можуть виявляти деяку повільність, особливо під час фонових оновлень або при встановленні значної кількості плагінів.

Однією з частин, яка призводить до деякої затримки в роботі Eclipse, є вбудований інкрементальний компілятор, який запускається при завантаженні файлу та при кожному оновленні його коду. Це, проте, сприяє виведенню невеликої кількості помилок під час введення.

Незалежно від системи збирання, проект Eclipse Java також підтримує модель його вмісту, яка включає інформацію про ієрархію типів, посилання та декларації елементів Java. Це також добре врівноважує декілька помічників із редагування та дозволяє навігації мати вигляд контуру.

Екосистема плагінів є однією з сильних сторін Eclipse, але також може призводити до випадкових розладів. На сьогоднішній день ринок Eclipse налічує понад 1600 рішень, і плагіни, що розробляються спільнотою, можуть або не можуть працювати так, як рекламується. Проте плагіни Eclipse охоплюють підтримку понад 100 мов програмування та майже 200 фреймворків для розробки додатків.

Більшість серверів Java підтримується: при визначенні нового підключення до сервера в Eclipse ви зможете перейти до списку

постачальників, де вас очікує близько 30 серверів додатків, включаючи дев'ять версій Apache Tomcat. Комерційні постачальники, як правило, об'єднують свої пропозиції: наприклад, у програмі Red Hat JBoss Middleware є лише один пункт, який включає WildFly та сервери EAP Server, а також JBoss AS.

Перший досвід розробника з Eclipse може викликати неспокій та заплутаність. Це пов'язано з тим, що потрібно адаптуватись до концептуальної архітектури робочих просторів, перспектив та видів Eclipse, функції яких визначаються тим, які плагіни встановлені. Наприклад, для розробки сервера Java ймовірно використовується перспектива перегляду Java, Java FX та Java; перегляд пакета провідника; перспектива налагодження; командна перспектива синхронізації; веб-інструменти; перспектива розвитку бази даних; і перспектива налагодження бази даних.

У Eclipse часто існує декілька способів виконати задане завдання. Наприклад, можна переглядати код за допомогою програми провідника та/або перспективи перегляду Java; яку обираємо.

Підтримка пошуку Java дозволяє знаходити декларації, посилання та вхідження Java-пакетів, типів та методів. Також можна скористатися швидким доступом для пошуку та швидкого перегляду для спливання таких речей, як контури класу.

Додавання методів та генерування класів підтримується анотаціями про помилки та вмістом. З шаблонів коду можна генерувати загальні шаблони коду, а Eclipse може автоматично генерувати та упорядковувати заяви про імпорт. Рефакторинг Java в Eclipse підтримує 23 операції, починаючи від загальних операцій з перейменування і закінчуючи більш незрозумілими перетвореннями прямо з книги Мартіна Фаулера [34]. Рефакторинг може виконуватися не тільки в інтерактивному режимі, але і за допомогою сценаріїв рефакторингу.

Eclipse підтримує налагодження як локально, так і віддалено, припускаючи, що ви використовуєте JVM, який підтримує віддалене

налагодження. Налагодження досить стандартне: зазвичай встановлюються точки перерви, а потім переглядаються змінні на вкладці перспективи налагодження, звичайно, можна переглядати свій код і оцінювати вирази.

Eclipse має велике співтовариство та документацію для різного віку, монетезації та корисності. Можна виявити, що в документацію входять зображення, які не відповідають поточній версії програмного забезпечення, або натискання клавіш для вашої операційної системи відрізняються від тих, що викликаються в довідці. Це одна з найпоширеніших проблем з проектами з відкритим кодом: документація може затримувати програмне забезпечення на місяці чи навіть роки. Оскільки екосистема настільки велика, Eclipse має більшу частку, ніж частка документації. (логотип середовища Eclipse на рис. 2.4) [11].



Рис. 2.4 – Логотип середовища Eclipse

NetBeans Java IDE почав своє існування як студентський проект в Празькому університеті в 1996 році. У 1997 році він перетворився на комерційний продукт, був придбаний компанією Sun у 1999 році і випущений з відкритим вихідним кодом у 2000 році, подарований Apache як інкубаційний проект.

Мовний редактор NetBeans виявляє помилки під час введення тексту і допомагає користувачам за допомогою спливаючих документацій та автоматичного заповнення смарт-коду. NetBeans пропонує повний спектр інструментів рефакторингу, які дозволяють реструктурувати код, не порушуючи його цілісності, виконує аналіз вихідного коду і надає широкий набір підказок для швидкого виправлення або оптимізації коду.

Також в рамках NetBeans інтегрований інструмент для розробки графічних інтерфейсів Swing, раніше відомий як "Project Matisse".

Інструмент Inspect & Transform дозволяє проводити перевірки по всій кодовій базі, автоматично вносячи виправлення в код.

NetBeans має ефективну підтримку для інструментів будування проектів Maven та Ant, а також включає плагін для Gradle.

Налагоджувач Java у NetBeans відзначається своєю зручністю, навіть якщо він звичайний. Також доступний окремий візуальний відладчик, який дозволяє робити знімки графічного інтерфейсу та візуально вивчати додатки на JavaFX та Swing. Профайлер NetBeans вражає зрозумілістю як для відслідковування роботи процесора, так і для аналізу використання пам'яті, забезпечуючи ефективні інструменти для виявлення витоків пам'яті (логотип середовища NetBeans IDE на рис. 2.5) [27].

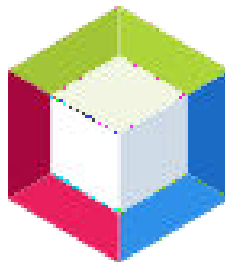


Рис. 2.5 – Логотип середовища NetBeans IDE

IntelliJ IDEA, яка є провідним інтегрованим середовищем розробки для Java як за функціональністю, так і за ціною, доступна у двох варіантах: безкоштовному виданні Community та платному виданні Ultimate, яке включає додаткові можливості.

Community Edition призначене для розробки на віртуальній машині Java (JVM) та для платформи Android. Воно підтримує такі мови, як Java, Kotlin, Groovy та Scala, а також платформи Android. Інструмент також

взаємодіє з іншими технологіями, такими як Maven, Gradle, SBT, а також системами контролю версій Git, SVN, Mercurial, CVS та TFS.

Ultimate Edition, спрямоване на розробку для Інтернету та підприємств, додатково підтримує систему контролю версій Perforce та інші, включає підтримку мов програмування JavaScript і TypeScript, а також фреймворків Java EE, Spring, GWT, Vaadin, Play, Grails та інших. Крім того, Ultimate включає інструменти для роботи з базами даних та підтримку мови SQL.

Комерційна версія Ultimate виправдовує своє вартість збільшенням продуктивності розробника на робочому столі професіонала. Ціни поступово знижуються для підприємств та стартапів і стають доступними для студентів, викладачів, чемпіонів Java та розробників з відкритим кодом.

IntelliJ IDEA надає зручний інтерфейс, забарвлення синтаксису та швидке автозавершення коду для Java, Kotlin та інших мов. Вона пропонує розширені можливості завершення коду, такі як "завершення ланцюга", яке враховує вашу особисту частоту використання символів. IntelliJ IDEA також взаємодіє з статичними членами та константами, автоматично додаючи необхідні заяви про імпорт, та намагається передбачити тип символу та вносити необхідні корекції в код.

Код на Java часто містить фрагменти інших мов у вигляді рядків. IntelliJ IDEA може вставляти у літературний рядок Java String фрагменти SQL, XPath, HTML, CSS або коду JavaScript. У зв'язку з цим, вона може проводити рефакторинг коду на кількох мовах, наприклад, при перейменуванні класу у операторі JPA, IDEA автоматично оновить відповідні вирази сутності та JPA. Рефакторинг фрагмента коду також охоплює всі його дублікати.

IntelliJ IDEA аналізує ваш код як під час завантаження, так і під час введення. Інструмент пропонує інспекції, які вказують на можливі проблеми.

Дизайн відладчика в IDEA є особливо приємним. Значення змінних відображаються безпосередньо поруч із відповідним вихідним кодом у вікні редактора. Змінюючи стан змінної, змінюється й колір її виділення.

IntelliJ IDEA пропонує єдиний інтерфейс для багатьох систем управління версіями, таких як Git, SVN, Mercurial, CVS, Perforce та TFS, дозволяючи виконувати всі керування змінами прямо в середовищі розробки.

IDEA інтегрує інструменти для збирання проекту, тестові бігуни та інструменти для визначення покриття коду, а також включає вбудоване вікно терміналу. Хоча IntelliJ IDEA не має свого власного профайлера, вона підтримує кілька сторонніх профілів через відповідні плагіни, такі як YourKit і VisualVM, остання з яких є варіантом профайлера NetBeans.

Розробка серверного застосунку на Java часто включає в себе взаємодію з базами даних, і тому IDEA Ultimate обладнана інструментами для роботи з базами даних SQL та NoSQL. IntelliJ IDEA підтримує всі основні сервери додатків JVM і може розгортати їх на серверах, а також налагоджувати, вирішуючи основні проблеми для розробників Enterprise Java. IDEA також включає підтримку Docker через спеціальний плагін, який додає вікно інструмента Docker.

Підтримка кодування в IDEA розширена для таких фреймворків, як Spring, Java EE, Grails, Play, Android, GWT, Vaadin, Thymeleaf, React, AngularJS та інших. Крім того, окрім Java, IntelliJ IDEA розуміє і підтримує багато інших мов програмування, включаючи Groovy, Kotlin, Scala, JavaScript, TypeScript та SQL. Якщо вам потрібно розширити функціональність, наразі існують сотні плагінів для різних мов IntelliJ, включаючи R, Elm, Go, Rust та D (логотип середовища IntelliJ IDEA представлено на рис. 2.6) [4].



Рис. 2.6 – Логотип середовища IntelliJ IDEA

Проаналізувавши середовища розробки, вибір було зупинено на Eclipse, так як саме це середовище розробки має велику кількість модулів, що дозволить з легкістю застосувати їх при розробці самого додатку.

2.3 Вибір бібліотеки графічного інтерфейсу

Swing – це інструментарій для віджетів GUI для Java та частиною Oracle «Java Foundation Classes (JFC)» – з API (Application programming interface) для надання графічного інтерфейсу користувача (GUI) для програм Java.

Swing було розроблено з метою забезпечення більш розширеного набору графічних компонентів у порівнянні з Інструментарієм абстрактних вікон (AWT), що існував перед ним. Основна ідея Swing полягала в створенні зовнішнього вигляду, який емулює зовнішність різних платформ, щоб програми виглядали однаково, незалежно від операційної системи. У порівнянні з AWT, Swing має потужніші та більш гнучкі компоненти. Окрім стандартних елементів, таких як кнопки, прапорці та мітки, Swing також пропонує вдосконалені компоненти, такі як панель з вкладками, панелі прокрутки, дерева, таблиці та списки.

Компоненти Swing, на відміну від AWT, не є реалізованими кодом, залежним від платформи. Вони повністю написані на мові Java, що робить їх незалежними від платформи. Термін "легкий" використовується для характеристики цих компонентів [26].

Internet Foundation Classes (IFC) була графічною бібліотекою для Java, розробленою спочатку Netscape Communications Corporation. 16 грудня 1996 року Sun Microsystems та Netscape Communications Corporation оголосили про намір включити IFC разом із іншими технологіями у створення Java

Foundation Classes. "Класи фундації Java" потім були перейменовані в "Swing".

Swing вніс механізм, який дозволяє змінювати зовнішній вигляд кожного компонента програми без зміни її коду. Введення підтримки для підключеного зовнішнього вигляду дозволяло компонентам Swing імітувати зовнішність нативних компонентів, зберігаючи при цьому переваги незалежності від платформи. Спочатку поширювався як окрема бібліотека, що завантажується окремо, Swing був включений до стандартної версії Java починаючи з випуску 1.2. Класи Swing та компоненти містяться в ієрархії пакетів `javax.swing`.

Swing представляє собою графічний інтерфейс GUI для «модель-перегляд-контролер» в Java, який працює за моделлю програмування з одним потоком. Цей фреймворк також надає абстракційний шар між структурою коду та графічною презентацією GUI на основі Swing.

Swing є незалежним від платформи, оскільки весь його код написаний на Java. Документацію для всіх класів Swing можна знайти в посібнику Java API для версії 6 або специфікації API платформи Java Platform Standard Edition 8 для версії 8 [1].

Архітектура Swing є високомодульною, що дозволяє "приєднувати" різні спеціалізовані реалізації рамкових інтерфейсів. Користувачі можуть надавати власну реалізацію цих компонентів для заміни виконання за замовчуванням за допомогою механізму успадкування в Java.

Swing базується на компонентній архітектурі, де компоненти походять від класу `javax.swing.JComponent`. Ці об'єкти асинхронно обробляють події, мають пов'язані властивості та реагують на документовані набір методів, що є характерними для компонентів. Компоненти Swing відповідають специфікаціям архітектури компонентів Java Beans.

Велика залежність Swing від механізмів виконання та непрямих моделей композиції дозволяє йому динамічно реагувати на фундаментальні

зміни в налаштуваннях під час виконання. Наприклад, додаток на основі Swing може адаптувати свій інтерфейс користувача прямо під час його роботи. Без програмних змін в коді програми користувачі можуть також надавати власний дизайн та реалізацію, що дозволяє їм рівномірно змінювати зовнішній вигляд існуючих додатків Swing.

Високий рівень гнучкості Swing виявляється у його здатності ефективно перекривати управління графічним інтерфейсом (GUI) з власною операційною системою (ОС) для відображення. Swing "малює" свої елементи керування, використовуючи API 2D Java, і не викликає стандартний набір інструментів користувацького інтерфейсу ОС. Таким чином, компонент Swing не обмежений вбудованим інтерфейсом ОС і може візуалізувати себе будь-яким можливим способом за допомогою базових графічних інтерфейсів.

Все ж, незважаючи на свою автономність, кожен компонент Swing все ще покладається на AWT контейнер, оскільки JComponent Swing розширює контейнер AWT. Це дозволяє Swing інтегруватися з фреймворками управління графічним інтерфейсом ОС, такими як вирішальне відображення пристрою/екрана та взаємодія з користувачем, така як натискання клавіш або рухи миші. Swing просто адаптує свою семантику (для незалежності від ОС) до базових (для ОС) компонентів. Наприклад, кожен компонент Swing малює своє зображення на графічному пристрої відповідно до виклику `.paint ()`, визначеного в AWT Container. Але на відміну від компонентів AWT, які передавали створення своїх зображень своїй власній ОС "важкої ваги", компоненти Swing відповідають за свою власну візуалізацію.

Цей перенос та вирішення не обмежуються лише візуальним аспектом, але також розповсюджуються на управління Swing та використання власної незалежної семантики операційної системи для подій, що виникають в межах її ієрархії обмежень компонентів. Загалом, архітектура Swing делегує завдання відображення різноманітних аспектів

семантики GUI операційних систем на просту, але узагальнену схему для контейнера AWT. Використовуючи цю узагальнену платформу, Swing формує власну багату та складну семантику GUI у формі JComponent моделі.

Бібліотека Swing активно використовує шаблон проектування Model/View/Controller (Модель/Вид/Контролер), який концептуально розділяє дані, які відображаються, від управління користувацьким інтерфейсом, через який ці дані переглядаються. З цього принципу більшість компонентів Swing мають асоційовані моделі (визначені з точки зору інтерфейсів Java), і розробники можуть використовувати різні реалізації за замовчуванням або надавати свої власні. Фреймворк забезпечує реалізацію інтерфейсів моделі за замовчуванням для всіх своїх конкретних компонентів.

Зазвичай, для звичайного використання фреймворку Swing не потрібно створювати користувацькі моделі, оскільки фреймворк має набір реалізацій за замовчуванням, які автоматично асоціюються з відповідними JComponent-дочірніми класами у бібліотеці Swing. Взагалі, лише для складних компонентів, таких як таблиці, дерева та іноді списки, може знадобитися реалізація спеціальної моделі для обробки структур даних, пов'язаних із додатками. Зазвичай об'єкти моделей Swing відповідають за надання зручного інтерфейсу, що визначає запуснені події та доступні властивості (концептуальної) моделі даних для використання асоційованим JComponent. У загальному розумінні, модель у рамках концепції MVC є слабо пов'язаною схемою взаємодії об'єктів, і вона забезпечує програмістам можливість приєднувати слухачів подій до об'єкта моделі даних. Зазвичай ці події орієнтовані на модель (наприклад, подія «вставлений рядок» у модель таблиці) і відображаються як значущі події для компонента GUI через спеціалізацію JComponent.

Наприклад, у JTable існує модель, що називається TableModel, яка визначає інтерфейс для того, як таблиця отримує доступ до табличних

даних. Реалізація цієї моделі за замовчуванням працює з двовимірним масивом.

Компонент перегляду Swing, або JComponent, є об'єктом, який використовується для графічного представлення концептуального управління графічним інтерфейсом. Однією з особливостей Swing як рамки графічного інтерфейсу є його залежність від програмно керованих графічних інтерфейсів управління, на відміну від використання власних елементів управління GUI операційних систем. До Java 6 Update 10 ця особливість становила виклик при змішуванні елементів управління AWT, які використовують нативні елементи управління, з елементами управління Swing у графічному інтерфейсі.

У кінці кінців, щодо візуальної структури та управління, Swing віддає перевагу відносним макетам (які визначають позиційні відносини між компонентами), на відміну від абсолютних макетів (які фіксують точне розташування та розмір компонентів). Цей підхід до "рідкого" візуального компоновання пояснюється його витоками в операційному середовищі аплетів та впливом на дизайн та розробку оригінального інструментарію Java GUI. Концептуально цей підхід до управління компонованням схожий на той, який використовується для вмісту HTML у браузерях, вирішуючи ті ж проблеми, що були присутні в AWT (див. логотип середовища Swing на рис. 2.7) [2].



Рис. 2.7 – Логотип середовища Swing

JavaFX дозволяє створювати додатки з багатою насиченою графікою завдяки використанню апаратного прискорення графіки та можливостям GPU. *JavaFX* дозволяє розробляти програми для різних операційних систем: Windows, MacOS, Linux, а також для різноманітних пристроїв, таких як десктопи, смартфони, планшети, вбудовані пристрої та ТВ. Додаток на *JavaFX* буде працювати у будь-якому місці, де встановлено виконавче середовище Java (JRE). *JavaFX* пропонує значні можливості порівняно з іншими подібними платформами, включаючи обширний набір елементів управління, функціонал для роботи з мультимедіа, двомірною і тривимірною графікою, можливість опису інтерфейсу декларативним методом за допомогою мови розмітки FXML, стилізацію інтерфейсу за допомогою CSS, інтеграцію з Swing та інші застосування.

Історія *JavaFX* фактично почалася в першій половині 2000-х років, коли Кріс Олівер, розробник, що працював у компанії SeeBeyond, створив нову мову F3 (Forms Follow Functions) для розробки графічних інтерфейсів. У 2005 році SeeBeyond була придбана Sun Microsystems, яка тоді активно розвивала мову Java, що стала придбаною пізніше компанією Oracle. F3 був перейменований в *JavaFX*, і Кріс Олівер продовжив роботу над новою платформою під патронатом Sun. У травні 2007 року Sun Microsystems анонсувала нову платформу для графічних додатків, а 4 грудня 2008 року вийшов *JavaFX 1.0 SDK*.

Після придбання Sun Microsystems компанією Oracle в 2010 році була оголошена версія *JavaFX 2.0*, яка вийшла в 2011 році. У першій версії *JavaFX* була скриптова мова, але вдруге версії підхід був повністю змінений. Скриптову мову видалено, і платформа була переписана фактично з нуля. Тепер для розробки додатків можна було використовувати будь-яку мову, яка підтримує JVM. Були введені нові API, інтеграція зі Swing та інші значущі оновлення.

Ключовими пунктами розвитку платформи стали *JavaFX 8* та особливо *JavaFX 9*, яка вийшла у вересні 2017 року, одночасно з Java 9 і

ввела модульність у платформу. Отже, JavaFX, яка раніше постачалася разом з Java SE, тепер виступає окремим модулем, відокремленим від основної функціональності Java SE. Остання версія фреймворку - JavaFX 17 – вийшла у вересні 2021 року.

На даний момент JavaFX називають одним з найбільших зручних способів створення графічних додатків за допомогою мови Java, на відміну від AWT. Також варто відзначити, що для роботи з JavaFX замість Java теоретично можна використовувати будь-яку мову програмування, яка підтримується JVM (Рис. 2.8). [35]



Рис. 2.8 – Логотип середовища JavaFX

2.4. Висновки до розділу 2

Проаналізувавши сучасні технології програмування та середовища розробки, для магістерського дослідження було обрано мову програмування Java через такі її переваги:

- мультіплатформеність, що є вимогою до ПЗ;
- велика кількість бібліотек, що дозволить з легкістю розширювати функціонал додатку;
- велика популярність та програмна підтримка, що дозволить швидко і легко вирішити проблеми під час написання коду в більшості випадків.

Для розробки програмного застосування обрано середовище розробки Eclipse, що має велику кількість модулів та дозволяє застосувати їх при програмуванні. Як інструмент створення графічного інтерфейсу

користувача, використано Swing за його простоту в багатьох аспектах та легкість при розгортанні.

РОЗДІЛ 3

ОПИС РЕАЛІЗАЦІЇ ТА ФУНКЦІОНАЛУ СИСТЕМИ «АВІЦЕНА»

3.1 Опис реалізації та застосування ООП підходу

Усі програми Java використовують об'єкти, тип об'єктів визначається їх класом або інтерфейсом. «Клас - це сукупність полів, що містять значення та методи, які оперують цими значеннями. Класи є найбільш фундаментальним структурним елементом усіх програм Java. Не можливо писати код Java без визначення певних класів. Усі оператори Java відображаються в методах, а всі методи реалізовані в класах» [21]. Існує декілька об'єктно-орієнтованих концепцій, що використовуються при реалізації системи керування ветеринарної клініки, деякі з основних концепцій - абстракція, інкапсуляція, успадкування та поліморфізм. Розглянемо реалізацію цих концепцій у розроблюваній системі «Авіцена».

3.1.1 Абстракція

Абстракція - це приховування та ігнорування певних деталей об'єкта і виділення лише важливих рис та функцій. Об'єкт - це екземпляр класу. В реалізації системи існує загалом 15 класів, що показано в UML діаграмі класів (рис 3.1).

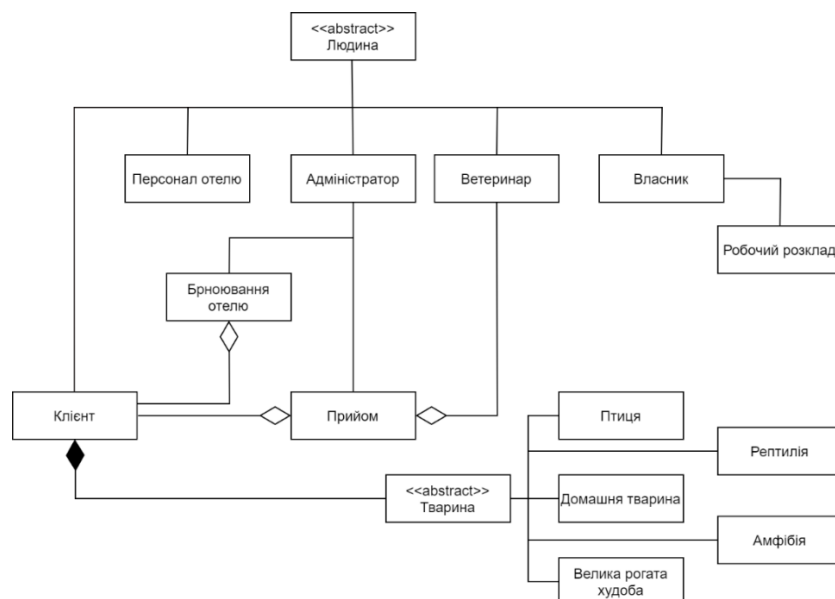


Рис. 3.1 – Діаграма класів

Ці класи є дуже хорошим прикладом абстракції, оскільки кожен клас містить поля та методи (які є важливими рисами об'єкта, який вони представляють), що описують його об'єкти.

На рисунку 3.2 показано один із класів у системі «Авіцена», який має всі поля, що описують важливі риси людини-користувача. Людина-користувач у цьому випадку представлений набором важливої інформації: ідентифікатором, ім'ям, ім'ям по батькові, прізвищем, паролем, номером телефону, адресою електронної пошти та домашньою адресою.

```

public abstract class Human implements Serializable{
    protected String ID;
    protected String firstName;
    protected String middleName;
    protected String lastName;
    protected String password;
    protected String tel;
    protected String email;
    protected String address;

    public Human() {

    }

    //Generate Staff ID
    public abstract String generateID();

    //Save generated Staff ID
    public abstract void saveGeneratedID();
  
```

Рис 3.2 – Приклад абстракції в класі Human

3.1.2 Інкапсуляція

Класи та їх об'єкти інкапсулюють атрибути та методи, де ті самі атрибути та методи тісно пов'язані. Інкапсуляція - це концепція, коли класи приховують свою інформацію від інших класів, з якими вони спілкуються та взаємодіють. Деталі реалізації об'єкта приховані в самих об'єктах, оскільки їм зазвичай не дозволяється знати, як реалізовані інші об'єкти [20].

На рисунку 3.3 наведено приклад інкапсуляції класу «Appointment» (Прийом) всередині системи. Атрибути класу встановлюються приватними, і єдиним способом змінити і прочитати значення атрибутів є виклик методів `get` і `set`. Таким чином атрибути класу інкапсульовані методами. Методи діють на бар'єр, який охоплює поля всередині класу. Крім цього, атрибут `ID` призначений лише для читання. Для нього не існує встановленого методу, оскільки ідентифікатор є унікальним і постійним для кожного призначеного прийому. Отже, це підкреслює одну з сильних сторін інкапсуляції, коли інші класи або користувачі не мають можливості змінити ідентифікатор зустрічі.

```
public class Appointment implements Serializable {

    private String ID;
    private Customer customer;
    private String petID;
    private Calendar appointmentTime;
    private String appointmentHour;
    private Vet vet;

    public String getID() {
        return this.ID;
    }

    public String getPetID() {
        return petID;
    }

    public void setPetID(String petID) {
        this.petID = petID;
    }

    public Calendar getAppointmentTime() {
        return appointmentTime;
    }

    public void setAppointmentTime(Calendar appointmentTime) {
        this.appointmentTime = appointmentTime;
    }
}
```

Рис 3.3 – Приклад інкапсуляції в класі «Appointment».

3.1.3 Успадкування

Успадкування - механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та функціональність батьківського класу переходять до класу нащадка (дочірнього). Успадкування економить час, дозволяє ефективно впроваджувати та підтримувати програму. Існуючий клас, який новий клас успадковує, називається суперкласом, де сам новий клас називається підкласом. Підклас може додавати поля та методи, яких суперклас не має, отже, він описує більш конкретну абстракцію, ніж його суперклас. Це ще називають спеціалізацією [20].

В системі «Авіцена» існує клас, а також клас Домашніх тварин, який є суперкласом. Під класом «Human» (Людина) існує 5 підкласів, що поширюють його, - це класи «Customer» (Клієнт), «Administrator», «BoardingStuff» (Персонал готелю), «Vet»(Ветеринар) та «Owner»(Власник). Для класу домашніх тварин існують класи земноводних, рогатої худоби, птахів, домашніх тварин та плазунів. Ці підкласи є більш конкретними, ніж їх суперклас, де вони успадковують усі атрибути та методи від свого суперкласу.

На рисунку 3.4 наведено приклад підкласу «Customer», який походить від суперкласу «Human». Перший рядок показує, що «Customer» має стосунки «є» з класом Людина. Як результат, набагато більше об'єктно-орієнтованої концепції можна досягти за допомогою успадкування, наприклад, поліморфізму.

```

public class Customer extends Human implements Serializable {

    private Pet[] pet = new Pet[10];
    private int numberOfPet = 1;

    public Customer() {

    }
  }

```

Рис. 3.4 – Приклад спадкування в класі «Customer»

3.1.4 Поліморфізм

Поліморфізм - це здатність об'єкта приймати різні форми. Поліморфна поведінка може відбуватися лише між підкласом та його суперкласом. «Поліморфізм означає, що фактичний тип об'єкта, що бере участь у виклику методу, визначає, який метод буде викликаний, а не тип змінної, що використовується для зберігання посилання на об'єкт» [22].

У системі «Авіцена» поліморфізм використовується в класі «Pet» та його підкласах. На рисунку 3.5, рядок 18 ілюструє змінну масиву типу «Pet», яка називається «pet».

```

16 public class Customer extends Human implements Serializable {
17
18     private Pet[] pet = new Pet[10];
19     private int numberOfPet = 1;
20
21     public Customer() {
22
23     }
24
25     public Customer(String petType, String species, String petName) {
26
27         if (petType.equalsIgnoreCase("Амфібія")) {
28             this.pet[0] = new Amphibian(petType, species, petName);
29
30         } else if (petType.equalsIgnoreCase("Птиця")) {
31             this.pet[0] = new Bird(petType, species, petName);
32
33         } else if (petType.equalsIgnoreCase("Ропата худоба")) {
34             this.pet[0] = new Fish(petType, species, petName);
35
36         } else if (petType.equalsIgnoreCase("Домашня тварина")) {
37             this.pet[0] = new HouseholdPet(petType, species, petName);
38
39         } else if (petType.equalsIgnoreCase("Рептилія")) {
40             this.pet[0] = new Reptile(petType, species, petName);
41
42         }
43     }

```

Рис. 3.5 – Приклад поліморфізму

У рядках 28, 31, 34, 37 і 40 показано, що змінна типу «Pet» здатна зберігати будь-які об'єкти, створені з підкласів класу «Pet». Перевагою цього є те економія часу та строк коду, дозволяючи класу «Customer» оголошувати лише одну змінну типу «Pet» замість того, щоб оголошувати 5 різних типів змінних. Крім того, він також забезпечує гнучкість, оскільки

дозволяє будь-якому методу, який повинен брати аргумент «Pet», писати лише один раз, замість того, щоб перевантажувати той самий метод, щоб вмістити всі 5 підкласів «Pet».

3.2 Опис функціоналу системи

3.2.1 Сторінка входу

Користувачі повинні вибрати свою посаду та ввести свій «ID користувача» як ім'я користувача та свій пароль, щоб перейти на відповідні сторінки (рис. 3.6).

Рис.3.6 – Сторінка входу

3.2.2 Сторінка адміністратора

Адміністратор може створити профіль клієнта та домашньої тварини на першій сторінці (рис. 3.7).

ID	Ім'я	Ім'я по бать...	Фамілія	Телефон	E-mail	Адреса	К. тварин	ID тварин	Тип тварини	Вид тварини	Ім'я тварини
C001	Микола	Олександрович	Птушкін	095368542	ptus@gmail.com	м.Білогірськ-Д...	1	P001	Пітбул	Пітбул Кранера	Альберт
C002	Олена	Максимівна	Колесник	0663215869	om.kolesnik@gmail.com	м.Білогірськ-Д...	2	P002,P003	Донація тва...	Кіт,Черепашка	Мураш,Жора
C003	Наталія	Олександрівна	Мілованова	0663215869	nlo_d@gmail.com	с.Хотів	2	P004,P005	Рогата худоб...	Голландська ...	Мілка,Ренс

Рис. 3.7 – Головна сторінка адміністратора

Для редагування профілю конкретного клієнта адміністратор повинен двічі клацнути на цього клієнта в таблиці на сторінці. Після чого з'явиться сторінка, що дозволяє редагувати профіль клієнта та домашньої тварини (рис. 3.8)

Редагування даних про клієнта та домашніх тварин

Редагувати клієнта Редагувати тварину

ID: C003

Ім'я: Наталія

Ім'я по батькові: Олексівна

Фамілія: Мілованова

Телефон: 0663215869

E-mail адреса: milo_n@gmail.com

Домашня адреса: с.Хотів

Редагувати профіль клієнта

Рис. 3.8 – Сторінка редагування даних профілю

Адміністратор також може назначити прийом та замовити місце для конкретного клієнта після його вибору, а потім натиснути кнопку «Назначити прийом» (рис. 3.9).

Сторінка реєстрації

Призначити прийом Забронювати проживання Редагувати прийом Відрегувати проживання

ID тварини: P004

Дата прийому: 12/05/2023

ID вет.: VET001

Час прийому: 0700

Додати

Назначені прийоми :

ID прийому	Дата прийому	Час прийому	ID тварини	ID ветеринара	Ім'я ветеринара
AP001	ср 12/05/2023	0900	P001	VET001	Василь Іванович
AP002	ср 12/05/2023	1000	P002	VET002	Іван Федорович
AP003	ср 12/05/2023	1100	P003	VET001	Василь Іванович
AP004	чт 13/05/2023	0900	P004	VET004	Геннадій Михай...
AP005	чт 13/05/2023	1000	P005	VET004	Геннадій Михай...

Розклад на поточну неділю :

Понеділок	Вівторок	Середа	Четвер	П'ятниця	Субота	Неділя
VET001	VET001	VET004	VET004	VET001	VET004	VET004
VET002	VET002	VET005	VET005	VET002	VET005	VET005
VET003	VET003	VET006	VET006	VET003	VET006	VET006

Інфо про ветеринара :

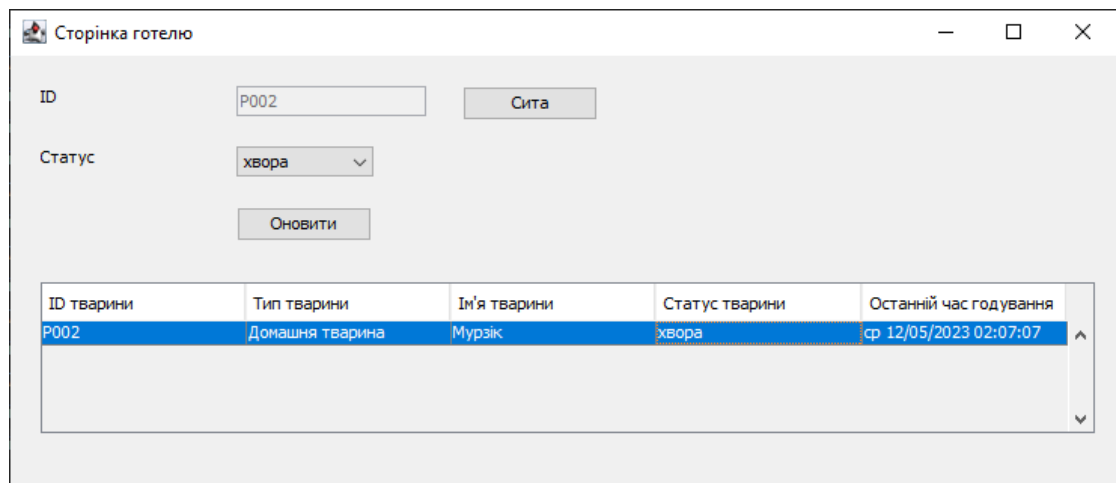
ID	Ім'я	Експертиза 1	Експертиза 2
VET001	Василь Іванович	Амфібія	Рептилія
VET002	Іван Федорович	Птиця	Рогата худоба
VET003	Іван Миколайович	Домашня тварина	Птиця

Рис. 3.9 – Сторінка призначення прийому

Адміністратору дозволено редагувати будь-яку послугу огляду або поселення на вкладках «Редагувати прийом» та «Редагувати бронювання» відповідно.

3.2.3 Сторінка персоналу готелю

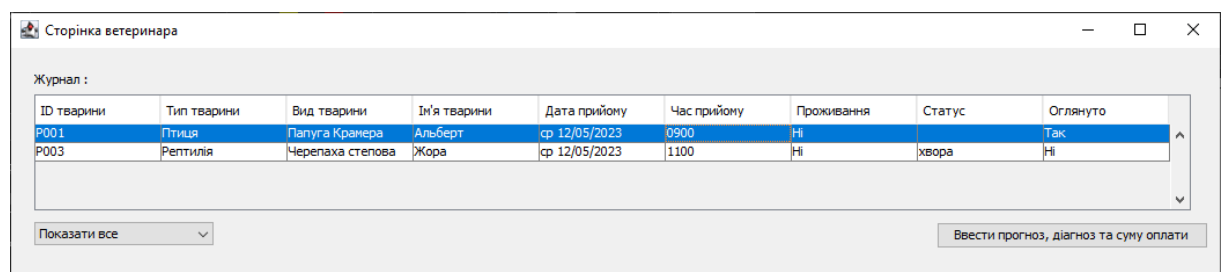
Персонал може обрати тварину з відповідної таблиці, щоб оновити її статус та час останнього годування натисканням кнопки «Сита» (рис. 3.10).



ID тварини	Тип тварини	Ім'я тварини	Статус тварини	Останній час годування
P002	Домашня тварина	Мурзік	хвора	сп 12/05/2023 02:07:07

Рис. 3.10 – Сторінка персоналу готелю

Сторінка ветеринара відображає журнал для кожного ветеринара. Ветеринар може перемикає журнал ветеринара, щоб показати всі прийоми або показати прийоми, призначені на поточний день (рис. 3.11).



ID тварини	Тип тварини	Вид тварини	Ім'я тварини	Дата прийому	Час прийому	Проживання	Статус	Оглянуто
P001	Птиця	Папуга Кранера	Альберт	сп 12/05/2023	0900	Ні		Так
P003	Рептилія	Черепаха степова	Жора	сп 12/05/2023	1100	Ні	хвора	Ні

Рис. 3.11 – Сторінка ветеринара

Крім цього, ветеринар може ввести прогноз, діагностику та зафіксувати оплату щодо будь-якої тварини. Натиснувши кнопку внизу

праворуч, з'явиться інша сторінка із текстовою областю, що дозволить ветеринару вводити свої висновки (рис. 3.12).

Прогноз, Діагноз і Оплата

Прогноз :

Діагноз :

При дотриманні режиму та доз вказаних в рецепті - оздоров

Авітаміноз

Показати попередні результати діагностики

Оплата : 0 грн

Зберегти прогноз и діагноз

Зберегти оплату

Рис. 3.12 – Сторінка прогнозу, діагнозу і оплати

3.2.4 Сторінка власника

Перша вкладка сторінки власника дозволяє керівникові закладу додавати або видаляти будь-якого працівника в системі (рис. 3.13).

Сторінка власника

Новий співробітник Новий робочий розклад Зетти

Посада : ☐ Адміністратор ☐ Персонал готелю ☒ Ветеринар

ID: VET003

Ім'я: Іван

Ім'я по батькові: Миколайович

Пароль:

Фамілія: Мороз

Телефон: 0501235468

E-mail адреса: example@gmail.com

Домашня адреса: м.Київ

Область знань:

ID	Ім'я	Ім'я по батькові	Фамілія	Телефон	E-mail	Адреса	Область знань 1	Область знань 2
VET001	Василь	Іванович	Петренко	31316516	example@gmail.c...	м.Київ	Амфібія	Рептилія
VET002	Іван	Федорович	Сренко	0508756423	example@gmail.c...	м.Київ	Птиця	Рогата худоба
VET003	Іван	Миколайович	Мороз	0501235468	example@gmail.c...	м.Київ	Домашня тварина	Птиця
VET004	Геннадій	Миколайович	Копилов	0672225468	example@gmail.c...	м.Київ	Домашня тварина	Рогата худоба
VET005	Михайло	Олександрович	Фісєнко	0661113468	example@gmail.c...	м.Київ	Домашня тварина	Рогата худоба
VET006	Артур	Олександрович	Фірташ	0661113468	example@gmail.c...	м.Київ	Рогата худоба	Амфібія
VET007	Олексій	Олександрович	Санойлов	0661113468	example@gmail.c...	м.Київ	Рептилія	Птиця

Рис. 3.13 – Сторінка власника (співробітники)

Друга вкладка дозволяє керівнику закладу створити нову робочу ротацию для ветеринарів (рис. 3.14).

Рис. 3.14 – Сторінка власника (розклад)

Керівник закладу також може переглянути звіт ветеринара та звіт про домашніх тварин, для аналізу та фіксації стану справ (рис. 3.15).

Рис. 3.15 – Сторінка власника (звіти)

3.3 Висновки до розділу 3

В процесі створення АІС «Авіцена» всі вимоги до системи були виконані. Охоплено всі функції, що повинні виконувати чотири види робіт ветеринарної клініки «Авіцена», якими є адміністратор, персонал готелю,

ветеринар та власник. Крім того, система «Авіцена» відповідає загально-технічним вимогам, написана на Java з чисто об'єктно-орієнтованим підходом. В ході розробки системи було проведено кілька тестових запусків та перевірку введення, для нівелювання помилок і надання можливості читати та записувати об'єкти у бінарні файли. Результати тестування показали, що система стабільна та вважається простою в обслуговуванні, оскільки вона в основному написана з концепцією успадкування та поліморфізму.

ВИСНОВКИ

В ході виконання магістерської роботи були вирішені всі поставлені завдання.

Розглянуто існуючі АІС ветеринарних організацій. Проведено аналіз функцій ветеринарної клініки «Авіцена», яка пропонує для тварин послуги вакцинації; хірургії; терапії; догляду під час проживання їх в спеціалізованому готелі. У ветеринарній клініці «Авіцена» облік і контроль діяльності організації не автоматизовано. Керівник організації не має можливості доступу до актуальної інформації про клініку: контролю наданих послуг; наявності клієнтів і проведених заходів (лікування, призначення) їх тваринам; можливості отримання консультації ветеринарного лікаря для аналізу і поліпшенню якості послуг і тим самим залученню нових клієнтів. Для розвитку даної сфери послуг населенню, підвищення якості та оперативного обслуговування, необхідна була автоматизація такої діяльності.

З урахуванням висунутих вимог і побажань наявності необхідних для замовника модулів було розроблено нову концепцію і розробку програми для ветеринарної клініки «Авіцена» з автоматизації процесів обліку і управління, яка буде мати можливості ведення обліку з матеріалів або препаратів, матиме набір необхідних звітів.

Проаналізувавши сучасні технології програмування та середовища розробки, для магістерського дослідження було обрано об'єктно - орієнтовану мову програмування Java через її мультиплатформеність, велику кількість бібліотек, та програмну підтримку. Для розробки самого додатку вибрано середовище розробки Eclipse, а для створення графічного інтерфейсу користувача - Swing.

В процесі розробки програмного застосування «Авіцена» всі вимоги до системи були виконані, охоплено всі функції, що виконують чотири види

робіт ветеринарної клініки, якими є адміністрування, обслуговування готелю для тварин, ветеринарія та керування. Система відповідає стандартам і вимогам, що пред'являються до сучасних систем подібного роду.

Крім того, розроблена система «Авіцена», має можливість доопрацювання в майбутньому.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Java Swing tutorial. URL:<http://zetcode.com/tutorials/javaswingtutorial/>. (Accessed: 5.11.2023).
2. Java2D: An Introduction and Tutorial. URL: <http://www.apl.jhu.edu/~hall/java/Java2D-Tutorial.html>. (Accessed: 10.05.2023).
3. JavaScript behavior. URL: <https://getbootstrap.com/docs/4.0/components/navs/#javascript-behavior>. (Accessed: 5.11.2023).
4. Kumaraswamipillai A., Arulkumaran S. Tutorial: Java, Maven 2, Eclipse and JSF. Lulu.com. 2010.
5. Modal. URL: <https://getbootstrap.com/docs/4.0/components/modal/>. (Accessed: 10.12.2023).
6. Navbar [Documentation and examples for Bootstrap's powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more, including support for our collapse plugin]. URL: <https://getbootstrap.com/docs/4.0/components/navbar/>. (Accessed: 5.11.2023).
7. Pdfmake - друк PDF на стороні клієнта / сервера у чистому JavaScript. URL: <https://github.com/bpampuch/pdfmake>. (дата звернення: 10.12.2023).
8. PDFMAKE [Install]. URL: <http://pdfmake.org/#/gettingstarted>. (Accessed: 10.12.2023).
9. Script inlining (JavaScript and Dart). URL: <https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#script-inlining-javascript-and-dart>. (дата звернення: 5.11.2023).
10. SeunMatt. MySQL-Backup4j [бібліотека для програмного експорту баз даних mysql]. URL: <https://github.com/SeunMatt/mysql-backup4j>. (Accessed: 10.12.2023).
11. Shah S. Maven for Eclipse. Packt Publishing, 2014. 158 p.
12. Spring Data JPA - Reference Documentation. [Query Creation]. URL:<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

#repositories.query- method.query-create (Accessed: 10.12.2023).

13. Spring Framework. URL: https://bg.wikipedia.org/wiki/Spring_Framework. (Accessed: 10.12.2023).

14. This is an outdated alpha release. URL: <https://v4-alpha.getbootstrap.com/components/forms/#form-groups>. (Accessed: 10.12.2023).

15. ШМИТТ, К. CSS. Рецепты программирования. Санкт-Петербург: БХВ-Петербург, 2011. 672 с.

16. Программы для ветеринарной клиники. URL: https://www.livemedical/tags/programmy_dlja_veterinarnoj_kliniki/(дата звернення: 17.04.2022).

17. Программа для ветеринарии. URL: <http://ususoft.com/veterinary.php>(дата звернення: 17.04.2022) – Назва з титулу екрана.

18. Проект наказу МОЗ України «Про затвердження Концепції інформатизації сфери охорони здоров'я України». URL: http://www.moz.gov.ua/ua/portal/dn_20121226_pp.html.

19. Chauhan, S., 2013. Understanding Association, Aggregation, Composition and Dependency relationship. URL: <http://www.dotnet-tricks.com/Tutorial/oops/T0Lb270813-Understanding-Association,-Aggregation,-Composition-and-Dependency-relationship.html>(Accessed: 10.12.2023).

20. Deitel, P. & Deitel, H., 2015. Java How to Program. 10th ed. London: Pearson.

21. Flanagan, D., 2005. Java In A Nutshell. 5th ed. s.l.:O'Reilly.

22. Horton, I., 2011. Beginning Java. 7th ed. Indianapolis: John Wiley & Sons, Inc. koirala, S., 2012. Understanding Association, Aggregation, and Composition - CodeProject. URL: <http://www.codeproject.com/Articles/330447/Understanding-Association-Aggregation-and-Composit> (Accessed 17 April 2023).

23. Lewis, J. & Loftus, W., 004. Java Software Solutions - Foundations of Program Design. 4th ed. s.l.:Addison Wesley.

24. Malik, D., 2012. Java Programming from Problem Analysis to

Program Design. 5th ed. Boston: Course Technology.

25. Oracle, 2015. Anonymous Classes (The Java™ Tutorials > Learning the Java Language > Classes and Objects). [Online] Available at: <https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html>.

(Accessed 17 April 2023).

26. Oracle, 2015. What Is an Exception? (The Java™ Tutorials > Essential Classes > Exceptions). [Online] Available at: <https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html>. (Accessed 17 April 2023).

27. Rouse, M., 2008. What is object-oriented programming (OOP)? - Definition from WhatIs.com. [Online] Available at: <http://searchsoa.techtarget.com/definition/object-oriented-programming>. (Accessed 17 April 2023).

28. Ветменеджер. URL: <http://www.vetmanager.ru>

29. CosmoZoo. URL: <https://cosmozoo.co> (application date: 10.11.2023).

30. VetDesk. URL: <http://vetdesk.ru/> (Accessed: 10.05.2022).

31. Sofie. URL: <https://sofie.com/> (Accessed: 10.05.2022).

32. Кристиан Нейгел. «C# 5.0 и платформа .NET 4.5 для профессионалов – Professional C# 5.0 and .NET 4.5.». М.: «Диалектика», 2013. 1440 с. ISBN 978-5-8459-1850-5

33. Лафоре Р. Объектно-ориентированное программирование в C++. Серия: Классика Computer Science. Издательство: Питер СПб. 2018. 928с.

34. Фаулер Мартин, Бек Кент, Брант Джон, Опдаик Уильям, Робертс Дон. Рефакторинг: улучшение проекта существующего кода. Спб: «Диалектика», 2019. 448 с. ISBN 978-5-9909445-1-0.

35. 3 Integrating JavaFX into Swing Applications. URL: <https://docs.oracle.com/javase/8/javafx/interoperability-tutorial/swing-fx-interoperability.htm>. (Accessed: 10.12.2023).

ДОДАТКИ

Додаток А. Код ReceptionPage.java

```
package server;

import javax.swing.*.*;
import java.awt.*.*;
import java.util.LinkedList;
import javax.swing.table.*.*;
import java.awt.event.*.*;
import java.util.*.*;
import com.toedter.calendar.*.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

public class ReceptionistPage extends JFrame {

    private JTabbedPane tab1;
    private JPanel panel1;

    //panel1
    private JLabel ID, firstName, lastName, middleName, tel, email, address, petType,
    petSpecies, petName;
    private JTextField tID, tFirstName, tLastName, tmiddleName, tTel, tEmail, tAddress,
    tPetSpecies, tPetName;
    private JRadioButton rNewCustomer, rRegisterPet;
    private ButtonGroup bg;
    private JComboBox cbPetType;
    private JButton addButton, makeAppointmentButton;
    private JTable table;
    private DefaultTableModel model;
    private JScrollPane scrollPane;

    public ReceptionistPage() {
        super("Сторінка адміністратора");

        //Tab 1
        tab1 = new JTabbedPane();
        panel1 = new JPanel();
        panel1.setLayout(null);

        rNewCustomer = new JRadioButton("Новий клієнт", true);
        rNewCustomer.setLocation(80, 20);
        rNewCustomer.setSize(rNewCustomer.getPreferredSize());
        panel1.add(rNewCustomer);

        rRegisterPet = new JRadioButton("Тільки тварина", false);
        rRegisterPet.setLocation(180, 20);
        rRegisterPet.setSize(rRegisterPet.getPreferredSize());
        panel1.add(rRegisterPet);
    }
}
```

```

bg = new ButtonGroup();
bg.add(rNewCustomer);
bg.add(rRegisterPet);

ID = new JLabel("ID");
ID.setLocation(20, 80);
ID.setSize(ID.getPreferredSize());
panel1.add(ID);

Customer cus = new Customer();
tID = new JTextField(cus.generateID());
tID.setColumns(15);
tID.setSize(tID.getPreferredSize());
tID.setLocation(150, 80);
tID.setEnabled(false);
panel1.add(tID);

firstName = new JLabel("Ім'я");
firstName.setLocation(20, 120);
firstName.setSize(firstName.getPreferredSize());
panel1.add(firstName);

tFirstName = new JTextField();
tFirstName.setColumns(15);
tFirstName.setSize(tFirstName.getPreferredSize());
tFirstName.setLocation(150, 120);
tFirstName.setToolTipText("Введіть ім'я");
panel1.add(tFirstName);

lastName = new JLabel("Ім'я по батькові");
lastName.setLocation(20, 160);
lastName.setSize(lastName.getPreferredSize());
panel1.add(lastName);

tLastName = new JTextField();
tLastName.setColumns(15);
tLastName.setSize(tLastName.getPreferredSize());
tLastName.setLocation(150, 160);
tLastName.setToolTipText("Введіть фамілію");
panel1.add(tLastName);

middleName = new JLabel("Фамілія");
middleName.setLocation(20, 200);
middleName.setSize(middleName.getPreferredSize());
panel1.add(middleName);

tmiddleName = new JTextField();
tmiddleName.setColumns(15);
tmiddleName.setSize(tmiddleName.getPreferredSize());
tmiddleName.setLocation(150, 200);
tmiddleName.setToolTipText("e.g. 940328086126");
panel1.add(tmiddleName);

```

```

tel = new JLabel("Телефон");
tel.setLocation(20, 240);
tel.setSize(tel.getPreferredSize());
panel1.add(tel);

tTel = new JTextField();
tTel.setColumns(15);
tTel.setSize(tTel.getPreferredSize());
tTel.setLocation(150, 240);
tTel.setToolTipText("e.g. 0169532287");
panel1.add(tTel);

email = new JLabel("E-mail адреса");
email.setLocation(20, 280);
email.setSize(email.getPreferredSize());
panel1.add(email);

tEmail = new JTextField();
tEmail.setColumns(15);
tEmail.setSize(tEmail.getPreferredSize());
tEmail.setLocation(150, 280);
tEmail.setToolTipText("e.g. eric.you@hotmail.com");
panel1.add(tEmail);

address = new JLabel("Домашня адреса");
address.setLocation(20, 320);
address.setSize(address.getPreferredSize());
panel1.add(address);

tAddress = new JTextField();
tAddress.setColumns(15);
tAddress.setSize(tAddress.getPreferredSize());
tAddress.setLocation(150, 320);
panel1.add(tAddress);

petType = new JLabel("Тип тварини");
petType.setLocation(20, 360);
petType.setSize(petType.getPreferredSize());
panel1.add(petType);

String[] petTypeList = {"Амфібія", "Птиця", "Рогата худоба", "Домашня тварина",
"Рептилія"};
cbPetType = new JComboBox(petTypeList);
cbPetType.setSize(cbPetType.getPreferredSize());
cbPetType.setLocation(150, 360);
panel1.add(cbPetType);

petSpecies = new JLabel("Вид тварини");
petSpecies.setLocation(20, 400);
petSpecies.setSize(petSpecies.getPreferredSize());
panel1.add(petSpecies);

```

```

tPetSpecies = new JTextField();
tPetSpecies.setColumns(15);
tPetSpecies.setSize(tPetSpecies.getPreferredSize());
tPetSpecies.setLocation(150, 400);
panel1.add(tPetSpecies);

petName = new JLabel("Ім'я тварини");
petName.setLocation(20, 440);
petName.setSize(petName.getPreferredSize());
panel1.add(petName);

tPetName = new JTextField();
tPetName.setColumns(15);
tPetName.setSize(tPetName.getPreferredSize());
tPetName.setLocation(150, 440);
panel1.add(tPetName);

addButton = new JButton("Додати");
addButton.setSize(90, 23);
addButton.setLocation(150, 480);
panel1.add(addButton);

makeAppointmentButton = new JButton("Призначити прийом");
makeAppointmentButton.setSize(makeAppointmentButton.getPreferredSize());
makeAppointmentButton.setLocation(1230, 420);
panel1.add(makeAppointmentButton);

table = new JTable(model);
table.setToolTipText("Подвійний клік для редагування");
java.util.List<Customer> customerList = new LinkedList();
Customer.populateList(customerList);

//ref: http://stackoverflow.com/questions/17368505/populating-jtable-using-list
String columnNames[] = {"ID", "Ім'я", "Ім'я по батькові", "Фамілія", "Телефон", "E-
mail", "Адреса", "К. тварин", "ID тварини", "Тип тварини", "Вид тварини", "Ім'я тварини"};
model = new DefaultTableModel(new String[0][0], columnNames);

for (Customer x : customerList) {

    String[] o = new String[12];
    o[0] = x.getID();
    o[1] = x.getFirstName();
    o[2] = x.getLastName();
    o[3] = x.getMiddleName();
    o[4] = x.getTel();
    o[5] = x.getEmail();
    o[6] = x.getAddress();
    o[7] = Integer.toString(x.getNumberOfPet());

    String ID, type, species, name;
    ID = x.getPet(0).getID();
    type = x.getPet(0).getType();
    species = x.getPet(0).getSpecies();

```

```

name = x.getPet(0).getName();

for (int i = 1; i < x.getNumberOfPet(); i++) {
    ID = ID + "," + x.getPet(i).getID();

    type = type + "," + x.getPet(i).getType();

    species = species + "," + x.getPet(i).getSpecies();

    name = name + "," + x.getPet(i).getName();

}
o[8] = ID;
o[9] = type;
o[10] = species;
o[11] = name;
model.addRow(o);
}
table.setModel(model);

scrollPane = new JScrollPane();
scrollPane.setViewportView(table);
scrollPane.setLocation(320, 10);
scrollPane.setSize(1040, 400);
scrollPane.setVerticalScrollBarPolicy(scrollPane.VERTICAL_SCROLLBAR_ALWAYS);
panel1.add(scrollPane);

tab1.addTab("Регістрація", panel1);
add(tab1);

//Event Handler
rNewCustomer.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            Customer c = new Customer();
            tID.setText(c.generateID());
        }
    }
);

rNewCustomer.addItemListener(
    new ItemListener() {
        public void itemStateChanged(ItemEvent e) {

            tFirstName.setEnabled(true);
            tLastName.setEnabled(true);
            tmiddleName.setEnabled(true);
            tTel.setEnabled(true);
            tEmail.setEnabled(true);
            tAddress.setEnabled(true);

            Customer c = new Customer();
            tID.setText(c.generateID());

```

```

    }
    }
);

rRegisterPet.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            tID.setText("");
            JOptionPane.showMessageDialog(null, "Будь ласка, спочатку виберіть
Клієнта з таблиці", "WARNING", JOptionPane.WARNING_MESSAGE);
        }
    }
);

rRegisterPet.addItemListener(
    new ItemListener() {
        public void itemStateChanged(ItemEvent event) {
            tID.setEnabled(false);
            tFirstName.setEnabled(false);
            tLastName.setEnabled(false);
            tmiddleName.setEnabled(false);
            tTel.setEnabled(false);
            tEmail.setEnabled(false);
            tAddress.setEnabled(false);
        }
    }
);

addButton.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            try {
                Receptionist r = new Receptionist();
                java.util.List<Customer> customerList = new LinkedList();

                if (rNewCustomer.isSelected()) {
                    if (tID.getText().equals("") || tFirstName.getText().equals("") ||
tLastName.getText().equals("") || tmiddleName.getText().equals("") || tTel.getText().equals("") ||
tEmail.getText().equals("") || tAddress.getText().equals("") || tPetSpecies.getText().equals("") ||
tPetName.getText().equals("")) {
                        throw new NullPointerException("Будь ласка, заповніть все поля!");
                    }

                    r.createCustomerProfile((String) cbPetType.getSelectedItem(),
tPetSpecies.getText(), tPetName.getText(), tFirstName.getText(), tLastName.getText(),
tmiddleName.getText(), tTel.getText(), tEmail.getText(), tAddress.getText());
                    JOptionPane.showMessageDialog(null, "Новий клієнт створений!");

                    Customer.populateList(customerList);
                    String columnNames[] = {"ID", "Ім'я", "Ім'я по батькові", "Фамілія",
"Телефон", "E-mail", "Адреса", "К. тварин", "ID тварини", "Тип тварини", "Вид тварини", "Ім'я
тварини"};

                    model = new DefaultTableModel(new String[0][0], columnNames);

```

```

for (Customer x : customerList) {

    String[] o = new String[12];
    o[0] = x.getID();
    o[1] = x.getFirstName();
    o[2] = x.getLastName();
    o[3] = x.getmiddleName();
    o[4] = x.getTel();
    o[5] = x.getEmail();
    o[6] = x.getAddress();
    o[7] = Integer.toString(x.getNumberOfPet());

    String ID, type, species, name;
    ID = x.getPet(0).getID();
    type = x.getPet(0).getType();
    species = x.getPet(0).getSpecies();
    name = x.getPet(0).getName();

    for (int i = 1; i < x.getNumberOfPet(); i++) {
        ID = ID + "," + x.getPet(i).getID();

        type = type + "," + x.getPet(i).getType();

        species = species + "," + x.getPet(i).getSpecies();

        name = name + "," + x.getPet(i).getName();

    }
    o[8] = ID;
    o[9] = type;
    o[10] = species;
    o[11] = name;
    model.addRow(o);
}
table.setModel(model);

} else if (rRegisterPet.isSelected()) {
    if (tID.getText().equals("")) {
        throw new NullPointerException("Виберіть клієнта з таблиці!");
    }

    if (tPetSpecies.getText().equals("") || tPetName.getText().equals("")) {
        throw new NullPointerException("Будь ласка, заповніть усі поля для
тварини!");
    }

    r.createPetProfile(tID.getText(), (String) cbPetType.getSelectedItem(),
tPetSpecies.getText(), tPetName.getText());
    JOptionPane.showMessageDialog(null, "Додано нову тварину ");

    Customer.populateList(customerList);
}

```

```

using-list
//ref: http://stackoverflow.com/questions/17368505/populating-jtable-
String columnNames[] = {"ID", "Ім'я", "Ім'я по батькові", "Фамілія",
"Телефон", "E-mail", "Адреса", "К. тварин", "ID тварини", "Тип тварини", "Вид тварини", "Ім'я
тварини"};

model = new DefaultTableModel(new String[0][0], columnNames);

for (Customer x : customerList) {

    String[] o = new String[12];
    o[0] = x.getID();
    o[1] = x.getFirstName();
    o[2] = x.getLastName();
    o[3] = x.getmiddleName();
    o[4] = x.getTel();
    o[5] = x.getEmail();
    o[6] = x.getAddress();
    o[7] = Integer.toString(x.getNumberOfPet());

    String ID, type, species, name;
    ID = x.getPet(0).getID();
    type = x.getPet(0).getType();
    species = x.getPet(0).getSpecies();
    name = x.getPet(0).getName();

    for (int i = 1; i < x.getNumberOfPet(); i++) {
        ID = ID + "," + x.getPet(i).getID();

        type = type + "," + x.getPet(i).getType();

        species = species + "," + x.getPet(i).getSpecies();

        name = name + "," + x.getPet(i).getName();

    }
    o[8] = ID;
    o[9] = type;
    o[10] = species;
    o[11] = name;
    model.addRow(o);
}
table.setModel(model);
}
} catch (NullPointerException e) {
    JOptionPane.showMessageDialog(null, e.getMessage());
} catch (NullPointerException e) {
    JOptionPane.showMessageDialog(null, e.getMessage());
}
}
}
);

table.addMouseListener(

```



```

        new MouseAdapter() {
            public void mouseClicked(MouseEvent event) {
                int row = table.getSelectedRow();
                tID.setText(table.getModel().getValueAt(row, 0).toString());
            }
        }
    };

    table.addMouseListener(
        new MouseAdapter() {
            public void mousePressed(MouseEvent event) {
                if (event.getClickCount() % 2 == 0) {
                    int row = table.getSelectedRow();

                    EditCustomerFrame ec = new
EditCustomerFrame(table.getModel().getValueAt(row, 0).toString());
                    ec.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                    ec.setSize(600, 450);
                    ec.setLocation(400, 80);
                    ec.setVisible(true);
                }
            }
        }
    );

    makeAppointmentButton.addActionListener(
        new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                try {
                    int row = table.getSelectedRow();

                    AppointmentPage ap = new
AppointmentPage(table.getModel().getValueAt(row, 0).toString());
                    ap.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                    ap.setSize(1350, 450);
                    ap.setVisible(true);
                } catch (ArrayIndexOutOfBoundsException e) {
                    JOptionPane.showMessageDialog(null, "Будь ласка, виберіть Клієнта з
таблиці");
                }
            }
        }
    );
}
}

```

Додаток Б. Код Vet.java

```

package server;

import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InvalidClassException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.io.Serializable;
import java.util.*;
import javax.swing.JOptionPane;

public class Vet extends Human implements Serializable {

    private String[] areaOfExpertise = new String[2];

    public Vet() {

    }

    public Vet(String areaOfExpertise1, String areaOfExpertise2) {

        this.areaOfExpertise[0] = areaOfExpertise1;
        this.areaOfExpertise[1] = areaOfExpertise2;

    }

    public void setVetID() {
        this.ID = this.generateID();
        this.saveGeneratedID();
    }

    //Generate Staff ID
    @Override
    public String generateID() {
        try {
            int ID = 0;

            File file = new File("VetID.txt");
            Scanner inputFile = new Scanner(file);

            while (inputFile.hasNext()) {
                ID = Integer.parseInt(inputFile.nextLine());
            }
            inputFile.close();

            ID++;

```

```

        this.ID = "VET" + String.format("%03d", ID);

    } catch (IOException e) {
        e.printStackTrace();
    }
    return ID;
}

//Save generated Staff ID
@Override
public void saveGeneratedID() {
    try {
        int ID = 0;
        File file = new File("VetID.txt");
        Scanner inputFile = new Scanner(file);

        while (inputFile.hasNext()) {
            ID = Integer.parseInt(inputFile.nextLine());
        }
        inputFile.close();

        ID++;

        FileWriter fw = new FileWriter("VetID.txt");
        PrintWriter pw = new PrintWriter(fw);

        pw.print(ID);

        pw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

//Search and return object
public static Vet searchObject(List<Vet> list, String ID) {
    Vet obj = null;

    for (Vet y : list) {
        if (y.getID().equalsIgnoreCase(ID)) {
            obj = y;
        }
    }

    return obj;
}

//Remove the selected Object
public static void removeObject(List<Vet> list, String ID) {
    try {
        Vet x = null;
        for (Vet y : list) {
            if (y.getID().equalsIgnoreCase(ID)) {

```

```

        x = y;
        break;
    }
}
list.remove(x);
} catch (ConcurrentModificationException e) {
    JOptionPane.showMessageDialog(null, e.getMessage());
}
}

//Append new object to the LinkedList
public static void addObject(List<Vet> list, Vet y) {
    list.add(y);
}

//Populate the LinkedList with the content of binary file
public static void populateList(List<Vet> list) {
    try {

        FileInputStream inStream = new FileInputStream("Vet.dat");
        ObjectInputStream objectInputFile = new ObjectInputStream(inStream);

        while (true) {

            Vet temp = (Vet) objectInputFile.readObject();

            list.add(temp);

        }

    } catch (EOFException e) {

    } catch (Exception e) {
        e.printStackTrace();
    }
}

//Replace the whole binary file content with the LinkedList
public static void overwriteBinaryFile(List<Vet> list) {

    try {
        FileOutputStream outStream = new FileOutputStream("Vet.dat");
        ObjectOutputStream objectOutputFile = new ObjectOutputStream(outStream);

        for (Vet y : list) {
            objectOutputFile.writeObject(y);
        }

        objectOutputFile.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

public void enterDiagnosisAndPrognosis(String petID, String Prognosis, String Diagnosis) {
    List<Customer> list = new LinkedList();
    Customer.populateList(list);

    for (Customer x : list) {
        for (int i = 0; i < x.getNumberOfPet(); i++) {
            if (x.getPet(i).getID().equalsIgnoreCase(petID)) {
                x.getPet(i).setPrognosis(Prognosis);
                x.getPet(i).setDiagnosis(Diagnosis);
                x.getPet(i).setSeen(Boolean.TRUE);
            }
        }
    }
    Customer.overwriteBinaryFile(list);
}

public void inputCharges(String petID, double charges) {
    List<Customer> list = new LinkedList();
    Customer.populateList(list);

    for (Customer x : list) {
        for (int i = 0; i < x.getNumberOfPet(); i++) {
            if (x.getPet(i).getID().equalsIgnoreCase(petID)) {
                x.getPet(i).setCharges(charges);
            }
        }
    }
    Customer.overwriteBinaryFile(list);
}

public void viewAppointments() {
    List<Appointment> list = new LinkedList();
    Appointment.populateList(list);
    for (Appointment x : list) {

    }
}

public String[] getAreaOfExpertise() {
    return areaOfExpertise;
}

public void setAreaOfExpertise(String areaOfExpertise1, String areaOfExpertise2) {
    this.areaOfExpertise[0] = areaOfExpertise1;
    this.areaOfExpertise[1] = areaOfExpertise2;
}
}

```