Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

З лабораторної роботи №3

Варіант – 3

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав: ст. гр. КІ-306

Братівник Д. А.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Завдання(Варіант 3)

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у
лабораторній роботі No2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі No2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування
та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її
виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

Варіант завдання: Піддослідний Пес

Код програми:

Файл DogApp.java:

```
package org.example;

import org.example.ResearchDog;

import java.io.FileNotFoundException;
/**
 * Проста програма для демонстрації використання класу
ResearchDog.
 */
public class DogApp {
    public static void main(String[] args) throws
FileNotFoundException {
        // Створення об'єкту ResearchDog для дослідження
        ResearchDog myResearchDog = new ResearchDog("Рекс", 3,
"Лабрадор", false);

        // Виведення інформації про собаку
        myResearchDog.displayInfo();
```

```
        // Проведення експерименту та збір даних
        myResearchDog.observeBehavior();
        myResearchDog.analyzeData();
        myResearchDog.performExperiment();
        myResearchDog.recordResults();
        // Закриття файлу журналу
        myResearchDog.closeLogFile();
    }
}
```

Файл Dog.java:

```
package org.example;

import java.io.*;

/**
 * Абстрактний клас, що представляє собаку з основними
характеристиками та поведінкою.
 */
public abstract class Dog {
    private String name;
    private int age;
    private String breed;
    private boolean isTrained;
    private PrintWriter logFile;

    /**
     * Конструктор з параметрами для створення собаки з вказаними
характеристиками.
     *
     * @param name      Ім'я собаки.
     * @param age       Вік собаки.
     * @param breed     Порода собаки.
     * @param isTrained Прапорець, що вказує, чи навчена собака.
     * @throws FileNotFoundException Виникає, якщо не вдається
створити файл журналу.
     */
    public Dog(String name, int age, String breed, boolean
isTrained) throws FileNotFoundException {
        this.name = name;
        this.age = age;
        this.breed = breed;
        this.isTrained = isTrained;
        logFile = new PrintWriter(new File("DogLog.txt"));
    }

    /**
     * Конструктор за замовчуванням для створення собаки з
ініціальними значеннями.
     *
     * @throws FileNotFoundException Виникає, якщо не вдається
створити файл журналу.
     */
```

```
    public Dog() throws FileNotFoundException {
        this.name = "Невідомо";
        this.age = 0;
        this.breed = "Невідома";
        this.isTrained = false;
        logFile = new PrintWriter(new File("DogLog.txt"));
    }

    /**
     * Метод для зміни імені собаки.
     *
     * @param name Нове ім'я собаки.
     */
    public void setName(String name) {
        this.name = name;
        logFile.println("Змінено ім'я собаки на: " + name);
        logFile.flush();
    }

    /**
     * Метод для отримання імені собаки.
     *
     * @return Ім'я собаки.
     */
    public String getName() {
        return name;
    }

    /**
     * Метод для зміни віку собаки.
     *
     * @param age Новий вік собаки.
     */
    public void setAge(int age) {
        this.age = age;
        logFile.println("Змінено вік собаки на: " + age);
        logFile.flush();
    }

    /**
     * Метод для отримання віку собаки.
     *
     * @return Вік собаки.
     */
    public int getAge() {
        return age;
    }

    /**
     * Метод для зміни породи собаки.
     *
     * @param breed Нова порода собаки.
     */
    public void setBreed(String breed) {
```

```
        this.breed = breed;
        logFile.println("Змінено породу собаки на: " + breed);
        logFile.flush();
    }

    /**
     * Метод для отримання породи собаки.
     *
     * @return Порода собаки.
     */
    public String getBreed() {
        return breed;
    }

    /**
     * Метод для навчання собаки.
     */
    public void train() {
        isTrained = true;
        logFile.println("Собаку навчено");
        logFile.flush();
    }

    /**
     * Метод для перевірки, чи навчена собака.
     *
     * @return true, якщо собака навчена; false, якщо не навчена.
     */
    public boolean isTrained() {
        return isTrained;
    }

    /**
     * Метод для запису в журнал завершення роботи з файлом.
     */
    public void closeLogFile() {
        logFile.close();
    }

    /**
     * Метод для відображення інформації про собаку.
     */
    public void displayInfo() {
        System.out.println("Ім'я собаки: " + name);
        System.out.println("Вік собаки: " + age);
        System.out.println("Порода собаки: " + breed);
        System.out.println("Навчена: " + (isTrained ? "Так" :
"Ні"));
        System.out.print("\n");
    }
    /**
     * Отримує файл журналу, використовуваний для записів.
     *
     * @return Файл журналу.
```

```
     */
    public PrintWriter getLogFile() {
        return logFile;
    }
}
```

Файл BehaviorResearch.java

```
package org.example;
/**
 * Інтерфейс для дослідження поведінки, який визначає методи для
спостереження за поведінкою та аналізу даних.
 */
public interface BehaviorResearch {
    /**
     * Спостерігає та реєструє поведінку.
     */
    void observeBehavior();
    /**
     * Аналізує зібрані дані.
     */
    void analyzeData();
    /**
     * Проводить експеримент з собакою.
     */
    void performExperiment();
    /**
     * Реєструє результати експерименту.
     */
    void recordResults();
    /**
     * Закриває файл журналу, використовуваний для записів.
     */
    void closeLogFile();
}
```

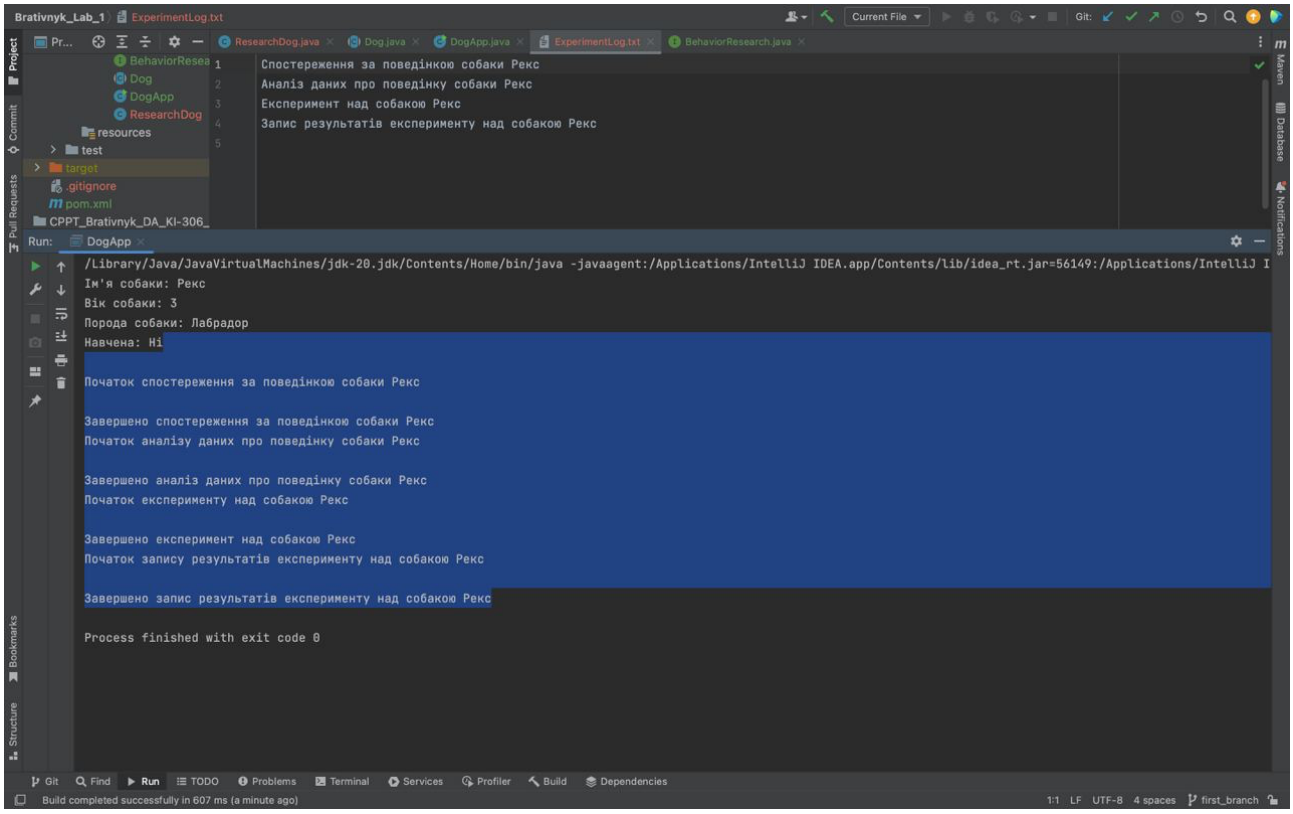
Файл ResearchDog.java

```
package org.example;

import java.io.*;
/**
 * Клас, який представляє дослідницьку собаку, який розширює клас
Dog та реалізує BehaviorResearch.
 */
public class ResearchDog extends Dog implements BehaviorResearch {
    private PrintWriter experimentLogFile;

    public ResearchDog(String name, int age, String breed, boolean
isTrained) throws FileNotFoundException {
        super(name, age, breed, isTrained);
        experimentLogFile = new PrintWriter(new
File("ExperimentLog.txt"));
```

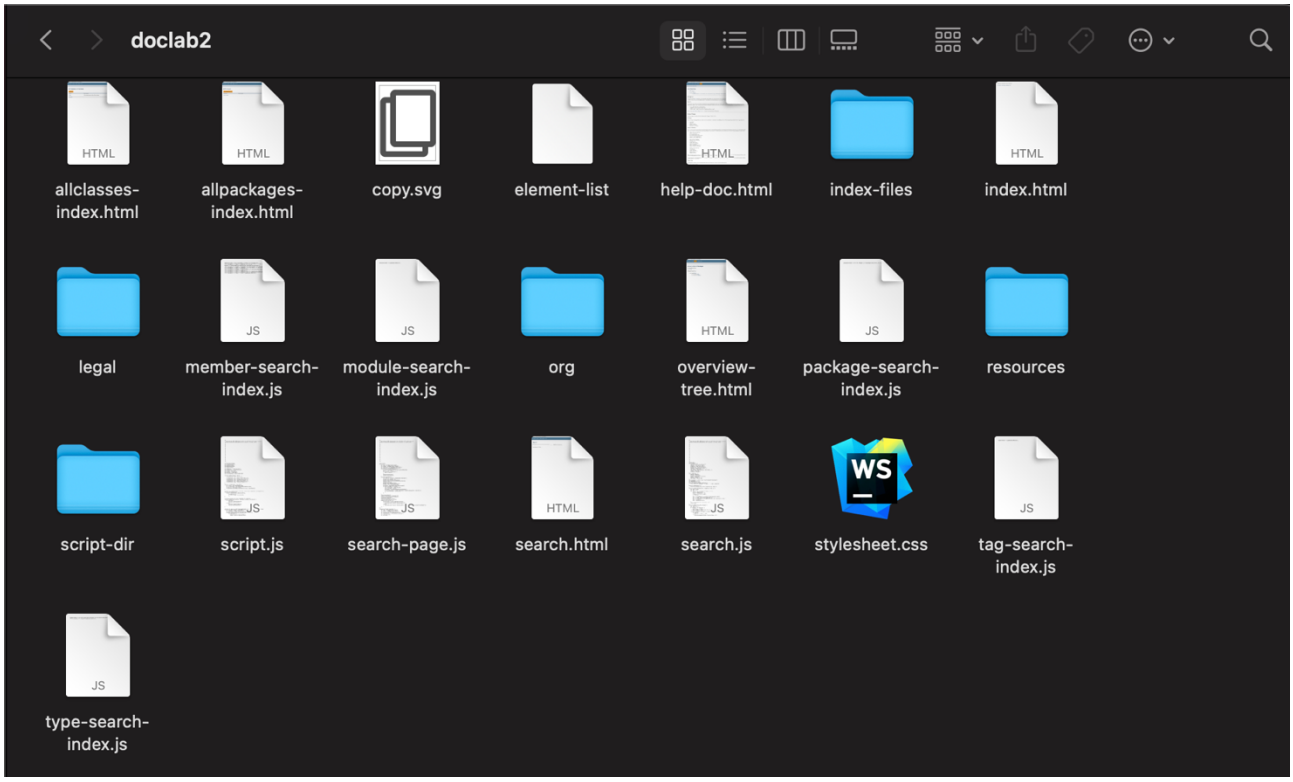
```
    }
    /**
     * Починає спостереження за поведінкою собаки.
     */
    @Override
    public void observeBehavior() {
        System.out.println("Початок спостереження за поведінкою
собаки " + getName() + "\n");
        // Тут реалізуйте спостереження за поведінкою собаки
        System.out.println("Завершено спостереження за поведінкою
собаки " + getName());
        experimentLogFile.println("Спостереження за поведінкою
собаки " + getName());
        experimentLogFile.flush();
    }
    /**
     * Аналізує дані, зібрані під час спостереження за поведінкою.
     */
    @Override
    public void analyzeData() {
        System.out.println("Початок аналізу даних про поведінку
собаки " + getName() + "\n");
        // Тут реалізуйте аналіз даних про поведінку собаки
        System.out.println("Завершено аналіз даних про поведінку
собаки " + getName());
        experimentLogFile.println("Аналіз даних про поведінку
собаки " + getName());
        experimentLogFile.flush();
    }
    /**
     * Проводить експеримент з собакою.
     */
    public void performExperiment() {
        System.out.println("Початок експерименту над собакою " +
getName() + "\n");
        // Тут реалізуйте дослід, який ви хочете провести над
собакою
        System.out.println("Завершено експеримент над собакою " +
getName());
        experimentLogFile.println("Експеримент над собакою " +
getName());
        experimentLogFile.flush();
    }
    /**
     * Реєструє результати експерименту.
     */
    public void recordResults() {
        System.out.println("Початок запису результатів
експерименту над собакою " + getName() + "\n");
        // Тут реалізуйте запис результатів експерименту
        System.out.println("Завершено запис результатів
експерименту над собакою " + getName());
        experimentLogFile.println("Запис результатів експерименту
над собакою " + getName());
```

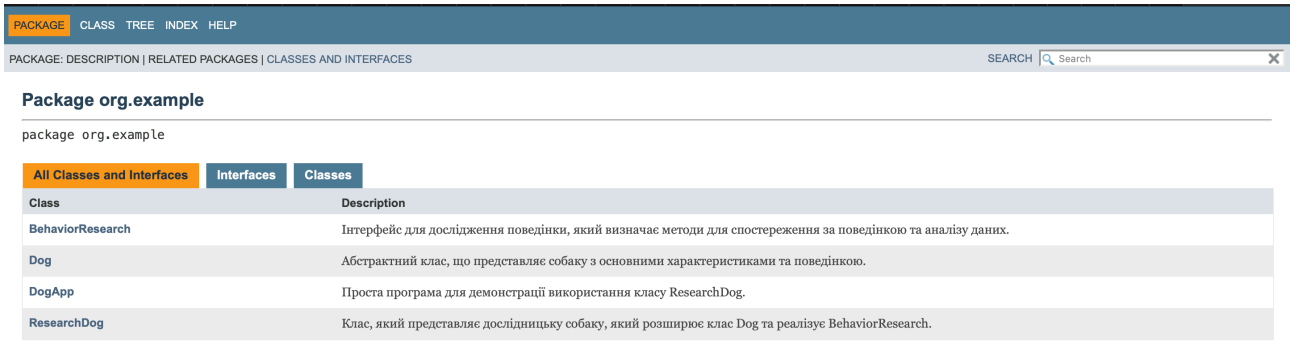
```
            experimentLogFile.flush();
        }
        /**
         * Закриває файл журналу, використовуваний для записів.
         */
        @Override
        public void closeLogFile() {
            super.closeLogFile();
            experimentLogFile.close();
        }
}
```

Результати роботи програми:



Фрагмент згенерованої документації





Відповіді на контрольні запитання

1. Синтаксис реалізації спадкування.
- class МійКлас implements Інтерфейс {

// тіло класу }

2. Що таке суперклас та підклас?
- суперклас - це клас, від якого інший клас успадковує властивості та методи.

Підклас - це клас, який успадковує властивості та методи від суперкласу.

3. Як звернутися до членів суперкласу з підкласу?
- super.назваМетоду([параметри]); // виклик методу суперкласу

super.назваПоля; // звернення до поля суперкласу

4. Коли використовується статичне зв’язування при виклику методу?
- Статичне зв’язування використовується, коли метод є приватним, статичним,

фінальним або конструктором. В таких випадках вибір методу відбувається на етапі компіляції.

5. Як відбувається динамічне зв’язування при виклику методу?
- вибір методу для виклику відбувається під час виконання програми на основі

фактичного типу об’єкта.

6. Що таке абстрактний клас та як його реалізувати?
- це клас, який має один або більше абстрактних методів (методів без реалізації).

Щоб створити абстрактний клас, використовується ключове слово abstract.
Приклад:
abstract class АбстрактнийКлас {

abstract void абстрактнийМетод(); }

7. Для чого використовується ключове слово instanceof?
- для перевірки, чи об'єкт належить до певного класу або інтерфейсу.

Синтаксис:
if (об'єкт instanceof Клас) {

// код, який виконується, якщо об'єкт належить до класу }

8. Як перевірити чи клас є підкласом іншого класу?
- В Java використовується ключове слово extends, щоб вказати, що клас є

підкласом іншого класу. Перевірити, чи один клас є підкласом іншого класу можна шляхом аналізу ієрархії успадкування.

9. Що таке інтерфейс?
- це абстрактний тип даних, який визначає набір методів, але не надає їх

реалізацію. Всі методи інтерфейсу є загальнодоступними та автоматично є public. Інтерфейси використовуються для створення контрактів, які класи повинні реалізувати.

10. Як оголосити та застосувати інтерфейс?

- - Для оголошення інтерфейсу використовується ключове слово interface.

  Синтаксис:
  interface Інтерфейс {

  // оголошення методів та констант }

- - Для застосування інтерфейсу в класі використовується ключове слово implements.

  Синтаксис:
  class МійКлас implements Інтерфейс {

  // реалізація методів інтерфейсу }

Висновок: У ході виконання даної лабораторної роботи, я отримав навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Ознайомившись з цими важливими аспектами об'єктно-орієнтованого програмування, я зрозумів їх роль у створенні більш структурованих і гнучких програм.