

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»



Звіт
з лабораторної роботи № 3
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Спадкування та інтерфейси»

Виконав:

студент групи КІ-306

Гапонова Дарина

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

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

Завдання (варіант № 4)

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну

область згідно варіанту. Програма має задовольняти наступним вимогам:

- програма має розміщуватися в пакеті Група.Прізвище.Lab2;
- клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
- клас має містити кілька конструкторів та мінімум 10 методів;
- для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;

- методи класу мають вести протокол своєї діяльності, що записується у файл;

- розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);

- програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

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.

Вихідний код програми

Файл CatApplication.java

```
package ki306.haponova.lab3;

import java.io.IOException;

/**
 * The {@code CatApplication} class is a simple program that demonstrates the
 * usage
 * of the {@link ExperimentalCat} class to create and interact with its object.
 *
 * @author Haponova Darina
 */
public class CatApplication {
    /**
     * The main entry point of the program.
     */
}
```

```

*
* @param args Command-line arguments (not used in this program).
*/
public static void main(String[] args) {
    ExperimentalCat myCat = new ExperimentalCat();

    myCat.setName("Fluffy");
    System.out.println("Cat's name: " + myCat.getName());
    myCat.setAge(4);
    System.out.println("Cat's age: " + myCat.getAge());
    myCat.setBreed("Persian");
    System.out.println("Cat's breed: " + myCat.getBreed());

    myCat.meow();
    myCat.run();
    myCat.jump();
    myCat.purr();
    myCat.sleep();
    myCat.eat("tuna");
    myCat.play("a ball");
    System.out.println(myCat.getSubjectType());

    try {
        myCat.dispose();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Файл Cat.java

```

package ki306.haponova.lab3;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**
 * The {@code Cat} abstract class represents a cat with various attributes and
 * behaviors.
 * It can log the cat's actions to a text file.
 *
 * @author Haponova Darina
 */
public abstract class Cat {
    private String name;
    private int age;
    private String breed;
    private String logFileName;
    private FileWriter logFileWriter;

    /**
     * Constructs a new cat with default values and initializes a log file.
     */
    public Cat() {
        this.name = "";
        this.age = 0;
        this.breed = "";
        logFileName = "lab3/catLog.txt";

        try {
            logFileWriter = new FileWriter(new File(logFileName));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    /**
     * Constructs a new cat with the specified name, age, and breed, and
     initializes a log file.
     *
     * @param name The name of the cat.
     * @param age The age of the cat.
     * @param breed The breed of the cat.
     */
    public Cat(String name, int age, String breed) {
        this.name = name;
        this.age = age;
        this.breed = breed;
        logFileName = "lab3/catLog.txt";

        try {
            logFileWriter = new FileWriter(new File(logFileName));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Sets the name of the cat.
     *
     * @param name The name of the cat.
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Sets the age of the cat.
     *
     * @param age The age of the cat.
     */
    public void setAge(int age) {
        this.age = age;
    }

    /**
     * Sets the breed of the cat.
     *
     * @param breed The breed of the cat.
     */
    public void setBreed(String breed) {
        this.breed = breed;
    }

    /**
     * Gets the name of the cat.
     *
     * @return The name of the cat.
     */
    public String getName() {
        return name;
    }

    /**
     * Gets the age of the cat.
     *
     * @return The age of the cat.
     */
    public int getAge() {
        return age;
    }

```

```

    }

    /**
     * Gets the breed of the cat.
     *
     * @return The breed of the cat.
     */
    public String getBreed() {
        return breed;
    }

    /**
     * Makes the cat meow and logs the action.
     */
    public void meow() {
        log(name + " says: Meow!");
    }

    /**
     * Makes the cat run and logs the action.
     */
    public void run() {
        log(name + " is running.");
    }

    /**
     * Makes the cat jump and logs the action.
     */
    public void jump() {
        log(name + " jumped.");
    }

    /**
     * Makes the cat purr and logs the action.
     */
    public void purr() {
        log(name + " is purring.");
    }

    /**
     * Makes the cat sleep and logs the action.
     */
    public void sleep() {
        log(name + " is sleeping.");
    }

    /**
     * Makes the cat eat the specified food and logs the action.
     *
     * @param food The food that the cat is eating.
     */
    public void eat(String food) {
        log(name + " is eating " + food + ".");
    }

    /**
     * Makes the cat play with the specified toy and logs the action.
     *
     * @param toy The toy that the cat is playing with.
     */
    public void play(String toy) {
        log(name + " is playing with " + toy + ".");
    }

    /**
     * Logs a message to the cat's log file.

```

```

    *
    * @param message The message to log.
    */
    public void log(String message) {
        try {
            logFileWriter.write(message + "\n");
            logFileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Closes the cat's log file.
     *
     * @throws IOException If an I/O error occurs while closing the log file.
     */
    public void dispose() throws IOException {
        logFileWriter.close();
    }
}

```

Файл ExperimentalSubject.java

```

package ki306.haponova.lab3;

/**
 * The {@code ExperimentalSubject} interface defines a contract for objects that
 * can be used as experimental subjects.
 *
 * @author Haponova Darina
 */
public interface ExperimentalSubject {
    /**
     * Gets the type of the experimental subject.
     *
     * @return The type of the experimental subject.
     */
    String getSubjectType();
}

```

Файл ExperimentalCat.java

```

package ki306.haponova.lab3;

/**
 * The {@code ExperimentalCat} class extends the {@link Cat} class and
 * implements
 * the {@link ExperimentalSubject} interface. It represents a cat that can be
 * used
 * as an experimental subject.
 *
 * @author Haponova Darina
 */
public class ExperimentalCat extends Cat implements ExperimentalSubject {
    /**
     * Gets the type of experimental subject, which is "Cat" in this case.
     *
     * @return The type of experimental subject.
     */
    @Override
    public String getSubjectType() {
        return "Cat";
    }
}

```

Результат виконання програми

CatLog.txt:



The screenshot shows an IDE with several tabs: CatApplication.java, catLog.txt (active), Cat.java, and Experimenta. The catLog.txt file contains the following text:

```
1 Fluffy says: Meow!
2 Fluffy is running.
3 Fluffy jumped.
4 Fluffy is purring.
5 Fluffy is sleeping.
6 Fluffy is eating tuna.
7 Fluffy is playing with a ball.
8
```

Below the code editor, a terminal window displays the output of the Java program:

```
"C:\Program Files\Java\jdk-17.0.3.1\bin\java.exe"
Cat's name: Fluffy
Cat's age: 4
Cat's breed: Persian
Cat
```

Фрагмент згенерованої документації

PACKAGE

CLASS

TREE

INDEX

HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

SEARCH:

Package ki306.haponova.lab3

package ki306.haponova.lab3

All Classes and Interfaces

Interfaces

Classes

| Class | Description |
|---------------------|---|
| Cat | The Cat abstract class represents a cat with various attributes and behaviors. |
| CatApplication | The CatApplication class is a simple program that demonstrates the usage of the ExperimentalCat class to create and interact with its object. |
| ExperimentalCat | The ExperimentalCat class extends the Cat class and implements the ExperimentalSubject interface. |
| ExperimentalSubject | The ExperimentalSubject interface defines a contract for objects that can be used as experimental subjects. |

Відповіді на контрольні запитання

- Синтаксис реалізації спадкування.
 - `class МійКлас implements Інтерфейс {
 // тіло класу
}`
- 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 отримала навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Ознайомившись з цими важливими аспектами об'єктно-орієнтованого програмування, я зрозуміла їх роль у створенні більш структурованих і гнучких програм.