

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



Звіт
з лабораторної роботи № 2
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Класи та пакети»

Виконав:

студент групи КІ-306

Гапонова Дарина

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Мета роботи: ознайомитися з процесом розробки класів та пакетів мовою Java.

Завдання (варіант № 4)

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.lab2;

import java.io.IOException;

/**
 * The {@code CatApplication} class is a simple program that demonstrates the
 * usage
 * of the {@link Cat} class to create and interact with a cat 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) {
    Cat myCat = new Cat("Whiskers", 3, "Siamese");

    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");

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

```

Файл Cat.java

```

package ki306.haponova.lab2;

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

/**
 * The {@code Cat} class represents a cat with various attributes and behaviors.
 * It can log the cat's actions to a text file.
 *
 * @author Haponova Darina
 */
public 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 = "lab2/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 = "lab2/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.

```

```

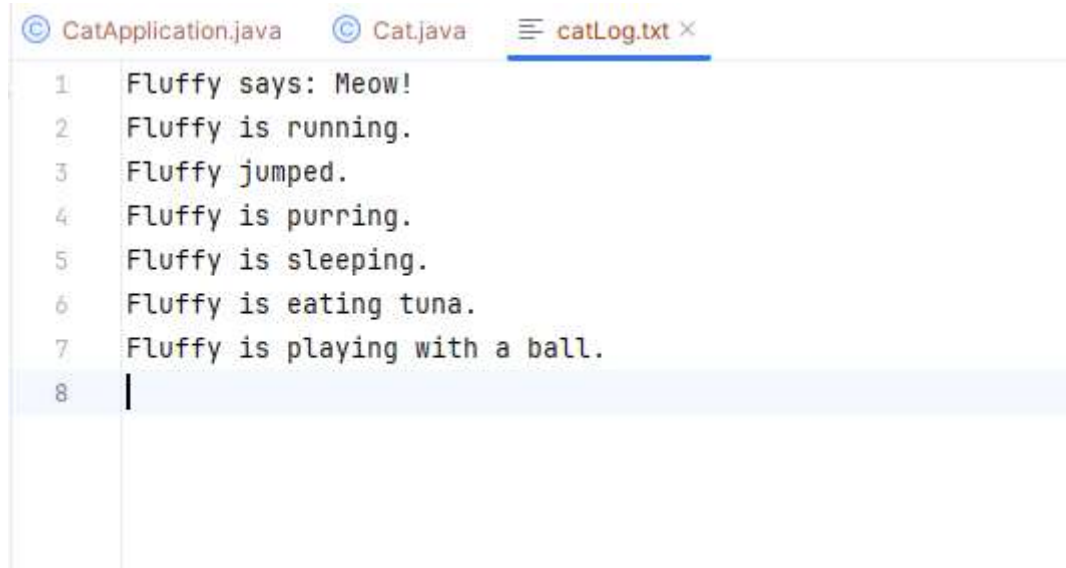
    *
    * @param message The message to log.
    */
    public void log(String message) {
        try {
            logFileWriter.write(message + "\n");
            logFileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 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();
    }
}

```

Результат виконання програми

catLog.txt:



The screenshot shows an IDE with three tabs: CatApplication.java, Cat.java, and catLog.txt. The catLog.txt tab is active and displays 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 |

```

Фрагмент згенерованої документації

PACKAGE CLASS TREE INDEX HELP	
PACKAGE: DESCRIPTION RELATED PACKAGES CLASSES AND INTERFACES	
SEARCH: <input type="text" value="Search"/>	
Package ki306.haponova.lab2	
package ki306.haponova.lab2	
Classes	
Class	Description
Cat	The Cat class represents a cat with various attributes and behaviors.
CatApplication	The CatApplication class is a simple program that demonstrates the usage of the Cat class to create and interact with a cat object.

Відповіді на контрольні запитання

1. Синтаксис визначення класу.

```
- public class ClassName {  
    // Class members (fields, methods, constructors)  
}
```

2. Синтаксис визначення методу.

```
- public returnType methodName(parameters) {  
    // Method body  
}
```

3. Синтаксис оголошення поля.

```
- accessModifier dataType fieldName;
```

4. Як оголосити та ініціалізувати константне поле?

```
- public static final dataType CONSTANT_NAME = initial_value;
```

5. Які є способи ініціалізації полів?

- Явна ініціалізація при оголошенні поля.
- Ініціалізація у конструкторі класу.
- Ініціалізація у блоку ініціалізації (конструкторі, статичному або звичайному).

6. Синтаксис визначення конструктора.

```
- public ClassName(parameters) {  
    // Constructor body  
}
```

7. Синтаксис оголошення пакету.

```
- package packageName.subpackage;
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

- Вказати повне ім'я класу перед використанням (наприклад, `java.util.Date today = new java.util.Date();`).
- Використовувати оператор `import` для підключення класів з інших пакетів, щоб уникнути повторення повного імені класу.

9. В чому суть статичного імпорту пакетів?

- Статичний імпорт дозволяє підключити статичні методи і поля класів без повного імені класу.
- Завдяки статичному імпорту, можна використовувати статичні члени класу, не додаючи перед ними ім'я класу.

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

- Назви пакетів повинні відповідати структурі каталогів.
- Назви загальнодоступних класів повинні співпадати з назвами файлів, де вони розміщені.
- Після компіляції ієрархія каталогів проекту повинна відповідати ієрархії пакетів.
- Для компіляції та запуску програми слід використовувати шляхи до файлів та пакетів.

Висновок

У ході виконання даної лабораторної роботи, я отримала цінні навички розробки класів та пакетів у мові програмування Java. Ця лабораторна робота надала мені можливість ознайомитися з базовими конструкціями Java, такими як оголошення класів, методів та полів. Я навчилася правильно структурувати свій код, визначати доступ до класів та їх членів, а також використовувати модифікатори доступу для керування видимістю.