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
Хмільовський С. Р.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 22)

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту **(Автомат)**. 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 AssaultRifle.java

```
//package AssaultRifle;

import java.io.IOException;
import java.io.PrintWriter;
import java.io.File;
```

```java
/**
 * The AssaultRifle class represents an assault rifle with various properties,
such as model, ammunition, caliber
 * weight, price, scope, muffler and enlargedMagazine.
 * It also provides methods for assault rifle operations like reloading,
 * shooting, changing caliber, installation of scope, installation of muffler,
installation of enlarged magazine,
 * and displaying gun information.
 *
 * @author Khmilovskiy Stanislaw
 * @version 1.0
 * @since 1.0
 */

public class AssaultRifle {
    private String model;
    private int ammunition;
    private double caliber;
    private double weight;
    private double price;
    private boolean scope;
    private boolean muffler;
    private boolean enlargedMagazine;
    private PrintWriter fout;

    /**
     * Constructs an AssaultRifle object with default values.
     *
     * @throws IOException if there is an error creating the log file.
     */
    public AssaultRifle() throws IOException {
        model = "";
        ammunition = 0;
        caliber = 0.0;
        weight = 0.0;
        price = 0.0;
        scope = false;
        muffler = false;
        enlargedMagazine = false;
        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Constructs an AssaultRifle object with the specified parameters.
     *
     * @param model              the model of the assault rifle.
     * @param ammunition         the initial ammunition count.
     * @param caliber            the caliber of the rifle.
     * @param weight             the weight of the rifle.
     * @param price              the price of the rifle.
     * @param scope              true if a scope is installed, false otherwise.
     * @param muffler            true if a muffler is installed, false
otherwise.
     * @param enlargedMagazine   true if an enlarged magazine is installed,
false otherwise.
     * @throws IOException if there is an error creating the log file.
     */
    public AssaultRifle(String model, int ammunition, double caliber, double
weight, double price, boolean scope, boolean muffler, boolean enlargedMagazine)
throws IOException {
        this.model = model;
        this.ammunition = ammunition;
        this.caliber = caliber;
        this.weight = weight;
        this.price = price;
```

```
        this.scope = scope;
        this.muffler = muffler;
        this.enlargedMagazine = enlargedMagazine;
        fout = new PrintWriter(new File("Log.txt"));
    }

    public void reload(int rounds) { //amount of bullets fired
        if (rounds > 0) {
            ammunition += rounds;
            System.out.println("Reloaded with " + rounds + " rounds.");
            logActivity("Reloaded with " + rounds + " rounds.");
        }
    }

    public void automaticFire(int bullets) {
        if (ammunition >= bullets) {
            ammunition -= bullets;
            for (int i = 0; i < bullets; i++) {
                System.out.println("Automatic Fire! (Bullet used: " + (i + 1) +
")");
            }
            System.out.println("Remaining ammunition: " + ammunition);
            logActivity("Fired " + bullets + " bullets in automatic mode.");
        } else {
            logActivity("Out of ammunition.");
        }
    }

    public void burstFire(int bursts) {
        int bulletsToFire = bursts * 3; // Кількість патронів для пострілу

        if (ammunition >= bulletsToFire) {
            ammunition -= bulletsToFire;

            for (int i = 0; i < bursts; i++) {
                System.out.println("Burst fire! (3 bullets used)");
            }
            System.out.println("Remaining ammunition: " + ammunition);
            logActivity("Fired " + bulletsToFire + " bullets in " + bursts + "
bursts.");
        } else {
            logActivity("Out of ammunition.");
        }
    }

    public void singleFire(int shots) {
        if (shots <= 0) {
            System.out.println("Invalid number of shots to fire.");
            return;
        }

        int bulletsFired = 0; // Лічильник кількості вистріляних куль

        if (ammunition >= shots) {
            ammunition -= shots;

            for (int i = 0; i < shots; i++) {
                System.out.println("Single shot! (1 bullet used)");
                bulletsFired++; // Додати 1 вистріл
            }
            System.out.println("Remaining ammunition: " + ammunition);
            logActivity("Fired " + bulletsFired + " bullets in " + shots + "
single shots.");
        } else {
            System.out.println("Not enough ammunition to fire " + shots + "
shots.");
```

```
            logActivity("Tried to fire " + shots + " shots, but ran out of
ammunition.");
        }
    }

    public void weaponScope(boolean scope, String scopeType) {
        if (scope) {
            System.out.println("Scope is installed: " + scopeType);
            logActivity("Scope is installed: " + scopeType);
        } else {
            System.out.println("No scope is installed.");
            logActivity("No scope is installed.");
        }
    }

    public void weaponMuffler(boolean muffler, String mufflerType) {
        if (muffler) {
            System.out.println("Muffler is installed: " + mufflerType);
            logActivity("Muffler is installed: " + mufflerType);
        } else {
            System.out.println("No muffler is installed.");
            logActivity("No muffler is installed.");
        }
    }

    public void weapoEnlargedMagazine(boolean enlargedMagazine, String
enlargedMagazineType) {
        if (enlargedMagazine) {
            System.out.println("Enlarged magazine is installed: " +
enlargedMagazineType);
            logActivity("Enlarged magazine is installed: " +
enlargedMagazineType);
        } else {
            System.out.println("No enlarged magazine is installed.");
            logActivity("No enlarged magazine is installed.");
        }
    }

    public void displayInfo() {
        System.out.println("\nModel: " + model);
        System.out.println("Ammunition: " + ammunition);
        System.out.println("Caliber: " + caliber);
        System.out.println("Weight: " + weight + " kg");
        System.out.println("Price: " + price + " $");
        System.out.println("Scope: " + scope);
        System.out.println("Muffler: " + muffler);
        System.out.println("Enlarged Magazine: " + enlargedMagazine);
    }

    public void setCaliber(double caliber) {
        this.caliber = caliber;
        System.out.println("Caliber set to " + caliber + " mm.");
        logActivity("Caliber set to " + caliber + " mm.");
    }

    private void logActivity(String message) {
        fout.println(message);
        fout.flush();
    }

    /**
     * Closes the log file.
     */
    public void closeLogFile() {
        fout.close();
    }
```

```
    public static void main(String[] args) throws IOException {
        AssaultRifle rifle = new AssaultRifle("AK-47", 30, 5.45, 1.2, 1299,
true, true, true);
        rifle.reload(10);
        rifle.automaticFire(10);
        System.out.println();
        rifle.burstFire(3);
        System.out.println();
        rifle.singleFire(21);
        System.out.println();
        rifle.setCaliber(7.62);
        rifle.weaponScope(true, "4x");
        rifle.weaponMuffler(true, "baffle");
        rifle.weapoEnlargedMagazine(true, "Default+10");
        rifle.closeLogFile();
    }
}
```

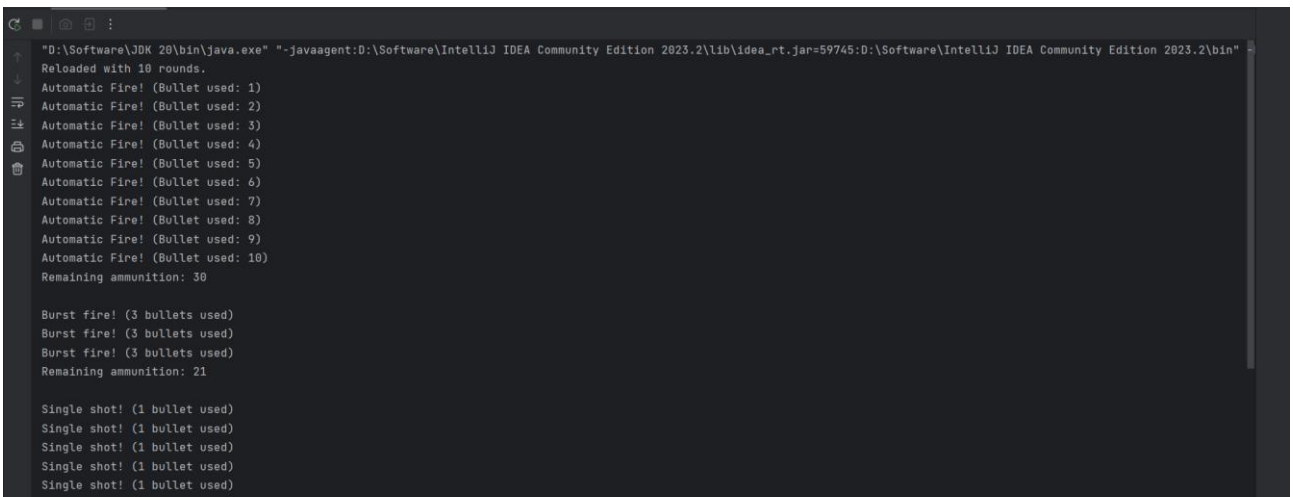
Файл AssaultRifleApp.java

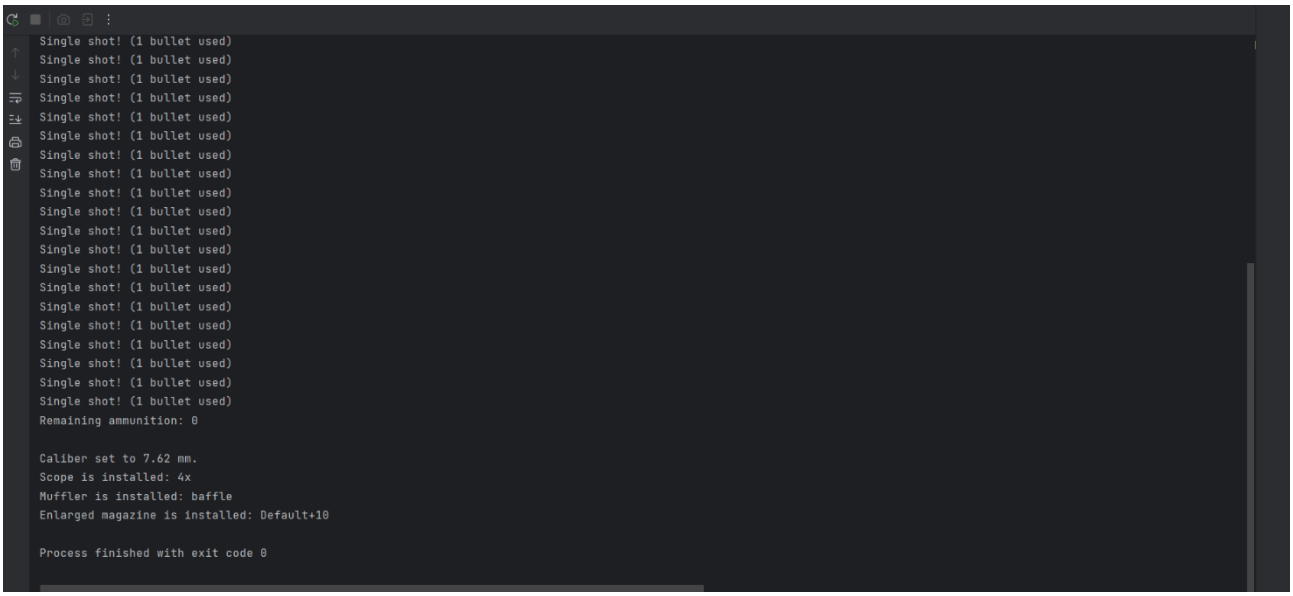
```
//package AssaultRifle;

import java.io.IOException;
import java.util.Scanner;

/**
 * The AssaultRifleApp class is a console application for simulating an assault
rifle.
 */

public class AssaultRifleApp {
    /**
     * The main method for running the AssaultRifleApp.
     *
     * @param args the command-line arguments (not used).
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Welcome to the Assault Rifle Simulator!\n");

        // Введення даних для створення об'єкта AssaultRifle
        System.out.print("Enter the assault rifle model: ");
        String model = scanner.nextLine();

        System.out.print("Enter the initial ammunition count: ");
        int ammunition = scanner.nextInt();

        System.out.print("Enter the caliber (e.g., 7.62): ");
        double caliber = scanner.nextDouble();

        System.out.print("Enter the weight (in kilograms): ");
        double weight = scanner.nextDouble();

        System.out.print("Enter the price (in dollars): ");
        double price = scanner.nextDouble();

        scanner.nextLine(); // Додатковий nextLine() для очищення буфера

        System.out.print("\nEnter the presence of a scope (true/false): ");
        String scopeInput = scanner.nextLine();
        boolean scope = Boolean.parseBoolean(scopeInput);

        System.out.print("Enter the presence of a muffler (true/false): ");
```

```
        String mufflerInput = scanner.nextLine();
        boolean muffler = Boolean.parseBoolean(mufflerInput);

        System.out.print("Enter the presence of a Enlarged Magazine
(true/false): ");
        String enlargedMagazineInput = scanner.nextLine();
        boolean enlargedMagazine = Boolean.parseBoolean(enlargedMagazineInput);

        // Створення об'єкту класу AssaultRifle з введеними даними
        AssaultRifle rifle = null;
        try {
            rifle = new AssaultRifle(model, ammunition, caliber, weight, price,
scope, muffler, enlargedMagazine);
        } catch (IOException e) {
            System.err.println("Error opening log file: " + e.getMessage());
            return;
        }

        // Основний цикл демонстрації
        boolean isRunning = true;
        while (isRunning) {
            System.out.println("\nAssault Rifle Simulator Menu:");
            System.out.println("1. Reload");
            System.out.println("2. Automatic Fire");
            System.out.println("3. Burst Fire");
            System.out.println("4. Single Fire");
            System.out.println("5. Set Caliber");
            System.out.println("6. Install scope");
            System.out.println("7. Install muffler");
            System.out.println("8. Install Enlarged Magazine");
            System.out.println("9. Display Info");
            System.out.println("0. Exit");
            System.out.print("\nEnter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter the number of bullets to reload: ");
                    int reloadRounds = scanner.nextInt();
                    rifle.reload(reloadRounds);
                    break;
                case 2:
                    System.out.print("Enter the number of rounds to fire: ");
                    int bulletsToFire = scanner.nextInt();
                    rifle.automaticFire(bulletsToFire);
                    break;
                case 3:
                    System.out.print("Enter the number of bursts to fire: ");
                    int bursts = scanner.nextInt();
                    rifle.burstFire(bursts);
                    break;
                case 4:
                    System.out.print("Enter the number of single shots to fire:
");
                    int shots = scanner.nextInt();
                    rifle.singleFire(shots);
                    break;
                case 5:
                    System.out.print("Enter the new caliber: ");
                    double newCaliber = scanner.nextDouble();
                    rifle.setCaliber(newCaliber);
                    break;
                case 6:
                    if (scope) {
                        System.out.print("Enter the type of new scope: ");
```

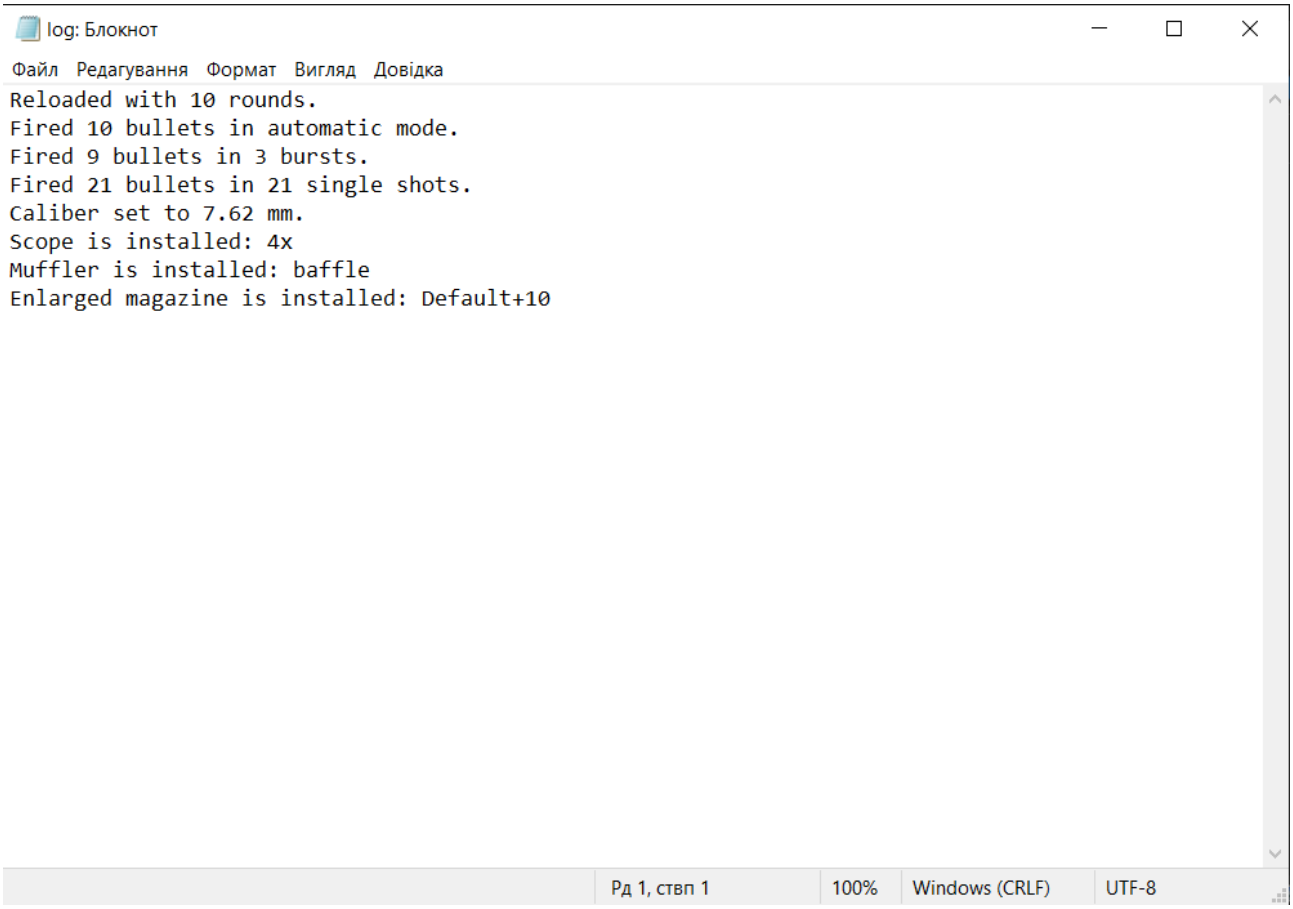
```
                        scanner.nextLine(); // Додатковий nextLine() для
очищення буфера
                        String newScope = scanner.nextLine();
                        rifle.weaponScope(true, newScope);
                    } else {
                        System.out.println("Cannot add a new scope because scope
is not installed.");
                    }
                    break;
                case 7:
                    if (muffler) {
                        System.out.print("Enter the type of new muffler: ");
                        scanner.nextLine(); // Додатковий nextLine() для
очищення буфера
                        String newMuffler = scanner.nextLine();
                        rifle.weaponMuffler(true, newMuffler);
                    } else {
                        System.out.println("Cannot add a new muffler because
muffler is not installed.");
                    }
                    break;
                case 8:
                    if (enlargedMagazine) {
                        System.out.print("Enter the type of new Enlarged
Magazine: ");
                        scanner.nextLine(); // Додатковий nextLine() для
очищення буфера
                        String newEnlargedMagazine = scanner.nextLine();
                        rifle.weapoEnlargedMagazine(true, newEnlargedMagazine);
                    } else {
                        System.out.println("Cannot add a new Enlarged Magazine
because Enlarged Magazine is not installed.");
                    }
                    break;
                case 9:
                    rifle.displayInfo();
                    break;
                case 0:
                    isRunning = false;
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }

        rifle.closeLogFile();
        scanner.close();
        System.out.println("Assault Rifle Simulator has exited.");
    }
}
```

Результат виконання програми:

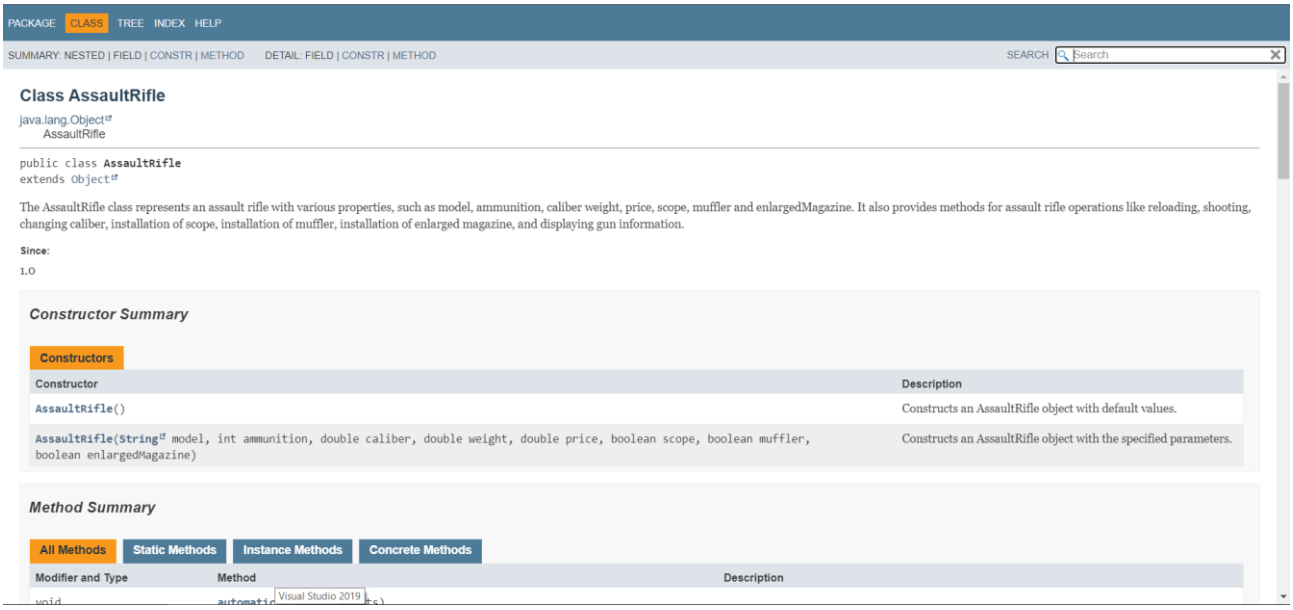




Вміст файлу log.txt:



Фрагмент згенерованої документації JavaDoc:



Відповіді на контрольні запитання:

1. Синтаксис визначення класу.

```
public class ClassName {
    // Вміст класу
}
```

2. Синтаксис визначення методу.

```
[модифікатори] [тип повернення] ім'яМетоду([параметри]) {
    // Тіло методу
}
```

3. Синтаксис оголошення поля.

```
[модифікатори] [тип] ім'яПоля;
```

4. Як оголосити та ініціалізувати константне поле?

```
[модифікатори] static final тип ІМЯ_КОНСТАНТИ = значення;
```

5. Які є способи ініціалізації полів?

- через конструктор
- в блоку ініціалізації

- при оголошенні або в методі.

6. Синтаксис визначення конструктора.

```
[модифікатори] Ім'яКласу([параметри]) {
   // Тіло конструктора
}
```

7. Синтаксис оголошення пакету.

```
package ім'яПакету;
```

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

Використовуючи імпорт: `import ім'яПакету.ім'яКласу;`

9. В чому суть статичного імпорту пакетів?

Суть статичного імпорту пакетів полягає в тому, що можна імпортувати статичні члени класу (методи або поля) і використовувати їх без префіксу класу.

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

Вимоги до файлів та каталогів при використанні пакетів включають в себе правильну структуру каталогів, іменування файлів та розташування пакетів у відповідних папках, що відповідають іменам пакетів.

Висновок:

На даній лабораторній роботі я отримав навички розробки класів та пакетів у мові програмування Java. Також на цій лабораторній роботі я ознайомився з базовими конструкціями Java, такими як оголошення класів, методів та полів. Було здобуто навички структурувати свій код, визначати доступ до класів та їх членів, а також використовувати модифікатори доступу для керування видимістю.