

ЗВІТ
до практичної роботи № 1
на тему:
"Навички об'єктно орієнтованого програмування."

Виконала:
студентка групи МІТ-21
Йовхимищ Діана

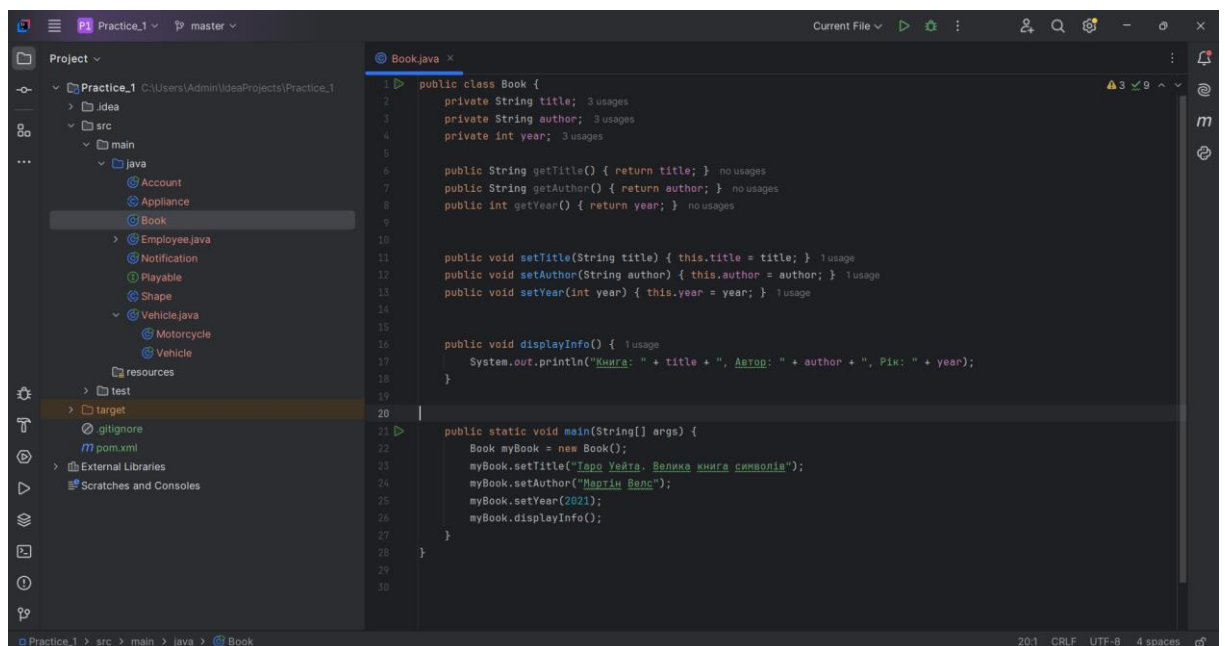
Допоміжні матеріали:

1. https://www.w3schools.com/java/java_oop.asp
2. https://docs.google.com/presentation/d/1kwpWhkCUPL_OGDqLUK8AB14p9I7iiq_t/edit#slide=id.p14

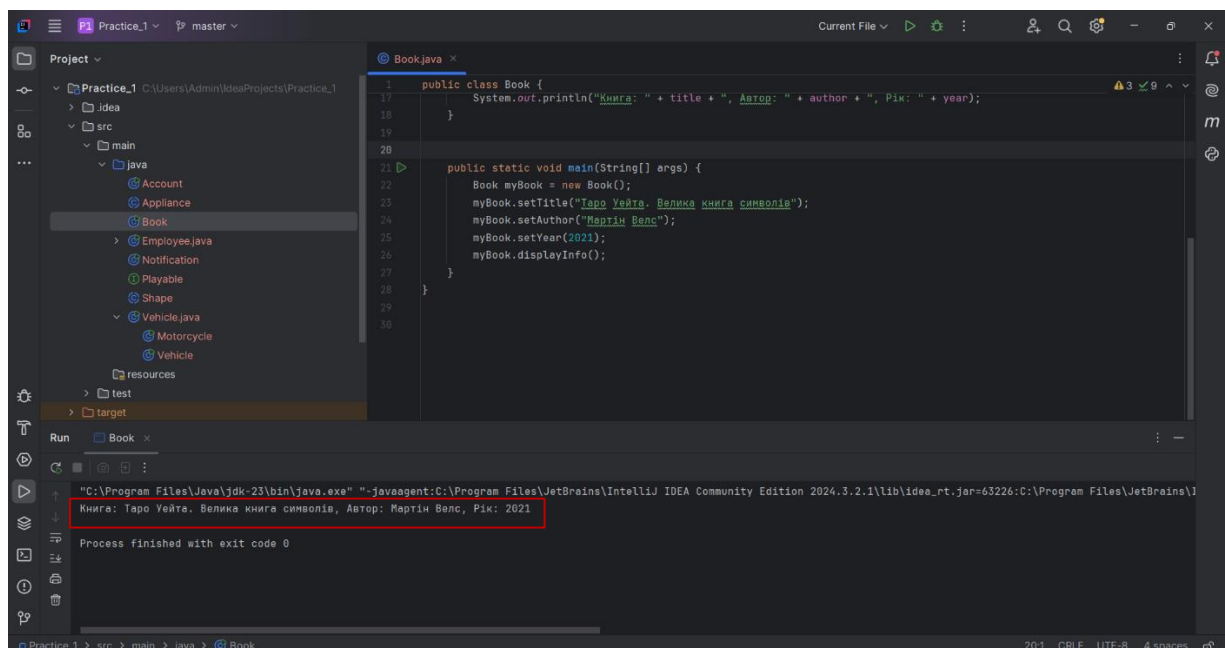
Завдання за темами:

1. Інкапсуляція

Завдання 1: Створити клас **Book** з приватними полями **title**, **author**, **year**. Реалізуйте геттери та сеттери для доступу до цих полів. Додайте метод для виведення інформації про книгу.



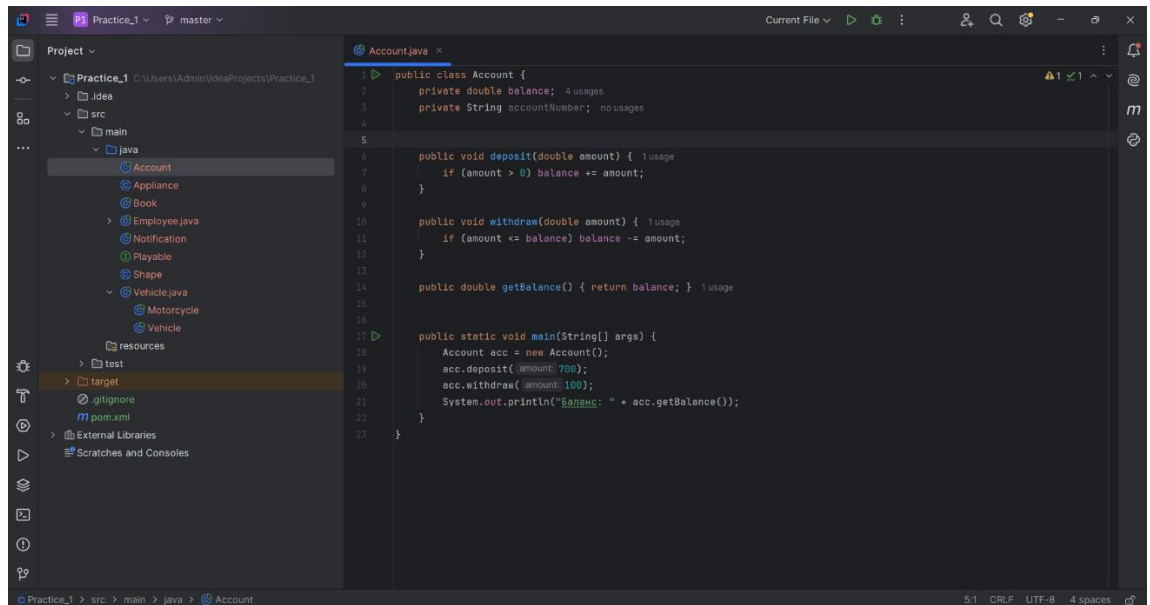
```
1 public class Book {
2     private String title;
3     private String author;
4     private int year;
5
6     public String getTitle() { return title; }
7     public String getAuthor() { return author; }
8     public int getYear() { return year; }
9
10    public void setTitle(String title) { this.title = title; }
11    public void setAuthor(String author) { this.author = author; }
12    public void setYear(int year) { this.year = year; }
13
14    public void displayInfo() {
15        System.out.println("Книга: " + title + ", Автор: " + author + ", Рік: " + year);
16    }
17
18    public static void main(String[] args) {
19        Book myBook = new Book();
20        myBook.setTitle("Таро Уейта. Велика книга символів");
21        myBook.setAuthor("Мартін Велс");
22        myBook.setYear(2021);
23        myBook.displayInfo();
24    }
25 }
```



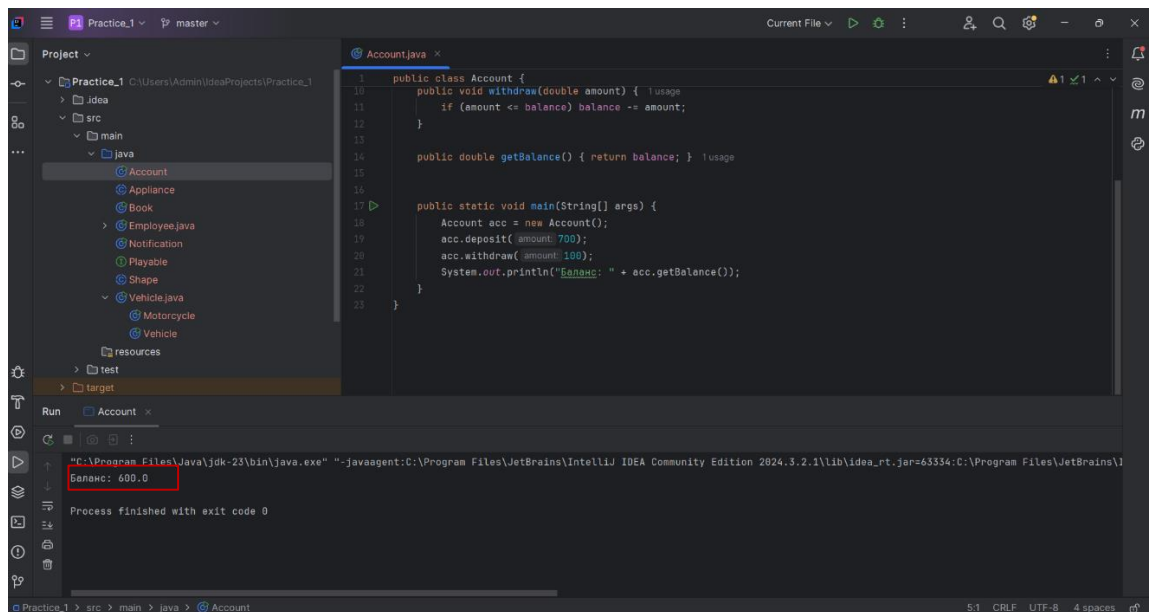
```
1 public class Book {
2     private String title;
3     private String author;
4     private int year;
5
6     public String getTitle() { return title; }
7     public String getAuthor() { return author; }
8     public int getYear() { return year; }
9
10    public void setTitle(String title) { this.title = title; }
11    public void setAuthor(String author) { this.author = author; }
12    public void setYear(int year) { this.year = year; }
13
14    public void displayInfo() {
15        System.out.println("Книга: " + title + ", Автор: " + author + ", Рік: " + year);
16    }
17
18    public static void main(String[] args) {
19        Book myBook = new Book();
20        myBook.setTitle("Таро Уейта. Велика книга символів");
21        myBook.setAuthor("Мартін Велс");
22        myBook.setYear(2021);
23        myBook.displayInfo();
24    }
25 }
```

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=63226:C:\Program Files\JetBrains\
Книга: Таро Уейта. Велика книга символів, Автор: Мартін Велс, Рік: 2021
Process finished with exit code 0
```

Завдання 2: Створіть клас **Account** з приватними полями **balance** та **accountNumber**. Додайте методи для поповнення, зняття коштів та перевірки балансу.



```
1 public class Account {
2     private double balance;
3     private String accountNumber;
4
5
6     public void deposit(double amount) {
7         if (amount > 0) balance += amount;
8     }
9
10    public void withdraw(double amount) {
11        if (amount <= balance) balance -= amount;
12    }
13
14    public double getBalance() { return balance; }
15
16
17    public static void main(String[] args) {
18        Account acc = new Account();
19        acc.deposit(700);
20        acc.withdraw(100);
21        System.out.println("Баланс: " + acc.getBalance());
22    }
23 }
```



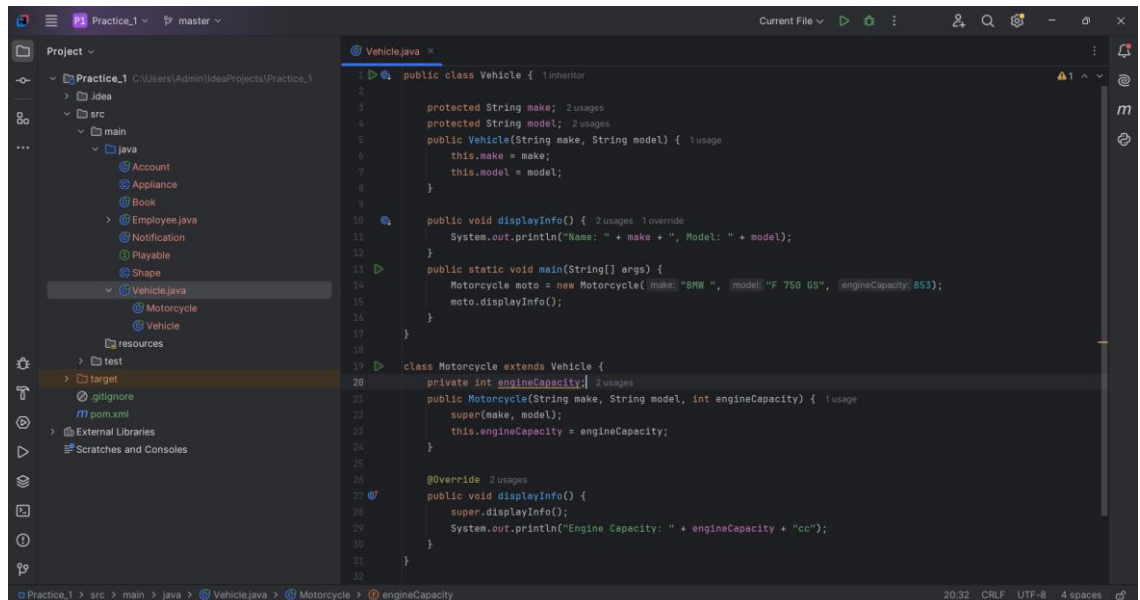
```
1 public class Account {
2     private double balance;
3     private String accountNumber;
4
5
6     public void deposit(double amount) {
7         if (amount > 0) balance += amount;
8     }
9
10    public void withdraw(double amount) {
11        if (amount <= balance) balance -= amount;
12    }
13
14    public double getBalance() { return balance; }
15
16
17    public static void main(String[] args) {
18        Account acc = new Account();
19        acc.deposit(700);
20        acc.withdraw(100);
21        System.out.println("Баланс: " + acc.getBalance());
22    }
23 }
```

Run Account

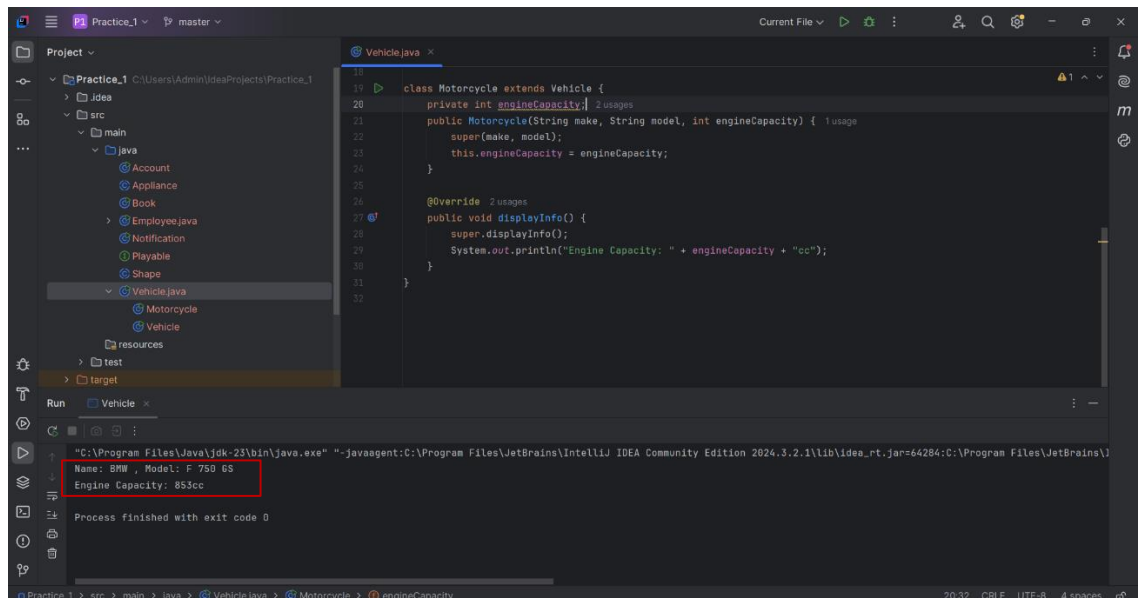
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=63334:C:\Program Files\JetBrains\I"
Баланс: 600.0
Process finished with exit code 0

2. Наслідування

Завдання 1: Створіть базовий клас **Vehicle** з полями **make**, **model** та методом **displayInfo()**. Створіть підклас **Motorcycle**, який успадковує **Vehicle** і додає поле **engineCapacity**. Перевизначте метод **displayInfo()** для виводу додаткової інформації. (@Override)



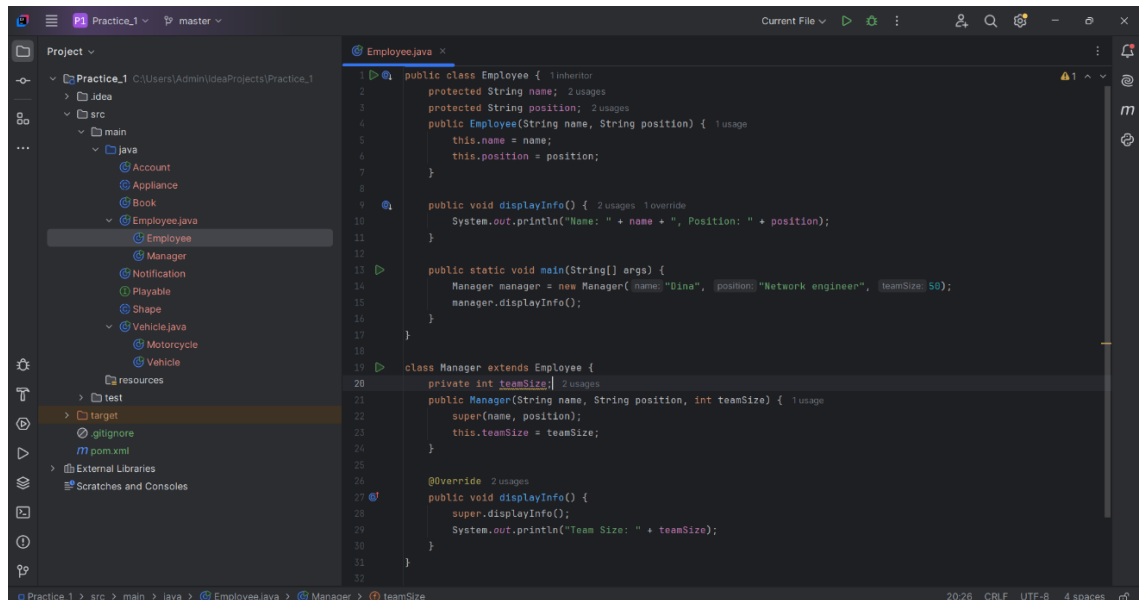
```
1 public class Vehicle { 1 inheritor
2
3     protected String make; 2 usages
4     protected String model; 2 usages
5     public Vehicle(String make, String model) { 1 usage
6         this.make = make;
7         this.model = model;
8     }
9
10    public void displayInfo() { 2 usages 1 override
11        System.out.println("Name: " + make + ", Model: " + model);
12    }
13    public static void main(String[] args) {
14        Motorcycle moto = new Motorcycle("BMW ", "F 750 GS", "engineCapacity: 853");
15        moto.displayInfo();
16    }
17 }
18
19 class Motorcycle extends Vehicle {
20     private int engineCapacity; 2 usages
21     public Motorcycle(String make, String model, int engineCapacity) { 1 usage
22         super(make, model);
23         this.engineCapacity = engineCapacity;
24     }
25
26     @Override 2 usages
27     public void displayInfo() {
28         super.displayInfo();
29         System.out.println("Engine Capacity: " + engineCapacity + "cc");
30     }
31 }
32
```



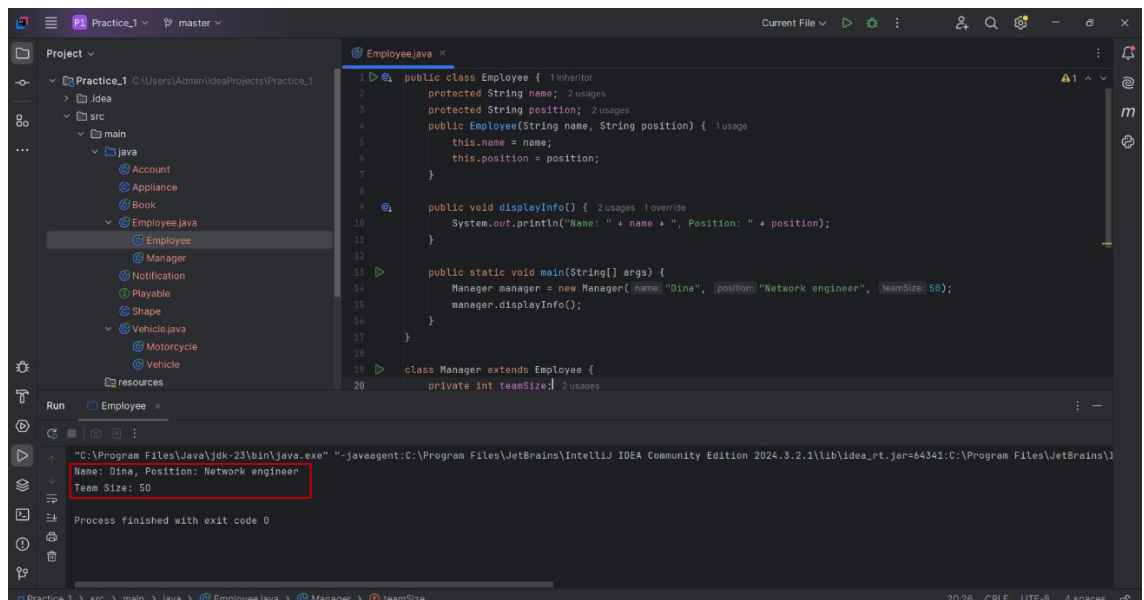
```
17
18
19 class Motorcycle extends Vehicle {
20     private int engineCapacity; 2 usages
21     public Motorcycle(String make, String model, int engineCapacity) { 1 usage
22         super(make, model);
23         this.engineCapacity = engineCapacity;
24     }
25
26     @Override 2 usages
27     public void displayInfo() {
28         super.displayInfo();
29         System.out.println("Engine Capacity: " + engineCapacity + "cc");
30     }
31 }
32
```

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2\lib\idea_rt.jar=64284:C:\Program Files\JetBrains\
Name: BMW , Model: F 750 GS
Engine Capacity: 853cc
Process finished with exit code 0
```

Завдання 2: Створіть клас **Employee** з полями **name**, **position**. Створіть підклас **Manager**, який додає поле **teamSize** та перевизначає метод **displayInfo()** для показу розміру команди.



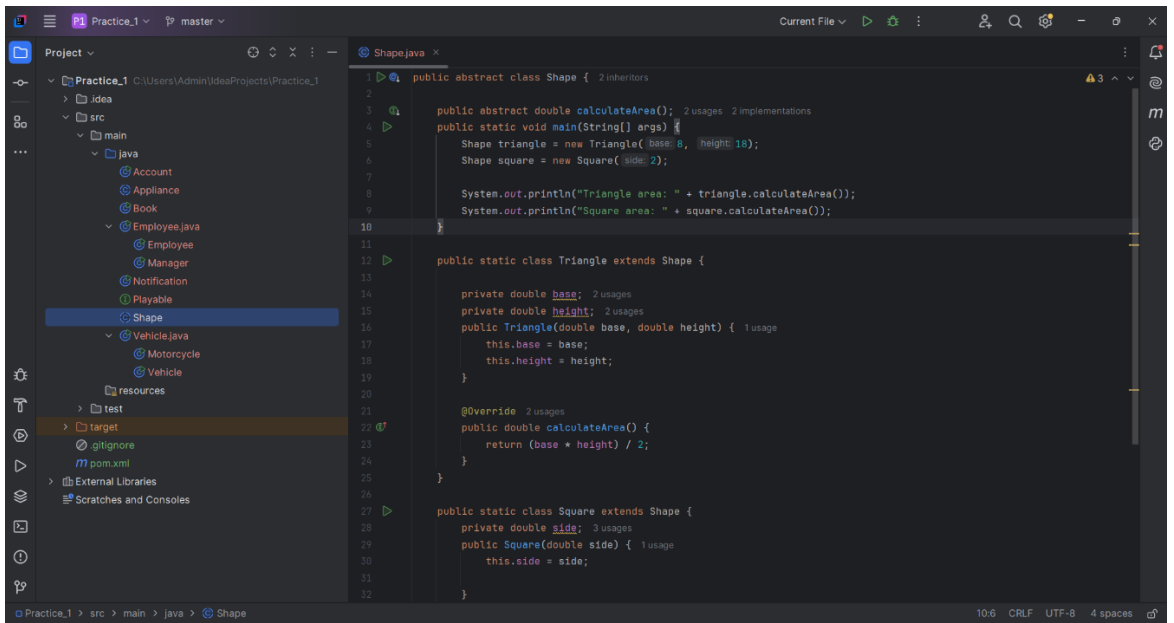
```
1 public class Employee { 1inheritor
2     protected String name; 2 usages
3     protected String position; 2 usages
4     public Employee(String name, String position) { 1usage
5         this.name = name;
6         this.position = position;
7     }
8
9     public void displayInfo() { 2 usages 1override
10        System.out.println("Name: " + name + ", Position: " + position);
11    }
12
13    public static void main(String[] args) {
14        Manager manager = new Manager("Dina", "Network engineer", teamSize: 50);
15        manager.displayInfo();
16    }
17
18 }
19
20 class Manager extends Employee {
21     private int teamSize; 2 usages
22     public Manager(String name, String position, int teamSize) { 1usage
23         super(name, position);
24         this.teamSize = teamSize;
25     }
26
27     @Override 2 usages
28     public void displayInfo() {
29         super.displayInfo();
30         System.out.println("Team Size: " + teamSize);
31     }
32 }
```



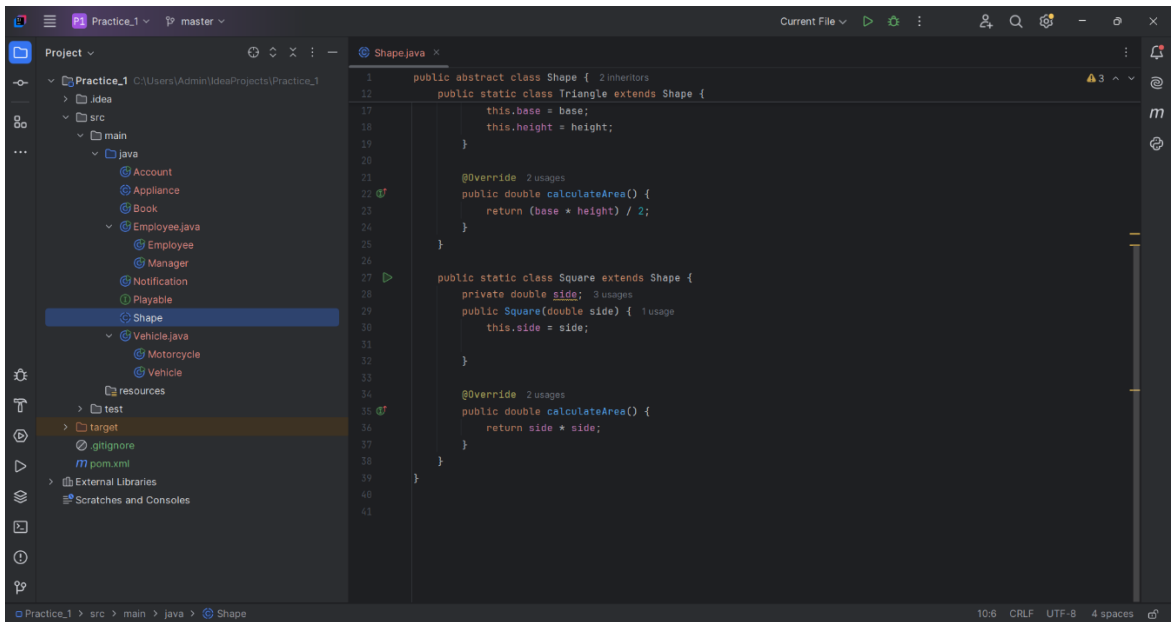
```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=64341:C:\Program Files\JetBrains\I
Team Size: 50
Process finished with exit code 0
```

3. Поліморфізм

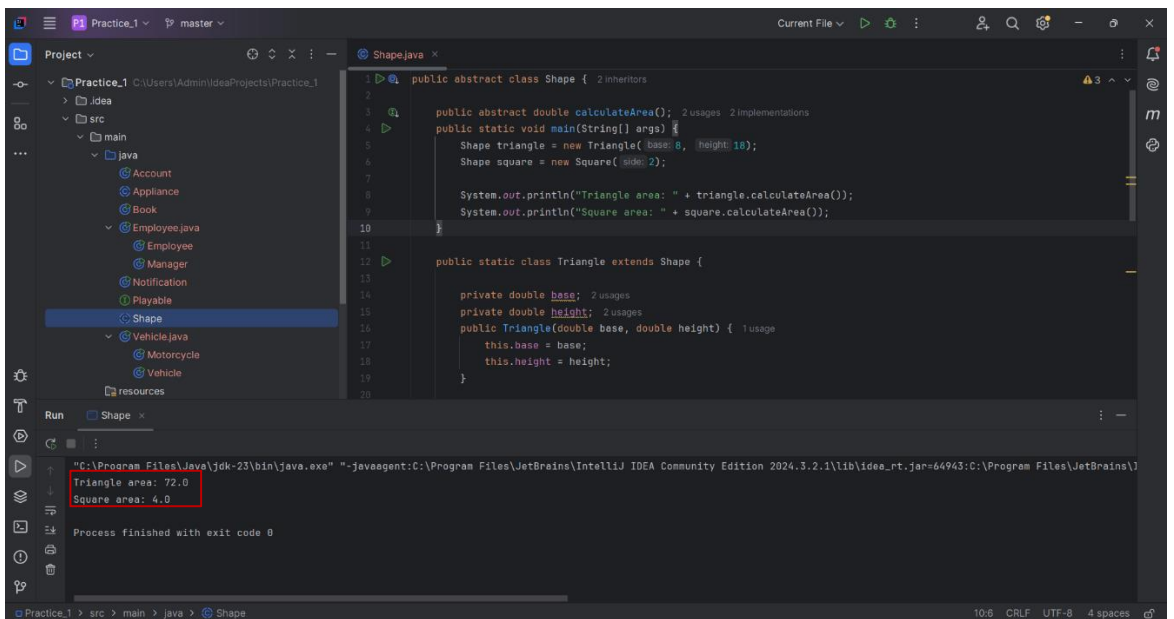
Завдання 1: Створіть базовий клас **Shape** з методом **calculateArea()**. Реалізуйте класи **Triangle** та **Square**, які успадковують **Shape** і реалізують власні версії методу для обчислення площі.



```
1 public abstract class Shape { 2 inheritors
2
3     public abstract double calculateArea(); 2 usages 2 implementations
4     public static void main(String[] args) {
5         Shape triangle = new Triangle(8, 18);
6         Shape square = new Square(2);
7
8         System.out.println("Triangle area: " + triangle.calculateArea());
9         System.out.println("Square area: " + square.calculateArea());
10    }
11
12    public static class Triangle extends Shape {
13
14        private double base; 2 usages
15        private double height; 2 usages
16        public Triangle(double base, double height) { 1 usage
17            this.base = base;
18            this.height = height;
19        }
20
21        @Override 2 usages
22        public double calculateArea() {
23            return (base * height) / 2;
24        }
25    }
26
27    public static class Square extends Shape {
28        private double side; 3 usages
29        public Square(double side) { 1 usage
30            this.side = side;
31        }
32    }
```



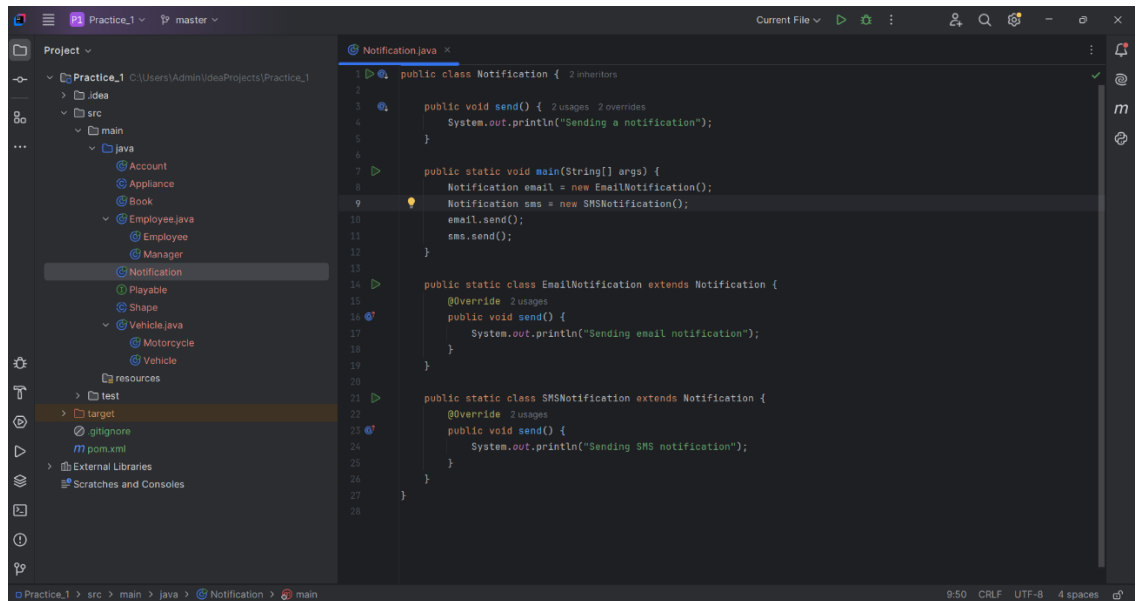
```
1 public abstract class Shape { 2 inheritors
2     public static class Triangle extends Shape {
3         this.base = base;
4         this.height = height;
5     }
6
7     @Override 2 usages
8     public double calculateArea() {
9         return (base * height) / 2;
10    }
11
12    public static class Square extends Shape {
13        private double side; 3 usages
14        public Square(double side) { 1 usage
15            this.side = side;
16        }
17
18        @Override 2 usages
19        public double calculateArea() {
20            return side * side;
21        }
22    }
23
24    }
25
26    }
27
28    }
29
30    }
31
32    }
33
34    }
35
36    }
37
38    }
39
40    }
41    }
```



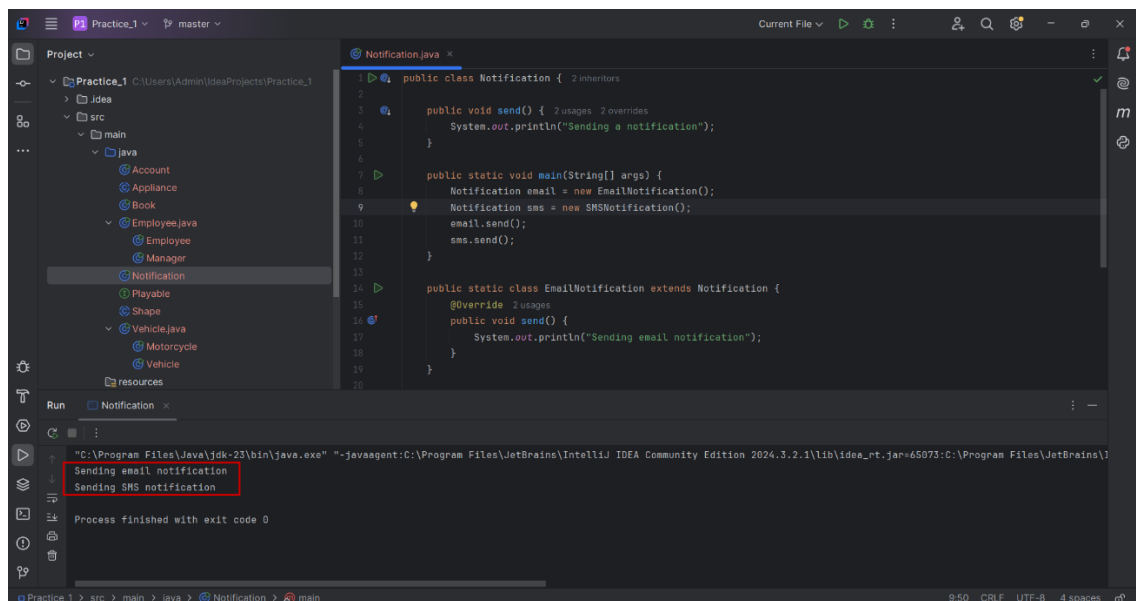
```
1 public abstract class Shape { 2 inheritors
2
3     public abstract double calculateArea(); 2 usages 2 implementations
4     public static void main(String[] args) {
5         Shape triangle = new Triangle(8, 18);
6         Shape square = new Square(2);
7
8         System.out.println("Triangle area: " + triangle.calculateArea());
9         System.out.println("Square area: " + square.calculateArea());
10    }
11
12    public static class Triangle extends Shape {
13
14        private double base; 2 usages
15        private double height; 2 usages
16        public Triangle(double base, double height) { 1 usage
17            this.base = base;
18            this.height = height;
19        }
20    }
```

```
Run Shape
C:\Program Files\Java\jdk-23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=64943:C:\Program Files\JetBrains\
Triangle area: 72.0
Square area: 4.0
Process finished with exit code 0
```

Завдання 2: Створіть клас **Notification** з методом **send()**. Створіть підкласи **EmailNotification** та **SMSNotification**, які перевизначають метод **send()** відповідно до типу повідомлення.



```
1 public class Notification { 2 inheritors
2
3     public void send() { 2 usages 2 overrides
4         System.out.println("Sending a notification");
5     }
6
7     public static void main(String[] args) {
8         Notification email = new EmailNotification();
9         Notification sms = new SMSNotification();
10        email.send();
11        sms.send();
12    }
13
14    public static class EmailNotification extends Notification {
15        @Override 2 usages
16        public void send() {
17            System.out.println("Sending email notification");
18        }
19    }
20
21    public static class SMSNotification extends Notification {
22        @Override 2 usages
23        public void send() {
24            System.out.println("Sending SMS notification");
25        }
26    }
27
28 }
```



```
1 public class Notification { 2 inheritors
2
3     public void send() { 2 usages 2 overrides
4         System.out.println("Sending a notification");
5     }
6
7     public static void main(String[] args) {
8         Notification email = new EmailNotification();
9         Notification sms = new SMSNotification();
10        email.send();
11        sms.send();
12    }
13
14    public static class EmailNotification extends Notification {
15        @Override 2 usages
16        public void send() {
17            System.out.println("Sending email notification");
18        }
19    }
20
21    public static class SMSNotification extends Notification {
22        @Override 2 usages
23        public void send() {
24            System.out.println("Sending SMS notification");
25        }
26    }
27
28 }
```

Run console output:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=65073:C:\Program Files\JetBrains\
Sending email notification
Sending SMS notification
Process finished with exit code 0
```

4. Абстрактні класи та інтерфейси

Завдання 1: Створіть абстрактний клас **Appliance** з абстрактним методом **turnOn()**. Створіть класи **WashingMachine** і **Microwave**, які реалізують цей метод.

```
1 public abstract class Appliance { 2 inheritors
2
3     public abstract void turnOn(); 2 usages 2 implementations
4
5     public static void main(String[] args) {
6         Appliance washingMachine = new WashingMachine();
7         Appliance microwave = new Microwave();
8         washingMachine.turnOn();
9         microwave.turnOn();
10    }
11
12    public static class Microwave extends Appliance {
13        @Override 2 usages
14        public void turnOn() {
15            System.out.println("Microwave is turned on");
16        }
17    }
18
19    public static class WashingMachine extends Appliance {
20        @Override 2 usages
21        public void turnOn() {
22            System.out.println("Washing machine is turned on");
23        }
24    }
25
26
27
28
29
```

```
1 public abstract class Appliance { 2 inheritors
12    public static class Microwave extends Appliance {
13    }
14
15
16
17
18
19
20    public static class WashingMachine extends Appliance {
21        @Override 2 usages
22        public void turnOn() {
23            System.out.println("Washing machine is turned on");
24        }
25    }
26
27
28
29
```

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.1\lib\idea_rt.jar=65127:C:\Program Files\JetBrains\I
Washing machine is turned on
Microwave is turned on
Process finished with exit code 0
```

Завдання 2: Створіть інтерфейс **Playable** з методом **play()**. Реалізуйте інтерфейс у класах **Guitar** та **Piano**, де метод **play()** відображатиме відповідне повідомлення.

The screenshot shows the IntelliJ IDEA IDE with a project named 'Practice_1'. The left sidebar displays the project structure, including a 'main' directory with a 'java' package containing several classes, including 'Playable'. The main editor window shows the 'Playable.java' file. It defines a 'Playable' interface with a 'play()' method. Two classes, 'Guitar' and 'Piano', implement this interface. The 'Guitar' class overrides 'play()' to print 'Guitar is playing', and the 'Piano' class overrides it to print 'Piano is playing'. A 'main' method in the 'Playable' interface creates instances of both classes and calls their 'play()' methods.

This screenshot shows the same IntelliJ IDEA IDE, but now the 'Run' console at the bottom is visible. It displays the output of the program execution: 'Guitar is playing' followed by 'Piano is playing'. The console also shows the command used to run the program and the exit code '0', indicating successful execution.

Висновок

У ході практичної роботи було реалізовано вісім завдань, що охоплюють основні принципи об'єктно-орієнтованого програмування: інкапсуляцію, наслідування, поліморфізм та абстракцію.

Завдяки використанню інкапсуляції вдалося приховати внутрішні дані класів, забезпечивши доступ до них через методи. Наслідування дозволило створювати похідні класи, які розширюють функціональність базових, а перевизначення методів забезпечило їхню гнучкість. Поліморфізм сприяв використанню спільного інтерфейсу для різних реалізацій, а абстракція допомогла виділити загальні характеристики об'єктів.

Під час виконання завдань виникли певні труднощі, зокрема у виборі між абстрактними класами та інтерфейсами, реалізації перевірки валідності даних та правильному

використанні `@Override`. Однак їхнє вирішення сприяло глибшому розумінню концепцій ООП.

Тож, виконана робота дозволила закріпити на практиці основні принципи ООП та їх ефективне використання в розробці програмних рішень.