

Міністерство освіти і науки України Національний
університет «Львівська політехніка»



Звіт

до лабораторної роботи №4

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Спадкування та інтерфейси»

Варіант 15

Виконав:
Ст. групи КІ-34
Ольховик О.С.

Прийняв:
к.т.н., доцент
Іванов Ю.С.

Львів 2022

Мета: ознайомитися зі спадкуванням та інтерфейсами на мові Java.

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант завдання: Фрегат

Код роботи

FreightApp.java

```
import java.util.Scanner;

import KI34.Olkhovik.Lab4.Freight;

public class FreightApp {

    public static void main(String[] args) {
        Freight freight = new Freight("ЧОВНИК", 25, 150, 23, 160);
        freight.GetAllInfo();
        freight.SetCurrentSpeed(23);
        System.out.println("Вирушаємо?");
        Scanner in = new Scanner(System.in);
        Boolean value = in.nextBoolean();
        if(value == true){
            System.out.println("Скільки годин рухатись?");
            int hours = in.nextInt();
            freight.Move(hours);
            freight.GetAllInfo();
            System.out.println("Пройдена дистанція в метрах/вузлах:");
            "+freight.GetPassedDistance());
        }
        else{
            System.out.println("Ну ні так ні");
        }
        in.close();
    }
}
```

Ship.java

```
package KI34.Olkhovyk.Lab4;

public abstract class Ship implements Movable{
    protected int maxSpeed;
    protected int currentSpeed;
    protected double maxFuelVolume;
    protected double CurrentFuelVolume;
    protected double fuelConsumption;
    protected double passedDistance;

    public Ship() {
        this.currentSpeed = 0;
    }

    public Ship(int maxSpeed) {
        this.maxSpeed = maxSpeed;
    }

    public Ship(int maxSpeed, double CurrentFuelVolume) {
        this(maxSpeed);
        this.CurrentFuelVolume = CurrentFuelVolume;
    }

    public Ship(int maxSpeed, double CurrentFuelVolume, double fuelConsumption) {
        this(maxSpeed, CurrentFuelVolume);
        this.fuelConsumption = fuelConsumption;
    }

    public Ship(int maxSpeed, double CurrentFuelVolume, double fuelConsumption,
double maxFuelVolume) {
        this(maxSpeed, CurrentFuelVolume, fuelConsumption);
        this.maxFuelVolume = maxFuelVolume;
    }

    public int GetMaxSpeed() {
        return this.maxSpeed;
    }

    public int GetCurrentSpeed() {
        return this.currentSpeed;
    }

    public double GetMaxFuelVolume() {
        return this.maxFuelVolume;
    }

    public double GetCurrentFuelVolume() {
        return this.CurrentFuelVolume;
    }

    public double GetFuelConsumption() {
        return this.fuelConsumption;
    }
}
```

```

    }

    public String GetPassedDistance() {
        return String.valueOf(this.passedDistance * 1852) + "/" +
String.valueOf(this.passedDistance);
    }

    public void SetCurrentSpeed(int speed) {
        if (speed > this.maxSpeed)
            this.currentSpeed = this.maxSpeed;
        this.currentSpeed = speed;
    }

    protected abstract void BurningFuel();

    protected abstract void SendData();
}

```

Fregate.java

```

package KI34.Olkhovyk.Lab4;

import static java.lang.System.out;
import java.io.FileWriter;
import java.io.IOException;

public class Fregate extends Ship {

    private String name;

    public Fregate(String name,int maxSpeed, double CurrentFuelVolume, double
fuelConsumption, double maxFuelVolume){
        super(maxSpeed, CurrentFuelVolume, fuelConsumption, maxFuelVolume);
        this.name = name;
    }

    public String GetName() { return this.name; }

    @Override
    protected void BurningFuel() {
        {
            if (this.currentSpeed != this.maxSpeed) {
                double coef = this.maxSpeed / this.currentSpeed;
                this.fuelConsumption /= coef;
            }
            this.CurrentFuelVolume -= this.fuelConsumption;
            if (this.CurrentFuelVolume <= 0)
                this.CurrentFuelVolume = 0;
        }
    }
}

```

```

    }
    public void GetAllInfo(){
        out.println("Назва фрегату: " + GetName());
        out.println("Максимальна швидкість: " + GetMaxSpeed());
        out.println("Максимальний об'єм палива: " + GetMaxFuelVolume());
        out.println("Об'єм палива на даний момент: " + GetCurrentFuelVolume());
    }
    @Override
    public void Move(int cycles) {
        while (cycles != 0) {
            BurningFuel();
            this.passedDistance += this.currentSpeed;
            if (this.CurrentFuelVolume == 0)
                break;
            cycles--;
        }
        SendData();
    }

    @Override
    public void SendData() {

        try (FileWriter writer = new FileWriter("Lab4.txt"))
        {
            writer.write("Пройдена дистанція: " + GetPassedDistance() + "\n");
            writer.write("Максимальна швидкість човна: " + GetMaxSpeed() + "\n");
            writer.write("Залишилось пального: " + GetCurrentFuelVolume() + "\n");
            writer.flush();
            writer.close();
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
}

```

Movable.java

```

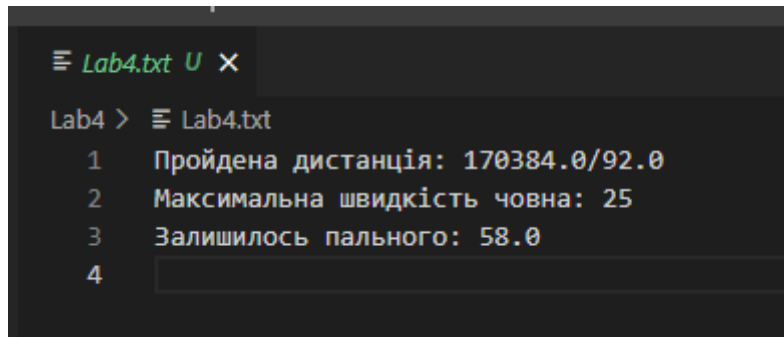
package KI34.Olkhovyk.Lab4;

public interface Movable {
    void Move(int cycles);
}

```

Результат виконання програми

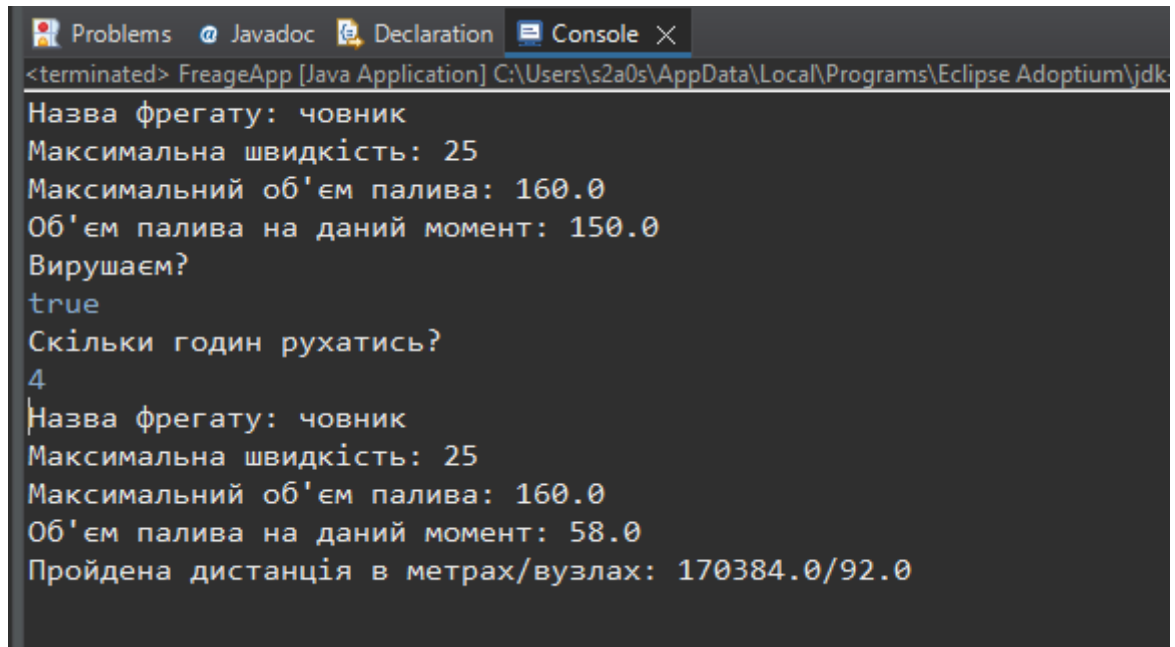
- Файл Lab4.txt



The screenshot shows a text editor window titled 'Lab4.txt'. The content of the file is as follows:

```
Lab4 > Lab4.txt
1  Пройдена дистанція: 170384.0/92.0
2  Максимальна швидкість човна: 25
3  Залишилось пального: 58.0
4  
```

- Консоль



The screenshot shows the Eclipse IDE console window. The title bar includes 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output is as follows:

```
<terminated> FreageApp [Java Application] C:\Users\s2a0s\AppData\Local\Programs\Eclipse Adoptium\jdk
Назва фрегату: човник
Максимальна швидкість: 25
Максимальний об'єм палива: 160.0
Об'єм палива на даний момент: 150.0
Вирушаєм?
true
Скільки годин рухатись?
4
Назва фрегату: човник
Максимальна швидкість: 25
Максимальний об'єм палива: 160.0
Об'єм палива на даний момент: 58.0
Пройдена дистанція в метрах/вузлах: 170384.0/92.0
```

Відповіді на контрольні запитання:

1. class Підклас extends Суперклас {
 Додаткові поля і методи
}
2. Найчастіше супер-клас – це базовий клас, а підклас – це похідний клас від суперкласу.
3. Для звернення до методів чи полів суперкласу з підкласу потрібно використати ключове слово *super*.
 super.назваМетоду([параметри]); // виклик методу суперкласу
 super.назваПоля // звертання до поля суперкласу
4. Статичне зв'язування використовується при поліморфізмі. (компіляція).
 Лише тоді, коли метод є приватним, статичним або конструктором.
5. Поліморфізм реалізується за допомогою механізму динамічного (пізнього) зв'язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається *не на етапі компіляції*, а під час виконання програми.
6. Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об'єкти свого класу. Вони реалізуються за допомогою ключового слова `abstract`. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити).
7. Використовується для визначення типу об'єкта в момент виконання програми. Посилання на базовий клас.
8. При наслідуванні у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних. Перевизначення методу полягає у визначенні у підкласі методу з сигнатурою методу суперкласу. При виклику такого методу з-під об'єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод

суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку.

9. Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. Синтаксис оголошення інтерфейсів:

```
[public] interface НазваІнтерфейсу {  
    Прототипи методів та оголошення констант інтерфейсу  
}
```

Висновок: я ознайомився зі спадкуванням та інтерфейсами на мові Java.