Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



**Лабораторна робота №3**

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Cпадкування та інтерфейси»  
Варіант № 2

Виконав: ст. гр. КІ-304

Гірняк О.Р.

Прийняв:

Олексів М.В.

Львів – 2023

**Мета:** Ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Контрольні питання:**

1. Синтаксис реалізації спадкування.

Відповідь:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

2. Що таке суперклас та підклас?

Відповідь: Суперклас – батьківський клас. Підклас – дочірній.

3. Як звернутися до членів суперкласу з підкласу?

Відповідь: super.назваМетоду([параметри]); super.назваПоля;

4. Коли використовується статичне зв’язування при виклику методу?

Відповідь: метод є приватним, статичним, фінальним або конструктором. Механізм статичного зв’язування передбачає визначення методу, який необхідно викликати, на етапі компіляції.

5. Як відбувається динамічне зв’язування при виклику методу?

Відповідь: метод, що необхідно викликати, визначається по фактичному типу

неявного параметру.

6. Що таке абстрактний клас та як його реалізувати?

Відповідь: Це клас який оголошений з ключовим словом abstract. Об’єкт такого класу не може бути створеним, може вміщати абстрактні методи.

7. Для чого використовується ключове слово instanceof?

Відповідь: Для встановлення чи є певний клас спадкоємцем другого.

8. Як перевірити чи клас є підкласом іншого класу?

Відповідь: використати ключове слово instanceof.

9. Що таке інтерфейс?

Відповідь: 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. Як оголосити та застосувати інтерфейс?

Відповідь: [public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

Застосувати можна імплементуючи його, або створюючи посилання на дочірній об’єкт класу.

**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.

**Код MultiplyStarShip.java**

package lab3Hirniak304;

import lab3Hirniak304.MultiplyStarShip.StarShipColor;

public class MultiplyStarShipApp {

    public static void main(String[] args) {

        MultiplyStarShip multiplyStarShip=new MultiplyStarShip();

        System.out.println(multiplyStarShip.getStarShipName());

        System.out.println(multiplyStarShip.getStarShipColor());

        System.out.println(multiplyStarShip.checkEngineWear());

        System.out.println(multiplyStarShip.checkFuel());

        multiplyStarShip.refuel();

        multiplyStarShip.repair();

        System.out.println(multiplyStarShip.checkEngineWear());

        System.out.println(multiplyStarShip.checkFuel());

        MultiplyStarShip multiplyStarShip1=new MultiplyStarShip("MotherShip",StarShipColor.RED,-1,-1);

        System.out.println(multiplyStarShip1.getStarShipName());

        System.out.println(multiplyStarShip1.getStarShipColor());

        System.out.println(multiplyStarShip1.checkEngineWear());

        System.out.println(multiplyStarShip1.checkFuel());

        multiplyStarShip1.refuel();

        multiplyStarShip1.repair();

        System.out.println(multiplyStarShip1.checkEngineWear());

        System.out.println(multiplyStarShip1.checkFuel());

    }

}

**Код ClimateControlDevice.java:**

package lab3Hirniak304;

interface EngineWearConrolInterface{

    double checkEngineWear();

}

interface FuelControlInterface{

    double checkFuel();

}

public class MultiplyStarShip implements EngineWearConrolInterface, FuelControlInterface{

    //Data fields

    private String starShipName;

    private double engineWear;

    private double fuel;

    private StarShipColor starShipColor;

    private static int StarShipNumber = 0;

    //Constructors

    public MultiplyStarShip(){

        starShipName = "MultiplyStarShip";

        starShipColor = StarShipColor.RED;

        fuel = 234.0;

        engineWear= 145.0;

        ++StarShipNumber;

    }

    public MultiplyStarShip(String starShipName){

        this.starShipName = starShipName;

        starShipColor = StarShipColor.RED;

        fuel = 1000.0;

        engineWear= 1000.0;

        ++StarShipNumber;

    }

    public MultiplyStarShip(String starShipName, StarShipColor starShipColor){

        this.starShipName = starShipName;

        this.starShipColor = starShipColor;

        fuel = 1000.0;

        engineWear= 1000.0;

        ++StarShipNumber;

    }

    public MultiplyStarShip(String starShipName, StarShipColor starShipColor, double fuel){

        this.starShipName = starShipName;

        this.starShipColor = starShipColor;

        this.fuel = fuel;

        engineWear= 1000.0;

        ++StarShipNumber;

    }

    public MultiplyStarShip(String starShipName, StarShipColor starShipColor, double fuel, double engineWear){

        this.starShipName = starShipName;

        this.starShipColor = starShipColor;

        this.fuel = fuel;

        this.engineWear= engineWear;

        ++StarShipNumber;

    }

    //enum

    public enum StarShipColor{

        WHITE, BLACK, RED, PINK, YELLOW, GREEN, BLUE

    }

    //Getter methods

    public String getStarShipName() {

        return "#"+StarShipNumber+" "+starShipName;

    }

    public double getFuel() {

        return fuel;

    }

    public double getEngineWear() {

        return engineWear;

    }

    public StarShipColor getStarShipColor() {

        return starShipColor;

    }

    //Overridden methods

    @Override

    public double checkFuel() {

        fuel -= 0.100;

        if(fuel - 0.100 < 0) {

            System.out.println("the starship has no fuel");

            return 0;

        }

        System.out.println("Engine wear: ");

        return fuel;

    }

    @Override

    public double checkEngineWear() {

        engineWear -= 0.100;

        if(engineWear - 0.100 < 0) {

            System.out.println("the starship has no fuel");

            return 0;

        }

        System.out.println("Fuel wear: ");

        return engineWear;

    }

    //refuel method

    public void refuel(){

        if(fuel == 1000.0)

            System.out.println("Is already full.");

        else

        {

            System.out.println("Fuel was reload ");

            fuel = 1000.0;

        }

    }

    public void repair(){

        if(engineWear == 1000.0)

            System.out.println("Is already full.");

        else

        {

            System.out.println("Engine was repair ");

            engineWear = 1000.0;

        }

    }

}

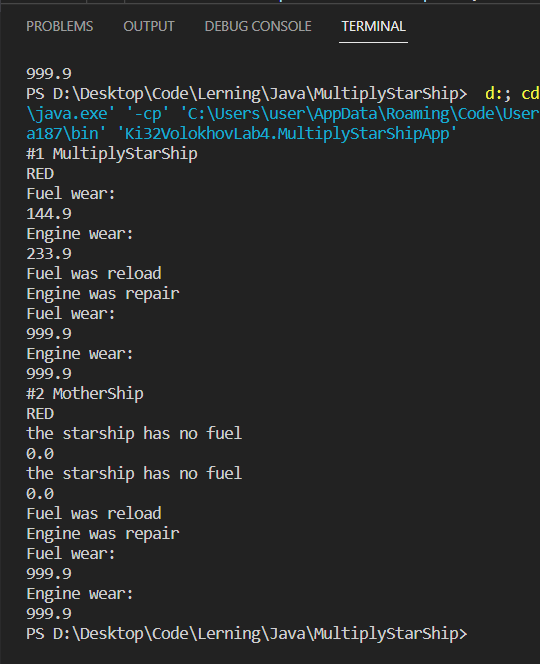


Рис. 1. Результат роботи програми.

**Висновок:** Я ознайомився з спадкуванням та інтерфейсами у мові Java.