Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

**Виконав:**

студент групи КІ-301

*Солтисяк О. Р.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2024

**Мета роботи:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Завдання (варіант № 23)**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі No2, для реалізації предметної області заданої варіантом **(Шлюпка 🡪Моторний човен)**. Суперклас, що реалізований у лабораторній роботі No2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
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:  
  
MotorBoatApp**

package Lab3;  
  
import static java.lang.System.*out*;  
import java.io.\*;  
  
public class MotorBoatApp {  
 public static void main(String[] args) throws FileNotFoundException {  
 Korpus korpus = new Korpus("Склопластик");  
 Sidinnya sidinnya = new Sidinnya(6);  
 Engine engine = new Engine(200);  
  
 MotorBoat motorBoat = new MotorBoat(korpus, sidinnya, engine);  
  
 motorBoat.start();  
 motorBoat.accelerate();  
 *out*.println("Поточна швидкість: " + motorBoat.getSpeed() + " км/год");  
 motorBoat.setDirection("Схід");  
 *out*.println("Напрямок руху: " + motorBoat.getDirection());  
 motorBoat.turnRight();  
 motorBoat.decelerate();  
 *out*.println("Поточна швидкість: " + motorBoat.getSpeed() + " км/год");  
 motorBoat.stop();  
  
 *out*.println("Матеріал корпусу: " + motorBoat.getHullMaterial());  
 *out*.println("Кількість сидінь: " + motorBoat.getNumberOfSeats());  
 *out*.println("Потужність двигуна: " + motorBoat.getEnginePower() + " к.с.");  
  
 motorBoat.dispose();  
 }  
}

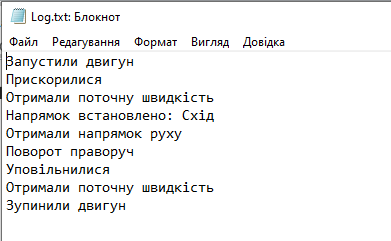
**Shlyubka**

package Lab3;  
  
import java.io.\*;  
  
public abstract class Shlyubka extends Object {  
 protected Korpus korpus;  
 protected Sidinnya sidinnya;  
 protected PrintWriter fout;  
 protected double speed;  
  
 public Shlyubka() throws FileNotFoundException {  
 korpus = new Korpus();  
 sidinnya = new Sidinnya();  
 speed = 0.0;  
 fout = new PrintWriter(new File("Log.txt"));  
 }  
  
 public abstract void start();  
 public abstract void stop();  
  
 public void turnLeft() {  
 fout.print("Поворот ліворуч\n");  
 fout.flush();  
 }  
  
 public void turnRight() {  
 fout.print("Поворот праворуч\n");  
 fout.flush();  
 }  
  
 public void accelerate() {  
 fout.print("Прискорилися\n");  
 fout.flush();  
 speed += 5.0;  
 }  
  
 public void decelerate() {  
 fout.print("Уповільнилися\n");  
 fout.flush();  
 speed -= 5.0;  
 if (speed < 0) speed = 0;  
 }  
  
 public double getSpeed() {  
 fout.print("Отримали поточну швидкість\n");  
 fout.flush();  
 return speed;  
 }  
  
 public String getHullMaterial() {  
 return korpus.getMaterial();  
 }  
  
 public int getNumberOfSeats() {  
 return sidinnya.getNumberOfSeats();  
 }  
  
 public void dispose() {  
 fout.close();  
 }  
}

**MotorBoat**

package Lab3;  
  
import java.io.\*;  
  
public class MotorBoat extends Shlyubka implements Navigable {  
 private final Engine engine;  
 private String direction;  
  
 public MotorBoat() throws FileNotFoundException {  
 super();  
 engine = new Engine();  
 direction = "Прямо";  
 }  
  
 public MotorBoat(Korpus k, Sidinnya s, Engine e) throws FileNotFoundException {  
 super();  
 korpus = k;  
 sidinnya = s;  
 engine = e;  
 direction = "Прямо";  
 }  
  
 @Override  
 public void start() {  
 fout.print("Запустили двигун\n");  
 fout.flush();  
 engine.startEngine();  
 speed = 10.0;  
 }  
  
 @Override  
 public void stop() {  
 fout.print("Зупинили двигун\n");  
 fout.flush();  
 engine.stopEngine();  
 speed = 0.0;  
 }  
  
 @Override  
 public void setDirection(String direction) {  
 this.direction = direction;  
 fout.print("Напрямок встановлено: " + direction + "\n");  
 fout.flush();  
 }  
  
 @Override  
 public String getDirection() {  
 fout.print("Отримали напрямок руху\n");  
 fout.flush();  
 return direction;  
 }  
  
 public int getEnginePower() {  
 return engine.getPower();  
 }  
}

**Вміст файлу log.txt:**



**Відповіді на контрольні запитання:**

1. Синтаксис реалізації спадкування.

class Subclass extends Superclass {

// код підкласу

}

1. Що таке суперклас та підклас?

Суперклас (або батьківський клас) - це клас, від якого спадковується інший клас, який називається підкласом (або дочірнім класом). Підклас успадковує всі поля і методи суперкласу і може розширювати їх або навіть перевизначити.

1. Як звернутися до членів суперкласу з підкласу?

Для звернення до членів суперкласу з підкласу можна використовувати ключове слово “super”. Наприклад, “super.methodName()” викликає метод суперкласу.

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

Статичне зв’язування (або раннє зв’язування) відбувається під час компіляції. Воно використовується, коли метод, який викликається, визначається на етапі компіляції на основі типу посилання.

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

Динамічне зв’язування (або пізнє зв’язування) відбувається під час виконання програми. Воно використовується, коли метод, який викликається, визначається на етапі виконання на основі реального типу об'єкта.

1. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас - це клас, який не може бути створений напряму, але може містити абстрактні (незавершені) методи. Щоб оголосити абстрактний клас, використовуйте ключове слово abstract.

Приклад реалізації:

abstract class MyAbstractClass {

// абстрактний метод

public abstract void myMethod();

}

1. Для чого використовується ключове слово instanceof?

Ключове слово instanceof використовується для перевірки, чи об'єкт є екземпляром певного класу або підкласу. Воно повертає true, якщо об'єкт є екземпляром класу, і false, якщо ні.

1. Як перевірити чи клас є підкласом іншого класу?

Використати ключове слово extends при оголошенні класу для показу спадкування.

1. Що таке інтерфейс?

Інтерфейс в Java - це абстрактний тип, який оголошує набір методів, але не містить їх реалізації. Класи можуть реалізовувати інтерфейси, надаючи конкретну реалізацію методів, вказаних в інтерфейсі.

1. Як оголосити та застосувати інтерфейс?

Щоб оголосити інтерфейс, використовуйте ключове слово interface.

Приклад реалізації:

interface MyInterface {

void myMethod();

}

Для застосування інтерфейсу в класі використовуйте ключове слово implements.

Приклад реалізації:

class MyClass implements MyInterface {

public void myMethod() {

// реалізація методу

}

}

**Висновок:**

На даній лабораторній роботі я отримав навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Також на цій лабораторній роботі я ознайомився з цими важливими аспектами об'єктно-орієнтованого програмування та зрозумів їх роль у створенні більш структурованих і гнучких програм.