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-301

Гавриляк Д. В.

Прийняв:

Майдан 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 №2 для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab2 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 програми**

package ki.havryliak.lab2;  
  
public interface FuelControlInterface {  
 double checkFuel();  
}

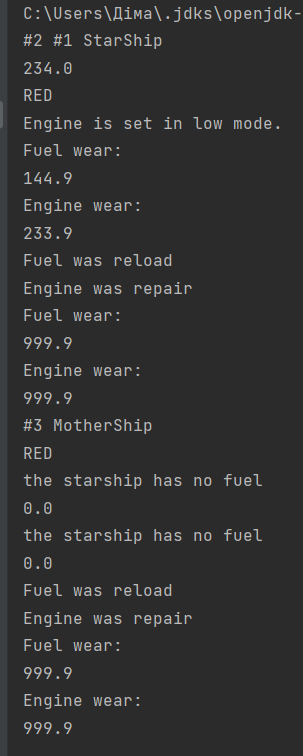
package ki.havryliak.lab2;  
  
public interface EngineWearConrolInterface {  
 double checkEngineWear();  
}

package ki.havryliak.lab2;  
  
import java.io.\*;  
  
public abstract class StarShip {  
 //Data fields  
 protected String starShipName;  
 protected StarShipColor starShipColor;  
 protected EngineStatus engineStatus;  
 protected static int *starShipNumber* = 1;  
 protected PrintWriter outputStream = new PrintWriter(String.*format*("StarShipLogger%s.txt", *starShipNumber*));  
  
 //Constructors  
 public StarShip() throws FileNotFoundException {  
 starShipName = String.*format*("#%s StarShip", *starShipNumber*);  
 starShipColor = StarShipColor.*RED*;  
 engineStatus = EngineStatus.*TURNED\_OFF*;  
 outputStream.println("Creating a StarShip");  
 ++*starShipNumber*;  
 }  
  
 public StarShip(String starShipName) throws FileNotFoundException {  
 this.starShipName = starShipName;  
 starShipColor = StarShipColor.*RED*;  
 engineStatus = EngineStatus.*TURNED\_OFF*;  
 outputStream.println("Creating a StarShip");  
 ++*starShipNumber*;  
 }  
  
 public StarShip(String starShipName, StarShipColor starShipColor) throws FileNotFoundException {  
 this.starShipName = starShipName;  
 this.starShipColor = starShipColor;  
 engineStatus = EngineStatus.*TURNED\_OFF*;  
 outputStream.println("Creating a StarShip");  
 ++*starShipNumber*;  
 }  
  
 public StarShip(String starShipName, StarShipColor starShipColor, EngineStatus engineStatus) throws FileNotFoundException {  
 this.starShipName = starShipName;  
 this.starShipColor = starShipColor;  
 this.engineStatus = engineStatus;  
 outputStream.println("Creating a StarShip");  
 ++*starShipNumber*;  
 }  
  
 //Enum  
 public enum EngineStatus {  
 *TURNED\_OFF*, *LOW*, *MEDIUM*, *HIGH* }  
  
 public enum StarShipColor {  
 *WHITE*, *BLACK*, *RED*, *PINK*, *YELLOW*, *GREEN*, *BLUE* }  
  
 //Getter methods  
 public String getStarShipName() {  
 outputStream.println("getStarShipName: " + starShipName);  
 return starShipName;  
 }  
  
 public EngineStatus getEngineStatus() {  
 outputStream.println("getEngineStatus: " + engineStatus);  
 return engineStatus;  
 }  
  
 public StarShipColor getStarShipColor() {  
 outputStream.println("getStarShipColor: " + starShipColor);  
 return starShipColor;  
 }  
  
 //Set methods  
  
 public abstract void turnOffEngine();  
  
 public abstract void setLowEngine();  
  
 public abstract void setMediumEngine();  
  
 public abstract void setHighEngine();  
  
 public void closeLoggerFile() {  
 outputStream.println("Close logger file.");  
 outputStream.close();  
 }  
}

package ki.havryliak.lab2;  
  
import java.io.FileNotFoundException;  
  
public class MultiplyStarShip extends StarShip implements EngineWearConrolInterface, FuelControlInterface {  
 //Data fields  
 private double engineWear;  
 private double fuel;  
  
 //Constructors  
 public MultiplyStarShip() throws FileNotFoundException {  
 super();  
 fuel = 234.0;  
 engineWear = 145.0;  
 }  
  
 public MultiplyStarShip(String starShipName) throws FileNotFoundException {  
 super();  
 this.starShipName = starShipName;  
 fuel = 1000.0;  
 engineWear = 1000.0;  
 }  
  
 public MultiplyStarShip(String starShipName, StarShipColor starShipColor) throws FileNotFoundException {  
 super();  
 this.starShipName = starShipName;  
 this.starShipColor = starShipColor;  
 fuel = 1000.0;  
 engineWear = 1000.0;  
 }  
  
 public MultiplyStarShip(String starShipName, StarShipColor starShipColor, double fuel) throws FileNotFoundException {  
 super();  
 this.starShipName = starShipName;  
 this.starShipColor = starShipColor;  
 this.fuel = fuel;  
 engineWear = 1000.0;  
 }  
  
 public MultiplyStarShip(String starShipName, StarShipColor starShipColor, double fuel, double engineWear) throws FileNotFoundException {  
 super();  
 this.starShipName = starShipName;  
 this.starShipColor = starShipColor;  
 this.fuel = fuel;  
 this.engineWear = engineWear;  
 }  
  
  
 //Getter methods  
 public String getStarShipName() {  
 return "#" + *starShipNumber* + " " + starShipName;  
 }  
  
 public double getFuel() {  
 return fuel;  
 }  
  
 public double getEngineWear() {  
 return engineWear;  
 }  
  
 //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;  
 }  
 }  
  
 @Override  
 public void turnOffEngine() {  
 if (engineStatus == EngineStatus.*TURNED\_OFF*) {  
 System.*out*.println("Engine is already turned off.");  
 outputStream.println("Engine is already turned off.");  
 } else {  
 System.*out*.println("Engine is turned off.");  
 outputStream.println("Engine is turned off.");  
 engineStatus = EngineStatus.*TURNED\_OFF*;  
 }  
 }  
  
 @Override  
 public void setLowEngine() {  
 if (engineStatus == EngineStatus.*LOW*) {  
 System.*out*.println("Engine is already in low mode.");  
 outputStream.println("Engine is already in low mode.");  
 } else {  
 System.*out*.println("Engine is set in low mode.");  
 outputStream.println("Engine is set in low mode.");  
 engineStatus = EngineStatus.*LOW*;  
 }  
 }  
  
 @Override  
 public void setMediumEngine() {  
 if (engineStatus == EngineStatus.*MEDIUM*) {  
 System.*out*.println("Engine is already in medium mode.");  
 outputStream.println("Engine is already in medium mode.");  
 } else {  
 System.*out*.println("Engine is set in medium mode.");  
 outputStream.println("Engine is set in medium mode.");  
 engineStatus = EngineStatus.*MEDIUM*;  
 }  
 }  
  
 @Override  
 public void setHighEngine() {  
 if (engineStatus == EngineStatus.*HIGH*) {  
 System.*out*.println("Engine is already in high mode.");  
 outputStream.println("Engine is already in high mode.");  
 } else {  
 System.*out*.println("Engine is set in high mode.");  
 outputStream.println("Engine is set in high mode.");  
 engineStatus = EngineStatus.*HIGH*;  
 }  
 }  
}

package ki.havryliak.lab2;  
  
import java.io.FileNotFoundException;  
  
public class MultiplyStarShipApp {  
 public static void main(String[] args) throws FileNotFoundException {  
 MultiplyStarShip multiplyStarShip = new MultiplyStarShip();  
 System.*out*.println(multiplyStarShip.getStarShipName());  
 System.*out*.println(multiplyStarShip.getFuel());  
 System.*out*.println(multiplyStarShip.getStarShipColor());  
 multiplyStarShip.setLowEngine();  
  
 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", MultiplyStarShip.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());  
 }  
}

**Результат роботи:**

****

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