Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ »

**Виконав:**

студент групи КІ-35

Ничай В.Б.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

   
Львів – 2022

 Мета: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Завдання

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.



Текст програми

Lab4.java

package Ship\_package;  
  
import java.io.FileNotFoundException;  
  
public class Lab4 {  
 */\*\*  
 \** ***@param*** *args  
 \** ***@throws*** *FileNotFoundException  
 \*/* public static void main(String []args) throws FileNotFoundException {  
 Frigate f1= new Frigate(4,5,2,2,9,100,7);  
 f1.AllInfo();  
 f1.ShotAtTheShip();  
 f1.AllInfo();  
 f1.printInfoAboutChase();  
  
 }  
}

Frigate.java

package Ship\_package;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
/\*  
 \* Class <code>Frigate</code> implements the count of people  
 \* @author Nychai Volodymyr KI-35  
 \* @version 1.0  
 \*/  
public class Frigate extends Ship implements Shot {  
 private int ammunition;  
 private int speed;  
 */\*\*  
 \* Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>ammunition</code>*** *count of ammunition  
 \** ***@param<code>speed</code>*** *value of speed  
 \*/* Frigate()throws FileNotFoundException{  
 this.ammunition=0;  
 this.speed=0;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>valuePower</code>*** *power of the engine  
 \** ***@param<code>valueVolume</code>*** *volume of the engine  
 \** ***@param<code>xLoc</code>*** *X location  
 \** ***@param<code>yLoc</code>*** *Y location  
 \** ***@param<code>count</code>*** *count of the crew  
 \** ***@param<code>ammunition</code>*** *count of ammunition  
 \** ***@param<code>speed</code>*** *value of speed  
 \*/* Frigate(int valuePower,int valueVolume,int xLoc, int yLoc,int valueCount,int ammunition,int speed) throws FileNotFoundException {  
 super( valuePower,valueVolume, xLoc, yLoc, valueCount);  
 this.ammunition=ammunition;  
 this.speed=speed;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>valueSpeed</code>*** *value of speed  
 \*/*Frigate(int valueSpeed) throws FileNotFoundException{  
 this.speed=valueSpeed;  
  
}  
 */\*\*  
 \* Method returns value of speed of the Frigate  
 \** ***@return*** *speed  
 \*/* public int getSpeed(){return this.speed;}  
  
 */\*\*  
 \* Method returns ammunition of the frigate  
 \** ***@return*** *ammunition  
 \*/* public int getAmmunition(){  
 return this.ammunition;  
 }  
  
 */\*\*  
 \* method print AllInfo  
 \*/* public void AllInfo(){  
 System.*out*.println("Power of ship is " + engine1.getPower());  
 System.*out*.println("Volume of ship is "+ engine1.getVolume());  
 System.*out*.println("Count of people = " + people.getCount());  
 System.*out*.println("X location is " + loc1.getXlocation());  
 System.*out*.println("Y location is " + loc1.getYlocation());  
 System.out.println("Ammunition is " + getAmmunition());  
 System.out.println("Speed is " + getSpeed());  
 }  
  
 */\*\*  
 \* method check ammunition and shot  
 \*/* public void ShotAtTheShip() {  
 if (this.ammunition>0){  
 System.out.println("You shot");  
 fout.print("You Shot\n");  
 fout.flush();  
 this.ammunition--;  
 }  
 else {  
 System.*out*.println("You don't have ammunition");  
 fout.print("You don't have ammunition");  
 fout.flush();  
 }  
 }  
  
 */\*\*  
 \* method print info about chase  
 \*/* public void printInfoAboutChase() {  
 System.*out*.println("You start Chase \t"+ "Your speed is " +getSpeed() + " \tYou have "+ getAmmunition()+ " ammunitions");  
 fout.println("You start Chase \t"+ "Your speed is " +getSpeed() + "You have "+ getAmmunition()+ " ammunitions");  
 fout.flush();  
 }  
}

Ship.java

package Ship\_package;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
/\*  
 \* Class <code>Frigate</code> implements the count of people  
 \* @author Nychai Volodymyr KI-35  
 \* @version 1.0  
 \*/  
public class Frigate extends Ship implements Shot {  
 private int ammunition;  
 private int speed;  
 */\*\*  
 \* Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>ammunition</code>*** *count of ammunition  
 \** ***@param<code>speed</code>*** *value of speed  
 \*/* Frigate()throws FileNotFoundException{  
 this.ammunition=0;  
 this.speed=0;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>valuePower</code>*** *power of the engine  
 \** ***@param<code>valueVolume</code>*** *volume of the engine  
 \** ***@param<code>xLoc</code>*** *X location  
 \** ***@param<code>yLoc</code>*** *Y location  
 \** ***@param<code>count</code>*** *count of the crew  
 \** ***@param<code>ammunition</code>*** *count of ammunition  
 \** ***@param<code>speed</code>*** *value of speed  
 \*/* Frigate(int valuePower,int valueVolume,int xLoc, int yLoc,int valueCount,int ammunition,int speed) throws FileNotFoundException {  
 super( valuePower,valueVolume, xLoc, yLoc, valueCount);  
 this.ammunition=ammunition;  
 this.speed=speed;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@throws*** *FileNotFoundException  
 \** ***@param<code>valueSpeed</code>*** *value of speed  
 \*/*Frigate(int valueSpeed) throws FileNotFoundException{  
 this.speed=valueSpeed;  
  
}  
 */\*\*  
 \* Method returns value of speed of the Frigate  
 \** ***@return*** *speed  
 \*/* public int getSpeed(){return this.speed;}  
  
 */\*\*  
 \* Method returns ammunition of the frigate  
 \** ***@return*** *ammunition  
 \*/* public int getAmmunition(){  
 return this.ammunition;  
 }  
  
 */\*\*  
 \* method print AllInfo  
 \*/* public void AllInfo(){  
 System.*out*.println("Power of ship is " + engine1.getPower());  
 System.*out*.println("Volume of ship is "+ engine1.getVolume());  
 System.*out*.println("Count of people = " + people.getCount());  
 System.*out*.println("X location is " + loc1.getXlocation());  
 System.*out*.println("Y location is " + loc1.getYlocation());  
 System.*out*.println("Ammunition is " + getAmmunition());  
 System.*out*.println("Speed is " + getSpeed());  
 }  
  
 */\*\*  
 \* method check ammunition and shot  
 \*/* public void ShotAtTheShip() {  
 if (this.ammunition>0){  
 System.*out*.println("You shot");  
 fout.print("You Shot\n");  
 fout.flush();  
 this.ammunition--;  
 }  
 else {  
 System.*out*.println("You don't have ammunition");  
 fout.print("You don't have ammunition");  
 fout.flush();  
 }  
 }  
  
 */\*\*  
 \* method print info about chase  
 \*/* public void printInfoAboutChase() {  
 System.*out*.println("You start Chase \t"+ "Your speed is " +getSpeed() + " \tYou have "+ getAmmunition()+ " ammunitions");  
 fout.println("You start Chase \t"+ "Your speed is " +getSpeed() + "You have "+ getAmmunition()+ " ammunitions");  
 fout.flush();  
 }  
}

Engine.java

package Ship\_package;  
*/\*\*  
 \* Class <code>Engine</code> implements the operation of the engine  
 \** ***@author*** *Nychai Volodymyr KI-35  
 \** ***@version*** *1.0  
 \*/*public class Engine {  
 //characteristics of engine  
 private int power;  
 private int volume;  
*/\*\*  
 \* Constructor  
 \** ***@param<code>power</code>*** *\** ***@param<code>volume</code>*** *volume of the ship  
 \*/* public Engine(){  
 this.power=100000;  
 this.volume=500;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@param<code>power</code>*** *power of the engine  
 \** ***@param<code>volume</code>*** *volume of the ship  
 \*/* public Engine(int power, int volume){  
 this.power=power;  
 this.volume=volume;  
 }  
 */\*\*  
 \* Method returns the power of engine  
 \** ***@return*** *power of engine  
 \*/* public int getPower(){  
 return this.power;  
 }  
 */\*\*  
 \* Method returns the volume of engine  
 \** ***@return*** *volume of engine  
 \*/* public int getVolume(){  
 return this.volume;  
 }  
 */\*\*  
 \* method reduse power  
 \** ***@param*** *value  
 \*/* public void redusePower(int value){  
 if (value>=this.power)  
 {  
 System.*out*.println("You enter wrong value");  
 }  
 this.power-=value;  
 showEngine();  
 }  
 */\*\*  
 \* method increase power  
 \** ***@param*** *value  
 \*/* public void increasePower(int value){  
 if (value<=0)  
 {  
 System.*out*.println("You enter wrong value");  
 }  
 this.power+=value;  
 showEngine();  
 }  
 */\*\*  
 \* method show info  
 \*/* public void showEngine(){  
 System.*out*.println("Power is " + this.power);  
 System.*out*.println("Volume is " + this.volume);  
 System.*out*.print("\n");  
 } //SHOW INFO  
}

Location.java

package Ship\_package;  
  
import java.util.IllegalFormatCodePointException;  
*/\*\*  
 \* Class <code>Location</code> implements the location X,Y  
 \** ***@author*** *Nychai Volodymyr KI-35  
 \** ***@version*** *1.0  
 \*/*public class Location {  
 private int x,y;  
  
 */\*\*  
 \* Constructor  
 \** ***@param<code>x</code>*** *X location  
 \** ***@param<code>y</code>*** *Y location  
 \*/* Location(){  
 this.x=0;  
 this.y=0;  
 }  
 */\*\*  
 \* Another Constructor  
 \** ***@param<code>x</code>*** *X location  
 \** ***@param<code>y</code>*** *Y location  
 \*/* Location(int xValue, int yValue){  
 this.x=xValue;  
 this.y=yValue;  
 }  
 */\*\*  
 \* Method returns X location  
 \** ***@return*** *X X  
 \*/* public int getXlocation(){  
 return this.x;  
 }  
 */\*\*  
 \* Method returns Y location  
 \** ***@return*** *Y Y  
 \*/* public int getYlocation(){  
 return this.y;  
 }  
  
 */\*\*  
 \* Method sets X location  
 \** ***@param*** *valueX X  
 \*/* public void setXLocation(int valueX){  
 this.x=valueX;  
 }  
 */\*\*  
 \* Method sets Y location  
 \** ***@param*** *valueY Y  
 \*/* public void setYLocation(int valueY){  
 this.y=valueY;  
 }  
 */\*\*  
 \* method show info about Location  
 \*/* public void showInfoAboutLocation(){  
 System.out.println("Location X is " + getXlocation());  
 System.out.println("Location Y is " + getYlocation());  
 }  
 */\*\*  
 \* method increase X location  
 \** ***@param*** *valueX X  
 \*/* public void increaseXLocation(int valueX){  
 if (valueX<0){  
 System.out.println("You entered wrong value");  
 }  
 else{  
 this.x+=valueX;  
 System.out.println("You change X location, now X =" + getXlocation());  
 }  
  
 }  
 */\*\*  
 \* method increase Y location  
 \** ***@param*** *valueY Y  
 \*/* public void increaseYLocation(int valueY){  
 if (valueY<0){  
 System.out.println("You entered wrong value");  
 }  
 else{  
 this.y+=valueY;  
 System.out.println("You change Y location, now Y =" + getYlocation());  
 }  
 }  
 */\*\*  
 \* method reduse Y location  
 \** ***@param*** *valueY Y  
 \*/* public void reduseYLocation(int valueY){  
 if (valueY>this.y){  
 System.out.println("You entered wrong value");  
 }  
 else{  
 this.y-=valueY;  
 System.*out*.println("You change Y location, now Y =" + getYlocation());  
 }  
 }  
 */\*\*  
 \* method reduse X location  
 \** ***@param*** *valueX X  
 \*/* public void reduseXLocation(int valueX){  
 if (valueX>this.x){  
 System.*out*.println("You entered wrong value");  
 }  
 else{  
 this.x-=valueX;  
 System.out.println("You change X location, now X =" + getXlocation());  
 }  
 }  
}

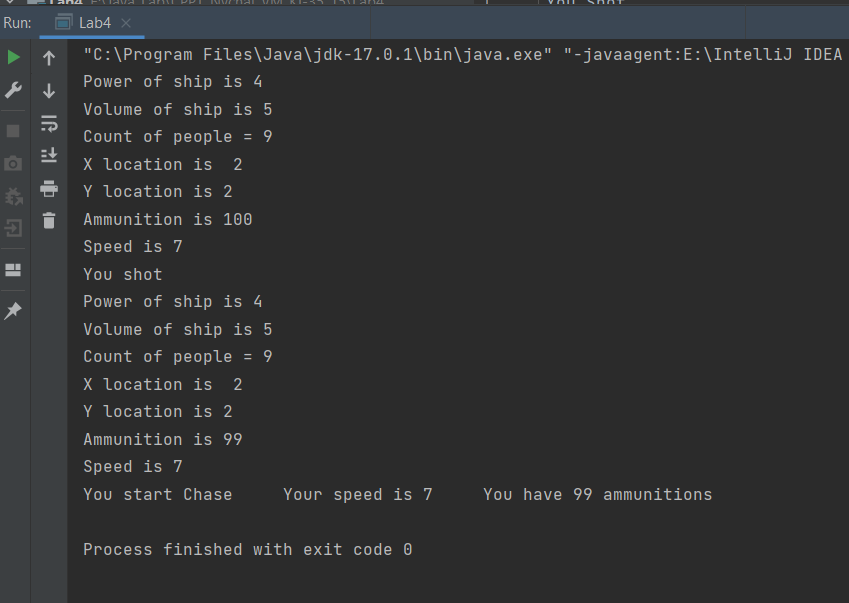
Crew.java

package Ship\_package;  
//екіпаж  
*/\*\*  
 \* Class <code>Crew</code> implements the count of people  
 \** ***@author*** *Nychai Volodymyr KI-35  
 \** ***@version*** *1.0  
 \*/*public class Crew {  
 private int count;  
  
 */\*\*  
 \* Constructor  
 \** ***@param<code>count</code>*** *count of crew  
 \*/* Crew(){  
 this.count=0;  
 }  
  
 */\*\*  
 \* Another Constructor  
 \** ***@param<code>count</code>*** *count of crew  
 \*/* Crew(int count){  
 this.count=count;  
 }  
  
 */\*\*  
 \* method print info about move  
 \*/* public void startMoving(){  
 if (count>50){  
 System.out.println("Ship is going");  
 }  
 else{  
 System.out.println("You don't start moving");  
 }  
 }  
 */\*\*  
 \* Method returns count of crew  
 \** ***@return*** *count  
 \*/* public int getCount(){  
 return count;  
 }  
 */\*\*  
 \* Method sets count of crew  
 \** ***@param*** *count count  
 \*/* public void setCount(int count){  
 this.count+=count;  
 }  
  
 */\*\*  
 \* method increase count of crew  
 \** ***@param*** *value crew  
 \*/* public void increaseCount(int value){  
 if (value<0){  
 System.*out*.println("You entered wrong number");  
 }  
 else {  
 this.count+=value;  
 }  
  
 }  
 */\*\*  
 \* method reduse count of crew  
 \** ***@param*** *value value  
 \*/* public void reduseCount(int value){  
 if (value>this.count){  
 System.out.println("There aren't that many people on the ship");  
 }  
 else {  
 this.count-=value;  
 }  
  
 }  
  
}

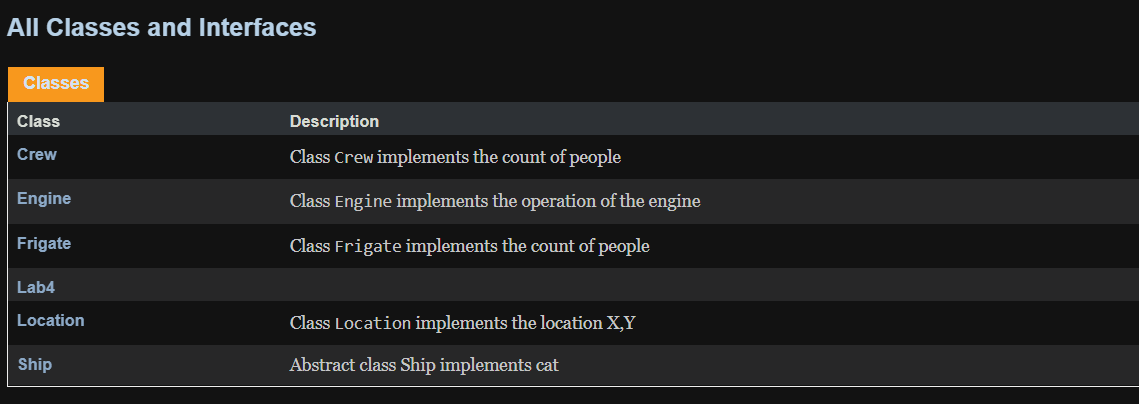
Interface.java

package Ship\_package;  
  
interface StartOfChase {  
 void printInfoAboutChase();  
}  
  
interface Shot extends StartOfChase{  
 void ShotAtTheShip();  
}

Результат роботи програми



**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання:**

1. Що таке абстрактний клас та як його реалізувати?

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об’єкти свого класу. Вони реалізуються за допомогою ключового слова abstract.

2. Що таке інтерфейс?

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

**Висновок**: на даній лабораторній роботі я ознайомився з принципами роботи успадкування на мові програмування Java. Ознайомився з принципами роботи інтерфейсів, створив новий клас, що наслідує клас з попередньої роботи. Покращив декілька методів.