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-306

Бокало П.М.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

**Завдання(варіант №2):**

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.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.

**Індивідуальне завдання:**



**Вихідний код програми:**

**Файл Lab3.Bokalo.Ki306.java:**

**package** KI306.Bokalo.Lab3;

**import** java.io.FileNotFoundException;

/\*\*

\* Дана програма тестує підклас написаний згідно до 2 варіанту - Багаторазовий космічний корабель

\* **@author** Petro Bokalo KI-306

\* **@version** 1.0

\* **@since** version 1.0

\*/

**public** **class** Lab3BokaloKI306

{

**public** **static** **void** main(String[] args) **throws** FileNotFoundException

{

Multi\_Starship ms = **new** Multi\_Starship("Boss", "Soreman");

ms.stats();

**for** (**int** i = 0; i <10; i++)

{

ms.fight();

}

ms.stats();

ms.repair();

ms.stats();

ms.call\_captain();

ms.closer();

}

}

**Файл Starship.java**

package KI306.Bokalo.Lab2;

import java.io.FileNotFoundException;

import java.io.\*;

/\*\*

\* Тут реалізовано класи Starship і Captain

\* @author Petro Bokalo KI-203

\* @version 1.0

\* @since version 1.0

\*/

public class Starship

{

private int fuel;

private String s\_name;

private int s\_health;

private int staff;

private Captain captain;

private boolean is\_fighting = false;

private PrintWriter fout;

public int getFuel() {

fout.write("\n public int getFuel() : was used\n");

return fuel;

}

public void setFuel(int fuel) {

this.fuel = fuel;

fout.write("\n public void setFuel(int fuel) : was used\n");

}

public String getS\_name() {

fout.write("\n public String getS\_name() : was used\n");

return s\_name;

}

public void setS\_name(String s\_name) {

fout.write("\n public void setS\_name() : was used\n");

this.s\_name = s\_name;

}

public int getS\_health() {

fout.write("\n public int getS\_health() : was used\n");

return s\_health;

}

public void setS\_health(int s\_health) {

fout.write("\n public void setS\_health(int s\_health) : was used\n");

this.s\_health = s\_health;

}

public int getStaff() {

fout.write("\n public int getStaff() : was used\n");

return staff;

}

public void setStaff(int staff) {

fout.write("\n public void setStaff(int staff)() : was used\n");

this.staff = staff;

}

public Starship()

{

fuel = 100;

s\_name = "void";

s\_health = 100;

staff = 10;

captain = new Captain();

try

{

fout = new PrintWriter(new BufferedWriter(new FileWriter("Starship\_log.txt", true)));

} catch (IOException e)

{

System.err.println("Помилка: файл Captain\_log.txt не може бути створений або записаний.");

}

fout.write("\n public Starship() : was used\n");

}

public Starship(String name, String c\_name)

{

fuel = 100;

this.s\_name = name;

s\_health = 100;

staff = 10;

captain = new Captain(c\_name);

try

{

fout = new PrintWriter(new BufferedWriter(new FileWriter("Starship\_log.txt", true)));

} catch (IOException e)

{

System.err.println("Помилка: файл Captain\_log.txt не може бути створений або записаний.");

}

fout.write("\n public Starship(String name, String c\_name) : was used\n");

}

public void closer()

{

fout.write("public void closer() : was used\n");

if (fout != null)

{

fout.close();

captain.close();

}

}

public void call\_captain()

{

captain.Captain\_stats();

}

public void move()

{

if(fuel > 10)

{

fuel -=10;

}

else

{

System.out.print("\n Can't move!!! \n");

}

fout.write("\n public move() : was used\n");

}

public void damage()

{

if(s\_health > 0 && staff > 0)

{

s\_health -= 10;

staff -= 1;

}else if (s\_health == 0)

{

System.out.print("\n Starship is critically damaged !!! \n");

}

fout.write("\n public damage() : was used\n");

}

public void repair()

{

if(s\_health < 100)

{

staff += 5;

int coins = captain.getCoins();

coins -= 50;

captain.setCoins(coins);

}

else

{

System.out.print("Repaired\n");

}

fout.write("\n public repair() : was used\n");

}

public void fight ()

{

is\_fighting = true;

damage();

int coins = captain.getCoins();

captain.setCoins(coins + 25);

fout.write("\n public fight() : was used\n");

}

public void retreat ()

{

is\_fighting = false;

fout.write("\n public retreat() : was used\n");

}

public void stats ()

{

System.out.print("Stats : \n");

System.out.print("Name : " + s\_name + "\n");

System.out.print("Fuel : " + fuel + "\n");

System.out.print("Health : " + s\_health + "\n");

System.out.print("Staff : " + staff + "\n");

System.out.print("Fighting : " + is\_fighting + "\n\n");

fout.write("\n public stats() : was used\n");

}

public void refuel()

{

fuel = 100;

int coins = captain.getCoins();

captain.setCoins(coins - 25);

fout.write("\n public refuel() : was used\n");

}

public void clearShip\_logs()

{

fout.write("public void clearShip\_logs() : was used\n");

try

{

PrintWriter clearWriter = new PrintWriter("Starship\_log.txt");

clearWriter.close();

} catch (FileNotFoundException e) {

System.err.println("Помилка: файл 'Starship\_log.txt' не може бути очищений.");

}

}

}

class Captain {

private String name;

private int coins;

private int health;

private static int id = 0;

private PrintWriter fout;

public String getName() {

fout.write("\n public String getName() : was used\n");

return name;

}

public void setName(String name) {

fout.write("\n public void setName(String name) : was used\n");

this.name = name;

}

public int getCoins() {

fout.write("\n public int getCoins() : was used\n");

return coins;

}

public void setCoins(int coins) {

fout.write("\n public void setCoins(int coins) : was used\n");

this.coins = coins;

}

public int getHealth() {

fout.write("\n public int getHealth() : was used\n");

return health;

}

public void setHealth(int health) {

fout.write("\n public void setHealth(int health) : was used\n");

this.health = health;

}

public Captain() {

this.name = "Noname";

this.coins = 100;

this.health = 100;

id++;

try {

fout = new PrintWriter(new BufferedWriter(new FileWriter("Captain\_log.txt", true)));

} catch (IOException e) {

System.err.println("Помилка: файл Captain\_log.txt не може бути створений або записаний.");

}

fout.write("\n public Captain() : was used\n");

Captain\_stats\_file();

}

public Captain(String name) {

this.name = name;

coins = 100;

health = 100;

id++;

try {

fout = new PrintWriter(new BufferedWriter(new FileWriter("Captain\_log.txt", true)));

} catch (IOException e) {

System.err.println("Помилка: файл Captain\_log.txt не може бути створений або записаний.");

}

fout.write("\n public Captain (String name) : was used\n");

Captain\_stats\_file();

}

public void Captain\_stats\_file() {

fout.write("\n public void Captain\_stats\_file() : was used \n\n");

fout.write("New Captain created :\n");

fout.write("ID : " + id + "\n");

fout.write("Name : " + name + "\n");

fout.write("Coins : " + coins + "\n");

fout.write("Health : " + health + "\n\n");

fout.flush(); // Очистити буфер і записати дані в файл

}

public void Captain\_stats() {

fout.write("public void Captain\_stats : was used \n");

System.out.print("Your captain has : \n");

System.out.print("Name : " + name + "\n");

System.out.print("Coins : " + coins + "\n");

System.out.print("Health : " + health + "\n\n");

}

public void close()

{

fout.write("public void close() : was used\n");

if (fout != null)

{

fout.close();

}

}

public void clearCaptain\_logs()

{

fout.write("public void clearCaptain\_logs() : was used\n");

try

{

PrintWriter clearWriter = new PrintWriter("Captain\_log.txt");

clearWriter.close();

} catch (FileNotFoundException e) {

System.err.println("Помилка: файл 'Captain\_log.txt' не може бути очищений.");

}

}

}

**Файл Multi\_Starship.java**

package KI306.Bokalo.Lab3;

/\*\*

\* Тут реалізовано підклас Multi\_Starship, який є підкласом Starship, а також імплементує інтерфейс Ships

\* @author Petro Bokalo KI-203

\* @version 1.0

\* @since version 1.0

\*/

public class Multi\_Starship extends Starship implements Ships

{

private int Life\_points;

public Multi\_Starship(String name, String c\_name)

{

super(name, c\_name);

Life\_points = 5;

}

public Multi\_Starship()

{

super();

Life\_points = 5;

}

public void Star\_jump()

{

for(int i = 0; i < 5; i++)

{

move();

}

}

public void repair()

{

if(s\_health == 0)

{

Life\_points = Life\_points > 0 ? Life\_points - 1 : Life\_points;

super.repair();

} else if(Life\_points == 0)

{

System.out.print("\n Out of life points!!! Get more \n");

}else if(s\_health < 100 && s\_health > 0)

{

super.repair();

}

}

public void stats()

{

super.stats();

System.out.print("Life Points : " + Life\_points + "\n\n");

}

}

**Файл Ships.java**

**package** KI306.Bokalo.Lab3;

/\*\*

\* Тут створено інтерфейс Ships

\* **@author** Petro Bokalo KI-203

\* **@version** 1.0

\* **@since** version 1.0

\*/

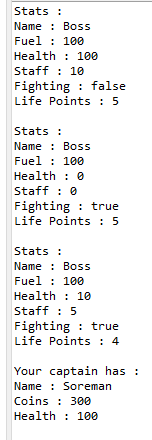
**public** **interface** Ships

{

**void** Star\_jump();

}

**Результат роботи програми:**



**Вміст файлів :**

**Файл Captain.txt:**

public Captain (String name) : was used

public void Captain\_stats\_file() : was used

New Captain created :

ID : 1

Name : Soreman

Coins : 100

Health : 100

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public int getCoins() : was used

public void setCoins(int coins) : was used

public void Captain\_stats : was used

public void close() : was used

**Файл Starship.txt**

public Starship(String name, String c\_name) : was used

public stats() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

public damage() : was used

public fight() : was used

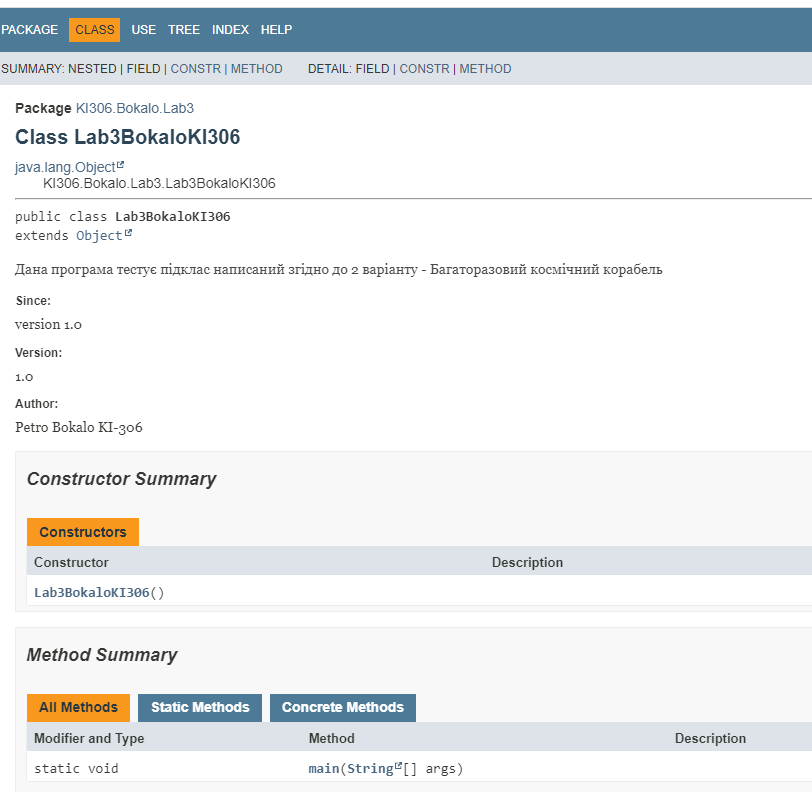
public stats() : was used

public repair() : was used

public stats() : was used

public void closer() : was used

**Фрагмент згенерованої документації:**



**Відповідь на контрольні питання**

**Синтаксис реалізації спадкування.**

1. Синтаксис реалізації спадкування: `class ChildClass extends ParentClass { ... }`.

**Що таке суперклас та підклас?**

2. Суперклас - це клас, від якого інший клас (підклас) успадковує властивості і методи. Підклас - це клас, який успадковує властивості і методи від суперкласу.

**Як звернутися до членів суперкласу з підкласу?**

3. До членів суперкласу з підкласу можна звертатися, використовуючи ключове слово `super`.

**Коли використовується статичне зв’язування при виклику методу?**

4. Статичне зв’язування використовується при виклику статичних методів або методів, які компілятор може визначити на етапі компіляції за типом посилання.

**Як відбувається динамічне зв’язування при виклику методу?**

5. Динамічне зв’язування відбувається під час виконання програми, коли викликається метод на об'єкті, і вибір методу залежить від типу об'єкта на етапі виконання.

**Що таке абстрактний клас та як його реалізувати?**

6. Абстрактний клас - це клас, який не може бути створений безпосередньо, і він містить абстрактні методи. Для його реалізації використовується ключове слово `abstract`.

**Для чого використовується ключове слово instanceof?**

7. Ключове слово `instanceof` використовується для перевірки, чи об'єкт є екземпляром певного класу або підкласу.

**Як перевірити чи клас є підкласом іншого класу?**

8. Для перевірки, чи клас є підкласом іншого класу, можна використовувати ключове слово `instanceof` або порівняння типів об'єктів.

**Що таке інтерфейс?**

9. Інтерфейс - це контракт, який визначає набір методів, які клас повинен реалізувати. Він не містить реалізації методів, тільки їхні сигнатури.

**Як оголосити та застосувати інтерфейс?**

10. Для оголошення інтерфейсу використовується ключове слово `interface`, і класи реалізують інтерфейс за допомогою ключового слова `implements`.

**Висновок:** на цій лабораторній роботі, я ознайомився з спадкуванням та інтерфейсами в мові програмування java. Написав програму згідно до свого варіанту. Навчився оголошувати і використовувати інтерфейси.