

Міністерство освіти і науки України

Національний університет «Львівська політехніка»



Звіт

З лабораторної роботи №3

З дисципліни «Кросплатформенні засоби програмування»

На тему: «Класи та пакети»

Виконав:
ст.гр. КІ-36
Литовко С.Г
Прийняв:
Іванов Ю.С.

Львів-2022

Мета роботи: ознайомитися з процесом розробки класів та пакетів мовою Java.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Класи

Мова Java є повністю об'єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об'єктно-орієнтованих парадигм програмування, що базуються на понятті класів.

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

```
[public] class НазваКласу  
{  
[конструктори]  
[методи]  
[поля]  
}
```

Приклад оголошення загальнодоступного класу:

```
public class StartClass  
{  
public StartClass()  
{  
str = "Hello";  
}  
public StartClass(String initString)  
{  
str = initString;  
}  
public void showMessage()  
{  
System.out.print(str);  
}  
private String str;  
}
```

Необов'язковий специфікатор доступу `public` робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім'я якого співпадає з назвою файлу, та безліч класів без специфікатора `public`.

Створення об'єкту класу складається з двох етапів: оголошення та ініціалізації посилання на об'єкт. Оголошення посилання на об'єкт класу має синтаксис:

НазваКласу *назваПосилання*;

Приклад оголошення посилання на об'єкт класу `StartClass`:

StartClass obj;

Ініціалізація посилання на об'єкт класу здійснюється за допомогою оператора `new` і вказування конструктора, який має збудувати об'єкт. Одержаний в результаті цих операцій об'єкт розташується у області оперативної пам'яті що зветься "куча". Ініціалізація посилання на об'єкт класу за допомогою конструктора за замовчуванням має такий синтаксис:

назваПосилання = *new НазваКонструктора()*;

Приклад ініціалізації посилання на об'єкт класу `StartClass`:

obj = *new StartClass()*;

При створенні об'єктів дозволяється суміщати оголошення та ініціалізацію об'єктів, а також створювати анонімні об'єкти. Якщо посилання на об'єкт не посилається на жоден об'єкт, то йому слід присвоїти значення `null`. На відміну від полів-посилань на об'єкти, локальні змінні-посилання на об'єкти не ініціалізуються значенням `null` при оголошенні. Для них ініціалізацію посилання слід проводити явно.

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:

- програма має розміщуватися в пакеті `Група.Прізвище.Lab3`;
- клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;

- клас має містити кілька конструкторів та мінімум 10 методів;
- для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
- методи класу мають вести протокол своєї діяльності, що записується у файл;
- розробити механізм коректного завершення роботи з файлом (не надіятися на метод `finalize()`);
- програма має 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.

Варіант: 6.Літак

Код програми:

Plane.java

```

Plane.java × App.java Chassis.java Engine.java
1 package lytovko.lab3;
2
3 import java.io.File;
4 import java.io.PrintWriter;
5 /**
6  * Class Plane
7  * @author Lytovko Sofia KI-36
8  * @version 1.0
9  */
10 public class Plane {
11
12     protected PrintWriter fout;
13     protected Chassis chassis = new Chassis();
14     private boolean isInTheAir = false;
15     private Engine engine;
16     /**
17      * Constructor
18      * @throws Exception
19      */
20     public Plane() throws Exception
21     {
22         fout = new PrintWriter(new File("lab3.txt"));
23         engine = new Engine(fout);
24     }
25     /**
26      * Method starts the engine
27      * @param sec
28      * @throws Exception
29      */

```

```

Plane.java × App.java Chassis.java Engine.java
30● public void startEngine(int sec) throws Exception
31 {
32     engine.run(sec);
33 }
34● /**
35  * Method refuels the engine
36  */
37● public void refuel()
38 {
39     message("Engine refuel...");
40     engine.refuel();
41 }
42● /**
43  * Method returns isOpen value
44  */
45● public boolean isChassisOpen() {
46     return chassis.isOpen();
47 }
48● /**
49  * Method closes the chassis
50  */
51● public void closeChassis() {
52     chassis.close();
53     message("Chassis closed");
54 }
55● /**
56  * Method opens the chassis
57  */
58● public void openChassis() {
59     chassis.open();
60     message("Chassis open");
61 }
62● /**
63  * Method for landing
64  */
65● public void land() throws Exception
66 {
67     if(!chassis.isOpen())
68     {
69         throw new Exception("Chassis isn't open");
70     }
71
72     isInTheAir = false;
73 }
74 }
75● /**
76  * Method starts to fly
77  */
78● public void startFly() {
79     isInTheAir = true;
80 }
81● ...
82● /**
83  * Method returns fuel level
84  */
85● public int getFuelLevel()
86 {
87     return engine.getFuelLevel();
88 }
89● /**
90  * Method closes the file
91  */
92● public void dispose()
93 {

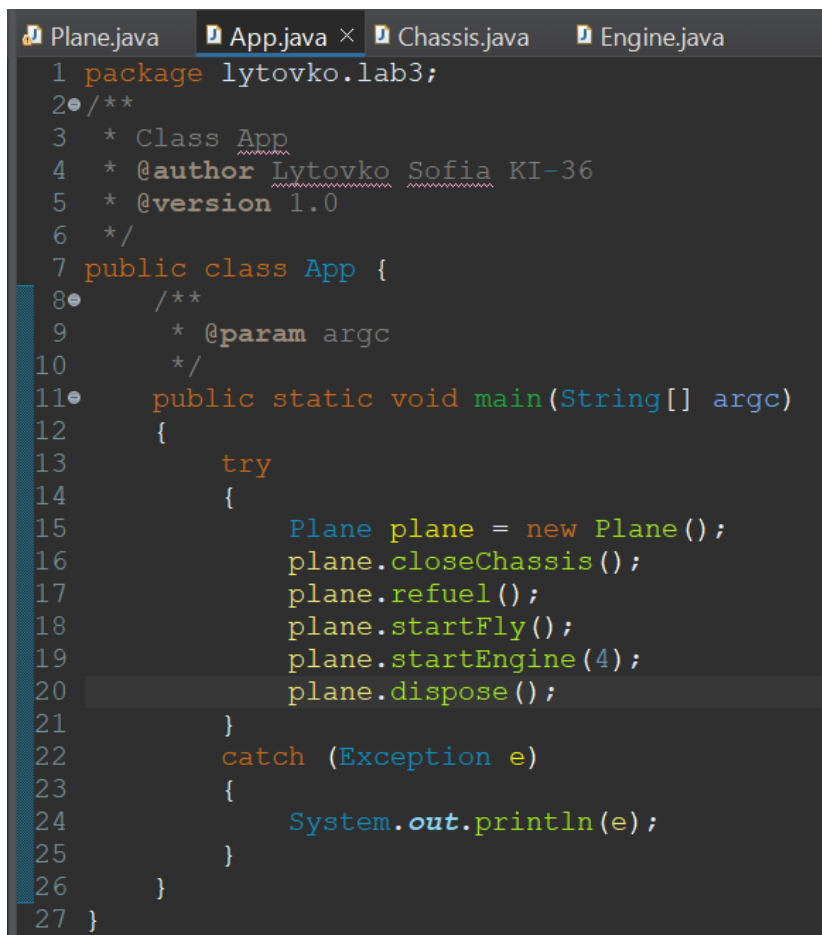
```

```

93         fout.flush();
94         fout.flush();
95         engine.dispose();
96     }
97
98     protected void message(String message)
99     {
100         System.out.println(message);
101         fout.println(message);
102     }
103 }

```

App.java



```

Plane.java  App.java ×  Chassis.java  Engine.java
1 package lytovko.lab3;
2 /**
3  * Class App
4  * @author Lytovko Sofia KI-36
5  * @version 1.0
6  */
7 public class App {
8     /**
9      * @param argc
10     */
11     public static void main(String[] argc)
12     {
13         try
14         {
15             Plane plane = new Plane();
16             plane.closeChassis();
17             plane.refuel();
18             plane.startFly();
19             plane.startEngine(4);
20             plane.dispose();
21         }
22         catch (Exception e)
23         {
24             System.out.println(e);
25         }
26     }
27 }

```

Chassis.java

```
Plane.java App.java Chassis.java × Engine.java
1 package lytovko.lab3;
2 /**
3  * Class Chassis
4  * @author Lytovko Sofia KI-36
5  * @version 1.0
6  */
7 public class Chassis {
8     private boolean isClosed = false;
9     /**
10     * Method returns isOpen value
11     */
12     public boolean isOpen() {
13         return !isClosed;
14     }
15     /**
16     * Method opens the chassis
17     */
18     public void open()
19     {
20         isClosed = false;
21     }
22     /**
23     * Method closes the chassis
24     */
25     public void close()
26     {
27         isClosed = true;
28     }
29 }
```

Engine.java

```
Plane.java App.java Chassis.java Engine.java ×
1 package lytovko.lab3;
2
3 import java.io.PrintWriter;
4 /**
5  * Class Engine
6  * @author Lytovko Sofia KI-36
7  * @version 1.0
8  */
9 public class Engine {
10
11     private int fuel = 0;
12     private final int FUEL_BANK_CAPACITY = 1000;
13     private PrintWriter fout;
14     /**
15      * Constructor
16      * @param fout
17      */
18     public Engine(PrintWriter fout)
19     {
20         this.fout = fout;
21     }
22     /**
23      * Method starts the engine
24      * @param sec
25      * @throws Exception
26      */
27     public void run(int sec) throws Exception {
28         for(int i = 0; i < sec; i++)
29         {
30             if(fuel < 1)
31             {
32                 throw new Exception("No fuel!");
33             }
34             fuel -= 10;
35             message("Remain fuel: " + fuel);
36             Thread.sleep(1000);
37         }
38     }
39     /**
40      * Method refuels the engine
41      */
42     public void refuel()
43     {
44         fuel = FUEL_BANK_CAPACITY;
45     }
46     /**
47      * Method closes the file
48      */
49     public void dispose()
50     {
51         fout.flush();
52         fout.close();
53     }
54 }
```

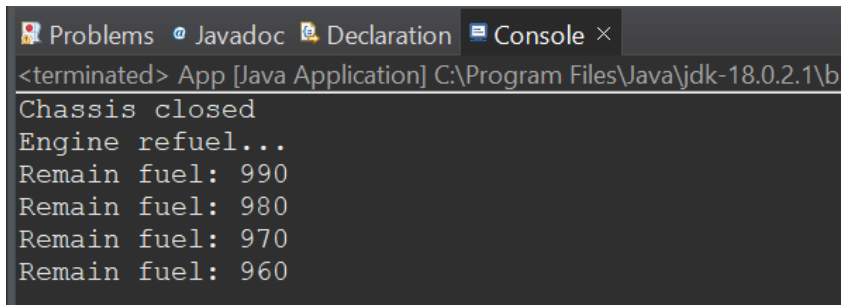


```

53     }
54     /**
55      * Method returns fuel level
56      */
57     public int getFuelLevel()
58     {
59         return fuel;
60     }
61
62     protected void message(String message)
63     {
64         System.out.println(message);
65         fout.println(message);
66     }
67
68 }

```

Результат програми:



```

<terminated> App [Java Application] C:\Program Files\Java\jdk-18.0.2.1\b
Chassis closed
Engine refuel...
Remain fuel: 990
Remain fuel: 980
Remain fuel: 970
Remain fuel: 960

```

Висновок: На цій лабораторній роботі я ознайомила з процесом розробки класів та пакетів мовою Java.