

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»



Звіт
з лабораторної роботи № 2
з дисципліни: «Кросплатформенні засоби програмування»
на тему: «Класи та пакети»

Виконав:
студент групи КІ-307
Бажулін С.В.

Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

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

Завдання (варіант № 2)

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.Lab2;
- клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області: космічний корабель;
- 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 розробленої програми.

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:

1. Синтаксис визначення класу.
2. Синтаксис визначення методу.
3. Синтаксис оголошення поля.
4. Як оголосити та ініціалізувати константне поле?
5. Які є способи ініціалізації полів?
6. Синтаксис визначення конструктора.
7. Синтаксис оголошення пакету.
8. Як підключити до програми класи, що визначені в зовнішніх пакетах?
9. В чому суть статичного імпорту пакетів?
10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

Вихідний код програми

SpaceshipApp.java

```
/**
 * Lab3 package
 */
package KI307.Bazhulin.Lab2;

import java.io.*;

import static java.lang.System.out;

/**
 * Spaceship Application class implement main methods for Spaceship class possibilities
 * demonstration
 */
```

```

* @author SERHIY BAZHULIN
* @version 1.0
*/

public class SpaceshipApp
{
    /**
     * @param args
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException
    {
        ControlPanel.Direction direction = null;
        Spaceship spaceship=new Spaceship();
        spaceship.RefuelSpaceship(50);
        out.print(spaceship.getFuelStatus());

        spaceship.CloseDoor();
        spaceship.StartSpaceship();
        spaceship.SetSpeed(100);

        spaceship.TurnRightSpaceship();
        direction = spaceship.getDirectionSpaceship();
        if(direction==ControlPanel.Direction.FORWARD)
            out.print("Forward direction\n");
        else if(direction == ControlPanel.Direction.LEFT)
            out.print("Left direction\n");
        else
            out.print("Right direction\n");

        spaceship.SwitchOffSapceship();
        spaceship.OpenDoor();
        spaceship.dispose();
    }
}

```

Spaceship.java

```

/**
 * Lab3 package
 */
package KI307.Bazhulin.Lab2;

import java.io.*;

/**
 * Class <code>Spaceship</code> implements spaceship
 * @author SERHIY BAZHULIN
 * @version 1.0
 */
public class Spaceship {
    private Engine engine;
    private ControlPanel controlPanel;
    private Door door;
    private PrintWriter fout;

    /**
     * Constructor
     * @throws FileNotFoundException
     */
    public Spaceship() throws FileNotFoundException
    {
        engine = new Engine();
        controlPanel = new ControlPanel();
    }
}

```

```

        door = new Door();

        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Constructor
     * @param <code>fuel</code> Fuel of engine
     * @throws FileNotFoundException
     */
    public Spaceship(double fuel) throws FileNotFoundException
    {
        engine = new Engine(fuel);
        controlPanel = new ControlPanel();
        door = new Door();

        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Method implements fuel of engine in spaceship
     * @param <code>fuel</code> Fuel of engine in spaceship
     */
    public void RefuelSpaceship(double fuel)
    {
        engine.Refuel(fuel);

        fout.print("Engine refuel: "+fuel+"\n");
    }

    /**
     * Method starts the engine
     */
    public void StartSpaceship()
    {
        if(engine.StartEngine())
            fout.print("Engine was started\n");
        else
            fout.print("Fuel is not enough for started\n");
    }

    /**
     * Method switches off the engine
     */
    public void SwitchOffSapceship()
    {
        engine.SwitchOffEngine();
        fout.print("Engine was switched off\n");
    }

    /**
     * Method returns fuel in engine
     * @return Fuel of engine
     */
    public double getFuelStatus()
    {
        double fuel = engine.getFuel();
        return fuel;
    }

    /**
     * Method returns status of engine
     * @return Engine status
     */

```

```

public boolean getEngineSpaceshipStatus()
{
    boolean status = engine.getEngineStatus();
    return status;
}

/**
 * Method turns left a spaceship
 */
public void TurnLeftSpaceship()
{
    controlPanel.TurnLeft();
    fout.print("Spaceship turned left\n");
}

/**
 * Method turns right a spaceship
 */
public void TurnRightSpaceship()
{
    controlPanel.TurnRight();
    fout.print("Spaceship turned right\n");
}

/**
 * Method forward a spaceship
 */
public void ForwardSpaceship()
{
    controlPanel.Forward();
    fout.print("Spaceship forward\n");
}

/**
 * Method sets a speed for spaceship
 * @param <code>speed</code> Speed for spaceship
 */
public void SetSpeed(int speed)
{
    controlPanel.SpeedChange(speed);
    fout.print("Spaceship speed set "+speed+"\n");
}

/**
 * Method returns speed of spaceship
 * @return Speed of spaceship
 */
public int getSpeedSpaceship()
{
    return controlPanel.getSpeed();
}

/**
 * Method opens door of spaceship
 */
public void OpenDoor()
{
    door.Open();
    fout.print("Door of spaceship was opened\n");
}

/**
 * Method closes door of spaceship
 */

```

```

    public void CloseDoor()
    {
        door.Close();
        fout.print("Door of spaceship was closed\n");
    }

    /**
     * Method returns status of door
     * @return Door status
     */
    public boolean getStatusDoor()
    {
        return door.getDoorStatus();
    }

    /**
     * Method returns spaceship direction
     * @return Direction of spaceship <code>ControlPanel.Direction</code> type
     */
    public ControlPanel.Direction getDirectionSpaceship()
    {
        return controlPanel.getDirection();
    }

    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout.close();
    }
}

/**
 * @author BAZHULIN SERHIY
 * Class <code>Engine</code> implements engine
 */
class Engine
{
    private double fuel;
    private boolean isStarted;

    /**
     * Constructor
     * @throws FileNotFoundException
     */
    public Engine() throws FileNotFoundException
    {
        fuel = 0;
        isStarted = false;
    }

    /**
     * Constructor
     * @param fuel
     * @throws FileNotFoundException
     */
    public Engine(double fuel) throws FileNotFoundException
    {
        this.fuel = fuel;
    }

    /**
     * Constructor

```

```

    * @param fuel
    * @param isStarted
    * @throws FileNotFoundException
    */
    public Engine(double fuel, boolean isStarted) throws FileNotFoundException
    {
        this.fuel=fuel;
        this.isStarted = isStarted;
    }

    /**
     * Method starts the engine
     */
    public boolean StartEngine()
    {
        if(fuel>0)
        {
            isStarted = true;
            return true;
        }
        else
            return false;
    }

    /**
     * Method switches off the engine
     */
    public void SwitchOffEngine()
    {
        isStarted = false;
    }

    /**
     * Method refuel the engine
     * @param fuel
     */
    public void Refuel(double fuel)
    {
        this.fuel = fuel;
    }

    /**
     * Method returns the fuel of engine
     * @return The fuel value
     */
    public double getFuel()
    {
        return fuel;
    }

    /**
     * Method returns a status of engine
     * @return A status of engine
     */
    public boolean getEngineStatus()
    {
        return isStarted;
    }
}

/**
 * @author BAZHULIN SERHIY
 * Class <code>ControlPanel</code> implements control panel
 */

```

```

class ControlPanel
{
    enum Direction {RIGHT, LEFT, FORWARD}

    private int speed;
    private Direction direction;

    /**
     * Constructor
     */
    public ControlPanel()
    {
        speed = 0;
        direction=Direction.FORWARD;
    }

    /**
     * Constructor
     * @param <code>speed</code>
     * @param <code>direction</code>
     */
    public ControlPanel(int speed, Direction direction)
    {
        this.speed = speed;
        this.direction = direction;
    }

    /**
     * Method set a speed for control panel
     * @param <code>speed</code>
     */
    public void SpeedChange(int speed)
    {
        this.speed=speed;
    }

    /**
     * Method returns the set speed on control panel
     * @return The speed value
     */
    public int getSpeed()
    {
        return speed;
    }

    /**
     * Method sets the right direction to control panel
     */
    public void TurnRight()
    {
        direction = Direction.RIGHT;
    }

    /**
     * Method sets the left direction to control panel
     */
    public void TurnLeft()
    {
        direction = Direction.LEFT;
    }

    /**
     * Method sets the forward direction to control panel
     */

```



```

    public void Forward()
    {
        direction = Direction.FORWARD;
    }

    /**
     * Method returns direction of control panel
     * @return Direction of control panel
     */
    public Direction getDirection()
    {
        return direction;
    }
}

/**
 * @author BAZHULIN SERHIY
 * Class <code>Door</code> implements door
 */
class Door
{
    private boolean isClosed;

    /**
     * Constructor
     * @throws FileNotFoundException
     */
    public Door() throws FileNotFoundException
    {
        isClosed = false;
    }

    /**
     * Constructor
     * @param isClosed
     * @throws FileNotFoundException
     */
    public Door(boolean isClosed) throws FileNotFoundException
    {
        this.isClosed = isClosed;
    }

    /**
     * Method opens door
     */
    public void Open()
    {
        isClosed = false;
    }

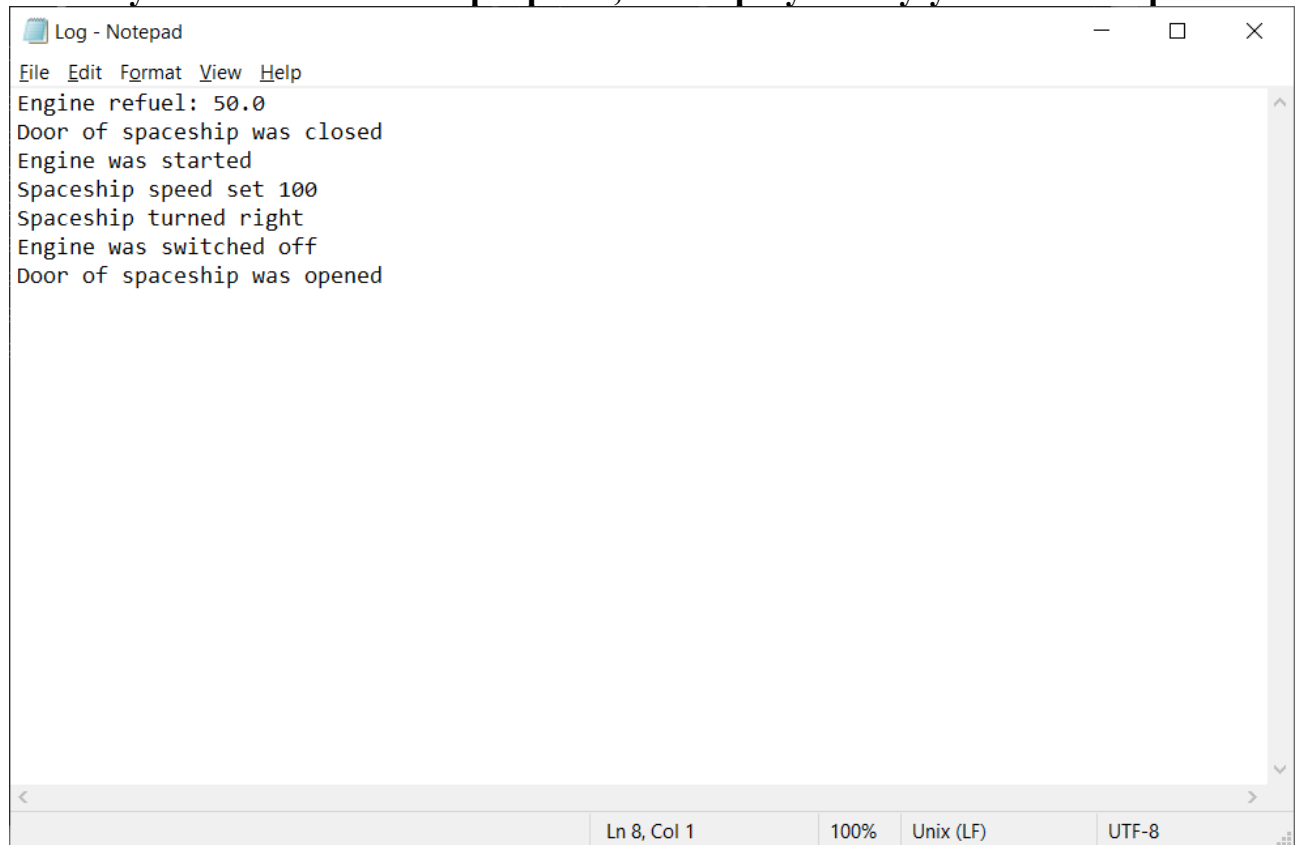
    /**
     * Method closes door
     */
    public void Close()
    {
        isClosed = true;
    }

    /**
     * Method returns status of door
     * @return door status
     */
    public boolean getDoorStatus()
    {

```

```
        return isClosed;  
    }  
}
```

Результат виконання програми, запис результату у текстовий файл



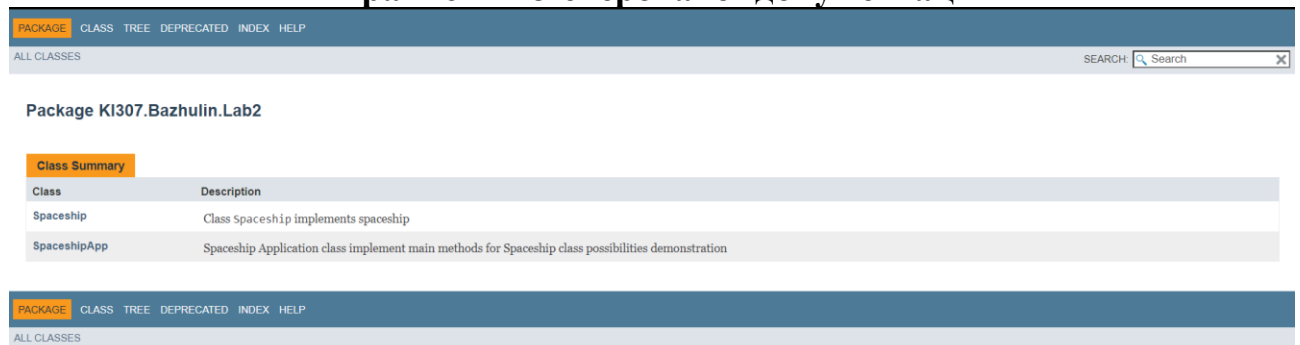
Log - Notepad

File Edit Format View Help

Engine refuel: 50.0
Door of spaceship was closed
Engine was started
Spaceship speed set 100
Spaceship turned right
Engine was switched off
Door of spaceship was opened

Ln 8, Col 1 100% Unix (LF) UTF-8

Фрагменти згенерованої документації



PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES SEARCH: Search

Package KI307.Bazhulin.Lab2

Class Summary

Class	Description
Spaceship	Class Spaceship implements spaceship
SpaceshipApp	Spaceship Application class implement main methods for Spaceship class possibilities demonstration

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

The top screenshot shows the documentation for the `Spaceship` class. It includes the package `KI307.Bazhulin.Lab2`, the class `Spaceship` extending `java.lang.Object`, and a description: "Class Spaceship implements spaceship". It lists two constructors: `Spaceship()` and `Spaceship(double fuel)`.

The bottom screenshot shows the documentation for the `SpaceshipApp` class. It includes the package `KI307.Bazhulin.Lab2`, the class `SpaceshipApp` extending `java.lang.Object`, and a description: "Spaceship Application class implement main methods for Spaceship class possibilities demonstration". It lists one constructor: `SpaceshipApp()` and one static method: `main(java.lang.String[] args)`.

Відповіді на контрольні запитання

1. Синтаксис визначення класу.

[специфікаторДоступу] class Ім'я_класу {}

2. Синтаксис визначення методу.

[специфікаторДоступу] Тип_Даних Ім'я_Методу([параметри]) [throws класи]
{
 [Тіло методу]
 [return [значення]];
}

3. Синтаксис оголошення поля.

[специфікаторДоступу] Тип_Даних Ім'я_Поля;

4. Як оголосити та ініціалізувати константне поле?

public class MyClass

{
 final int pi = 3.14;
}

5. Які є способи ініціалізації полів?

- Ініціалізація при оголошенні;

- Ініціалізація в конструкторі;
- У блоці ініціалізації.

6. Синтаксис визначення конструктора.

```
public Ім'я_Класу([параметри])
{
}
```

7. Синтаксис оголошення пакету.

```
package [Назва_Пакету].[Назва_Підпакетів];
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

За допомогою ключового слова import;

9. В чому суть статичного імпорту пакетів?

В тому щоб спростити використання полів та методів з класів, без використання назви класу перед цим полем чи методом.

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

- *Кожен пакет повинен мати свій корінь в системі файлів, який відповідає його імені.*
- *Ім'я каталогу повинно точно відповідати імені пакету.*
- *Кожний файл класу, який належить до пакету, повинен містити інструкцію package, яка вказує ім'я пакету, до якого він належить.*

Висновок: У цій лабораторній роботі, я ознайомився з класами та пакетами в мові програмування java. Написав програму згідно свого варіанту. Навчився оголошувати і використовувати класи, поля, методи та пакети.