Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №3

Варіант – 6

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-35

Павлик С. В.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2022

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

**Завдання(Варіант 6)**

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:

1. Як оголосити та ініціалізувати константне поле?
2. Синтаксис оголошення пакету.

**Варіант завдання: 6. Літак**

**Код програми:**

**Файл Main.java**

package LAB\_03;

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException{

PlaneApp planeApp = new PlaneApp();

planeApp.setVisible(true);

}

}

**Файл Plane\_component.java**

**package LAB\_03;**

**import java.awt.\*;**

**import java.awt.geom.AffineTransform;**

**import javax.swing.JOptionPane;**

**import javax.swing.ImageIcon;**

**/\*\***

**\* Class Pilot implements main method for Plane class possibilities moving**

**\***

**\* @author Pavlyk Serhii**

**\* @version 1.0**

**\*/**

**class Pilot**

**{/\*\***

**\* Constructor**

**\*/**

**public Pilot()**

**{**

**this.image = new ImageIcon("PlaneModel.png").getImage();**

**}**

**public static final double pilot\_size = 64;**

**private double pilot\_x,pilot\_y;**

**private float angle =0;**

**private final Image image;**

**/\* Method змінює кут нахилу літака**

**@Param angle кут на який змінюється значення**

**\*/**

**public void ChangeAngle(float angle)**

**{**

**if (angle<0) angle = 359;**

**else if(angle > 359) angle = 0;**

**this.angle = angle;**

**}**

**/\*\***

**\* Method sets the Y position**

**\* \*/**

**public void setPilot\_y(double pilot\_y) {**

**this.pilot\_y = pilot\_y;**

**}**

**/\*\***

**\* Method sets the X position**

**\* \*/**

**public void setPilot\_x(double pilot\_x) {**

**this.pilot\_x = pilot\_x;**

**}**

**/\*\***

**\* Method return the Y position**

**\* \*/**

**public double getPilot\_y() {**

**return pilot\_y;**

**}**

**/\*\***

**\* Method return the X position**

**\* \*/**

**public double getPilot\_x() {**

**return pilot\_x;**

**}**

**/\*\***

**\* Method створює графічну можесь літака**

**\* \*/**

**public void BuildPilot(Graphics2D d2)**

**{**

**AffineTransform old = d2.getTransform();**

**d2.translate(pilot\_x, pilot\_y);**

**AffineTransform transform = new AffineTransform();**

**transform.rotate(Math.toRadians(angle+90),pilot\_size/2,pilot\_size/2);**

**d2.drawImage(image, transform, null);**

**d2.setTransform(old);**

**}**

**/\*\***

**\* Method задає конкретну позицію**

**\*/**

**public void move(double x, double y)**

**{**

**this.pilot\_x = x;**

**this.pilot\_y = y;**

**}**

**/\*\***

**\* Method return the angle position**

**\* \*/**

**public float getAngle() {**

**return angle;**

**}**

**/\*\***

**\* Method set the angle position**

**\* \*/**

**public void setAngle(float angle) {**

**this.angle = angle;**

**}**

**}**

**/\*\***

**\* Class Helm implements main method for Plane class possibilities get signal from user**

**\***

**\* @author Pavlyk Serhii**

**\* @version 1.0**

**\*/**

**class Helm**

**{**

**private boolean go\_AngleUp;**

**private boolean go\_AngleDown;**

**private boolean acceleration;**

**/\*\***

**\* Method return the Acceleration position**

**\* \*/**

**public boolean getAcceleration() {**

**return acceleration;**

**}**

**/\*\***

**\* Method set the acceleration position**

**\* \*/**

**public void setAcceleration(boolean acceleration) {**

**this.acceleration = acceleration;**

**}**

**/\*\***

**\* Method return the AngleUp position**

**\* \*/**

**public boolean getGo\_AngleUp() {**

**return go\_AngleUp;**

**}**

**/\*\***

**\* Method set the AngleUp position**

**\* \*/**

**public void setGo\_AngleUp(boolean go\_AngleUp) {**

**this.go\_AngleUp = go\_AngleUp;**

**}**

**/\*\***

**\* Method return the AngleDown position**

**\* \*/**

**public boolean getGo\_AngleDown() {**

**return go\_AngleDown;**

**}**

**/\*\***

**\* Method set the AngleDown position**

**\* \*/**

**public void setGo\_AngleDown(boolean go\_AngleDown) {**

**this.go\_AngleDown = go\_AngleDown;**

**}**

**}**

**/\*\***

**\* Class Engine implements main method for Plane class possibilities control Fuel level and speedUp or SpeedDown**

**\***

**\* @author Pavlyk Serhii**

**\* @version 1.0**

**\*/**

**class Engine**

**{**

**/\*\***

**\* Constructor**

**\*/**

**public Engine ()**

**{**

**MAX\_speed = 1f;**

**this.VolumeOfFuel = 8f;**

**}**

**/\*\***

**\* Constructor**

**\* @param Max\_speed максимальна швидкість**

**\* @param VolumeOfFuel вмстиме баку з паливом у літрах**

**\*/**

**public Engine (float Max\_speed,float VolumeOfFuel)**

**{**

**this.MAX\_speed = Max\_speed;**

**this.VolumeOfFuel = VolumeOfFuel;**

**}**

**/\*\***

**\* Constructor**

**\* @param VolumeOfFuel вмстиме баку з паливом у літрах**

**\*/**

**public Engine (float VolumeOfFuel) throws OutOfMemoryError**

**{**

**if (VolumeOfFuel < MaxVolumeOfFuel) this.VolumeOfFuel = VolumeOfFuel;**

**else {**

**System.out.print("the filling level has been exceeded!");**

**throw new OutOfMemoryError("Exception message");**

**}**

**}**

**private float VolumeOfFuel;**

**private boolean RanOutFuel = false;**

**private float MaxVolumeOfFuel = 9f;**

**private float MAX\_speed = 1f;**

**private float speed = 0f;**

**/\*\***

**\* Method контролює прискорення**

**\* \*/**

**public void SpeedUp()**

**{**

**if(speed>MAX\_speed) speed = MAX\_speed;**

**else speed += 0.02f;**

**if (VolumeOfFuel < 0.5f && !RanOutFuel && VolumeOfFuel != 0)**

**{**

**JOptionPane.showMessageDialog(null, "The fuel is fuel is running out!","Warning", JOptionPane.INFORMATION\_MESSAGE);**

**RanOutFuel = true;**

**}**

**if (RanOutFuel)**

**{**

**VolumeOfFuel = 0;**

**}**

**else if (VolumeOfFuel > 0) VolumeOfFuel -= 0.005f;**

**}**

**/\*\***

**\* Method контролює сповільнення**

**\* \*/**

**public void SpeedDown()**

**{**

**if (speed < 0) speed = 0;**

**else speed -= 0.005f;**

**if (VolumeOfFuel < 0.5f && !RanOutFuel && VolumeOfFuel != 0)**

**{**

**JOptionPane.showMessageDialog(null, "The fuel is fuel is running out!","Warning", JOptionPane.INFORMATION\_MESSAGE);**

**RanOutFuel = true;**

**}**

**if (RanOutFuel)**

**{**

**VolumeOfFuel = 0;**

**}**

**else if (VolumeOfFuel > 0) VolumeOfFuel -= 0.0005f;**

**}**

**/\*\***

**\* Method return the RanOutFuel position**

**\* \*/**

**public boolean getRanOutFuel() {**

**return RanOutFuel;**

**}**

**/\*\***

**\* Method return the Speed position**

**\* \*/**

**public float getSpeed() {**

**return speed;**

**}**

**}**

**Файл Plane.java**

**/\*\***

**\* LAB\_03 package**

**\*/**

**package LAB\_03;**

**import javax.swing.JComponent;**

**import java.awt.Color;**

**import java.awt.Graphics;**

**import java.awt.Graphics2D;**

**import java.awt.RenderingHints;**

**import java.awt.event.KeyAdapter;**

**import java.awt.event.KeyEvent;**

**import java.awt.image.BufferedImage;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.swing.JOptionPane;**

**/\*\***

**\* Class Plane implements game simulator of Plane**

**\***

**\* @author Pavlyk Serhii**

**\* @version 1.0**

**\*/**

**public class Plane extends JComponent {**

**private int height;**

**private int width;**

**private Thread thread;**

**private boolean ready = true;**

**public boolean isReady() {**

**return ready;**

**}**

**private Graphics2D graphics2D;**

**private BufferedImage image;**

**private Pilot pilot;**

**private Helm helm;**

**private Engine engine;**

**private float VolumeOfFuel;**

**private float MaxSpeed;**

**private PrintWriter fout;**

**/\*\***

**\* Constructor**

**\* @Param volumeOfFuel volume of fuel**

**\* @param MaxSpeed - максимальна швидкість літака**

**\* @param fout - робота з файлом**

**\* @throws IOException**

**\*/**

**Plane (float volumeOfFuel, float MaxSpeed) throws IOException**

**{**

**this.VolumeOfFuel = volumeOfFuel;**

**this.MaxSpeed = MaxSpeed;**

**this.fout = new PrintWriter((new FileWriter("Log.txt")));**

**}**

**/\*\***

**\* Constructor**

**\* @Param volumeOfFuel volume of fuel**

**\* @param MaxSpeed - максимальна швидкість літака**

**\* @param fout - робота з файлом**

**\* @throws IOException**

**\*/**

**Plane (float volumeOfFuel) throws IOException**

**{**

**this.VolumeOfFuel = volumeOfFuel;**

**this.MaxSpeed = 1;**

**this.fout = new PrintWriter((new FileWriter("Log.txt")));**

**}**

**/\*\***

**\* Method запускає роботу гри**

**\* @param height висота поля додатку**

**\* @param width ширина поля додатку**

**\* @param thread окремий потік для роботи гри**

**\*/**

**public void start()**

**{**

**height = getHeight();**

**width = getWidth();**

**fout.write("Height = ");**

**fout.write(String.valueOf(height));**

**fout.write(" Width = ");**

**fout.write(String.valueOf(width));**

**fout.write(" \n");**

**image = new BufferedImage(width,height,BufferedImage.TYPE\_INT\_ARGB);**

**graphics2D = image.createGraphics();**

**graphics2D.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);**

**graphics2D.setRenderingHint(RenderingHints.KEY\_INTERPOLATION, RenderingHints.VALUE\_INTERPOLATION\_BILINEAR);**

**thread = new Thread(new Runnable() {**

**@Override**

**public void run()**

**{**

**while (ready)**

**{**

**BuildBackground();**

**BuildPlane();**

**Render();**

**}**

**}**

**});**

**callPilot();**

**try {BuildHelmAndEngine(VolumeOfFuel); //fill up at 8l;**

**} catch (OutOfMemoryError e)**

**{**

**JOptionPane.showMessageDialog(null, "You try to fill up more fuel than it can be!","Error", JOptionPane.ERROR\_MESSAGE);**

**fout.write("An exception was thrown\n");**

**}**

**if (!ready)**

**{**

**fout.close();**

**}**

**thread.start();**

**}**

**/\*\***

**\* Method запускає в сон потік, у якому викликаний**

**\* @param time час на який потік засинає у мілісекундах**

**\*/**

**private void GoToSleep(long time)**

**{**

**try {**

**Thread.sleep(time);**

**} catch (InterruptedException e) {**

**System.err.println(e);**

**}//exception**

**}**

**/\*\***

**\* Method створює Background**

**\*/**

**private void BuildBackground()**

**{**

**graphics2D.setColor(new Color(255, 165, 0));//orange**

**graphics2D.fillRect(0, height-100, width, height);**

**graphics2D.setColor(new Color(137, 207, 240));**

**graphics2D.fillRect(0, 0, width, height-100);**

**fout.write("BuildBackground was successful\n");**

**}**

**/\*\***

**\* Method створює літак**

**\*/**

**private void BuildPlane()**

**{**

**pilot.BuildPilot(graphics2D);**

**fout.write("BuildPlane was successful\n");**

**}**

**/\*\***

**\* Method створює об'єкт класу Pilot**

**\*/**

**private void callPilot()**

**{**

**pilot = new Pilot();**

**pilot.move(Pilot.pilot\_size, height-100);//start pos**

**fout.write("callPilot was successful\n");**

**}**

**/\*\***

**\* Method створює об'єкт класу Helm та Engine**

**\*/**

**private void BuildHelmAndEngine(float VolumeOfFuel)**

**{**

**helm = new Helm();**

**engine = new Engine(MaxSpeed,VolumeOfFuel);**

**requestFocus();**

**addKeyListener(new KeyAdapter(){**

**@Override**

**public void keyPressed(KeyEvent e)**

**{**

**if (e.getKeyCode() == KeyEvent.VK\_R) helm.setGo\_AngleUp(true);**

**else if (e.getKeyCode() == KeyEvent.VK\_F) helm.setGo\_AngleDown(true);**

**else if (e.getKeyCode() == KeyEvent.VK\_SPACE) helm.setAcceleration(true);**

**}**

**@Override**

**public void keyReleased(KeyEvent e)**

**{**

**if (e.getKeyCode() == KeyEvent.VK\_R) helm.setGo\_AngleUp(false);**

**else if (e.getKeyCode() == KeyEvent.VK\_F) helm.setGo\_AngleDown(false);**

**else if (e.getKeyCode() == KeyEvent.VK\_SPACE) helm.setAcceleration(false);**

**}**

**});**

**new Thread (new Runnable() {**

**@Override**

**public void run ()**

**{**

**boolean gameOver = false;**

**float delta = 0.5f;**

**while(ready)**

**{**

**if(gameOver) ready = false;**

**float angle = pilot.getAngle();**

**if(helm.getGo\_AngleUp()) angle -=delta;**

**else if (helm.getGo\_AngleDown()) angle +=delta;**

**else if (helm.getAcceleration()) engine.SpeedUp();**

**else if (!helm.getAcceleration()) engine.SpeedDown();**

**GoMoving();**

**OutOfScren();**

**pilot.ChangeAngle(angle);**

**GoToSleep(4);**

**if (engine.getRanOutFuel())**

**{**

**JOptionPane.showMessageDialog(null, "Ran out of fuel","Disaster", JOptionPane.INFORMATION\_MESSAGE);**

**gameOver = fall();**

**GoToSleep(100);**

**};**

**}**

**}**

**}).start();**

**fout.write("BuildHelmAndEngine was successful\n");**

**}**

**/\*\***

**\* Method закінчує гру, якщо в літака закінчилося паливо і повертає його в початкову позицію**

**\*/**

**private boolean fall()**

**{**

**pilot.move(Pilot.pilot\_size, height-100);**

**pilot.ChangeAngle(0);**

**fout.write("fall was successful\n");**

**return true;**

**}**

**/\*\***

**\* Method виконує перевірки чи не виходить літак за межі додатку, якщо виходить то коректує його коодинати**

**\*/**

**private void OutOfScren()**

**{**

**double x,y;**

**x = pilot.getPilot\_x();**

**y = pilot.getPilot\_y();**

**if ( x >= width ) pilot.setPilot\_x(Pilot.pilot\_size);**

**else if ( x<0 ) pilot.setPilot\_x(width);**

**else if (y >= height) pilot.setPilot\_y(0);**

**else if (y <0) pilot.setPilot\_y(height);**

**fout.write("OutOfScren was successful\n");**

**}**

**/\*\***

**\* Method генерує графічні об'єкти**

**\*/**

**private void Render()**

**{**

**Graphics graphaics = getGraphics();**

**graphaics.drawImage(image, 0, 0, null);**

**graphaics.dispose();**

**fout.write("Render was successful\n");**

**}**

**/\*\***

**\* Method, який відповідає за рух літака**

**\*/**

**public void GoMoving()**

**{**

**double x,y;**

**x = pilot.getPilot\_x();**

**x += Math.cos(Math.toRadians(pilot.getAngle())) \* engine.getSpeed();**

**pilot.setPilot\_x(x);**

**y = pilot.getPilot\_y();**

**y += Math.sin(Math.toRadians(pilot.getAngle()))\* engine.getSpeed();**

**pilot.setPilot\_y(y);**

**fout.write("GoMoving was successful\n");**

**}**

**/\*\***

**\* Method releases used recourses**

**\*/**

**public void dispose() throws IOException**

**{**

**fout.close();**

**}**

**}**

**Файл PlaneApp.java**

**/\*\***

**\* LAB\_03 package**

**\*/**

**package LAB\_03;**

**import javax.swing.JFrame;**

**import java.awt.BorderLayout;**

**import java.awt.event.WindowAdapter;**

**import java.awt.event.WindowEvent;**

**import java.io.IOException;**

**/\*\***

**\* Class PlaneApp implements main method for Plane class possibilities demonstration**

**\***

**\* @author Pavlyk Serhii**

**\* @version 1.0**

**\*/**

**public class PlaneApp extends JFrame {**

**/\*\***

**\* Constructor**

**\* @throws IOException**

**\*/**

**public PlaneApp ()throws IOException{**

**SetParameters();**

**}**

**/\*\***

**\* SetParameters - починає встановює параметри для робити додатку**

**\* @throws IOException**

**\*/**

**private void SetParameters ()throws IOException**

**{**

**setTitle("PLane");**

**setSize(1280,720);**

**setLocationRelativeTo(null);**

**setResizable(false);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLayout(new BorderLayout());**

**Plane plane = new Plane(8,1);**

**add(plane);**

**addWindowListener(new WindowAdapter()**

**{**

**@Override**

**public void windowOpened(WindowEvent e) {**

**plane.start();**

**}**

**});**

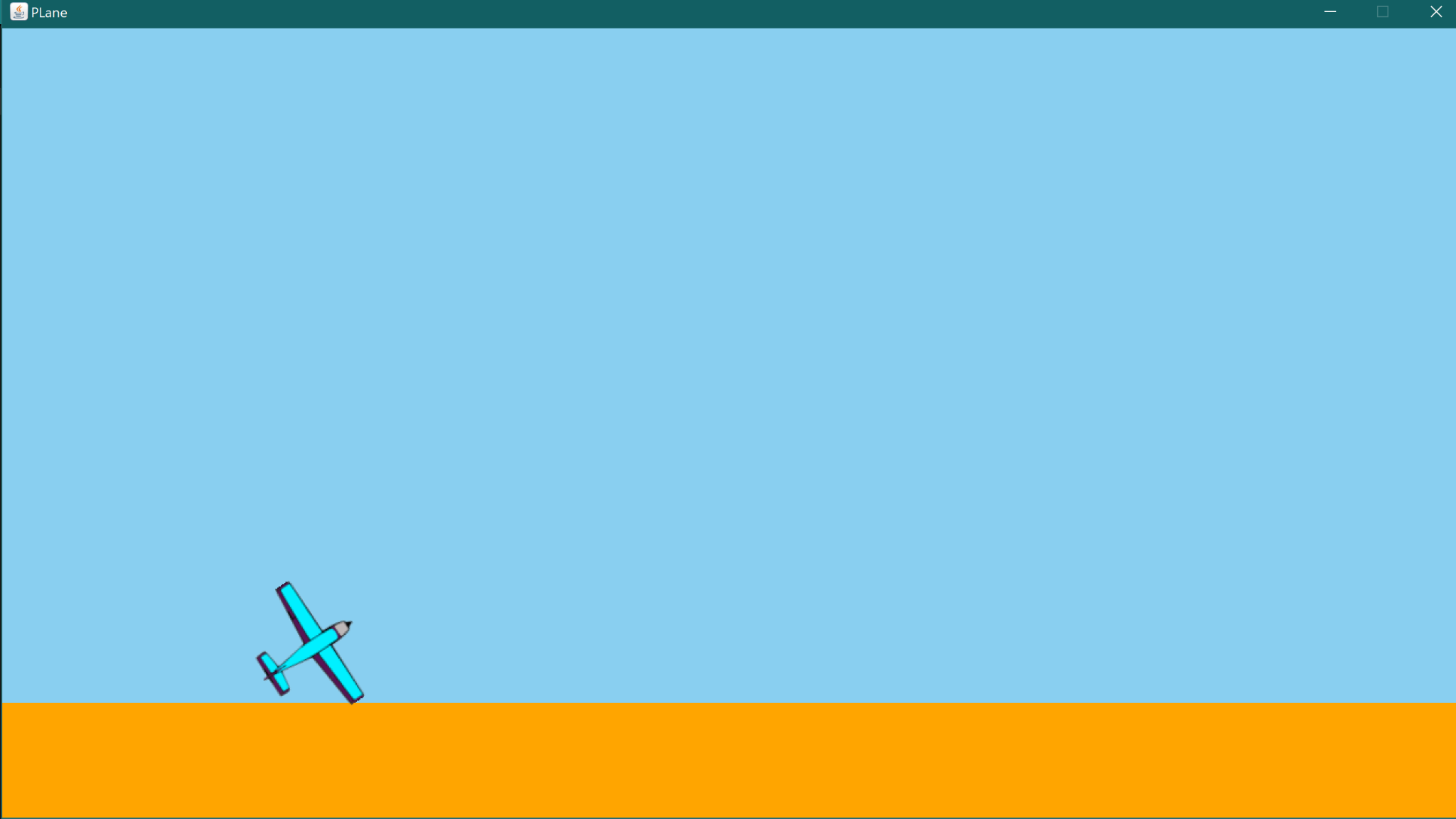
**if (!plane.isReady())**

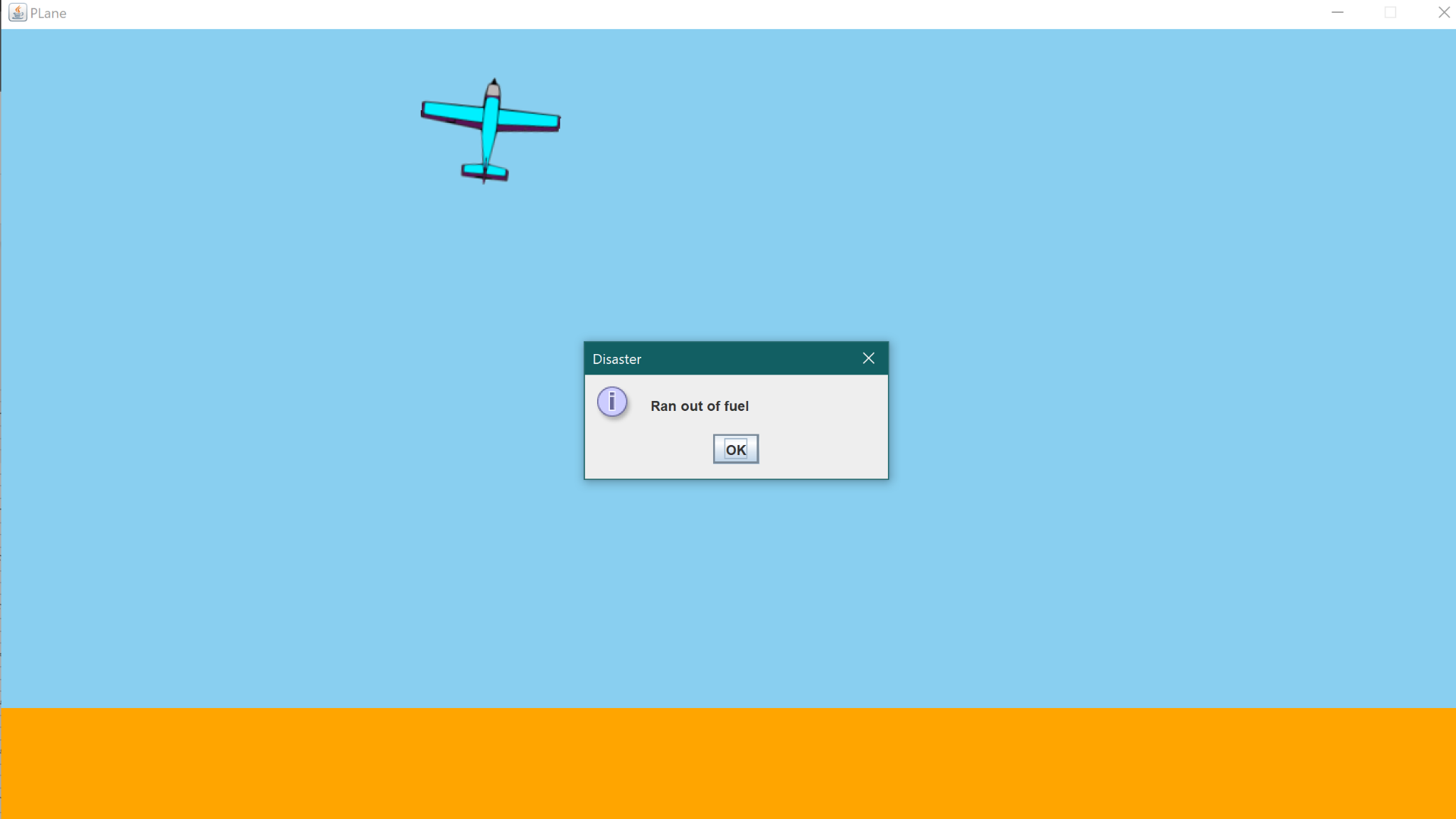
**plane.dispose();**

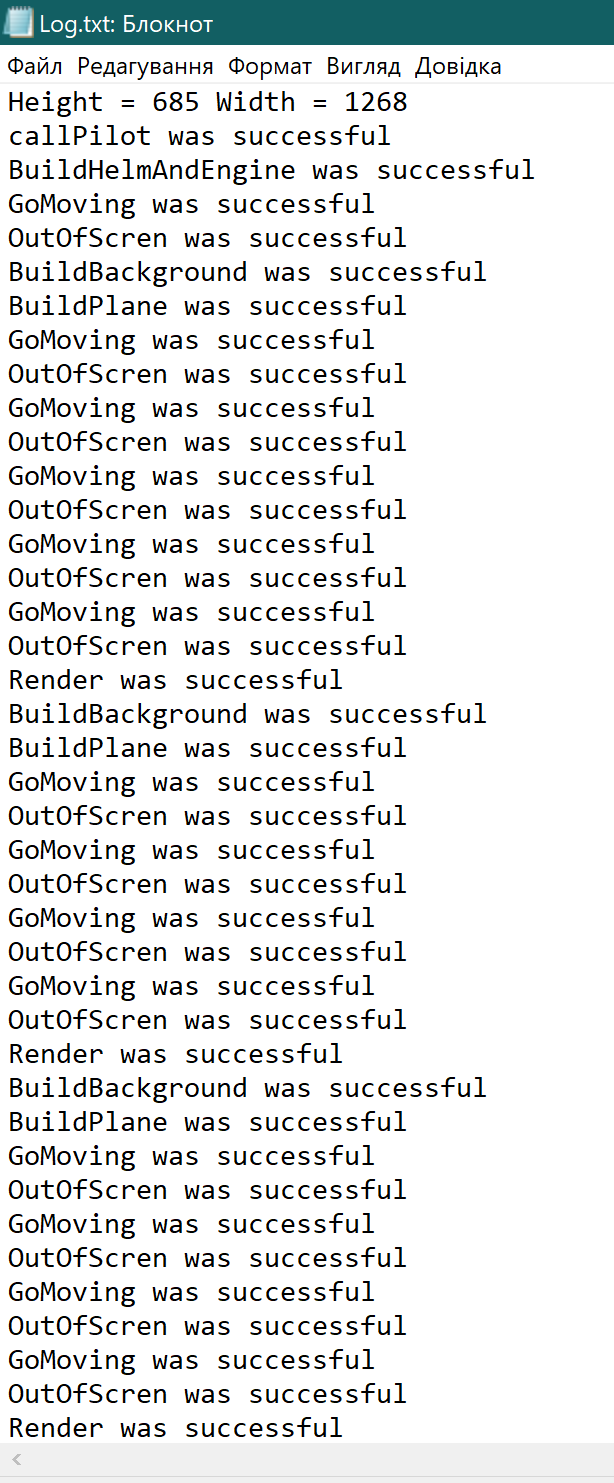
**}**

**}**

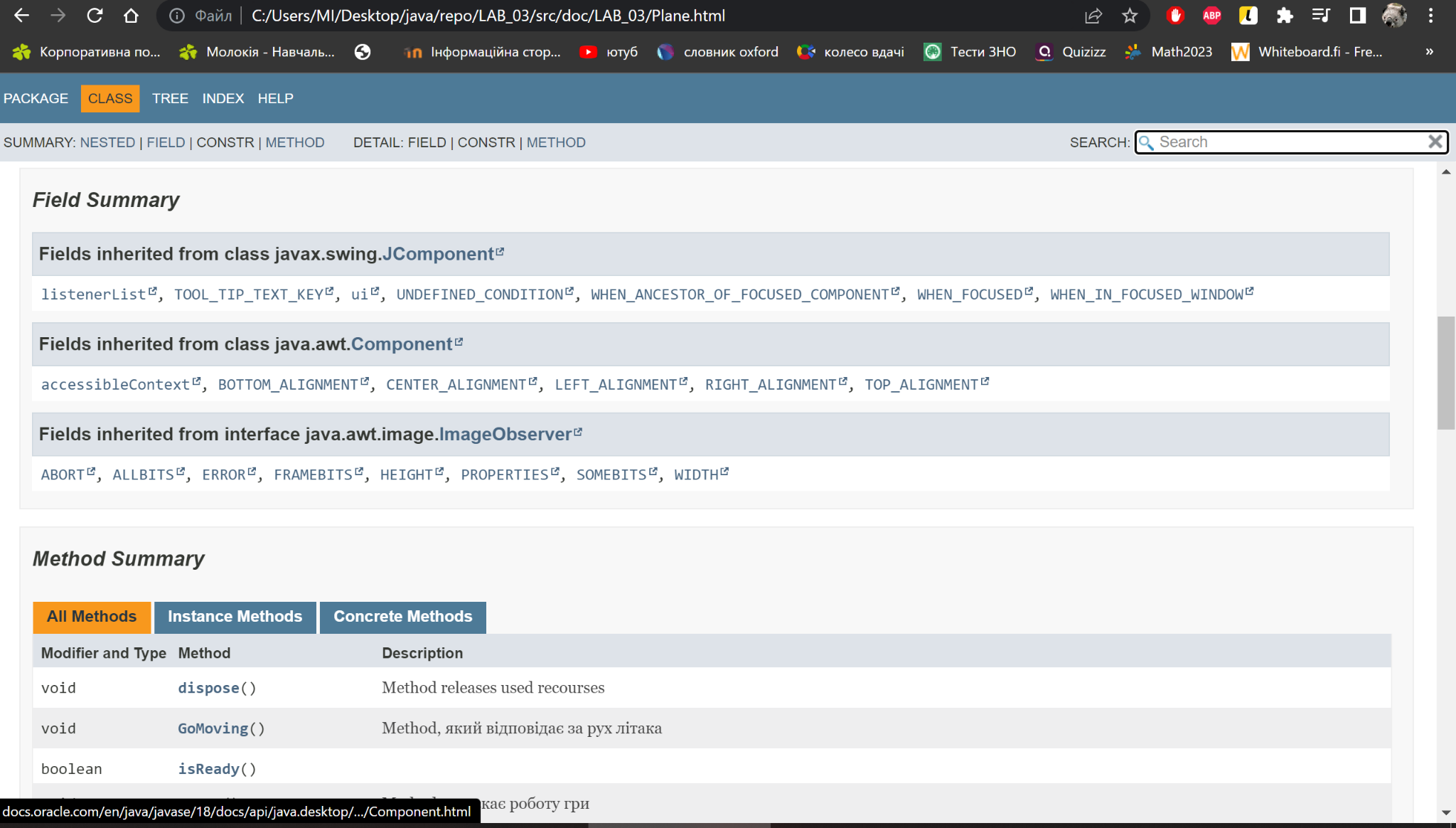
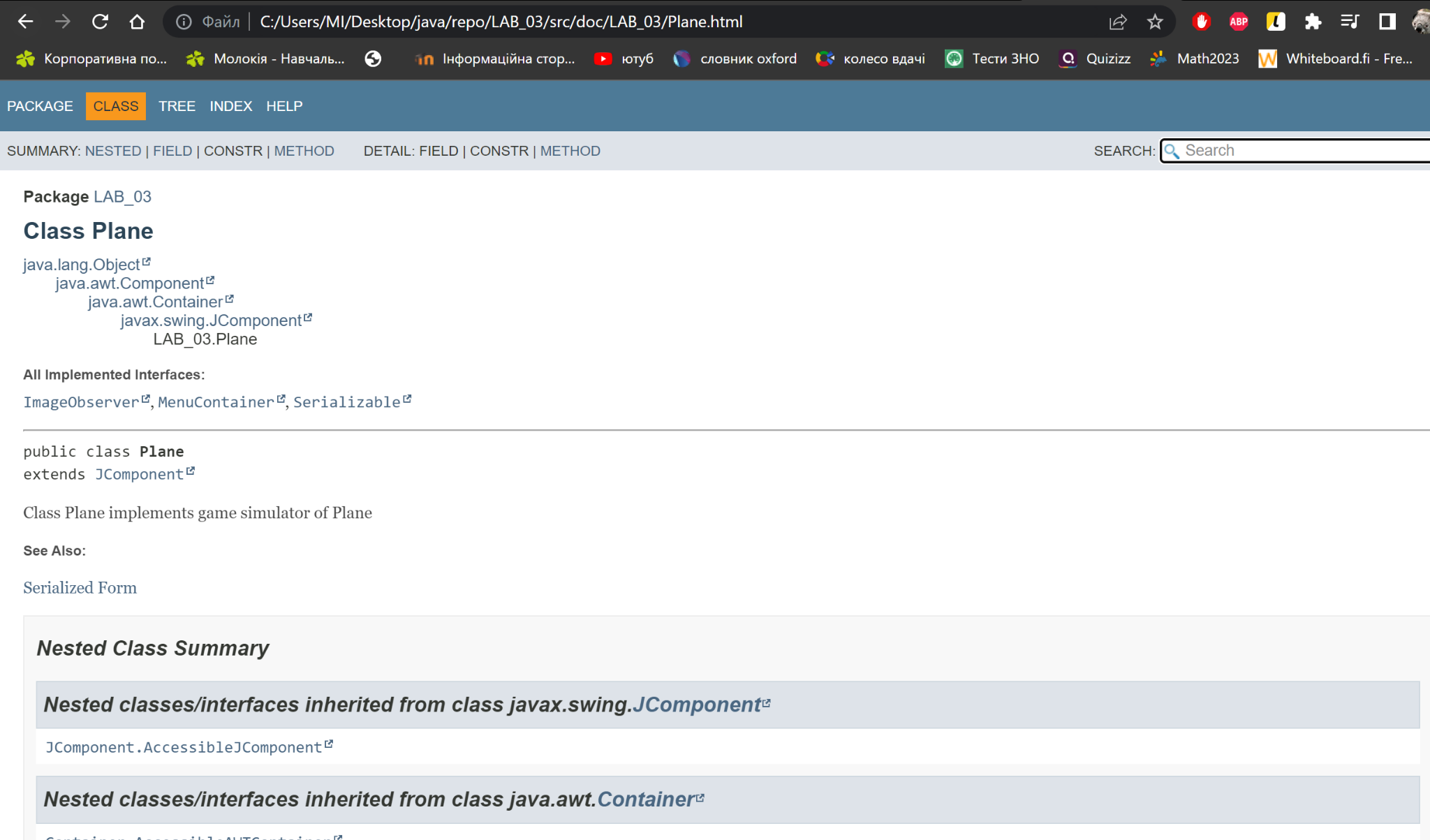
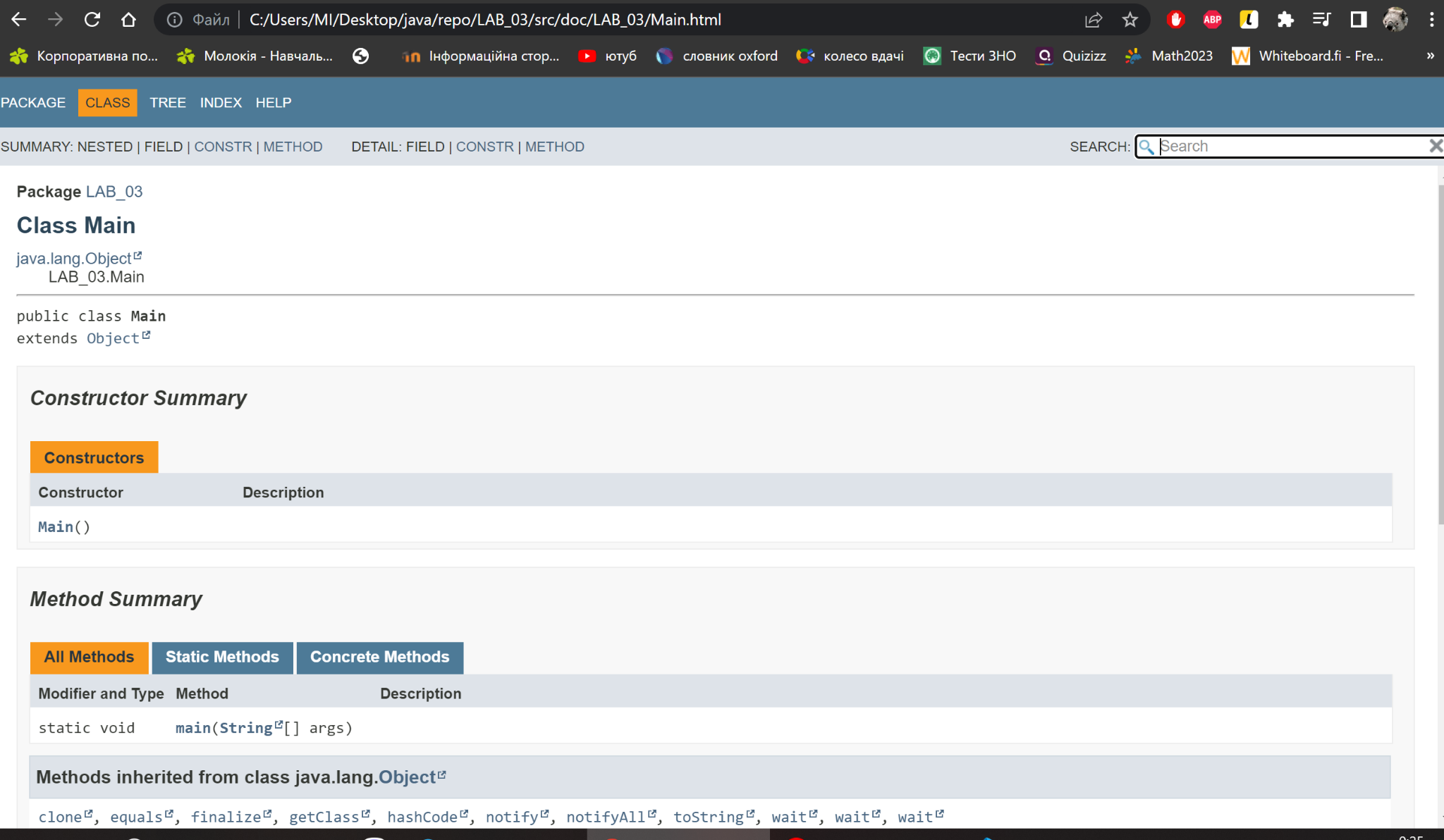
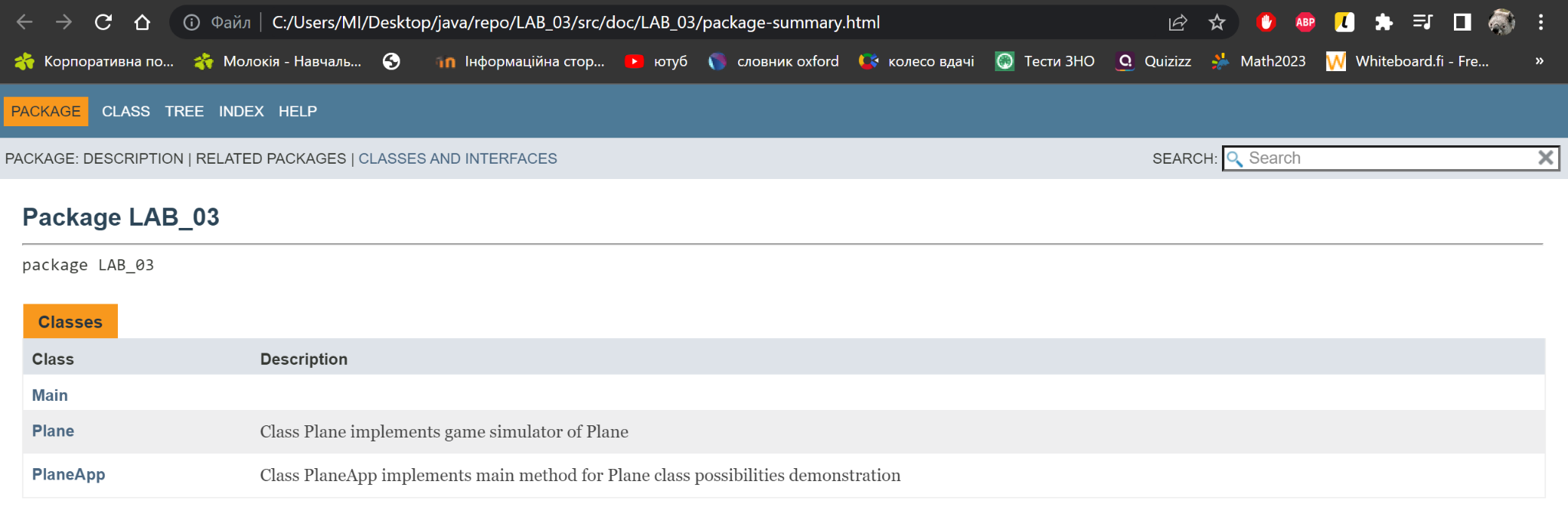
**Результати роботи програми:**

****

****

****

**Фрагмент згенерованої документації**

****

**Відповіді на контрольні запитання**

1. Поле (властивість) – це дані-члени класу, що призначені для зберігання стану об’єкту. Поле може бути статичним (в цьому випадку воно називається полем класу), незмінним (константне поле), простим типом чи об’єктом та мати різні рівні доступу, що визначаються специфікатором доступу. Синтаксис оголошення поля наступний: private final int i;
2. Створення пакетів відбувається за допомогою оператора package з вказуванням назв пакету і під пакетів (за необхідності), що розділені крапкою. Оператор package вказується на початку тексту програми перед операторами import та визначенням класу. Синтаксис оператора package:

package НазваПакету{.НазваПідпакету};

**Висновок:** Я ознайомився з процесом розробки класів та пакетів мовою Java.