Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №4

Варіант – 6

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав: ст. гр. КІ-35

Павлик С. В.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2022

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

**Завдання(Варіант 6)**

1. 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.

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. Що таке суперклас та підклас?

2. Що таке інтерфейс?

**Варіант завдання: 6. Бомбардувальник**

**Код програми:**

**Файл Main.java**

/\*\*

\* LAB\_04 package

\*/

package LAB\_04;

import java.io.IOException;

/\*\*

\* Class Main realize main method for Plane class possibilities demonstration

\*

\* @author Pavlyk Serhii

\* @version 1.0

\*/

/\*\*

\* Constructor

\* @throws IOException

\*/

public class Main {

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

PlaneApp planeApp = new PlaneApp();

planeApp.setVisible(true);

}

}

**Файл Plane\_component.java**

/\*\*

\* LAB\_04 package

\*/

package LAB\_04;

import java.awt.\*;

import java.awt.geom.AffineTransform;

import javax.swing.JOptionPane;

import javax.swing.ImageIcon;

/\*\*

\* Class Pilot realize component of Plane as Pilot

\* whith responsible for Plane class possibilities moving

\*

\* @author Pavlyk Serhii

\* @version 1.0s

\*/

class Pilot

{/\*\*

\* Constructor load image of an airplane

\*/

public Pilot()

{

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

}

/\*

\* pilot\_size static field equal 64

\*/

public static final double pilot\_size = 64;

private double pilot\_x,pilot\_y;

private float angle =0;

private final Image image;

/\*\* Method ChangeAngle changes the angle of inclination of the plane

\* @param angle the angle by which the value changes

\*/

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 BuildPilot creates a graphic model of an airplane

\* @param d2 Graphics2D

\* \*/

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 move sets a specific position to move the aircraft to

\* @param x set x position

\* @param y set y position

\*

\*/

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 realize component of Plane as Helm

\* whith responsible 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;

private boolean StartShoot;

private boolean EndShoot;

/\*\*

\* 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;

}

/\*\*

\* Method return the getStartShoot

\* \*/

public boolean getStartShoot() {

return StartShoot;

}

/\*\*

\* Method set the setStartShoot

\* \*/

public void setStartShoot(boolean startShoot) {

StartShoot = startShoot;

}

/\*\*

\* Method return the getEndShoot

\* \*/

public boolean getEndShoot() {

return EndShoot;

}

/\*\*

\* Method set the setEndShoot

\* \*/

public void setEndShoot(boolean endShoot) {

EndShoot = endShoot;

}

}

/\*\*

\* Class Engine realize component of Plane as Engine

\* whith responsible 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 maximum speed

\* @param VolumeOfFuel hold the fuel tank in liters

\*/

public Engine (float Max\_speed,float VolumeOfFuel)

{

this.MAX\_speed = Max\_speed;

this.VolumeOfFuel = VolumeOfFuel;

}

/\*\*

\* Constructor

\* @param VolumeOfFuel hold the fuel tank in liters

\* @throws OutOfMemoryError

\*/

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 SpeedUp controls SpeedUp

\* \*/

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 SpeedDown controls the SpeedDown

\* \*/

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\_04 package

\*/

package LAB\_04;

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 java.util.ArrayList;

import java.util.List;

import javax.swing.JOptionPane;

/\*\*

\* Class Plane realize 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;

private Graphics2D graphics2D;

private BufferedImage image;

private Pilot pilot;

private Helm helm;

private Engine engine;

private float VolumeOfFuel;

private float MaxSpeed;

private PrintWriter fout;

private List<Bomb> bombs;

private int shootTime;

/\*\*

\* Constructor

\* @param volumeOfFuel volume of fuel

\* @param MaxSpeed - maximum speed of the aircraft

\* @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

\* @throws IOException

\*/

Plane (float volumeOfFuel) throws IOException

{

this.VolumeOfFuel = volumeOfFuel;

this.MaxSpeed = 1;

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

}

/\*\*

\* Method start starts the game

\*/

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();

loadBombs();

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 GoToSleep puts to sleep the thread in which it is called

\* @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 BuildBackground create 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 BuildPlane creates an airplane and use Bomb

\*/

private void BuildPlane()

{

pilot.BuildPilot(graphics2D);

for(int i = 0; i < bombs.size(); i++)

{

Bomb bomb = bombs.get(i);

if(bomb != null)

{

bomb.draw(graphics2D);

fout.write("Draw bomb was successful\n");

}

}

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

}

/\*\*

\* Method creates an object of the class Pilot

\*/

private void callPilot()

{

pilot = new Pilot();

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

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

}

/\*\*

\* Method creates an object of the class Helm та Engine

\* @param VolumeOfFuel hold the fuel tank in liters

\*/

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);

else if (e.getKeyCode() == KeyEvent.VK\_T) helm.setStartShoot(true);

else if (e.getKeyCode() == KeyEvent.VK\_G) helm.setEndShoot(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);

else if (e.getKeyCode() == KeyEvent.VK\_T) helm.setStartShoot(false);

else if (e.getKeyCode() == KeyEvent.VK\_G) helm.setEndShoot(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.getStartShoot() || helm.getEndShoot())

{

if(shootTime == 0)

{

if(helm.getStartShoot())

bombs.add(0,new Bomb(pilot.getPilot\_x(), pilot.getPilot\_y(), pilot.getAngle(), 9, 1.5f));

else

bombs.add(0,new Bomb(pilot.getPilot\_x()+Pilot.pilot\_size, pilot.getPilot\_y()+Pilot.pilot\_size, pilot.getAngle(), 20, 2f));

}

shootTime++;

if (shootTime == 15) shootTime =0;

}

else shootTime = 0;

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 fall ends the game if the plane runs out of fuel and returns it to its starting position

\*/

private boolean fall()

{

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

pilot.ChangeAngle(0);

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

return true;

}

/\*\*

\* Method performs checks to see if the plane does not go beyond the limits of the application,

\* if it does, then corrects its coordinates

\*/

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 Render generates graphic objects

\*/

private void Render()

{

Graphics graphaics = getGraphics();

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

graphaics.dispose();

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

}

/\*\*

\* Method GoMoving responsible for the movement of the Plane

\*/

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();

}

/\*\*

\* Method loadBombs creates bombs so that the plane can shoot

\*/

private void loadBombs()

{

bombs = new ArrayList<>();

new Thread (new Runnable() {

@Override

public void run()

{

while(ready)

{

for (int i = 0; i < bombs.size(); i++)

{

Bomb bomb = bombs.get(i);

if(bomb != null)

{

bomb.update();

if(!bomb.Cheeck(width, height))

{

bombs.remove(bomb);

}

}

else bombs.remove(bomb);

}

GoToSleep(1);

}

}

}).start();

}

/\*\*

\* Method return the ready

\* \*/

public boolean isReady()

{

return ready;

}

}

**Файл PlaneApp.java**

/\*\*

\* LAB\_04 package

\*/

package LAB\_04;

import javax.swing.JFrame;

import java.awt.BorderLayout;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.IOException;

/\*\*

\* Class PlaneApp responsible for Plane class possibilities demonstration

\*

\* @author Pavlyk Serhii

\* @version 1.0

\*/

public class PlaneApp extends JFrame {

/\*\*

\* Constructor

\* @throws IOException

\*/

public PlaneApp ()throws IOException{

SetParameters();

}

/\*\*

\* Method SetParameters - починає встановює параметри для робити додатку

\* @throws IOException

\*/

private void SetParameters ()throws IOException

{

setTitle("BomberPLane");

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();

}

}

**Файл Bomb.java**

/\*\*

\* LAB\_04 package

\*/

package LAB\_04;

import java.awt.\*;

import java.awt.geom.Ellipse2D;

import java.awt.geom.AffineTransform;

/\*\*

\* abstract interface Bomb\_color declaration

\* @param color - set Bomb color

\*/

abstract interface Bomb\_color

{

Color color = new Color(4, 49, 107);

}

/\*\*

\* interface Bomb\_color declaration extends Bomb\_color

\* update - update sth

\* draw - draw sth by Graphics2D

\*/

interface Bomb\_things extends Bomb\_color

{

void update();

void draw( Graphics2D d2);

}

/\*\*

\* Class Bomb implements interface Bomb\_things realize Bomb

\* @author Pavlyk Serhii

\* @version 1.0

\*/

public class Bomb implements Bomb\_things {

private double bomb\_x;

private double bomb\_y;

private final Shape shape;

private final float bomb\_angele;

private double size;

private float speed = 1f;

/\*\*

\* Constructor

\* @param x X coordinate of Bomb

\* @param y Y coordinate of Bomb

\* @param angele Angle of Bomb

\* @param size sizr of Bomb

\* @param speed fall speed of Bomb

\*/

public Bomb(double x, double y, float angele,double size,float speed)

{

x += Pilot.pilot\_size /2 - size/2;

y += Pilot.pilot\_size /2 - size/2;

this.bomb\_x = x;

this.bomb\_y = y;

this.bomb\_angele = angele;

this.size = size;

this.shape = new Ellipse2D.Double(0,0,size,size);

this.speed = speed;

}

/\*

\* Method update() update position of Bomb

\* Realize method from interfase Bomb\_things

\*/

@Override

public void update() {

bomb\_x += Math.cos(Math.toRadians(bomb\_angele))\*speed;

bomb\_y += Math.sin(Math.toRadians(bomb\_angele))\*speed;

}

/\*

\* Method draw() draw Bomb by Graphics2D

\* Realize method from interfase Bomb\_things

\*/

@Override

public void draw(Graphics2D d2) {

AffineTransform oldTransform = d2.getTransform();

d2.setColor(color);

d2.translate(bomb\_x,bomb\_y);

d2.fill(shape);

d2.setTransform(oldTransform);

}

/\*\*

\* Method return the position of Bomb X

\*/

public double GetX()

{

return bomb\_x;

}

/\*\*

\* Method return the position of Bomb Y

\*/

public double GetY()

{

return bomb\_y;

}

/\*\*

\* Method return the size of Bomb

\*/

public double GetSize()

{

return size;

}

/\*\*

\* Method Cheeck checks if the bomb has gone off the screen

\*/

public boolean Cheeck(int wight, int height)

{

if(bomb\_x <= - size|| bomb\_y <= - size || bomb\_x > wight || bomb\_y >height)

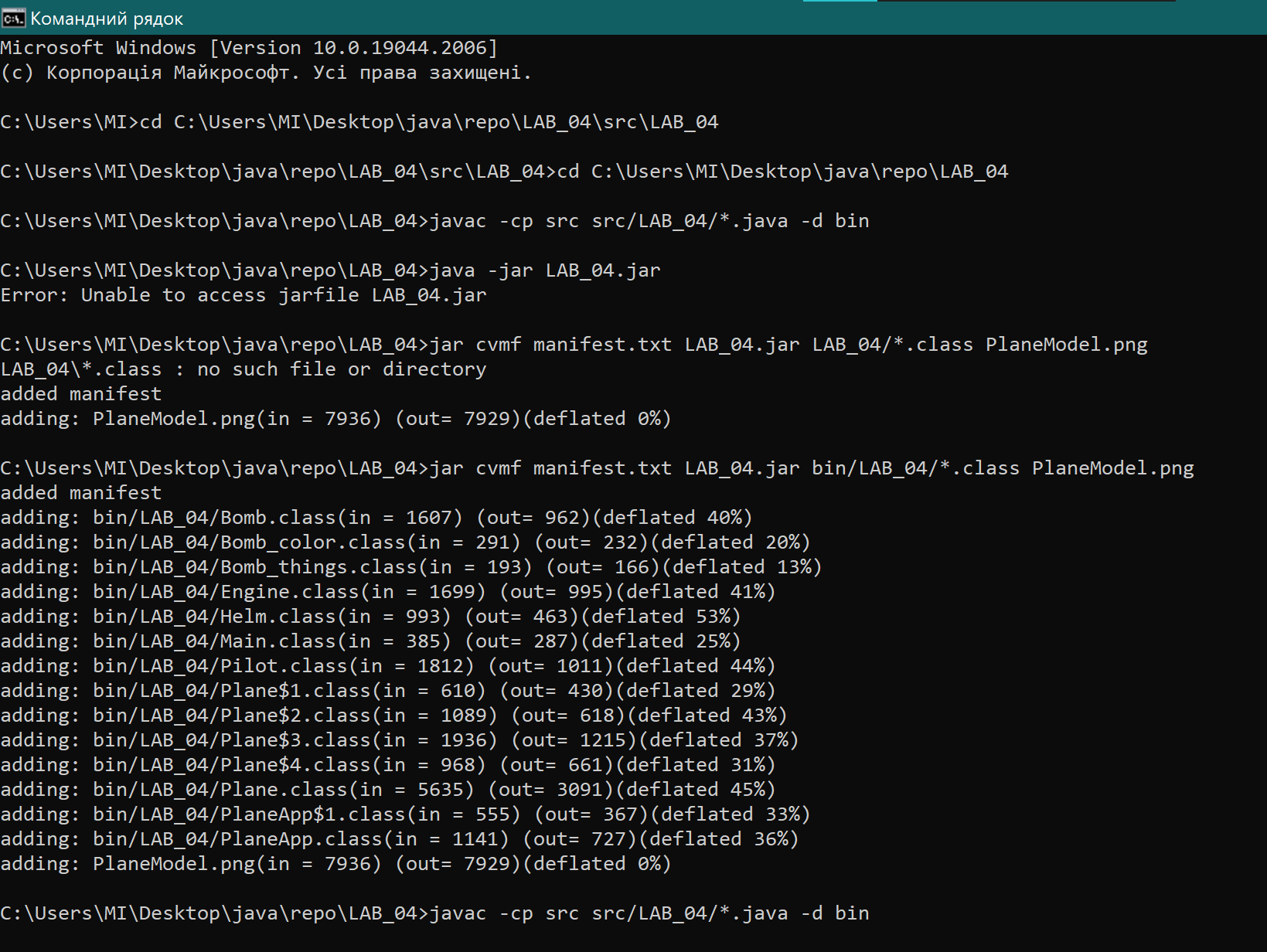
return false;

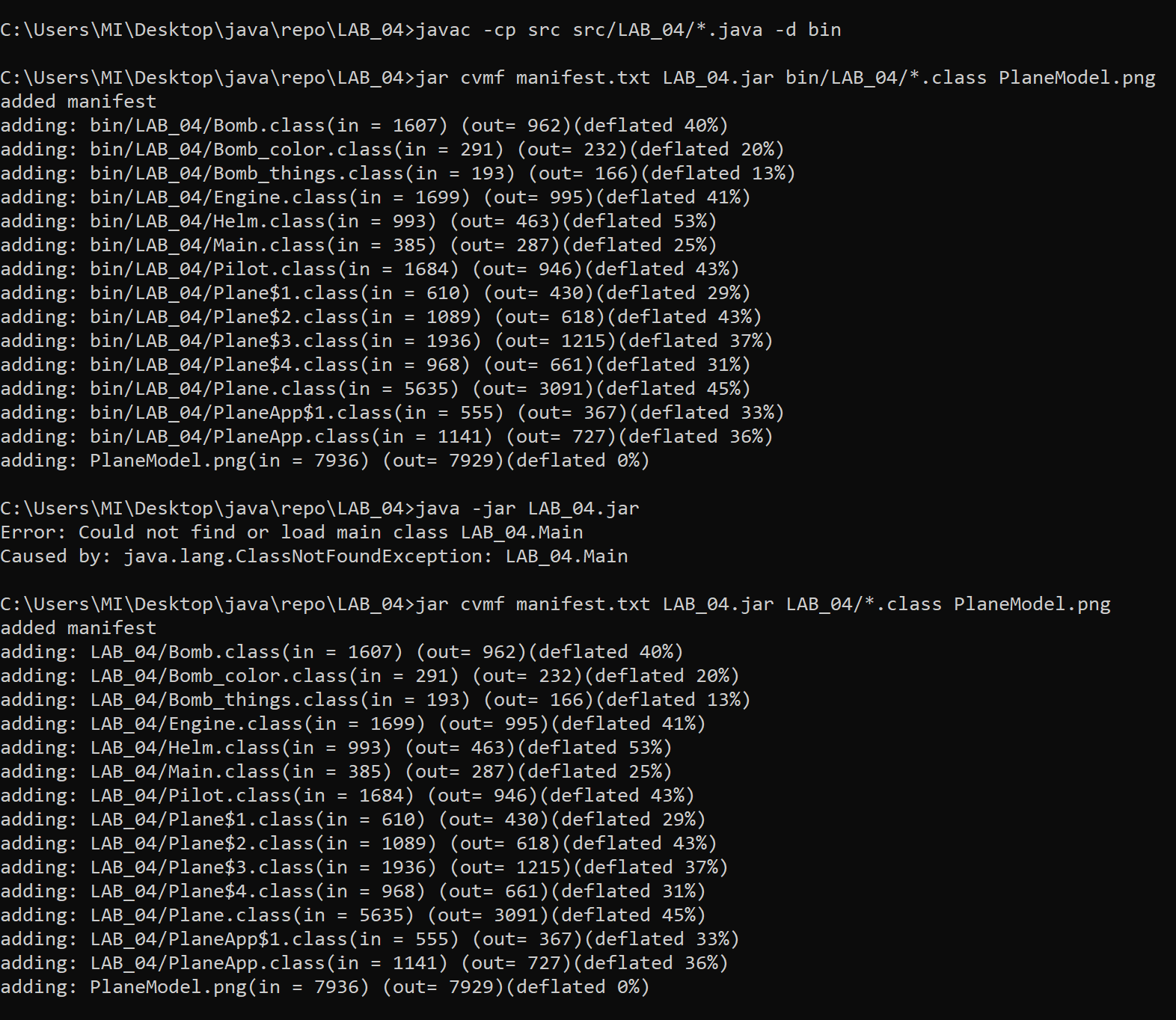
else return true;

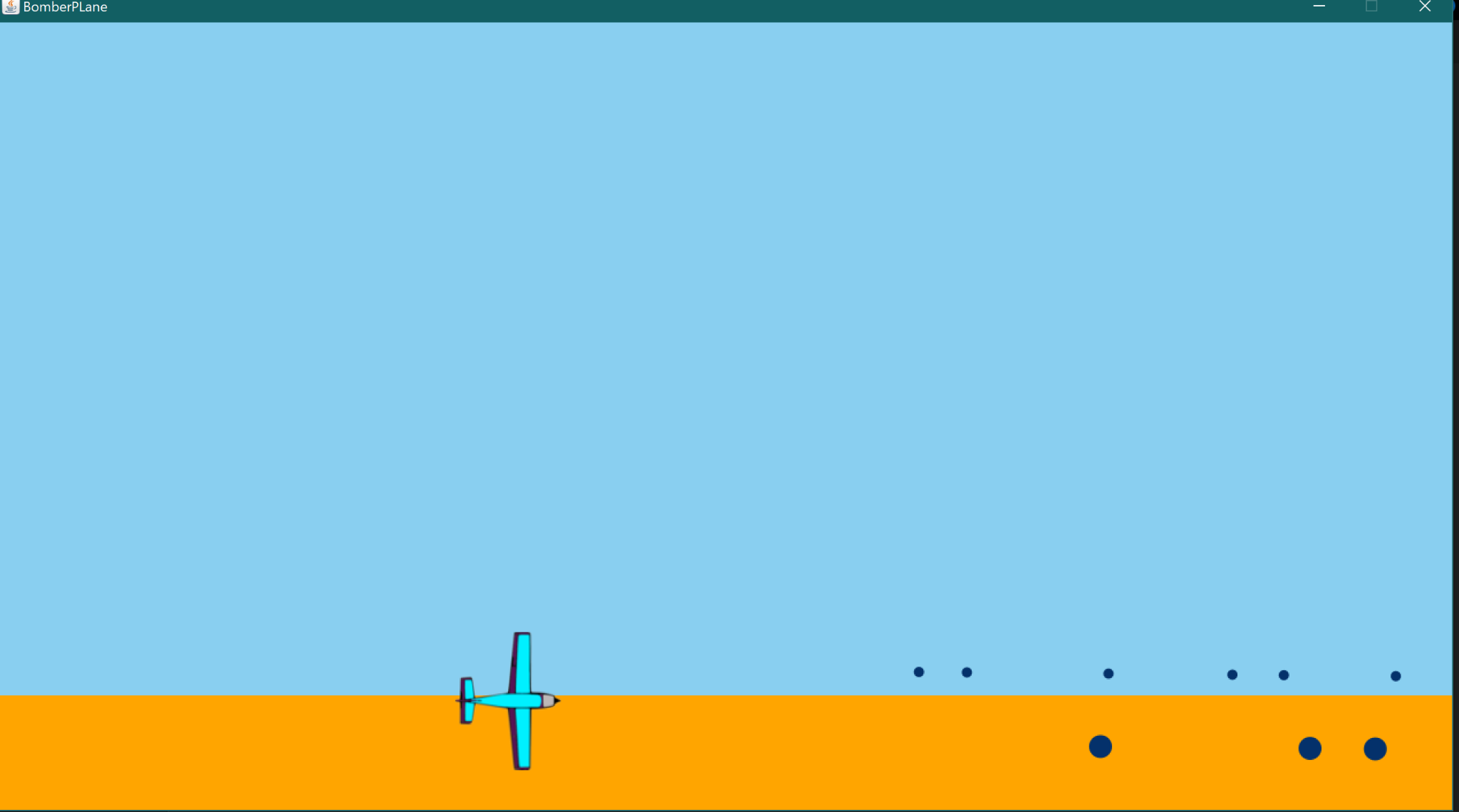
}

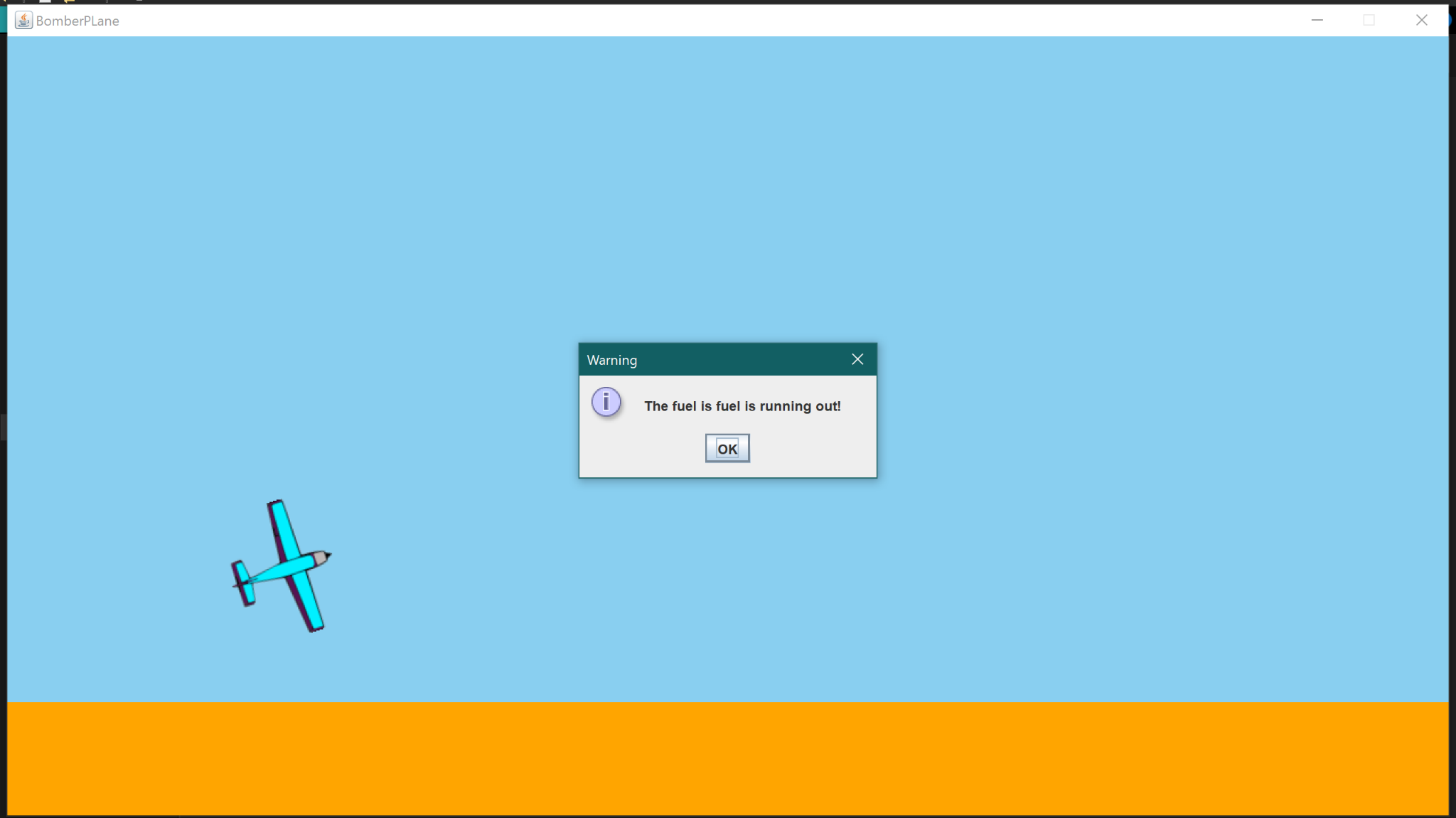
}

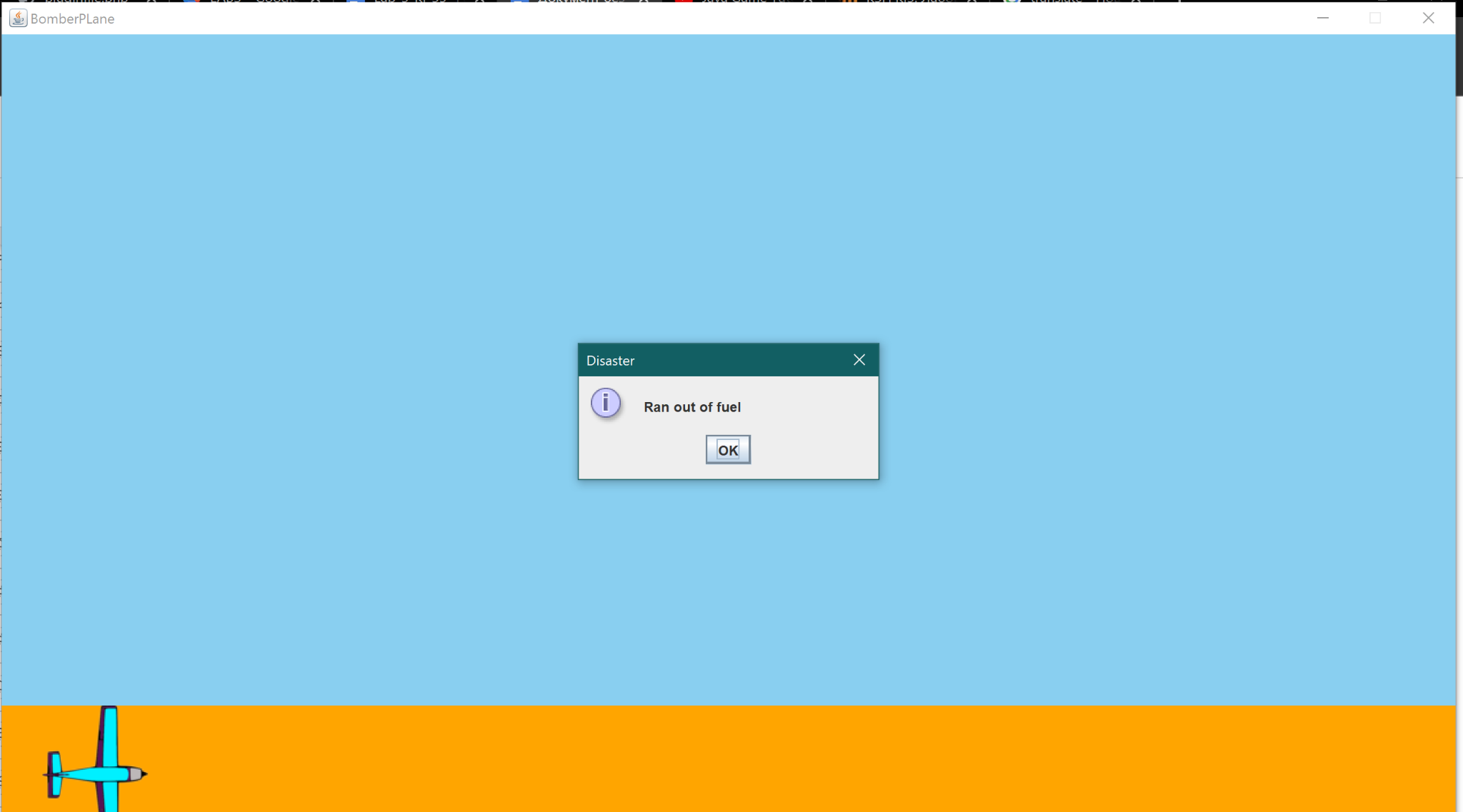
**Результати роботи програми:**

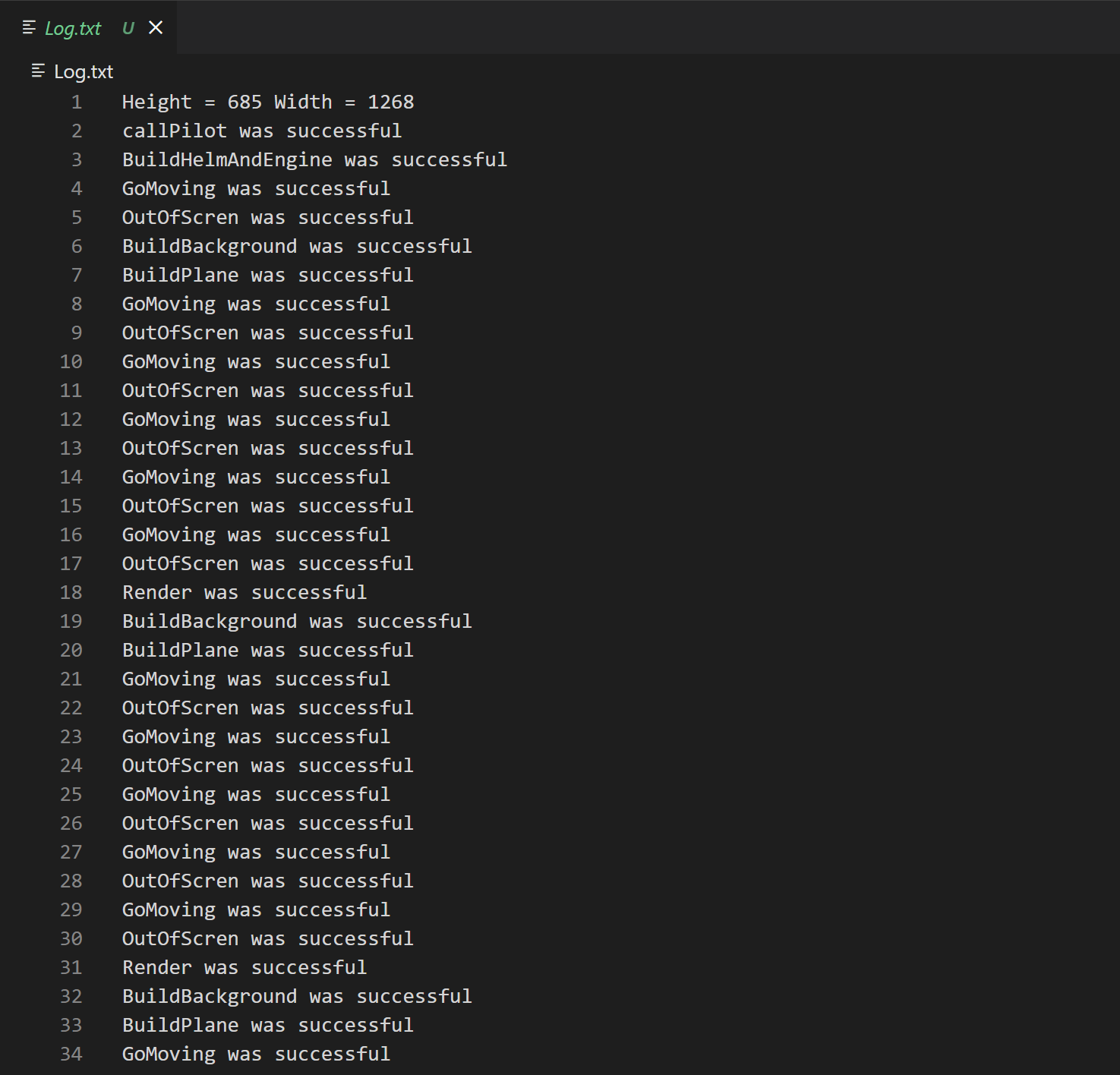




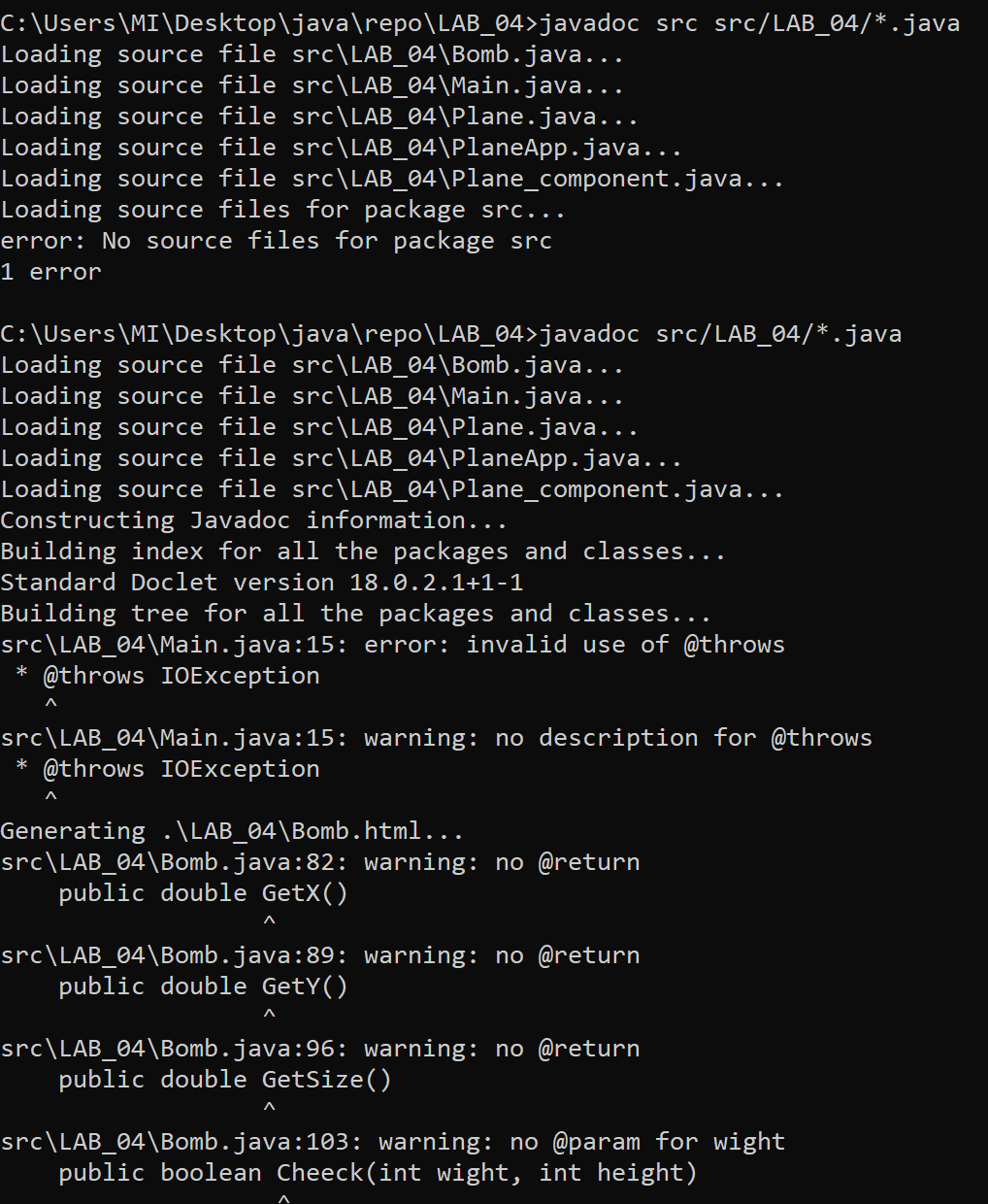
****

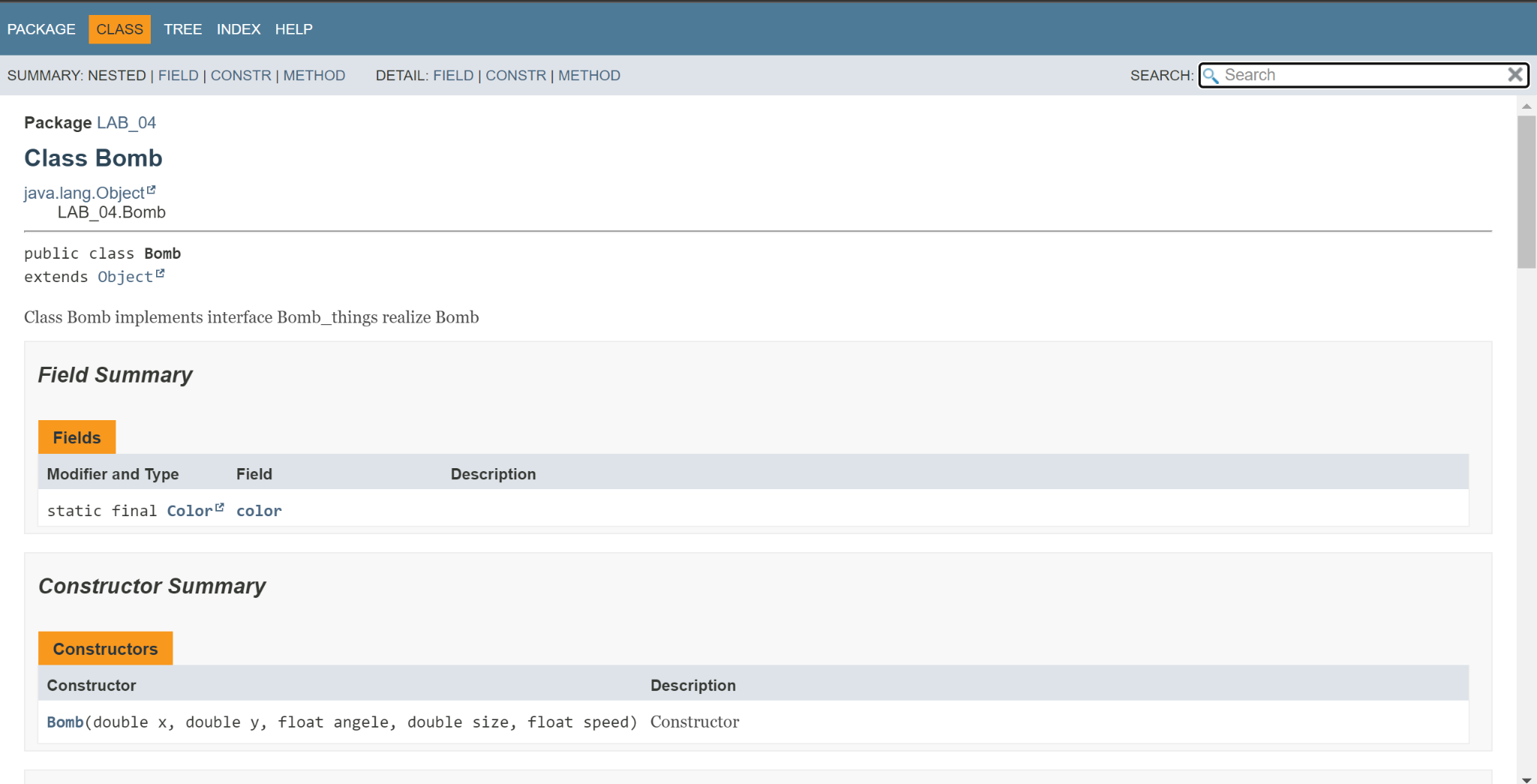
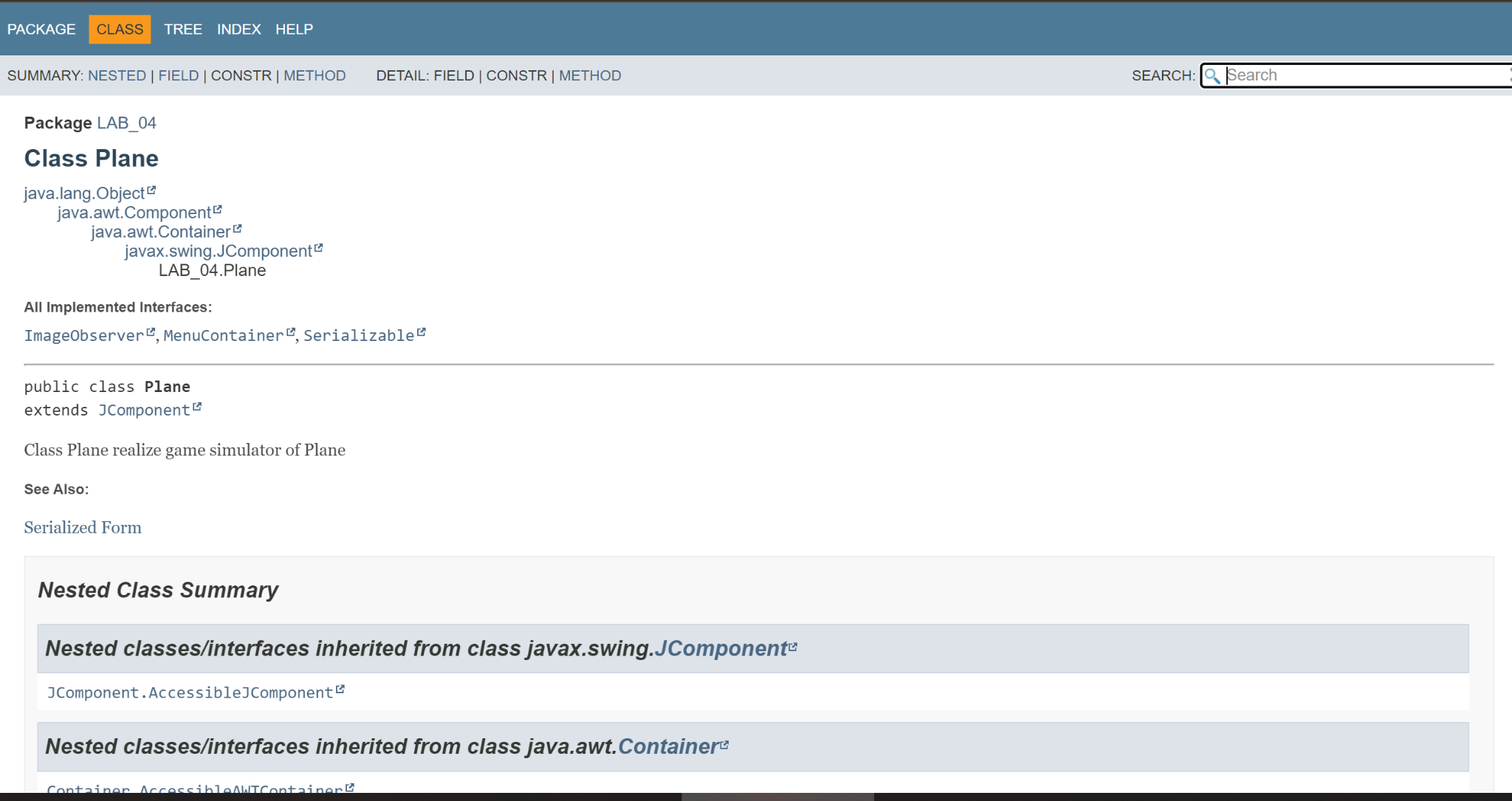
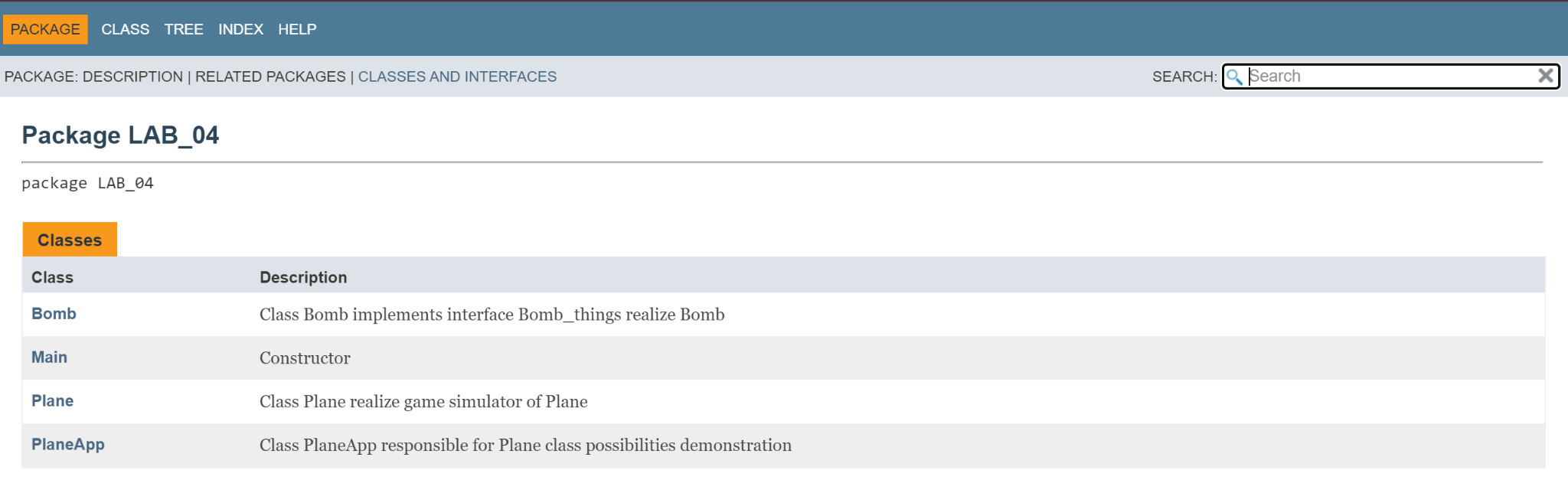
****

****

****

**Фрагмент згенерованої документації**



****

**Відповіді на контрольні запитання**

1. Базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супер множині.
2. 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:** Я ознайомився з спадкуванням та інтерфейсами у мові Java.