

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

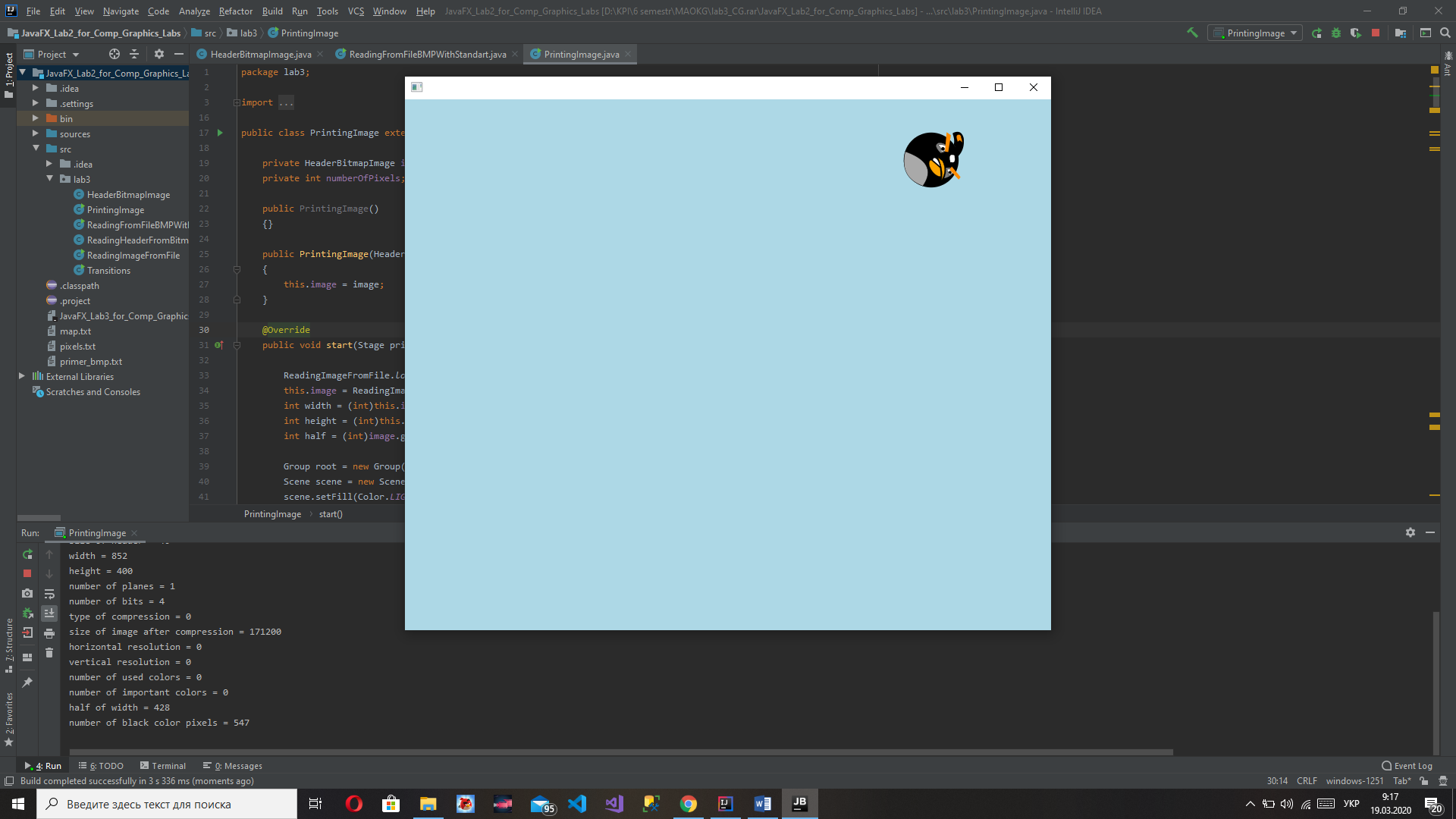
**Лабораторна робота № 3**

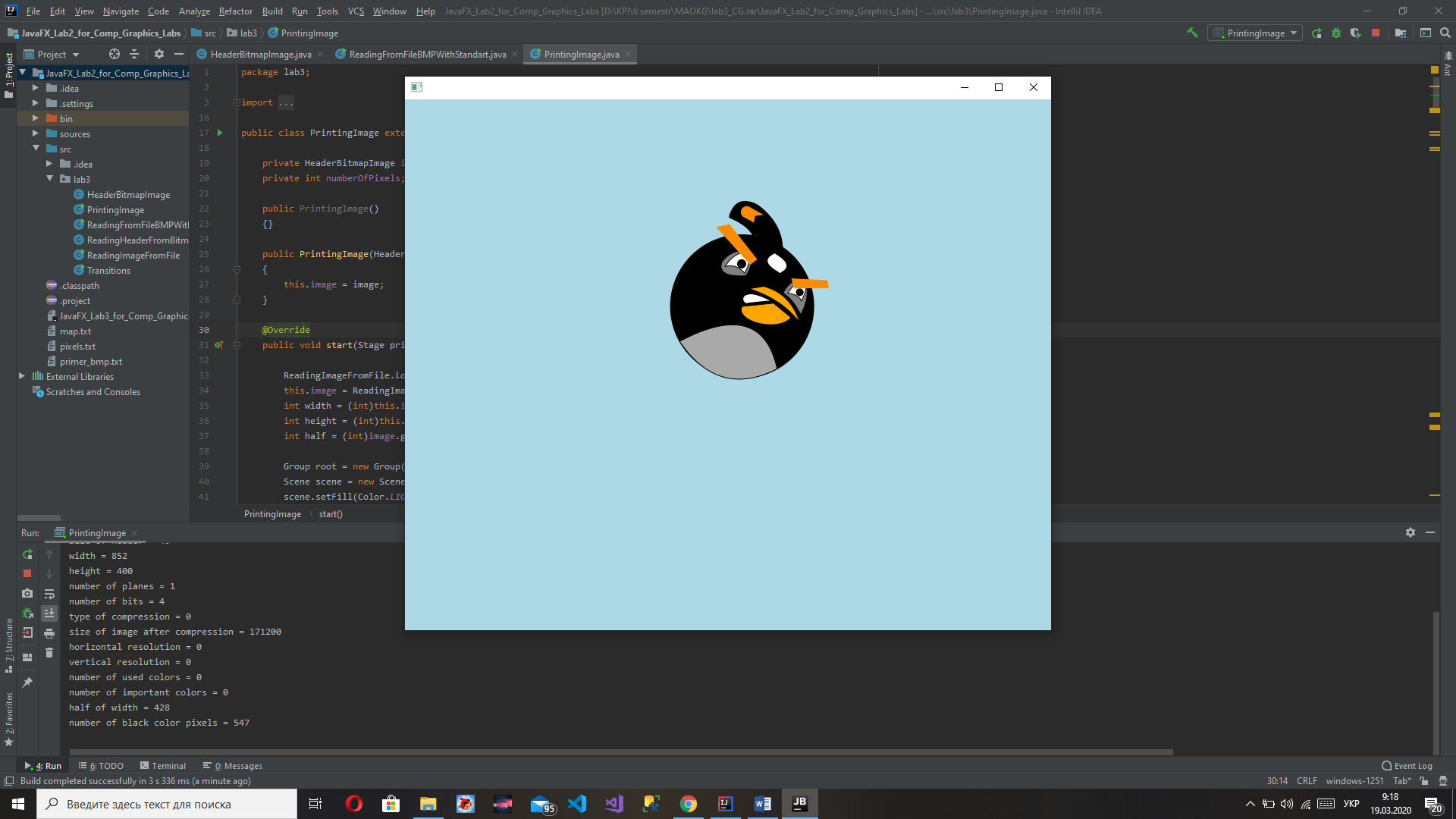
з дисципліни “ МАОКГ”

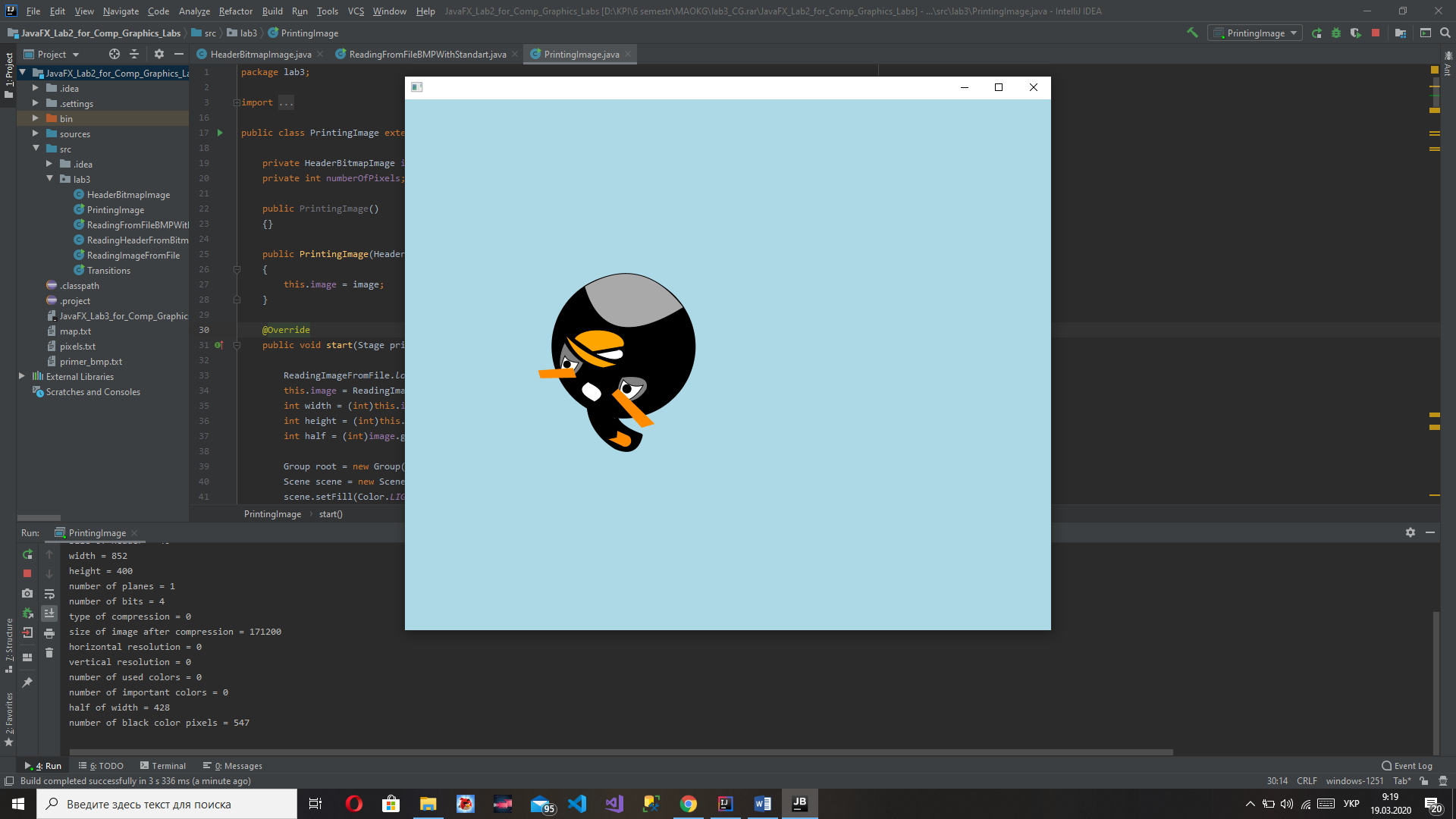
|  |  |  |
| --- | --- | --- |
| Виконав  студент III курсу  групи КП-72  Чумак Михайло Сергійович  (*прізвище, ім’я, по батькові*)  Варіант № 21 |  | Зарахована  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2019  р.  викладачем  Шкурат О. С.  (*прізвище, ім’я, по батькові*) |

Київ-2019

Результат :







Лістинг коду програми (class PrintingImage):

package lab3;  
  
import java.io.BufferedInputStream;  
import java.io.BufferedOutputStream;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
  
import javafx.animation.\*;  
import javafx.application.Application;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.\*;  
import javafx.stage.Stage;  
import javafx.util.Duration;  
  
public class PrintingImage extends Application{  
  
 private HeaderBitmapImage image; // приватне поле, яке зберігає об'єкт з інформацією про заголовок зображення  
 private int numberOfPixels; // приватне поле для збереження кількості пікселів з чорним кольором  
   
 public PrintingImage()  
 {}  
   
 public PrintingImage(HeaderBitmapImage image) // перевизначений стандартний конструктор  
 {  
 this.image = image;  
 }  
   
 @Override  
 public void start(Stage primaryStage) throws Exception {  
   
 ReadingImageFromFile.*loadBitmapImage*("D:/KPI/6 semestr/MAOKG/lab3\_CG.rar/JavaFX\_Lab2\_for\_Comp\_Graphics\_Labs/sources/trajectory.bmp");  
 this.image = ReadingImageFromFile.*pr*.image;  
 int width = (int)this.image.getWidth();  
 int height = (int)this.image.getHeight();  
 int half = (int)image.getHalfOfWidth();  
   
 Group root = new Group();   
 Scene scene = new Scene (root, width, 300 + height);  
 scene.setFill(Color.*LIGHTBLUE*);  
 Circle cir;  
   
 int let = 0;  
 int let1 = 0;  
 int let2 = 0;  
 char[][] map = new char[width][height];  
 // виконуємо зчитування даних про пікселі  
 BufferedInputStream reader = new BufferedInputStream (new FileInputStream("pixels.txt"));  
   
   
 for(int i=0;i<height;i++) // поки не кінець зображення по висоті  
 {   
 for(int j=0;j<half;j++) // поки не кінець зображення по довжині  
 {  
 let = reader.read(); // зчитуємо один символ з файлу  
 let1=let;  
 let2=let;   
 let1=let1&(0xf0); // старший байт - перший піксель  
 let1=let1>>4; // зсув на 4 розряди   
 let2=let2&(0x0f); // молодший байт - другий піксель   
 if(j\*2<width) // так як 1 символ кодує 2 пікселі нам необхідно пройти до середини ширини зображення  
 {   
 cir = new Circle ((j)\*2,(height-1-i),1, Color.*valueOf*((returnPixelColor(let1)))); // за допомогою стандартного   
 // примітива Коло радіусом в 1 піксель та кольором визначеним за допомогою методу returnPixelColor малюємо піксель  
 //root.getChildren().add(cir); //додаємо об'єкт в сцену   
 if (returnPixelColor(let1) == "BLACK") // якщо колір пікселя чорний, то ставимо в масиві 1  
 {  
 map[j\*2][height-1-i] = '1';  
 numberOfPixels++; // збільшуємо кількість чорних пікселів  
 }  
 else  
 {  
 map[j\*2][height-1-i] = '0';   
 }  
 }  
 if(j\*2+1<width) // для другого пікселя  
 {   
 cir = new Circle ((j)\*2+1,(height-1-i),1,Color.*valueOf*((returnPixelColor(let2))));  
 //root.getChildren().add(cir);  
 if (returnPixelColor(let2) == "BLACK")  
 {  
 map[j\*2+1][height-1-i] = '1';  
 numberOfPixels++;  
 }  
 else  
 {   
 map[j\*2+1][height-1-i] = '0';   
 }  
 }  
 }  
 }  
 primaryStage.setScene(scene); // ініціалізуємо сцену  
 primaryStage.show(); // візуалізуємо сцену  
   
 reader.close();  
   
 int[][] black;  
 black = new int[numberOfPixels][2];   
 int lich = 0;  
   
 BufferedOutputStream writer = new BufferedOutputStream (new FileOutputStream("map.txt")); // записуємо карту для руху по траекторії в файл  
 for(int i=0;i<height;i++) // поки не кінець зображення по висоті  
 {   
 for(int j=0;j<width;j++) // поки не кінець зображення по довжині  
 {  
 if (map[j][i] == '1')  
 {  
 black[lich][0] = j;  
 black[lich][1] = i;  
 lich++;  
 }  
 writer.write(map[j][i]);  
 }  
 writer.write(10);  
 }  
 writer.close();  
   
 System.*out*.println("number of black color pixels = " + numberOfPixels);  
   
 Path path2 = new Path();  
 for (int l=0; l<numberOfPixels-1; l++)  
 {  
 path2.getElements().addAll(  
 new MoveTo(black[l][0],black[l][1]),  
 new LineTo (black[l+1][0],black[l+1][1])  
 );  
 }  
  
  
 //animation  
 PathTransition pathTransition = new PathTransition();  
 pathTransition.setDuration(Duration.*millis*(5000));  
 pathTransition.setPath(path2);  
  
  
 //Body  
 {  
 Ellipse ellipse = new Ellipse();  
 ellipse.setCenterX(115);  
 ellipse.setCenterY(140);  
 ellipse.setRadiusX(95);  
 ellipse.setRadiusY(95);  
 ellipse.setFill(Color.*BLACK*);  
 root.getChildren().add(ellipse);  
 //pathTransition.setNode(ellipse);  
 }  
  
  
  
 // Hair  
 {  
 Path p = new Path();  
 MoveTo mv = new MoveTo(135, 50);  
 QuadCurveTo qt1 = new QuadCurveTo(122, 20, 85, 10);  
 QuadCurveTo qt2 = new QuadCurveTo(56, 7, 58, 35);  
 QuadCurveTo qt3 = new QuadCurveTo(84, 36, 95,46);  
 p.setStrokeWidth(5);  
 p.setStrokeLineJoin(StrokeLineJoin.*ROUND*);  
 p.setStrokeLineCap(StrokeLineCap.*ROUND*);  
 p.setStroke(Color.*BLACK*);  
 p.setFill(Color.*BLACK*);  
 p.getElements().addAll(mv, qt1, qt2, qt3);  
 root.getChildren().add(p);  
  
 }  
  
 // Lower gray ellipse  
 {  
 Path p = new Path();  
 MoveTo mv = new MoveTo(55, 213);  
 QuadCurveTo qt1 = new QuadCurveTo(135, 120, 188, 200);  
 QuadCurveTo qt2 = new QuadCurveTo(130, 265, 55, 213);  
 p.setFill(Color.*DARKGRAY*);  
 p.getElements().addAll(mv, qt1, qt2);  
 root.getChildren().add(p);  
 }  
  
 // Hairstyle  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(88, 23),  
 new LineTo(99, 17),  
 new LineTo(78, 14),  
 new ArcTo(1, 1, 0, 73, 30, false, false),  
 new LineTo(90, 32),  
 new LineTo(88, 23)  
 );  
 p.setStrokeWidth(1);  
 p.setFill(Color.*DARKORANGE*);  
 root.getChildren().add(p);  
 }  
  
 //Head white round  
 {  
 Path p1 = new Path();  
 MoveTo m = new MoveTo(145,80);  
 QuadCurveTo qt = new QuadCurveTo(160, 65, 130, 60);  
 p1.setFill(Color.*WHITE*);  
 p1.setStroke(Color.*WHITE*);  
 p1.getElements().addAll(m, qt);  
  
 Line l = new Line();  
 l.setStartX(145); l.setStartY(79);  
 l.setEndX(130); l.setEndY(62);  
 l.setStrokeWidth(3);  
 l.setStroke(Color.*WHITE*); l.setFill(Color.*WHITE*);  
  
 Path p = new Path();  
 MoveTo mv = new MoveTo(145, 80);  
 QuadCurveTo qt1 = new QuadCurveTo(114, 76, 130, 60);  
 p.setFill(Color.*WHITE*);  
 p.setStroke(Color.*WHITE*);  
 p.getElements().addAll(mv, qt1);  
 root.getChildren().addAll(p, l, p1);  
 }  
  
 //mouth lower part  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(115, 140),  
 new LineTo(152, 121),  
 new LineTo(180, 130),  
 new ArcTo(6, 3, -15, 115, 140, false, true)  
 );  
 p.setStrokeWidth(1);  
 p.setFill(Color.*ORANGE*);  
 p.setStrokeLineCap(StrokeLineCap.*ROUND*);  
 p.setStrokeLineJoin(StrokeLineJoin.*ROUND*);  
 root.getChildren().add(p);  
 }  
  
 //mouth white part  
 {  
 Path p = new Path();  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*WHITE*);  
 p.setFill(Color.*WHITE*);  
 p.getElements().add(new MoveTo(145, 120));  
 p.getElements().add(new QuadCurveTo(122, 118, 116, 124));  
 //p.getElements().add(new LineTo(116, 124));  
 p.getElements().add(new ArcTo(3, 3, 0, 120, 133, false, false));  
 root.getChildren().add(p);  
 }  
  
 //mouth upper part  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(115, 115),  
 new LineTo(132, 105),  
 new QuadCurveTo(169, 101, 193, 132),  
 new QuadCurveTo(160, 112, 115, 115)  
 );  
 p.setStrokeWidth(1);  
 p.setFill(Color.*ORANGE*);  
 root.getChildren().add(p);  
 }  
  
 //gray part  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(70, 95),  
 new ArcTo(4, 6,20, 104, 100, false, true),  
 new QuadCurveTo(70,115, 70, 95)  
 );  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*GRAY*);  
 p.setFill(Color.*GRAY*);  
 root.getChildren().add(p);  
  
 Path path = new Path();  
 path.getElements().addAll(  
 new MoveTo(160, 103),  
 new ArcTo(3, 6,0, 190, 120, false, true),  
 new QuadCurveTo(176,107, 160, 103)  
 );  
  
 path.setStrokeWidth(1);  
 path.setStroke(Color.*GRAY*);  
 path.setFill(Color.*GRAY*);  
 root.getChildren().add(path);  
 }  
  
 //white part of eyes  
 {  
 Path p = new Path();  
 p.getElements().addAll(  
 new MoveTo(102, 98),  
 new ArcTo(3, 6,15, 75, 99, false, false),  
 new QuadCurveTo(89,90, 102, 98)  
 );  
 p.setStrokeWidth(1);  
 p.setStroke(Color.*BLACK*);  
 p.setFill(Color.*WHITE*);  
 root.getChildren().add(p);  
  
 Path path = new Path();  
 path.getElements().addAll(  
 new MoveTo(165, 102),  
 new ArcTo(4, 10,9, 185, 105, false, true),  
 new QuadCurveTo(179,98, 165, 102)  
 );  
 path.setStrokeWidth(1);  
 path.setStroke(Color.*BLACK*);  
 path.setFill(Color.*WHITE*);  
 root.getChildren().add(path);  
 }  
  
 //eyes  
 {  
 Circle circle = new Circle(94,88, 6);  
 circle.setFill(Color.*BLACK*);  
  
 Circle circle1 = new Circle(179, 95, 5);  
 circle1.setFill(Color.*BLACK*);  
  
 root.getChildren().addAll(circle, circle1);  
 }  
  
 {  
 Polygon polygon = new Polygon();  
 polygon.getPoints().addAll(new Double[]{  
 111.0, 75.0,  
 105.0, 85.0,  
 45.0, 55.0,  
 60.0, 45.0});  
 polygon.setFill(Color.*DARKORANGE*);  
 root.getChildren().add(polygon);  
 }  
  
 {  
 Polygon polygon = new Polygon();  
 polygon.getPoints().addAll(new Double[]{  
 162.0, 82.0,  
 170.0, 93.0,  
 213.0, 76.0,  
 207.0, 67.0  
 });  
 polygon.setFill(Color.*DARKORANGE*);  
 root.getChildren().add(polygon);  
 }  
  
  
  
 pathTransition.setNode(root);  
  
 //an  
 int cycleCount = 3;  
 int time = 2000;  
  
 RotateTransition rotateTransition = new RotateTransition(Duration.*millis*(time), root);  
 rotateTransition.setByAngle(360f);  
 rotateTransition.setCycleCount(cycleCount);  
 rotateTransition.setAutoReverse(true);  
  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*millis*(time), root);  
 scaleTransition.setToX(-1);  
 scaleTransition.setToY(-1);  
 scaleTransition.setAutoReverse(true);  
  
 ParallelTransition parallelTransition = new ParallelTransition();  
 parallelTransition.getChildren().addAll(  
 rotateTransition,  
 scaleTransition,  
 pathTransition  
 );  
 parallelTransition.setCycleCount(Timeline.*INDEFINITE*);  
 parallelTransition.play();  
  
  
 }  
   
 private String returnPixelColor (int color) // метод для співставлення кольорів 16-бітного зображення  
 {  
 String col = "BLACK";  
 switch(color)  
 {  
 case 0: return "BLACK"; //BLACK;  
 case 1: return "LIGHTCORAL"; //LIGHTCORAL;  
 case 2: return "GREEN"; //GREEN  
 case 3: return "BROWN"; //BROWN  
 case 4: return "BLUE"; //BLUE;  
 case 5: return "MAGENTA"; //MAGENTA;  
 case 6: return "CYAN"; //CYAN;  
 case 7: return "LIGHTGRAY"; //LIGHTGRAY;  
 case 8: return "DARKGRAY"; //DARKGRAY;  
 case 9: return "RED"; //RED;  
 case 10:return "LIGHTGREEN";//LIGHTGREEN  
 case 11:return "YELLOW"; //YELLOW;  
 case 12:return "LIGHTBLUE"; //LIGHTBLUE;  
 case 13:return "LIGHTPINK"; //LIGHTMAGENTA  
 case 14:return "LIGHTCYAN"; //LIGHTCYAN;  
 case 15:return "WHITE"; //WHITE;  
 }  
 return col;  
 }  
   
 public static void main (String args[])   
 {  
 *launch*(args);  
 }  
  
}