import java.applet.Applet;

import java.awt.Canvas;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.util.Random;

/\* Вариант 20

\* В следующих заданиях выполнить соответствующий рисунок в окне апплета:

\* 1. Создать классы Point и Line. Объявить массив из n объектов класса Point.

\* Для объекта класса Line определить какие из объектов Point лежат на одной стороне

\* от прямой линии и какие - на другой. Реализовать ввод данных для объекта Line и

\* случайное задание данных для объектов Point.

\*/

class Point extends Canvas {

private static final long serialVersionUID = 1L;

Color clr;

int x, y;

Dimension dim = new Dimension(20, 20);

public Point (int appWidth, int appHeight) {

super();

setMaximumSize(dim);

setBackground(null);

setBounds(0, 0, dim.width, dim.height);

clr = Color.black;

x = new Random().nextInt(appWidth - dim.width) + dim.width / 2;

y = new Random().nextInt(appHeight - dim.height) + dim.height / 2;

}

public void paint (Graphics g) {

g.setColor(clr);

g.fillOval(0, 0, dim.width, dim.height);

}

public void setColor (Color clr) {

this.clr = clr;

}

public Dimension getMinimumSize () { return dim; }

public Dimension getPreferredSize () { return dim; }

}

class Line {

Color clr;

int x1, y1, x2, y2;

public Line (Color c, int x1, int y1, int x2, int y2) {

clr = c;

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

}

public int compare (Point p) {

double tg = (double)(y2 - y1) / (x2 - x1);

int xt = (int)(x1 - (double)(y1 - p.y) / tg);

if ( p.x < xt )

return -1;

else if ( p.x > xt )

return 1;

int yt = (int)(y1 - tg \* (x1 - p.x));

if ( p.y < yt )

return 1;

else if ( p.y > yt )

return -1;

return 0;

}

}

public class task\_20 extends Applet {

private static final long serialVersionUID = 1L;

static final int CX = 600, CY = 500;

Canvas points[];

Line line;

public Color getHtmlColor (String rgb, Color def) {

if ( rgb == null || rgb.charAt(0) != '#' )

return def;

try {

return new Color(Integer.parseInt(rgb.substring(1), 16));

} catch (NumberFormatException e) {

return def;

}

}

public void init () {

setSize(CX, CY);

setLayout(null);

setBackground(Color.white);

Color clr = Color.BLACK;

int x1 = 0, y1 = 0, x2 = 0, y2 = 0, n = 0;

try {

clr = getHtmlColor(getParameter("lineColor"), Color.BLACK);

x1 = Integer.parseInt((getParameter("x1") == null)?"50":getParameter("x1"));

y1 = Integer.parseInt((getParameter("y1") == null)?"50":getParameter("y1"));

x2 = Integer.parseInt((getParameter("x2") == null)?"550":getParameter("x2"));

y2 = Integer.parseInt((getParameter("y2") == null)?"450":getParameter("y2"));

n = Integer.parseInt((getParameter("n") == null)?"10":getParameter("n"));

} catch (Exception e) {}

line = new Line(clr, x1, y1, x2, y2);

points = new Point[n];

for (Canvas p: points) {

p = new Point(this.getWidth(), this.getHeight());

int ans = ((Line)line).compare((Point)p);

Color cl;

switch (ans) {

case -1: cl = Color.red; break;

case 1: cl = Color.green; break;

default: cl = Color.yellow;

}

((Point)p).setColor(cl);

p.setLocation(((Point)p).x, ((Point)p).y);

add(p);

}

repaint();

}

public void paint (Graphics g) {

g.setColor(line.clr);

g.drawLine(line.x1, line.y1, line.x2, line.y2);

}

}