import java.applet.Applet;

import java.awt.Canvas;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

/\* Вариант 21

\* Изобразить в окне апплета гармонические колебания точки вдоль некоторого

\* горизонтального отрезка. Если длина отрезка равна q, то расстояние от точки до левого

\* конца в момент времени t можно считать равным q(1+cos(wt))/2,где w - некоторая константа.

\* Указанные величины вводятся через параметры апплета.

\*/

class MyCanv extends Canvas {

private static final long serialVersionUID = 1L;

int q, w, x, t;

Dimension dim;

private int func () {

return (int)(0.5 \* (double)q \* (1. + Math.cos(1.0 \* w \* t)));

}

public MyCanv (int q, int w) {

super();

this.q = q;

this.w = w;

t = 0;

x = func();

dim = new Dimension(q, 10);

setBackground(Color.WHITE);

setMaximumSize(dim);

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

}

public void paint (Graphics g) {

g.setColor(Color.RED);

g.drawLine(0, dim.height / 2, q, dim.height / 2);

g.setColor(Color.black);

g.fillOval(x - 5, 0, 10, dim.height);

}

public Dimension getMinimumSize () { return dim; }

public Dimension getPreferredSize () { return dim; }

public void doMove () {

t += 10;

x = func();

}

}

public class task\_21 extends Applet implements Runnable {

private static final long serialVersionUID = 1L;

static final int CX = 600, CY = 500;

Canvas canv;

Thread t = null;

public void init () {

setSize(CX, CY);

setLayout(null);

setBackground(Color.WHITE);

int q = 0, w = 0;

try {

q = Integer.parseInt((getParameter("q") == null)?"150":getParameter("q"));

w = Integer.parseInt((getParameter("w") == null)?"2":getParameter("w"));

} catch (Exception e) {

e.printStackTrace();

System.exit(0);

}

canv = new MyCanv(q, w);

canv.setLocation(this.getWidth() / 2 - ((MyCanv)canv).q / 2, this.getHeight() / 2);

add(canv);

if ( t == null )

t = new Thread(this);

t.start();

}

void doMove () {

((MyCanv)canv).doMove();

canv.setLocation(this.getWidth() / 2 - ((MyCanv)canv).q / 2, this.getHeight() / 2);

canv.repaint();

}

public void run () {

mySuspend();

while (true) {

try {

Thread.sleep(100);

} catch (InterruptedException e) {

break;

}

if ( !isSuspended() )

doMove();

}

}

public void start () {

myResume();

}

public void stop () {

mySuspend();

}

public void destroy () {

if ( t != null ) {

t.interrupt();

t = null;

}

}

volatile boolean f\_suspend = false;

boolean isSuspended () { return f\_suspend; }

void mySuspend () { f\_suspend = true; }

void myResume () { f\_suspend = false; }

}