**1.1. Класс Arcnode.java**

**package** hse.visualizer;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** hse.fortune.VCanvas;  
**import** hse.utilities.Utilities;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
  
  
**public class** VDFrame **extends** JFrame **implements** Runnable {  
  
 **private static final int *FRAME\_WIDTH*** = 1080;  
 **private static final int *FRAME\_HEIGHT*** = 720;  
 **private static final** String ***aboutText*** = **"Voronoi Diagram Visualizer app allows you to watch how Voronoi Diagram\n"**

+  
 **"is being built step by step with Fortune algorithm to the defined set of "** + **"points\n"** + **"Designed by: Nikita Marinosyan\n"** + **"Email: nikita.marinosyan@gmail.com\n"** + **"Higher School of Economics, Faculty of Computer Science\n"** + **"2017"**;  
  
 **private** VCanvas **canvas**;  
 **private** Controls **controls**;  
 **private** Thread **thread**;  
  
 **private** VDFrame()  
 {  
 init();  
  
 start();  
 **controls**.**buttons**[5].doClick();  
 **controls**.**buttons**[0].doClick();  
 }  
  
 */\*\*  
 \* Method to init the frame  
 \*/* **private void** init()  
 {  
 *// Set default properties* onPaint();  
  
 *// Create menu* createMenu();  
  
 *// Create the canvas, controls and import settings* setLayout(**new** BorderLayout());  
  
 **canvas** = **new** VCanvas(**this**.getWidth(), **this**.getHeight() - 115, 32);  
  
 add(BorderLayout.***NORTH***, **new** Settings(**canvas**));  
 add(BorderLayout.***SOUTH***, **controls** = **new** Controls(**canvas**));  
 add(BorderLayout.***CENTER***, **canvas**);  
 }  
  
 */\*\*  
 \* Method that starts the thread  
 \*/* **private void** start()  
 {  
 **if**(**thread** == **null**)  
 {  
 **controls**.**thread** = **new** Thread(**this**);  
 **thread** = **controls**.**thread**;  
 **thread**.start();  
 }  
 }  
  
 @SuppressWarnings(**"InfiniteLoopStatement"**)  
 **public void** run()  
 {  
 **if**(**thread** != **null**)  
 {  
 **while**(**true**)  
 {  
 **canvas**.init();  
 **while**(**canvas**.singleStep())  
 {  
 **try** {  
 Thread.*sleep*(25L);  
 }  
 **catch**(InterruptedException \_ex)  
 {  
 System.***out***.print(**"Program has been interrupted"**);  
 }  
 }  
 **controls**.threadRunning(**false**);  
 }  
 }  
 }  
  
 */\*\*  
 \* Method to set the default values to the frame's properties  
 \*/* **private void** onPaint()  
 {  
  
 setTitle(**"Voronoi Diagram Visualizer (Fortune Algorithm)"**);  
 setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 setSize(**new** Dimension(***FRAME\_WIDTH***, ***FRAME\_HEIGHT***));  
 setResizable(**false**);  
 setLayout(**null**);  
 setLocationRelativeTo(**null**);  
 setIconImage(**new** ImageIcon(getClass().getResource(**"/icon.png"**)).getImage());  
 }  
  
 */\*\*  
 \* Method which creates the menu bar  
 \*/* **private void** createMenu()  
 {  
  
 JMenuBar menuBar = **new** JMenuBar();  
 setJMenuBar(menuBar);  
  
 */\*--------------------------------------------- Create separator ---------------------------------------------\*/* JSeparator separator = **new** JSeparator(SwingConstants.***VERTICAL***);  
 separator.setMaximumSize(**new** Dimension(10, 50));  
  
 */\*------------------------------------------- Create help menu -----------------------------------------------\*/* JMenu helpMenu = **new** JMenu(**"Help"**);  
 menuBar.add(helpMenu);  
  
 JMenuItem manual = **new** JMenuItem(**"Manual"**);  
 JFrame manualWindow = **new** JFrame(**"Manual"**);  
 JTextPane manualText = **new** JTextPane();  
 manualText.setEditable(**false**);  
 manualText.setContentType(**"text/html"**);  
 manualText.setText(Utilities.*readAllTextFromResource*(**"manual.html"**));  
 manualText.setCaretPosition(0);  
 manualWindow.setDefaultCloseOperation(WindowConstants.***DISPOSE\_ON\_CLOSE***);  
 manualWindow.setSize(700, 700);  
 manualWindow.setResizable(**false**);  
 JScrollPane jsp = **new** JScrollPane(manualText);  
  
 manualWindow.add(jsp);  
 manualWindow.setLocation(**this**.getX() + 380, **this**.getY());  
 manual.addActionListener(e -> openManual(manualWindow));  
 helpMenu.add(manual);  
  
 JMenuItem about = **new** JMenuItem(**"About the program"**);  
 helpMenu.add(about).addActionListener(e -> openAbout());  
  
 menuBar.add(separator);  
 }  
  
 */\*\*  
 \* Method to handle manual menu button pressing  
 \* Opens new window with complete manual on the app  
 \*/* **private void** openManual(JFrame manualWindow)  
 {  
 **controls**.**buttons**[0].doClick();  
 manualWindow.setVisible(**true**);  
 }  
  
 */\*\*  
 \* Method to handle aboutMenu button pressing  
 \* Opens new window with brief info about the app  
 \*/* **private void** openAbout()  
 {  
 *// Pause visualization* **controls**.**buttons**[0].doClick();  
  
 *// Show about frame* JOptionPane.*showMessageDialog*(**this**, ***aboutText***);  
 }  
  
 **public static void** main(String[] args)  
 {  
 VDFrame vdFrame = **new** VDFrame();  
 vdFrame.setVisible(**true**);  
 }  
}

**1.2. Класс ArcTree.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
  
**class** ArcTree  
{  
 **private** ArcNode **Arcs**;  
  
 **void** insert (VPoint mypoint, **double** d, VEventQueue eventqueue)  
 {  
 **if**(**Arcs** == **null**)  
 {  
 **Arcs** = **new** ArcNode(mypoint);  
 **return**;  
 }  
  
 **try** {  
 ParabolaPoint parabolapoint = **new** ParabolaPoint(mypoint);  
 parabolapoint.init(d);  
 **Arcs**.init(d);  
 **Arcs**.insert(parabolapoint, d, eventqueue);  
 }  
 **catch**(Throwable \_ex)  
 {  
 System.***out***.println(**"\*\*\* error: No parabola intersection during fortune)"**);  
 }  
 }  
  
 **void** checkBounds (VCanvas vCanvas, **double** d)  
 {  
 **if**(**Arcs** != **null**)  
 {  
 **Arcs**.init(d);  
 **Arcs**.checkBounds(vCanvas, d);  
 }  
 }  
  
 **void** paint (Graphics g, **double** d, **boolean** flag, **boolean** drawBeach)  
 {  
 **if**(**Arcs** != **null**)  
 {  
 **Arcs**.init(d);  
 **Arcs**.paint(g, d, 0.0D, flag, drawBeach);  
 }  
 }  
}

**1.3. Класс CirclePoint.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
  
**class** CirclePoint **extends** VEventPoint  
{  
 **private double radius**;  
 **private** ArcNode **arc**;  
  
 CirclePoint(**double** X, **double** Y, ArcNode arcnode)  
 {  
 **super**(X, Y);  
 **arc** = arcnode;  
 **radius** = getDistance(arcnode);  
 **x** += **radius**;  
 }  
  
 **public void** paint(Graphics g)  
 {  
 **super**.paint(g);  
 **double** d = **radius**;  
 g.drawOval((**int**)(**x** - 2D \* d), (**int**)(**y** - d), (**int**)(2D \* d), (**int**)(2D \* d));  
 }  
  
 **public void** performAction(VCanvas vCanvas)  
 {  
 ArcNode arcNode1 = **arc**.**Prev**;  
 ArcNode arcNode2 = **arc**.**Next**;  
 VPoint vPoint = **new** VPoint(**x** - **radius**, **y**);  
  
 **arc**.completeTrace(vCanvas, vPoint);  
 arcNode1.completeTrace(vCanvas, vPoint);  
 arcNode1.**startOfTrace** = vPoint;  
 arcNode1.**Next** = arcNode2;  
 arcNode2.**Prev** = arcNode1;  
  
 **if**(arcNode1.**circlePoint** != **null**)  
 {  
 vCanvas.**Events**.remove(arcNode1.**circlePoint**);  
 arcNode1.**circlePoint** = **null**;  
 }  
 **if**(arcNode2.**circlePoint** != **null**)  
 {  
 vCanvas.**Events**.remove(arcNode2.**circlePoint**);  
 arcNode2.**circlePoint** = **null**;  
 }  
 arcNode1.checkCircle(vCanvas.**Events**);  
 arcNode2.checkCircle(vCanvas.**Events**);  
 }  
}

**1.4. Класс ParabolaPoint.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***class** ParabolaPoint **extends** VPoint  
{  
 **double a**, **b**, **c**;  
  
 ParabolaPoint(VPoint vPoint)  
 {  
 **super**(vPoint);  
 }  
  
 @SuppressWarnings(**"SuspiciousNameCombination"**)  
 **double** realX()  
 {  
 **return y**;  
 }  
  
 **private double** realY(**double** d)  
 {  
 **return** d - **x**;  
 }  
  
 CirclePoint calculateCenter(VPoint vPoint1, ArcNode arcnode, VPoint vPoint2)  
 {  
 CirclePoint circlepoint = **null**;  
 VPoint vPoint3 = **new** VPoint(arcnode.**x** - vPoint1.**x**, arcnode.**y** - vPoint1.**y**);  
 VPoint vPoint4 = **new** VPoint(vPoint2.**x** - arcnode.**x**, vPoint2.**y** - arcnode.**y**);  
 **if**(vPoint4.**y** \* vPoint3.**x** > vPoint4.**x** \* vPoint3.**y**)  
 {  
 **double** d = -vPoint3.**x** / vPoint3.**y**;  
 **double** d1 = (vPoint1.**y** + vPoint3.**y** / 2D) - d \* (vPoint1.**x** + vPoint3.**x** / 2D);  
 **double** d2 = -vPoint4.**x** / vPoint4.**y**;  
 **double** d3 = (arcnode.**y** + vPoint4.**y** / 2D) - d2 \* (arcnode.**x** + vPoint4.**x** / 2D);  
 **double** d4;  
 **double** d5;  
 **if**(vPoint3.**y** == 0.0D)  
 {  
 d4 = vPoint1.**x** + vPoint3.**x** / 2D;  
 d5 = d2 \* d4 + d3;  
 } **else  
 if**(vPoint4.**y** == 0.0D)  
 {  
 d4 = arcnode.**x** + vPoint4.**x** / 2D;  
 d5 = d \* d4 + d1;  
 } **else** {  
 d4 = (d3 - d1) / (d - d2);  
 d5 = d \* d4 + d1;  
 }  
 circlepoint = **new** CirclePoint(d4, d5, arcnode);  
 }  
 **return** circlepoint;  
 }  
  
 **void** init(**double** d)  
 {  
 **double** d1 = realX();  
 **double** d2 = realY(d);  
 **a** = 1.0D / (2D \* d2);  
 **b** = -d1 / d2;  
 **c** = (d1 \* d1) / (2D \* d2) + d2 / 2D;  
 }  
  
 **double** F(**double** d)  
 {  
 **return** (**a** \* d + **b**) \* d + **c**;  
 }  
  
 **double**[] solveQuadratic(**double** d, **double** d1, **double** d2) **throws** Throwable  
 {  
 **double** ad[] = **new double**[2];  
 **double** d3 = d1 \* d1 - 4D \* d \* d2;  
  
 **if**(d3 < 0.0D)  
 **throw new** Throwable();  
 **if**(d == 0.0D)  
 {  
 **if**(d1 != 0.0D)  
 ad[0] = -d2 / d1;  
 **else  
 throw new** Throwable();  
 }  
 **else** {  
 **double** d4 = Math.*sqrt*(d3);  
 **double** d5 = -d1;  
 **double** d6 = 2D \* d;  
 ad[0] = (d5 + d4) / d6;  
 ad[1] = (d5 - d4) / d6;  
 }  
 **return** ad;  
 }  
}

**1.5. Класс VCanvas.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.\*;  
**import** java.awt.event.MouseEvent;  
**import** java.awt.event.MouseListener;  
**import** java.awt.image.BufferedImage;  
  
**public class** VCanvas **extends** Canvas **implements** MouseListener  
{  
 **private** Graphics **offScreenGraphics**;  
 **private** BufferedImage **offScreenImage**;  
 **int XPos**;  
 VoronoiClass **Voronoi**;  
 **public boolean drawCircles**, **drawBeach**, **drawVoronoiLines**;  
 VEventQueue **Events**;  
 ArcTree **Arcs**;  
 **private int width**, **height**;  
  
 **public** VCanvas(**int** width, **int** height, **int** numberOfSites)  
 {  
 **drawCircles** = **false**;  
 **drawBeach** = **true**;  
 **drawVoronoiLines** = **true**;  
 addMouseListener(**this**);  
 **Voronoi** = **new** VoronoiClass(width, height, numberOfSites);  
 **this**.**width** = width;  
 **this**.**height** = height;  
 }  
  
 **public synchronized void** init()  
 {  
 **offScreenImage** = **new** BufferedImage(**width**, **height**, 1);  
 **offScreenGraphics** = **offScreenImage**.getGraphics();  
 **XPos** = 0;  
 **Arcs** = **new** ArcTree();  
 **Events** = **new** VEventQueue();  
 **Voronoi**.clear();  
  
 **for**(**int** i = 0; i < **Voronoi**.size(); i++)  
 **Events**.insert(**new** VEventPoint((VPoint)**Voronoi**.elementAt(i)));  
 }  
  
 **public synchronized void** paint(Graphics g)  
 {  
 g.setColor(Color.***WHITE***);  
 g.fillRect(0, 0, getBounds().**width**, getBounds().**height**);  
  
 g.setColor(Color.***BLACK***);  
 ((Graphics2D)g).setStroke(**new** BasicStroke(3));  
 **Voronoi**.paint(g, **drawVoronoiLines**);  
  
 g.setColor(Color.***RED***);  
 ((Graphics2D)g).setStroke(**new** BasicStroke(3));  
 g.drawLine(**XPos**, 0, **XPos**, getBounds().**height**);  
  
 **if**(**Events** != **null** && **Arcs** != **null**)  
 {  
 g.setColor(Color.***BLUE***);  
 ((Graphics2D) g).setStroke(**new** BasicStroke(3));  
 **Events**.paint(g, **drawCircles**);  
  
 g.setColor(**new** Color(13, 136, 0));  
 ((Graphics2D) g).setStroke(**new** BasicStroke(3));  
 **Arcs**.paint(g, **XPos**, **drawVoronoiLines**, **drawBeach**);  
 }  
 }  
  
 **public void** update(Graphics g)  
 {  
 **offScreenGraphics**.setClip(g.getClipBounds());  
 paint(**offScreenGraphics**);  
 g.drawImage(**offScreenImage**, 0, 0, **this**);  
 }  
  
 **public synchronized boolean** singleStep()  
 {  
 **if**(**Events**.**Events** == **null** || (**double**)**XPos** < **Events**.**Events**.**x**)  
 **XPos**++;  
  
 **while**(**Events**.**Events** != **null** && (**double**)**XPos** >= **Events**.**Events**.**x**)  
 {  
 VEventPoint vEventPoint = **Events**.pop();  
 **XPos** = Math.*max*(**XPos**, (**int**)vEventPoint.**x**);  
 vEventPoint.performAction(**this**);  
 **Arcs**.checkBounds(**this**, **XPos**);  
 }  
  
 **if**(**XPos** > getBounds().**width** && **Events**.**Events** == **null**)  
 **Arcs**.checkBounds(**this**, **XPos**);  
  
 repaint();  
 **return Events**.**Events** != **null** || **XPos** < 1000 + getBounds().**width**;  
 }  
  
 **public synchronized void** step()  
 {  
 VEventPoint vEventPoint = **Events**.pop();  
 **if**(vEventPoint != **null**)  
 {  
 **XPos** = Math.*max*(**XPos**, (**int**)vEventPoint.**x**);  
 vEventPoint.performAction(**this**);  
 } **else  
 if**(**XPos** < getBounds().**width**)  
 {  
 **XPos** = getBounds().**width**;  
 } **else** {  
 init();  
 }  
 **Arcs**.checkBounds(**this**, **XPos**);  
 repaint();  
 }  
  
 **public synchronized void** clear()  
 {  
 **Voronoi** = **new** VoronoiClass(getBounds().**width**, getBounds().**height**, 0);  
 restart();  
 }  
  
 **public synchronized void** restart()  
 {  
 init();  
 repaint();  
 }  
  
 @SuppressWarnings(**"unchecked"**)  
 **public synchronized void** mousePressed(MouseEvent mouseEvent)  
 {  
 VPoint vPoint = **new** VPoint(mouseEvent.getPoint());  
 **if**(vPoint.**x** > (**double**)**XPos**)  
 {  
 **Voronoi**.addElement(vPoint);  
 **Voronoi**.checkDegenerate();  
 **Events**.insert(**new** VEventPoint(vPoint));  
 repaint();  
 }  
 }  
  
 */\*\*  
 \* Methods below are empty because we need to implement MouseListener  
 \*/* **public void** mouseClicked(MouseEvent mouseevent)  
 {  
 }  
  
 **public void** mouseReleased(MouseEvent mouseevent)  
 {  
 }  
  
 **public void** mouseEntered(MouseEvent mouseevent)  
 {  
 }  
  
 **public void** mouseExited(MouseEvent mouseevent)  
 {  
 }  
}

**1.6. Класс VEventPoint.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***class** VEventPoint **extends** VPoint  
{  
 VEventPoint **Prev**, **Next**;  
  
 VEventPoint(VPoint vPoint)  
 {  
 **super**(vPoint);  
 }  
  
 VEventPoint(**double** x, **double** y)  
 {  
 **super**(x, y);  
 }  
  
 **void** insert(VEventPoint vEventPoint)  
 {  
 **if**(vEventPoint.**x** > **x** || vEventPoint.**x** == **x** && vEventPoint.**y** > **y**)  
 {  
 **if**(**Next** != **null**)  
 {  
 **Next**.insert(vEventPoint);  
 **return**;  
 } **else** {  
 **Next** = vEventPoint;  
 vEventPoint.**Prev** = **this**;  
 **return**;  
 }  
 }  
 **if**(vEventPoint.**x** != **x** || vEventPoint.**y** != **y** || (vEventPoint **instanceof** CirclePoint))  
 {  
 vEventPoint.**Prev** = **Prev**;  
 vEventPoint.**Next** = **this**;  
 **if**(**Prev** != **null**)  
 **Prev**.**Next** = vEventPoint;  
 **Prev** = vEventPoint;  
 }  
 **else** {  
 vEventPoint.**Prev** = vEventPoint;  
 System.***out***.println(**"Double point ignored: "** + vEventPoint.toString());  
 }  
 }  
  
 **public void** performAction(VCanvas mycanvas)  
 {  
 mycanvas.**Arcs**.insert(**this**, mycanvas.**XPos**, mycanvas.**Events**);  
 }  
}

**1.7. Класс VEventQueue.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
  
**class** VEventQueue  
{  
 VEventPoint **Events**;  
  
 */\*\*  
 \* Inserts the defined event to the queue  
 \** ***@param vEventPoint*** *- the event to be inserted  
 \*/* **void** insert (VEventPoint vEventPoint)  
 {  
 **if**(**Events** != **null**)  
 **Events**.insert(vEventPoint);  
  
 **if**(vEventPoint.**Prev** == **null**)  
 **Events** = vEventPoint;  
 }  
  
 */\*\*  
 \* Removes the event from the queue  
 \** ***@param vEventPoint*** *- the event to be removed  
 \*/* **void** remove (VEventPoint vEventPoint)  
 {  
 **if**(vEventPoint.**Next** != **null**)  
 vEventPoint.**Next**.**Prev** = vEventPoint.**Prev**;  
  
 **if**(vEventPoint.**Prev** != **null**)  
 vEventPoint.**Prev**.**Next** = vEventPoint.**Next**;  
 **else Events** = vEventPoint.**Next**;  
 }  
  
 */\*\*  
 \* Pops the event from the queue  
 \** ***@return*** *popped event  
 \*/* VEventPoint pop ()  
 {  
 VEventPoint vEventPoint = **Events**;  
 **if**(vEventPoint != **null**)  
 {  
 **Events** = **Events**.**Next**;  
 **if**(**Events** != **null**)  
 {  
 **Events**.**Prev** = **null**;  
 }  
 }  
 **return** vEventPoint;  
 }  
  
 **void** paint(Graphics g, **boolean** flag)  
 {  
 **for**(VEventPoint vEventPoint = **Events**; vEventPoint != **null**; vEventPoint = vEventPoint.**Next**)  
 {  
 **if**(flag || !(vEventPoint **instanceof** CirclePoint))  
 vEventPoint.paint(g);  
 }  
  
 }

**1.8. Класс VLine.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
  
**class** VLine **implements** VPaintable  
{  
 **private** VPoint **P1**, **P2**;  
  
 VLine(VPoint vPoint1, VPoint vPoint2)  
 {  
 **P1** = vPoint1;  
 **P2** = vPoint2;  
 }  
  
 **public void** paint(Graphics g)  
 {  
 g.drawLine((**int**)**P1**.**x**, (**int**)**P1**.**y**, (**int**)**P2**.**x**, (**int**)**P2**.**y**);  
 }  
}

**1.9. Класс VoronoiClass.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
**import** java.util.Vector;  
  
**class** VoronoiClass **extends** Vector<VPaintable>  
{  
 VoronoiClass (**int** width, **int** height, **int** numberOfSites)  
 {  
 **if**(numberOfSites > 0)  
 {  
 **boolean** indent = **true**;  
 **for** (**int** x = 100; x < width; x += width / 8)  
 {  
 **int** indentY = (indent ^= **true**) ? 0 : width / 16;  
 **for** (**int** y = 40 + indentY; y < height; y += height / 4)  
 addElement(**new** VPoint(x, y));  
 }  
 }  
  
 checkDegenerate();  
 }  
  
 **void** checkDegenerate ()  
 {  
 **if**(size() > 1)  
 {  
 VPoint min = (VPoint)elementAt(0), next = min;  
 **for**(**int** i = 1; i < size(); i++)  
 {  
 Object element = elementAt(i);  
 **if**(element **instanceof** VPoint)  
 {  
 **if**(((VPoint)element).**x** <= min.**x**)  
 {  
 next = min;  
 min = (VPoint)element;  
 }  
 **else if**(((VPoint)element).**x** <= min.**x**)  
 {  
 next = (VPoint)element;  
 }  
 }  
 }  
  
 **if**(min.**x** == next.**x** && min != next)  
 {  
 min.**x**--;  
 System.***out***.println(**"Moved point: "** + next.**x** + **" -> "** + min.**x**);  
 }  
 }  
 }  
  
 **void** paint (Graphics g, **boolean** flag)  
 {  
 **for**(**int** i = 0; i < size(); i++)  
 {  
 **if**(flag || !(elementAt(i) **instanceof** VLine))  
 (elementAt(i)).paint(g);  
 }  
 }  
  
 **public void** clear()  
 {  
 **for**(**int** i = 0; i < size(); i++)  
 {  
 **if**(elementAt(i) **instanceof** VLine)  
 removeElementAt(i--);  
 }  
 }  
}

**1.10. Класс VPoint.java**

**package** hse.fortune;*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
**import** java.awt.Point;  
  
**class** VPoint **implements** VPaintable  
{  
 *// Thread safe coordinates* **volatile double x**, **y**;  
  
 */\* Various constructors \*/* VPoint(**double** X, **double** Y)  
 {  
 **x** = X;  
 **y** = Y;  
 }  
  
 VPoint(VPoint other)  
 {  
 **x** = other.**x**;  
 **y** = other.**y**;  
 }  
  
 VPoint(Point other)  
 {  
 **x** = other.**x**;  
 **y** = other.**y**;  
 }  
  
 **public void** paint(Graphics g)  
 {  
 g.fillOval((**int**)(**x** - 3.0), (**int**)(**y** - 3.0), 7, 7);  
 }  
  
 */\*\*  
 \* Method to calculate the distance  
 \* between current VPoint other one  
 \** ***@param other*** *- second fortuneurn the distance (in double format) between these two points  
 \*/* **double** getDistance(VPoint other)  
 {  
 **double** s1 = other.**x** - **x**;  
 **double** s2 = other.**y** - **y**;  
 **return** Math.*sqrt*(s1 \* s1 + s2 \* s2);  
 }

**1.11. Класс Utilities.java**

*/\*  
 \* @author Nikita Marinosyan  
 \* @date 17.01.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***package** hse.utilities;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.io.\*;  
**import** java.net.URI;  
  
*/\*\*  
 \* Utility class with some handy routine  
 \*/***public class** Utilities {  
  
 */\*\*  
 \* Opens webpage in browser if it is supported  
 \** ***@param uri*** *- web address to be opened  
 \** ***@param component*** *- GUI component where the link is placed  
 \*/* **public static void** openWebpage(URI uri, JComponent component)  
 {  
 Desktop desktop = Desktop.*isDesktopSupported*() ? Desktop.*getDesktop*() : **null**;  
 **if** (desktop != **null** && desktop.isSupported(Desktop.Action.***BROWSE***))  
 {  
 **try** {  
 desktop.browse(uri);  
 }  
 **catch** (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 **else** {  
 JOptionPane.*showMessageDialog*(component, **"Browser is not supported on you system!"**,  
 **"Error"**, JOptionPane.***ERROR\_MESSAGE***);  
 }  
 }  
  
 */\*\*  
 \* Reads the entire resource text file.  
 \* \* @param resourceName Name of the resource text file  
 \* \* @return Content of the resource text file.  
 \*/* **public static** String readAllTextFromResource(String resourceName)  
 {  
 String s = **""**;;  
 InputStream inputStream =  
 Thread.*currentThread*().getContextClassLoader().getResourceAsStream(resourceName);  
 **try** {  
 ByteArrayOutputStream result = **new** ByteArrayOutputStream();  
 **byte**[] buffer = **new byte**[1024];  
 **int** length;  
 **while** ((length = inputStream.read(buffer)) != -1) {  
 result.write(buffer, 0, length);  
 }  
 s = result.toString(**"UTF-8"**);  
 } **catch** (Exception e) {  
 *// Omitted.* } **finally** {  
 **try** {  
 inputStream.close();  
 } **catch** (IOException e) {  
 *// Omitted.* }  
 }  
 **return** s;  
 }

**1.12. Класс Controls.java**

**package** hse.visualizer;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
**import** hse.fortune.VCanvas;  
  
**class** Controls **extends** Panel **implements** ActionListener  
{  
  
 **private** VCanvas **canvas**;  
 Thread **thread**;  
 **private boolean running**;  
 JButton **buttons**[];  
  
 Controls(VCanvas vCanvas)  
 {  
 **running** = **true**;  
 **canvas** = vCanvas;  
 String as[] = { **"Pause"**, **"Start/Resume"**, **"Next event"**, **"Step forward"**,  
 **"Clear"**, **"Restart"** };  
  
 **buttons** = **new** JButton[as.**length**];  
 **for**(**int** i = 0; i < as.**length**; i++)  
 {  
 **buttons**[i] = **new** JButton(as[i]);  
 **if** ((i % 2 == 0 && i != 0))  
 add(Box.*createRigidArea*(**new** Dimension(70, 0)));  
 **buttons**[i].addActionListener(**this**);  
 add(**buttons**[i]);  
 **buttons**[i].setFocusPainted(**false**);  
 }  
  
 **buttons**[1].setEnabled(**false**);  
 **buttons**[3].setEnabled(**false**);  
 }  
  
 **public void** actionPerformed(ActionEvent actionevent) {  
  
 String s = actionevent.getActionCommand();  
  
 **switch** (s)  
 {  
 **case "Pause"**:  
 threadRunning(**false**);  
 **break**;  
 **case "Start/Resume"**:  
 threadRunning(**true**);  
 **break**;  
 **case "Next event"**:  
 **canvas**.step();  
 **return**;  
 **case "Step forward"**:  
 **canvas**.singleStep();  
 **return**;  
 **case "Clear"**:  
 threadRunning(**false**);  
 **canvas**.clear();  
 **return**;  
 **case "Restart"**:  
 **canvas**.restart();  
 threadRunning(**true**);  
 **break**;  
 }  
 }  
  
 @SuppressWarnings(**"deprecation"**)  
 **void** threadRunning(**boolean** flag)  
 {  
 **if**(flag != **running**)  
 {  
 **if**(**running** = flag)  
 {  
 **buttons**[0].setEnabled(**true**);  
 **buttons**[1].setEnabled(**false**);  
 **buttons**[3].setEnabled(**false**);  
 **thread**.resume();  
 **return**;  
 }  
 **buttons**[0].setEnabled(**false**);  
 **buttons**[1].setEnabled(**true**);  
 **buttons**[3].setEnabled(**true**);  
 **thread**.suspend();  
 }  
 }  
}

**1.13. Класс Settings.java**

**package** hse.visualizer;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** hse.fortune.VCanvas;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.ItemEvent;  
**import** java.awt.event.ItemListener;  
  
**class** Settings **extends** Panel **implements** ItemListener  
{  
 **private** VCanvas **canvas**;  
  
 Settings(VCanvas vCanvas)  
 {  
 **canvas** = vCanvas;  
 String as[] = { **"Circle events"**, **"Beachline"**, **"Completed diagram"** };  
 Checkbox[] boxes = **new** Checkbox[as.**length**];  
 **for**(**int** i = 0; i < as.**length**; i++)  
 {  
 boxes[i] = **new** Checkbox(as[i]);  
 boxes[i].addItemListener(**this**);  
 add(boxes[i]);  
 }  
  
 boxes[1].setState(**true**);  
 boxes[2].setState(**true**);  
  
 ImageIcon legend1 = **new** ImageIcon(getClass().getResource(**"/legend1.png"**));  
 ImageIcon legend2 = **new** ImageIcon(getClass().getResource(**"/legend2.png"**));  
 JLabel legend1Label = **new** JLabel(**""**, legend1, JLabel.***CENTER***);  
 JLabel legend2Label = **new** JLabel(**""**, legend2, JLabel.***CENTER***);  
 add(legend1Label);  
 add(legend2Label);  
 }  
  
 **public void** itemStateChanged (ItemEvent itemEvent)  
 {  
 String s = itemEvent.getItem().toString();  
 **boolean** flag = itemEvent.getStateChange() == ItemEvent.***SELECTED***;  
 **switch** (s) {  
 **case "Circle events"**:  
 **canvas**.**drawCircles** = flag;  
 **break**;  
 **case "Beachline"**:  
 **canvas**.**drawBeach** = flag;  
 **break**;  
 **case "Completed diagram"**:  
 **canvas**.**drawVoronoiLines** = flag;  
 **break**;  
 }  
 **canvas**.repaint();  
 }  
}

**1.14. Класс VDFrame.java**

**package** hse.visualizer;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** hse.fortune.VCanvas;  
**import** hse.utilities.Utilities;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
  
  
**public class** VDFrame **extends** JFrame **implements** Runnable {  
  
 **private static final int *FRAME\_WIDTH*** = 1080;  
 **private static final int *FRAME\_HEIGHT*** = 720;  
 **private static final** String ***aboutText*** = **"Voronoi Diagram Visualizer app allows you to watch how Voronoi Diagram\n"** +  
 **"is being built step by step with Fortune algorithm to the defined set of "** + **"points\n"** + **"Designed by: Nikita Marinosyan\n"** + **"Email: nikita.marinosyan@gmail.com\n"** + **"Higher School of Economics, Faculty of Computer Science\n"** + **"2017"**;  
  
 **private** VCanvas **canvas**;  
 **private** Controls **controls**;  
 **private** Thread **thread**;  
  
 **private** VDFrame()  
 {  
 init();  
  
 start();  
 **controls**.**buttons**[5].doClick();  
 **controls**.**buttons**[0].doClick();  
 }  
  
 */\*\*  
 \* Method to init the frame  
 \*/* **private void** init()  
 {  
 *// Set default properties* onPaint();  
  
 *// Create menu* createMenu();  
  
 *// Create the canvas, controls and import settings* setLayout(**new** BorderLayout());  
  
 **canvas** = **new** VCanvas(**this**.getWidth(), **this**.getHeight() - 115, 32);  
  
 add(BorderLayout.***NORTH***, **new** Settings(**canvas**));  
 add(BorderLayout.***SOUTH***, **controls** = **new** Controls(**canvas**));  
 add(BorderLayout.***CENTER***, **canvas**);  
 }  
  
 */\*\*  
 \* Method that starts the thread  
 \*/* **private void** start()  
 {  
 **if**(**thread** == **null**)  
 {  
 **controls**.**thread** = **new** Thread(**this**);  
 **thread** = **controls**.**thread**;  
 **thread**.start();  
 }  
 }  
  
 @SuppressWarnings(**"InfiniteLoopStatement"**)  
 **public void** run()  
 {  
 **if**(**thread** != **null**)  
 {  
 **while**(**true**)  
 {  
 **canvas**.init();  
 **while**(**canvas**.singleStep())  
 {  
 **try** {  
 Thread.*sleep*(25L);  
 }  
 **catch**(InterruptedException \_ex)  
 {  
 System.***out***.print(**"Program has been interrupted"**);  
 }  
 }  
 **controls**.threadRunning(**false**);  
 }  
 }  
 }  
  
 */\*\*  
 \* Method to set the default values to the frame's properties  
 \*/* **private void** onPaint()  
 {  
  
 setTitle(**"Voronoi Diagram Visualizer (Fortune Algorithm)"**);  
 setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 setSize(**new** Dimension(***FRAME\_WIDTH***, ***FRAME\_HEIGHT***));  
 setResizable(**false**);  
 setLayout(**null**);  
 setLocationRelativeTo(**null**);  
 setIconImage(**new** ImageIcon(getClass().getResource(**"/icon.png"**)).getImage());  
 }  
  
 */\*\*  
 \* Method which creates the menu bar  
 \*/* **private void** createMenu()  
 {  
  
 JMenuBar menuBar = **new** JMenuBar();  
 setJMenuBar(menuBar);  
  
 */\*--------------------------------------------- Create separator ---------------------------------------------\*/* JSeparator separator = **new** JSeparator(SwingConstants.***VERTICAL***);  
 separator.setMaximumSize(**new** Dimension(10, 50));  
  
 */\*------------------------------------------- Create help menu -----------------------------------------------\*/* JMenu helpMenu = **new** JMenu(**"Help"**);  
 menuBar.add(helpMenu);  
  
 JMenuItem manual = **new** JMenuItem(**"Manual"**);  
 JFrame manualWindow = **new** JFrame(**"Manual"**);  
 JTextPane manualText = **new** JTextPane();  
 manualText.setEditable(**false**);  
 manualText.setContentType(**"text/html"**);  
 manualText.setText(Utilities.*readAllTextFromResource*(**"manual.html"**));  
 manualText.setCaretPosition(0);  
 manualWindow.setDefaultCloseOperation(WindowConstants.***DISPOSE\_ON\_CLOSE***);  
 manualWindow.setSize(700, 700);  
 manualWindow.setResizable(**false**);  
 JScrollPane jsp = **new** JScrollPane(manualText);  
  
 manualWindow.add(jsp);  
 manualWindow.setLocation(**this**.getX() + 380, **this**.getY());  
 manual.addActionListener(e -> openManual(manualWindow));  
 helpMenu.add(manual);  
  
 JMenuItem about = **new** JMenuItem(**"About the program"**);  
 helpMenu.add(about).addActionListener(e -> openAbout());  
  
 menuBar.add(separator);  
 }  
  
 */\*\*  
 \* Method to handle manual menu button pressing  
 \* Opens new window with complete manual on the app  
 \*/* **private void** openManual(JFrame manualWindow)  
 {  
 **controls**.**buttons**[0].doClick();  
 manualWindow.setVisible(**true**);  
 }  
  
 */\*\*  
 \* Method to handle aboutMenu button pressing  
 \* Opens new window with brief info about the app  
 \*/* **private void** openAbout()  
 {  
 *// Pause visualization* **controls**.**buttons**[0].doClick();  
  
 *// Show about frame* JOptionPane.*showMessageDialog*(**this**, ***aboutText***);  
 }  
  
 **public static void** main(String[] args)  
 {  
 VDFrame vdFrame = **new** VDFrame();  
 vdFrame.setVisible(**true**);  
 }  
}

**1.15. Интерфейс VPaintable.java**

**package** hse.fortune;  
  
*/\*  
 \* @author Nikita Marinosyan  
 \* @date 20.04.2017  
 \* Higher School of Economics,  
 \* Faculty of Computer Science,  
 \* Department of Software Engineering  
 \*/***import** java.awt.Graphics;  
  
**interface** VPaintable  
{  
 **void** paint(Graphics g);  
}