**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Профессор  департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. М. Баканов «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |
| --- | --- |
| ***Подп. и дата*** |  |
| ***Инв. № дубл.*** |  |
| ***Взам. инв. №*** |  |
| ***Подп. и дата*** |  |
| ***Инв. № подл*** | RU.17701729.04.01-01 12 01-1-ЛУ |

**ПРОГРАММА МОДЕЛИРОВАНИЯ И ВЫБОРА ОПТИМАЛЬНЫХ ПАРАМАТРОВ ДВИЖЕНИЯ ПЛОСКИХ МЕХАНИЗМОВ**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 12 01-1-ЛУ**

Исполнитель  
студент группы БПИ172  
\_\_\_\_\_\_\_\_\_\_ / Пучков А. М. /  
«\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Москва 2019**

УТВЕРЖДЕНRU.17701729.04.01-01 12 01-1-ЛУ

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл | RU.17701729.04.01-01 12 01-1 |

**ПРОГРАММА МОДЕЛИРОВАНИЯ И ВЫБОРА ОПТИМАЛЬНЫХ ПАРАМАТРОВ ДВИЖЕНИЯ ПЛОСКИХ МЕХАНИЗМОВ**

**Текст программы**

**RU.17701729.04.01-01 12 01-1**

**Листов 65**

**Москва 2019**

Содержание

[1. Текст программы 3](#_Toc482207421)

[1.1. Класс LearningActivity.java 3](#_Toc482207422)

[1.2. Класс MainActivity.java 9](#_Toc482207423)

[1.3. Класс MainHomeFragment.java 13](#_Toc482207424)

[1.4. Класс MainRatingFragment.java 16](#_Toc482207425)

[1.5. Класс GameActivity.java 19](#_Toc482207426)

[1.6. Класс Storage.java 26](#_Toc482207427)

[1.7. Класс Game.java 28](#_Toc482207428)

[1.8. Класс Tour.java 31](#_Toc482207429)

[1.9. Класс Round.java 32](#_Toc482207430)

[1.10. Класс NeuralNetworkConcurrentLearning.java 34](#_Toc482207431)

[1.10.1. В приложении – ru.hse.x21.logic.neuro. NeuralNetworkConcurrentLearning 34](#_Toc482207432)

[1.10.2. В экспериментальном модуле – x2plus1game.neuralexps. NeuralNetworkConcurrentLearning 37](#_Toc482207433)

[1.11. Класс NeuralNetworkPlaying.java 40](#_Toc482207434)

[1.12. Класс NeuralNetwork.java 47](#_Toc482207435)

[1.13. Класс BackPropagationLearning.java 55](#_Toc482207436)

[1.14. Класс Sigmoid.java 62](#_Toc482207437)

[1.15. Класс Main.java экспериментального модуля (x2plus1game.neuralexps.Main) 63](#_Toc482207438)

# 1. Текст программы

Программа написана на языке Java (использована среда IntelliJ IDEA). Программа состоит из 14 классов (полностью автоматически сгенерированные классы не учтены). В данном документе содержится только вручную написанный исходный код программы, код из полностью автоматически сгенерированных исходных файлов в данном документе не представлен.

## 1.1. Класс DrawPanel.java

|  |
| --- |
| package Graphics; |
|  |  |
|  | import Mechanics.\*; |
|  |  |
|  | import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.awt.geom.Ellipse2D; |
|  | import java.awt.geom.Line2D; |
|  | import java.awt.geom.Point2D; |
|  | import java.util.ConcurrentModificationException; |
|  | import java.util.LinkedList; |
|  |  |
|  | public class DrawPanel extends JPanel { |
|  | boolean drawPath; |
|  | Mechanism mech; |
|  | LinkedList<Point2D> positions; |
|  | DrawPanel(Mechanism m) |
|  | { |
|  | setBackground(Color.lightGray); |
|  | updateMech(m); |
|  | } |
|  |  |
|  | void updateMech(Mechanism m){ |
|  | this.mech = m; |
|  | positions = m.positions; |
|  | } |
|  |  |
|  | @Override |
|  | synchronized public void paint(Graphics g) { |
|  | super.paint(g); |
|  | Graphics2D g2 = (Graphics2D) g; |
|  |  |
|  | if (drawPath) |
|  | drawPath(g2); |
|  |  |
|  | drawMech(g2, mech); |
|  |  |
|  | } |
|  |  |
|  | private void drawMech(Graphics2D g2, Mechanism m){ |
|  | for (MechPart edge : m.contents) { |
|  | if (edge instanceof Edge){ |
|  | Line2D line = new Line2D.Double(edge.getStartX(), edge.getStartY(), |
|  | edge.getEndX(), edge.getEndY()); |
|  | g2.setColor(Color.BLACK); |
|  | g2.setStroke(new BasicStroke(4)); |
|  |  |
|  | g2.draw(line); |
|  | } |
|  |  |
|  | else if (edge instanceof TriEdge){ |
|  | Line2D line1 = new Line2D.Double(edge.getStartX(), edge.getStartY(), |
|  | ((TriEdge)edge).getCenterX(), ((TriEdge)edge).getCenterY()); |
|  | Line2D line2 = new Line2D.Double(((TriEdge)edge).getCenterX(), |
|  | ((TriEdge)edge).getCenterY(), edge.getEndX(), edge.getEndY()); |
|  | Line2D line3 = new Line2D.Double(edge.getStartX(), edge.getStartY(), |
|  | edge.getEndX(), edge.getEndY()); |
|  |  |
|  | g2.setColor(Color.BLACK); |
|  | g2.setStroke(new BasicStroke(4)); |
|  |  |
|  | g2.draw(line1); |
|  | g2.draw(line2); |
|  | g2.draw(line3); |
|  | } |
|  | } |
|  |  |
|  | for (MyPoint p : m.joints) { |
|  | Ellipse2D point = new Ellipse2D.Double(p.getCenterX() - 5, |
|  | p.getCenterY() - 5, 10, 10); |
|  |  |
|  | if (p.isConstant()) |
|  | g2.setColor(Color.RED); |
|  | else |
|  | g2.setColor(Color.BLUE); |
|  |  |
|  | g2.draw(point); |
|  | g2.fill(point); |
|  | } |
|  | } |
|  |  |
|  | private synchronized void drawPath(Graphics2D g2) { |
|  | g2.setColor(Color.green); |
|  | try { |
|  | for (Point2D p : positions) { |
|  | g2.draw(new Ellipse2D.Double(p.getX() - 2.5, p.getY() - 2.5, 5, 5)); |
|  | g2.fill(new Ellipse2D.Double(p.getX() - 2.5, p.getY() - 2.5, 5, 5)); |
|  | } |
|  | } catch (ConcurrentModificationException cme) {} |
|  | } |
|  | } |

## 1.2. Класс MechControls.java

|  |
| --- |
| package Graphics; |
|  |  |
|  | import Mechanics.Mechanism; |
|  |  |
|  | import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.awt.event.ActionEvent; |
|  | import java.awt.geom.Point2D; |
|  | import java.util.LinkedList; |
|  |  |
|  | class MechControls extends JPanel { |
|  |  |
|  | private Mechanism mech; |
|  | private DrawPanel drawPanel; |
|  |  |
|  | private boolean rotateOn = false; |
|  |  |
|  | private JLabel sliderLabel; |
|  | private JSlider slider; |
|  | private JLabel rightLabel; |
|  | private JButton rotBut; |
|  | private JButton showPath; |
|  | private JButton deletePath; |
|  | private JButton startRotating; |
|  | private JButton stopRotating; |
|  |  |
|  | MechControls(Mechanism m, DrawPanel dp){ |
|  | mech = m; |
|  | drawPanel = dp; |
|  | Font font = new Font("Default", Font.PLAIN, 14); |
|  |  |
|  | sliderLabel = new JLabel("angle of single rotation: -2"); |
|  | sliderLabel.setFont(font); |
|  | add(sliderLabel); |
|  |  |
|  | slider = new JSlider(JSlider.HORIZONTAL, -2,10,2); |
|  | add(slider); |
|  |  |
|  | rightLabel = new JLabel("10"); |
|  | rightLabel.setFont(font); |
|  | add(rightLabel); |
|  |  |
|  | rotBut = new JButton("rotate once"); |
|  | rotBut.setFont(font); |
|  | add(rotBut); |
|  | rotBut.addActionListener((ActionEvent event) -> { |
|  | dp.mech.moveMech(slider.getValue()); |
|  |  |
|  | addPoint(4, dp.positions); |
|  | dp.repaint(); |
|  | }); |
|  |  |
|  | showPath = new JButton("show point's path"); |
|  | showPath.setFont(font); |
|  | add(showPath); |
|  | showPath.addActionListener(event -> { |
|  | dp.drawPath = true; |
|  | dp.repaint(); |
|  | }); |
|  |  |
|  | deletePath = new JButton("hide path"); |
|  | deletePath.setFont(font); |
|  | add(deletePath); |
|  | deletePath.addActionListener(event -> { |
|  | dp.drawPath = false; |
|  | dp.repaint(); |
|  | }); |
|  |  |
|  | Thread rotater = new Thread(() -> { |
|  | while (true){ |
|  | if (rotateOn) { |
|  | dp.mech.moveMech(slider.getValue()); |
|  |  |
|  | addPoint(4, dp.positions); |
|  | dp.repaint(); |
|  | } |
|  | try { |
|  | Thread.sleep(50); |
|  | } catch (InterruptedException e) { |
|  | notify(); |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  | }); |
|  | rotater.setDaemon(true); |
|  |  |
|  | startRotating = new JButton("start rotation"); |
|  | startRotating.setFont(font); |
|  | add(startRotating); |
|  | startRotating.addActionListener(event ->{ |
|  | rotateOn = true; |
|  | if (!rotater.isAlive()) |
|  | rotater.start(); |
|  | }); |
|  |  |
|  | stopRotating = new JButton("stop rotation"); |
|  | stopRotating.setFont(font); |
|  | add(stopRotating); |
|  | stopRotating.addActionListener(event -> rotateOn = false); |
|  | } |
|  |  |
|  | synchronized void addPoint(int pointNum, LinkedList<Point2D> positions){ |
|  | int index = positions.size()%360; |
|  | if (positions.size() >= 360) |
|  | positions.set(index, new Point2D.Double( |
|  | drawPanel.mech.joints[pointNum].getCenterX(), drawPanel.mech.joints[pointNum].getCenterY())); |
|  | else |
|  | positions.add(index, new Point2D.Double( |
|  | drawPanel.mech.joints[pointNum].getCenterX(), drawPanel.mech.joints[pointNum].getCenterY())); |
|  | } |
|  | } |

## 1.3. Класс OptimisationControls.java

|  |
| --- |
| package Graphics; |
|  |  |
|  | import Helpers.WriterReader; |
|  | import Mechanics.Mechanism; |
|  | import Optimization.LegionCreator; |
|  | import Optimization.OutsourceAttractor; |
|  |  |
|  | import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.io.\*; |
|  | import java.nio.file.Paths; |
|  |  |
|  | class OptimisationControls extends JPanel { |
|  | private DrawPanel drawPanel; |
|  | private JLabel sliderLabel; |
|  | private JSlider slider; |
|  | private JLabel rightLabel; |
|  | private JButton createTrainSamples; |
|  | private JTextField num; |
|  | private JButton loadMech; |
|  | private JButton train; |
|  | OptimisationControls(DrawPanel dp){ |
|  | drawPanel = dp; |
|  | setBackground(Color.gray); |
|  |  |
|  | Font font = new Font("Default", Font.PLAIN, 14); |
|  |  |
|  |  |
|  | sliderLabel = new JLabel("number of samples in training set: 100"); |
|  | sliderLabel.setFont(font); |
|  | add(sliderLabel); |
|  |  |
|  | slider = new JSlider(JSlider.HORIZONTAL, 100,500,200); |
|  | slider.setPreferredSize(new Dimension(150, 25)); |
|  | add(slider); |
|  |  |
|  | rightLabel = new JLabel("500"); |
|  | rightLabel.setFont(font); |
|  | add(rightLabel); |
|  |  |
|  | createTrainSamples = new JButton("create training samples"); |
|  | createTrainSamples.setFont(font); |
|  | add(createTrainSamples); |
|  | createTrainSamples.addActionListener(event -> { |
|  | String lastPath = Paths.get(".").toAbsolutePath().normalize().toString(); |
|  | lastPath = lastPath.substring(lastPath.length()-11); |
|  | String path = ""; |
|  | if (lastPath.equals("Project\_jar")) |
|  | path += "..\\..\\..\\"; |
|  | path += "data\\mechs.csv"; |
|  | if (new File(path).delete()) |
|  | System.out.println("Deleted mechs.csv, writing new one"); |
|  | new LegionCreator(path, slider.getValue()); |
|  | }); |
|  |  |
|  | num = new JTextField(); |
|  | num.setPreferredSize(new Dimension(30, 25)); |
|  | num.setFont(font); |
|  | add(num); |
|  |  |
|  | loadMech = new JButton("load created mech by №"); |
|  | loadMech.setFont(font); |
|  | add(loadMech); |
|  |  |
|  | loadMech.addActionListener(event -> loadMech()); |
|  |  |
|  | train = new JButton("find optimal mechanism"); |
|  | train.setFont(font); |
|  | add(train); |
|  | train.addActionListener( event ->{ |
|  | OutsourceAttractor linReg = new OutsourceAttractor(); |
|  | linReg.runPython(); |
|  | }); |
|  | } |
|  |  |
|  | private void loadMech(){ |
|  | if (num.getText().equals("")) { |
|  | System.out.println("ERROR: no number entered"); |
|  | return; |
|  | } |
|  | int mechNo = -1; |
|  | try { |
|  | mechNo = Integer.parseInt(num.getText()); |
|  | } catch (NumberFormatException nfe){ |
|  | System.out.println("ERROR: no parsable number found"); |
|  | return; |
|  | } |
|  | if (mechNo < 0){ |
|  | System.out.println("ERROR: number is negative"); |
|  | return; |
|  | } |
|  |  |
|  | String lastPath = Paths.get(".").toAbsolutePath().normalize().toString(); |
|  | lastPath = lastPath.substring(lastPath.length()-11); |
|  | String path = ""; |
|  | if (lastPath.equals("Project\_jar")) |
|  | path += "..\\..\\..\\"; |
|  |  |
|  | WriterReader mechReader = new WriterReader(path+"data\\mechs.csv"); |
|  | mechReader.readSingle(); |
|  |  |
|  | if (mechNo >= mechReader.features.length){ |
|  | System.out.println("ERROR: no mech with that number found"); |
|  | return; |
|  | } |
|  | double[] m = new double[4]; |
|  | System.arraycopy(mechReader.features[mechNo], 0, m, 0, 4); |
|  | Mechanism mech = Mechanism.createMech(m); |
|  |  |
|  | drawPanel.updateMech(mech); |
|  | drawPanel.repaint(); |
|  | } |
|  | } |

## 1.4. Класс SwingFrame.java

|  |
| --- |
| package Graphics; |
|  |  |
|  | import Mechanics.\*; |
|  |  |
|  | import javax.swing.\*; |
|  | import java.awt.\*; |
|  | import java.nio.file.Paths; |
|  | import java.util.LinkedList; |
|  |  |
|  | public class SwingFrame extends JFrame { |
|  | private DrawPanel canvas; |
|  | private MechControls gui; |
|  | private OptimisationControls opt; |
|  | private Mechanism mech; |
|  |  |
|  | private Mechanism createMech(){ |
|  | MyPoint A = new MyPoint(400,420, true); |
|  | MyPoint B = new MyPoint(600,270); |
|  | MyPoint C = new MyPoint(700, 420, true); |
|  | MyPoint D = new MyPoint(800, 420); |
|  | MyPoint E = new MyPoint(400, 120); |
|  |  |
|  |  |
|  | MyPoint[] points = new MyPoint[]{A, B, C, D, E}; |
|  |  |
|  | LinkedList<int[]> edges = new LinkedList<>(); |
|  | edges.add(new int[]{2, 3}); //rotater |
|  | edges.add(new int[]{3, 1, 4}); //longest |
|  |  |
|  | edges.add(new int[]{0, 1}); //short left |
|  |  |
|  | return new Mechanism(edges, points); |
|  | } |
|  |  |
|  | public static void main(String[] args){ |
|  | new SwingFrame(); |
|  | } |
|  |  |
|  | private SwingFrame(){ |
|  | setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE); |
|  |  |
|  | String lastPath = Paths.get(".").toAbsolutePath().normalize().toString(); |
|  | lastPath = lastPath.substring(lastPath.length()-11); |
|  | String path = ""; |
|  | if (lastPath.equals("Project\_jar")) |
|  | path += "..\\..\\..\\"; |
|  |  |
|  | setIconImage(new ImageIcon(path + "icon.png").getImage()); |
|  |  |
|  | Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize(); |
|  | int x = (int)(dimension.getWidth() / 3); |
|  | int y = (int)(dimension.getHeight() / 3); |
|  | setLocation(x/2, y/2); |
|  | setSize(x\*2,y\*2); |
|  |  |
|  | mech = createMech(); |
|  |  |
|  | canvas = new DrawPanel(mech); |
|  | gui = new MechControls(mech, canvas); |
|  | opt = new OptimisationControls(canvas); |
|  |  |
|  | gui.setBackground(Color.gray); |
|  | getContentPane().setLayout(new BorderLayout()); |
|  | getContentPane().add(canvas, BorderLayout.CENTER); |
|  | getContentPane().add(gui, BorderLayout.NORTH); |
|  | getContentPane().add(opt, BorderLayout.SOUTH); |
|  |  |
|  | setVisible(true); |
|  | } |
|  | } |

## 1.5. Класс DoublePair.java

|  |
| --- |
| Package Helpers; |
|  |  |
|  | public class DoublePair{ |
|  | public double first; |
|  | public double second; |
|  |  |
|  | public DoublePair(double a, double b){ |
|  | first = a; |
|  | second = b; |
|  | } |
|  |  |
|  | public DoublePair add(DoublePair other){ |
|  | return new DoublePair(this.first + other.first, this.second + other.second); |
|  | } |
|  |  |
|  | @Override |
|  | public String toString(){ |
|  | return this.first + "; " + this.second; |
|  | } |
|  | } |

## 1.6. Класс Utils.java

|  |
| --- |
|  |
| package Helpers; |
|  |  |
|  | import Mechanics.MechPart; |
|  | import Mechanics.MyPoint; |
|  |  |
|  | public class Utils { |
|  | public static double distance(DoublePair a, DoublePair b) { |
|  | return Math.sqrt( |
|  | (a.first-b.first) |
|  | \*(a.first-b.first) + |
|  | (a.second-b.second) |
|  | \*(a.second-b.second)); |
|  | } |
|  |  |
|  | private static double distance(MyPoint a, DoublePair b){ |
|  | return Math.sqrt( |
|  | (a.getCenterX()-b.first)\* |
|  | (a.getCenterX()-b.first) + |
|  | (a.getCenterY()-b.second)\* |
|  | (a.getCenterY()-b.second)); |
|  | } |
|  |  |
|  | public static double distance(MyPoint a, MyPoint b){ |
|  | return Math.sqrt( |
|  | (a.getCenterX()-b.getCenterX()) \* |
|  | (a.getCenterX()-b.getCenterX()) + |
|  | (a.getCenterY()-b.getCenterY()) \* |
|  | (a.getCenterY()-b.getCenterY())); |
|  | } |
|  |  |
|  |  |
|  | public static DoublePair Pif2Pol(DoublePair pair){ |
|  | double x = pair.first; |
|  | double y = pair.second; |
|  |  |
|  | double ro = Math.sqrt(x\*x + y\*y); |
|  | double fi = Math.atan2(y, x); |
|  |  |
|  | return new DoublePair(ro, fi); |
|  | } |
|  |  |
|  | public static DoublePair Pol2Pif(DoublePair pair){ |
|  | double ro = pair.first; |
|  | double fi = pair.second; |
|  |  |
|  | double x = ro \* Math.cos(fi); |
|  | double y = ro \* Math.sin(fi); |
|  |  |
|  | return new DoublePair(x, y); |
|  | } |
|  |  |
|  | public static double asDegrees(double rad){ |
|  | return rad\*180/Math.PI; |
|  | } |
|  |  |
|  | public static double asRadians(double degrees){ |
|  | return degrees\*Math.PI/180; |
|  | } |
|  |  |
|  | public static double distanceToLine(double X, double Y, double k, double b){ |
|  | return (X\*k + b - Y)/Math.sqrt(k\*k+1); |
|  | } |
|  |  |
|  | public static boolean distanceEquals (MyPoint point, DoublePair pair, double distance){ |
|  | return Math.abs(Utils.distance(point, pair) - distance) < 5; |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |

## 1.7. Класс WriterReader.java

|  |
| --- |
| package Helpers; |
|  |  |
|  | import java.io.\*; |
|  | import java.util.ArrayList; |
|  |  |
|  | public class WriterReader { |
|  | private String path; |
|  | public double[][] features; |
|  |  |
|  | public WriterReader(String path) { |
|  | this.path = path; |
|  | } |
|  |  |
|  | public void readSingle(){ |
|  | String row; |
|  | ArrayList<double[]> feats = new ArrayList<>(); |
|  | try { |
|  | BufferedReader csvReader = new BufferedReader(new FileReader(path)); |
|  | while ((row = csvReader.readLine()) != null) { |
|  | String[] data = row.split(","); |
|  | double[] pData = new double[data.length]; |
|  | for (int i = 0; i < data.length; i++) |
|  | pData[i] = Double.parseDouble(data[i]); |
|  | feats.add(pData); |
|  | } |
|  |  |
|  | features = new double[feats.size()][]; |
|  | for (int i = 0; i < feats.size(); i++) |
|  | features[i] = new double[feats.get(i).length]; |
|  |  |
|  |  |
|  | for (int i = 0; i < feats.size(); i++) |
|  | System.arraycopy(feats.get(i), 0, |
|  | features[i], 0, feats.get(i).length); |
|  |  |
|  |  |
|  | csvReader.close(); |
|  | } catch (FileNotFoundException fnfe) { |
|  | fnfe.printStackTrace(); |
|  | } catch (IOException ioe) { |
|  | ioe.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | public void write(double[] numbers, double dev){ |
|  | try{ |
|  | FileWriter fw = new FileWriter(path, true); |
|  | for (double num : numbers) |
|  | fw.write((int)num + ", "); |
|  | fw.write((int)dev + "\n"); |
|  | fw.close(); |
|  |  |
|  | } catch (Exception e) {e.printStackTrace();} |
|  |  |
|  | // System.out.println("Written mech successfully"); |
|  | } |
|  |  |
|  | public void write(double[] numbers){ |
|  | try{ |
|  | FileWriter fw = new FileWriter(path, true); |
|  | for (double num : numbers) |
|  | fw.write((num == numbers[0]?"":", ") + (int)num ); |
|  | fw.write("\n"); |
|  | fw.close(); |
|  |  |
|  | } catch (Exception e) {e.printStackTrace();} |
|  | } |
|  | } |

## 1.8. Класс Edge.java

|  |
| --- |
| package Mechanics; |
|  |  |
|  | import Helpers.DoublePair; |
|  | import Helpers.Utils; |
|  |  |
|  | public class Edge implements MechPart{ |
|  | private MyPoint beginning; |
|  | private MyPoint end; |
|  | private DoublePair[] possiblePositions = new DoublePair[360]; |
|  | private double length = 0d; |
|  |  |
|  | public double Length(){ |
|  | length = Math.sqrt((getEndX()-getStartX())\*(getEndX()-getStartX()) + (getEndY()-getStartY())\*(getEndY()-getStartY())); |
|  | return length; |
|  | } |
|  |  |
|  | @Override |
|  | public double Angle(){ |
|  | return Math.atan2(getEndY()-getStartY(), getEndX()-getStartX()); |
|  | } |
|  |  |
|  | public void setLength(double length){ |
|  | double rads = Angle(); |
|  | if (getEnd().isConstant()){ |
|  | setStartX(getEndX() - Math.cos(rads)\*length); |
|  | setStartY(getEndY() - Math.sin(rads)\*length); |
|  | } |
|  | else |
|  | setEndX(getStartX() + Math.cos(rads)\*length); |
|  | setEndY(getStartY() + Math.sin(rads)\*length); |
|  | } |
|  |  |
|  | public String toString(){ |
|  | String res = "(" + getStartX() + "; " + getStartY() + ") to (" + getEndX() + "; " + getEndY() + "); length:" + Length(); |
|  | return res; |
|  | } |
|  |  |
|  | public MyPoint getBeginning() { return beginning; } |
|  |  |
|  | public MyPoint getEnd() { return end; } |
|  |  |
|  | public double getStartX() { return beginning.getCenterX(); } |
|  | private void setStartX(double value) { beginning.setCenterX(value); } |
|  |  |
|  | public double getStartY() { return beginning.getCenterY(); } |
|  | private void setStartY(double value) { beginning.setCenterY(value); } |
|  |  |
|  | public double getEndX() { return end.getCenterX(); } |
|  | private void setEndX(double value) { end.setCenterX(value); } |
|  |  |
|  | public double getEndY() { return end.getCenterY(); } |
|  | private void setEndY(double value) { end.setCenterY(value); } |
|  |  |
|  | public void setStart(double x, double y){ |
|  | setStartX(x); |
|  | setStartY(y); |
|  | beginning.move(x, y); |
|  | recalculatePositions(); |
|  | } |
|  |  |
|  | public void setFinish(double x, double y){ |
|  | setEndX(x); |
|  | setEndY(y); |
|  | end.move(x, y); |
|  | } |
|  |  |
|  | public void setFinish(int k){ |
|  | setEndX(possiblePositions[k].first); |
|  | setEndY(possiblePositions[k].second); |
|  | } |
|  |  |
|  | public DoublePair getStart(){ |
|  | return new DoublePair(getStartX(), getStartY()); |
|  | } |
|  |  |
|  | public DoublePair getFinish() { |
|  | return new DoublePair(getEndX(), getEndY()); |
|  | } |
|  |  |
|  | //thumb |
|  | public DoublePair getCenter(){return getFinish();} |
|  |  |
|  | Edge(MyPoint start, MyPoint finish){ |
|  | beginning = start; |
|  | end = finish; |
|  | recalculatePositions(); |
|  | } |
|  |  |
|  | //get all possible positions of the edge |
|  | public void recalculatePositions(){ |
|  | double radius = Length(); |
|  | DoublePair startPosition = getStart(); |
|  | DoublePair finishPosition = new DoublePair(radius, Utils.asRadians(180)); //finish is lower |
|  |  |
|  | for (int i = 0; i < 360; i++){ |
|  | finishPosition.second += Utils.asRadians(0.5d); //rotate by 1/2 degree |
|  | possiblePositions[i] = startPosition.add(Utils.Pol2Pif(finishPosition)); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public int getPointWithDistance(MechPart source, boolean finish, double distance) { |
|  | int index = 0; |
|  |  |
|  | if(!finish) |
|  | distance /= 2; |
|  |  |
|  | double minDist = Math.abs(Utils.distance(possiblePositions[0], source.getFinish()) - distance); |
|  | for (int i = 1; i < 360; i++) |
|  | if (minDist > Math.abs(Utils.distance(possiblePositions[i], source.getFinish()) - distance)){ |
|  | index = i; |
|  | minDist = Math.abs(Utils.distance(possiblePositions[i], source.getFinish()) - distance); |
|  | } |
|  | return index; |
|  | } |
|  |  |
|  |  |
|  | public void move(double x1, double y1, double x2, double y2){ |
|  | setStart(x1, y1); |
|  | setFinish(x2, y2); |
|  | } |
|  |  |
|  | public void move(double dx, double dy){ |
|  | setStart(getStartX() + dx, getStartY() + dy); |
|  | setFinish(getEndX() + dx, getEndY() + dy); |
|  | } |
|  |  |
|  | public void rotate(double angle){ |
|  | double length = Length(); |
|  | DoublePair pif = new DoublePair(getEndX() - getStartX(), |
|  | getEndY() - getStartY()); |
|  | DoublePair pol = Utils.Pif2Pol(pif); |
|  |  |
|  | pol.second += Utils.asRadians(angle); |
|  |  |
|  | DoublePair newPif = Utils.Pol2Pif(pol); |
|  |  |
|  | setFinish(getStartX() + newPif.first, |
|  | getStartY() + newPif.second); |
|  |  |
|  | setLength(length); |
|  | } |
|  | } |

## 1.9. Класс Mechpart.java

|  |
| --- |
| package Mechanics; |
|  |  |
|  | import Helpers.DoublePair; |
|  |  |
|  | public interface MechPart { |
|  | double getStartX(); |
|  | double getStartY(); |
|  | double getEndX(); |
|  | double getEndY(); |
|  |  |
|  | MyPoint getBeginning(); |
|  | MyPoint getEnd(); |
|  |  |
|  | DoublePair getStart(); |
|  | DoublePair getFinish(); |
|  | DoublePair getCenter(); |
|  |  |
|  | void setStart(double x, double y); |
|  | void setFinish(double x, double y); |
|  | void setFinish(int k); |
|  |  |
|  | void move(double dx, double dy); |
|  | void move(double x1, double y1, double x2, double y2); |
|  |  |
|  | void rotate(double angle); |
|  | double Length(); |
|  |  |
|  | double Angle(); |
|  |  |
|  | void setLength(double length); |
|  |  |
|  | void recalculatePositions(); |
|  |  |
|  | int getPointWithDistance(MechPart source, boolean finish, double dist); |
|  | } |

## 1.10. Класс Mechanism.java

|  |
| --- |
| package Mechanics; |
|  |  |
|  | import Helpers.DoublePair; |
|  | import Helpers.Utils; |
|  |  |
|  | import java.awt.geom.Point2D; |
|  | import java.util.Comparator; |
|  | import java.util.LinkedList; |
|  |  |
|  | public class Mechanism { |
|  | public MechPart[] contents; |
|  | public MyPoint[] joints; |
|  | public LinkedList<Point2D> positions = new LinkedList<>(); |
|  |  |
|  | public Mechanism(LinkedList<int[]> edges, MyPoint[] points) { |
|  | joints = points; |
|  | contents = new MechPart[edges.size()]; |
|  | for (int i = 0; i < edges.size(); i++){ |
|  | if(edges.get(i).length == 2) |
|  | contents[i] = new Edge( |
|  | points[edges.get(i)[0]], |
|  | points[edges.get(i)[1]]); |
|  |  |
|  | else |
|  | contents[i] = new TriEdge( |
|  | points[edges.get(i)[0]], |
|  | points[edges.get(i)[1]], |
|  | points[edges.get(i)[2]]); |
|  | } |
|  | } |
|  |  |
|  | public static Mechanism createMech(double[] distances) { |
|  | MyPoint[] joints = new MyPoint[5]; |
|  | MyPoint A = new MyPoint(); |
|  | MyPoint B = new MyPoint(); |
|  | MyPoint C = new MyPoint(); |
|  | MyPoint D = new MyPoint(); |
|  | MyPoint E = new MyPoint(); |
|  |  |
|  | C = new MyPoint(700, 400, true); |
|  | D = new MyPoint(700 + distances[0], 400); |
|  | A = new MyPoint(700 - distances[3], 400, true); |
|  |  |
|  |  |
|  | boolean ok = false; |
|  | for (int i = 0; i < 1800; i++){ |
|  | if (ok) break; |
|  | for (int j = 0; j < 700; j++) { |
|  | DoublePair d = new DoublePair(i, j); |
|  | if (Utils.distanceEquals(A, d, distances[2]) && |
|  | Utils.distanceEquals(D, d, distances[1] / 2)) { |
|  | B = new MyPoint(d.first, d.second); |
|  | ok = true; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | if (!ok) { |
|  | return null; |
|  | } |
|  |  |
|  | E = new MyPoint(B.getCenterX()\*2 - D.getCenterX(), |
|  | B.getCenterY()\*2 - D.getCenterY()); |
|  |  |
|  | joints[0] = A; |
|  | joints[1] = B; |
|  | joints[2] = C; |
|  | joints[3] = D; |
|  | joints[4] = E; |
|  |  |
|  |  |
|  | LinkedList<int[]> edges = new LinkedList<>(); |
|  | edges.add(new int[]{2, 3}); //rotater |
|  | edges.add(new int[]{3, 1, 4}); //longest |
|  |  |
|  | edges.add(new int[]{0, 1}); //short left |
|  |  |
|  | return new Mechanism(edges, joints); |
|  | } |
|  |  |
|  | public void moveMech(double angle){ |
|  | double length1 = contents[1].Length(); |
|  | double length2 = contents[2].Length(); |
|  | contents[0].rotate(angle); |
|  | int newPosition = contents[2].getPointWithDistance(contents[0], false, length1); |
|  |  |
|  | contents[2].setFinish(newPosition); |
|  | contents[2].setLength(length2); |
|  |  |
|  | ((TriEdge)contents[1]).inlineFinish(); |
|  | contents[1].setLength(length1); |
|  | } |
|  |  |
|  | private double[] toArray(){ |
|  | double[] result = new double[4]; |
|  | result[0] = contents[0].Length(); |
|  | result[1] = contents[1].Length(); |
|  | result[2] = contents[2].Length(); |
|  | result[3] = Utils.distance(joints[0], joints[2]); |
|  | return result; |
|  | } |
|  |  |
|  | public boolean checkFunctionality(){ |
|  | double[] arr = toArray(); |
|  | return (arr[3] < arr[1]/2 + arr[2] - arr[0]) && (arr[0] < arr[3]) && (arr[0]\*2 < arr[1]); |
|  | } |
|  |  |
|  | public void initPositions(){ |
|  | for (int i = 0; i < 360; i++){ |
|  | this.moveMech(1); |
|  |  |
|  | //addPoint |
|  | int index = positions.size()%360; |
|  | if (positions.size() >= 360) |
|  | positions.set(index, new Point2D.Double( |
|  | this.joints[4].getCenterX(), this.joints[4].getCenterY())); |
|  | else |
|  | positions.add(index, new Point2D.Double( |
|  | this.joints[4].getCenterX(), this.joints[4].getCenterY())); |
|  | } |
|  | } |
|  | } |

## 1.11. Класс MyPoint.java

Начало формы

Конец формы

Начало формы

Конец формы

|  |  |
| --- | --- |
|  | package Mechanics; |
|  |  |
|  | public class MyPoint { |
|  | private boolean constant; |
|  | private double centerX; |
|  | private double centerY; |
|  |  |
|  | public double getCenterX() { return centerX; } |
|  | void setCenterX(double value) { centerX = value; } |
|  |  |
|  | public double getCenterY() { return centerY; } |
|  | void setCenterY(double value) { centerY = value; } |
|  |  |
|  | MyPoint(){} |
|  |  |
|  | public MyPoint(double x, double y){ |
|  | setCenterX(x); |
|  | setCenterY(y); |
|  | } |
|  | public MyPoint(double x, double y, boolean fixed){ |
|  | constant = fixed; |
|  | setCenterX(x); |
|  | setCenterY(y); |
|  | } |
|  |  |
|  | void move(double x, double y){ |
|  | setCenterX(x); |
|  | setCenterY(y); |
|  | } |
|  |  |
|  | public boolean isConstant() { return constant; } |
|  | } |

## 1.12. Класс TriEdge.java

|  |
| --- |
| package Mechanics; |
|  |  |
|  | import Helpers.DoublePair; |
|  | import Helpers.Utils; |
|  |  |
|  | public class TriEdge implements MechPart { |
|  | private Edge first, second, main; |
|  | private MyPoint beginning, center, finish; |
|  | DoublePair[] possibleFinishPositions = new DoublePair[720]; |
|  | DoublePair[] possibleMiddlePositions = new DoublePair[720]; |
|  |  |
|  | TriEdge(MyPoint start, MyPoint mid, MyPoint end){ |
|  | beginning = start; |
|  | center = mid; |
|  | finish = end; |
|  | first = new Edge(beginning, center); |
|  | second = new Edge(center, finish); |
|  | main = new Edge(beginning, finish); |
|  | } |
|  |  |
|  | @Override |
|  | public double getStartX() { |
|  | return beginning.getCenterX(); |
|  | } |
|  |  |
|  | @Override |
|  | public double getStartY() { |
|  | return beginning.getCenterY(); |
|  | } |
|  |  |
|  | @Override |
|  | public double getEndX() { |
|  | return finish.getCenterX(); |
|  | } |
|  |  |
|  | @Override |
|  | public double getEndY() { |
|  | return finish.getCenterY(); |
|  | } |
|  |  |
|  | public MyPoint getBeginning(){ |
|  | return beginning; |
|  | } |
|  | private MyPoint getMiddle() {return center; } |
|  | public MyPoint getEnd(){ |
|  | return finish; |
|  | } |
|  |  |
|  | private void setCenter(double x, double y){ |
|  | center.setCenterX(x); |
|  | center.setCenterY(y); |
|  | recalculatePositions(); |
|  | } |
|  |  |
|  | public double getCenterX() { return center.getCenterX(); } |
|  | public double getCenterY() { return center.getCenterY(); } |
|  |  |
|  | public DoublePair getCenter(){ |
|  | return new DoublePair(getCenterX() , getCenterY()); |
|  | } |
|  |  |
|  | public void setStart(double x, double y){ |
|  | beginning.move(x, y); |
|  | setCenter(getStartX()/2 + getEndX()/2, getStartY()/2 + getEndY()/2); |
|  | recalculatePositions(); |
|  | } |
|  |  |
|  | public void setFinish(double x, double y){ |
|  | finish.move(x, y); |
|  | setCenter(getStartX()/2 + getEndX()/2, getStartY()/2 + getEndY()/2); |
|  | } |
|  |  |
|  | @Override |
|  | public void setFinish(int k) { |
|  | setFinish(possibleFinishPositions[k].first, possibleFinishPositions[k].second); |
|  | } |
|  |  |
|  | public DoublePair getStart(){ |
|  | return new DoublePair(getStartX(), getStartY()); |
|  | } |
|  |  |
|  | public DoublePair getFinish() { |
|  | return new DoublePair(getEndX(), getEndY()); |
|  | } |
|  |  |
|  | public void move(double x1, double y1, double x2, double y2){ |
|  | setStart(x1, y1); |
|  | setFinish(x2, y2); |
|  | setCenter(x1/2+x2/2, y1/2+y2/2); |
|  | } |
|  |  |
|  | @Override |
|  | public void rotate(double angle) { |
|  | DoublePair pif = new DoublePair(getEndX() - getStartX(), |
|  | getEndY() - getStartY()); |
|  | DoublePair pol = Utils.Pif2Pol(pif); |
|  |  |
|  | pol.second += Utils.asRadians(angle); |
|  |  |
|  | DoublePair newPif = Utils.Pol2Pif(pol); |
|  |  |
|  | setFinish(getStartX() + newPif.first, |
|  | getStartY() + newPif.second); |
|  |  |
|  | setCenter(getStartX()/2 + getEndX()/2, getStartY()/2 + getEndY()/2); |
|  | } |
|  |  |
|  | public void move(double dx, double dy){ |
|  | setCenter(getCenterX() + dx, getCenterY() + dy); |
|  | setStart(getStartX() + dx, getStartY() + dy); |
|  | setFinish(getEndX() + dx, getEndY() + dy); |
|  | } |
|  |  |
|  | private void inline(){ |
|  | setCenter(getStartX()/2 + getEndX()/2, getStartY()/2 + getEndX()/2); |
|  | } |
|  |  |
|  | void inlineFinish(){ |
|  | setFinish(getCenterX()\*2 - getStartX(), getCenterY()\*2 - getStartY()); |
|  | } |
|  |  |
|  | @Override |
|  | public double Angle(){ |
|  | return Math.atan2(getEndY()-getStartY(), getEndX()-getStartX()); |
|  | } |
|  |  |
|  | @Override |
|  | public double Length() { |
|  | return Math.sqrt( |
|  | (getEndX()-getStartX())\*(getEndX()-getStartX())+ |
|  | (getEndY()-getStartY())\*(getEndY()-getStartY()) |
|  | ); |
|  | } |
|  |  |
|  | @Override |
|  | public void setLength(double length) { |
|  | double rads = Angle(); |
|  | if (getEnd().isConstant()){ |
|  | setStart(getEndX() - Math.cos(rads)\*length, |
|  | getEndY() - Math.sin(rads)\*length); |
|  | inline(); |
|  | } |
|  | else if (getMiddle().isConstant()) { |
|  | setFinish(getCenterX() + Math.cos(rads) \* length / 2, |
|  | getCenterY() + Math.sin(rads) \* length / 2); |
|  | setStart(getCenterX() - Math.cos(rads) \* length / 2, |
|  | getCenterY() - Math.sin(rads) \* length / 2); |
|  | } |
|  | else { |
|  | setCenter(getStartX() + Math.cos(rads)\*length/2, |
|  | getStartY() + Math.sin(rads)\*length/2); |
|  | setFinish(getStartX() + Math.cos(rads)\*length, |
|  | getStartY() + Math.sin(rads)\*length); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | @Override |
|  | public void recalculatePositions() { |
|  | double radius = Length(); |
|  | DoublePair startPosition = getStart(); |
|  | DoublePair finishPosition = new DoublePair(radius, 0); //finish is lower |
|  | finishPosition = Utils.Pif2Pol(finishPosition); |
|  | for (int i = 0; i < 720; i++){ |
|  | finishPosition.second += 0.5; |
|  | possibleFinishPositions[i] = startPosition.add(Utils.Pol2Pif(finishPosition)); |
|  | possibleMiddlePositions[i] = |
|  | new DoublePair(startPosition.first/2 + possibleFinishPositions[i].first/2, |
|  | startPosition.second/2 + possibleFinishPositions[i].second/2); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public int getPointWithDistance(MechPart source, boolean finish, double distance) { |
|  | int index = 0; |
|  | if (!finish) |
|  | distance /= 2; |
|  |  |
|  | double minDist = Math.abs(Utils.distance(possibleFinishPositions[0], source.getFinish()) - distance); |
|  |  |
|  | for (int i = 1; i < 360; i++) |
|  | if (minDist > Math.abs(Utils.distance(possibleFinishPositions[i], source.getFinish()) - distance)) { |
|  | index = i; |
|  | minDist = Math.abs(Utils.distance(possibleFinishPositions[i], source.getFinish()) - distance); |
|  | } |
|  | return index; |
|  | } |
|  | } |

## 1.13. Класс LegionCreator.java

|  |
| --- |
| package Optimization; |
|  |  |
|  | import Helpers.WriterReader; |
|  | import Mechanics.Mechanism; |
|  |  |
|  | import java.awt.geom.Point2D; |
|  | import java.nio.file.Paths; |
|  | import java.util.Random; |
|  |  |
|  | public class LegionCreator { |
|  | private static Random ran = new Random(); |
|  |  |
|  | public LegionCreator(String path, int number){ |
|  | WriterReader wr = new WriterReader(path); |
|  | double[] params = new double[4]; |
|  | for (int i = 0; i < number; i++){ |
|  | params[0] = 50 + ran.nextInt(100); |
|  | params[1] = 50 + ran.nextInt(200); |
|  | params[2] = 50 + ran.nextInt(200); |
|  | params[3] = 50 + ran.nextInt(200); |
|  |  |
|  | try { |
|  | Mechanism mech = Mechanism.createMech(params); |
|  |  |
|  | if (mech == null || !mech.checkFunctionality()) |
|  | i--; |
|  | else { |
|  |  |
|  | wr.write(params); |
|  | System.out.println(String.format("Written mech %d of %d", i + 1, number)); |
|  |  |
|  | mech.initPositions(); |
|  | String lastPath = Paths.get(".").toAbsolutePath().normalize().toString(); |
|  | lastPath = lastPath.substring(lastPath.length()-15); |
|  | String prepath = ""; |
|  | if (lastPath.equals("TestProject\_jar")) |
|  | prepath += "..\\..\\..\\"; |
|  |  |
|  | WriterReader posWrite = new WriterReader(String.format(prepath + "data\\positions%d.csv", i)); |
|  | for (Point2D p : mech.positions) |
|  | posWrite.write(new double[]{p.getX()}, p.getY()); |
|  |  |
|  | } |
|  | } |
|  | catch (ArrayIndexOutOfBoundsException aiobe){ |
|  | aiobe.printStackTrace(); |
|  | } |
|  | } |
|  | System.out.println("Learning samples created at " + path); |
|  | } |
|  | } |

## 1.14. Класс OutsourceAttractor.java

|  |
| --- |
| package Optimization; |
|  |  |
|  | import Helpers.WriterReader; |
|  |  |
|  | import java.io.\*; |
|  | import java.nio.file.Paths; |
|  |  |
|  | public class OutsourceAttractor { |
|  | public OutsourceAttractor(){} |
|  |  |
|  | public void runPython(){ |
|  | String lastPath = Paths.get(".").toAbsolutePath().normalize().toString(); |
|  | lastPath = lastPath.substring(lastPath.length()-11); |
|  | String path = ""; |
|  | if (lastPath.equals("Project\_jar")) |
|  | path += "..\\..\\..\\"; |
|  |  |
|  | WriterReader wr = new WriterReader(path + "data\\mechs.csv"); |
|  | wr.readSingle(); |
|  | try { |
|  | // System.out.println("python LinReg.py " + wr.features.length); |
|  | String num = Integer.toString(wr.features.length); |
|  | new ProcessBuilder("python", path + "LinReg.py", num).start(); |
|  | } catch (IOException e) { |
|  | e.printStackTrace(); |
|  | } |
|  | } |
|  |  |
|  | public static void main(String[] args){ |
|  | OutsourceAttractor os = new OutsourceAttractor(); |
|  | os.runPython(); |
|  | } |
|  | } |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов | | | | Всего листов (страниц) в документе | № документа | Входя- щий № сопро- водит. докум. и дата | Под- пись | Дата |
| изменен- ных | заменен- ных | новых | аннулиро- ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |