Факултет инжењерских наука Универзитета у Крагујевцу

Софтверски инжењеринг

- Апликација која рачуна површину затворене контуре -

Студент:
Никола Вуловић, 570/2015

Предметни наставник:
Проф. др Ненад Филиповић

Крагујевац, 2018

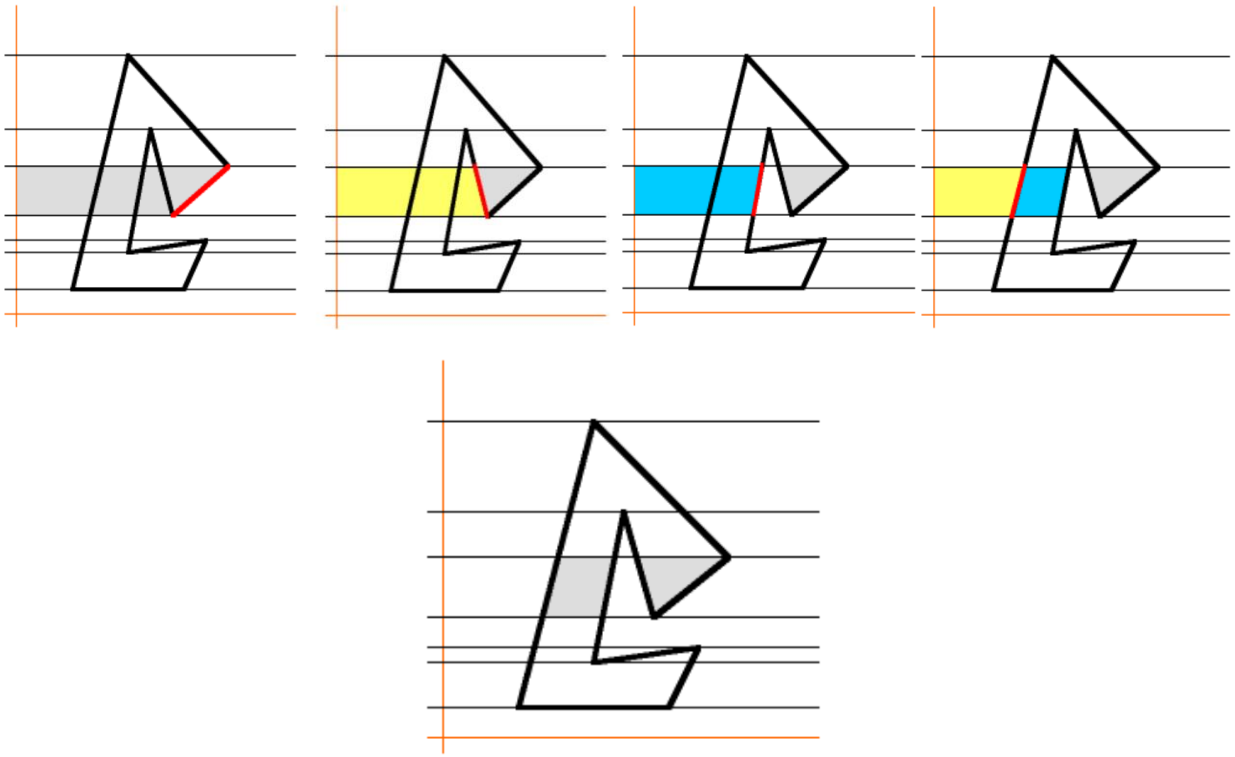
Садржај

1. УВОД 3
2. РЕАЛИЗАЦИЈА.....4
3. UML ДИЈАГРАМИ.....9
ЛИТЕРАТУРА.....12

1. УВОД

Циљ пројеката “Апликација која рачуна површину затворене контуре “ је да се направи апликација која служи да брзо и ефикасно кориснику израчуна површину полигона који он нацрта мишем у апликацији.

Проблем рачунања површине задате контуре се решава алгоритмом који рачуна и препознаје темена задате контуре. Алгоритам претпоставља уобичајену математичку конвенцију да је Y оса позитивна када јој је смер на горе. У већини рачунарских системима где позитивна Y оса има смер на доле најлакше је излистати темена у смеру супротном казаљке на сату користећи координате “позитивне Y осе усмерене на доле”. Алгоритам функционише тако што подели полигон на мање полигоне хоризонталним линијама које пролазе кроз свако теме. Ако рачунамо да је Y оса позитивна на доле, и смер страница у смеру казаљке на сату, алгоритам ће израчунати површину тако што ће све десно од странице рачунати као површину а све лево одузети и онда ће страница која има смер на горе одсећи површину странице са њене десне стране којој је смер на доле.



Слика 1: *Визуелни приказ алгоритма за проналажење површине полигона*

2. РЕАЛИЗАЦИЈА

За реализацију програма је коришћен програмски језик Java у NetBeans IDE 8.2 развојном окружењу. Програм се састоји из класа GUI која је главна класа и графички кориснички интерфејс и класе Povrsina_figure у којој је имплементиран алгоритам који израчунава површину нацртаног полигона. Библиотеке за графички кориснички интерфејс су javax.swing и java.awt.

Када се покрене апликација, иницијализује се GUI класа која учитава графичке компоненте JPanel, JFrame, JButton, JtextArea, Container, BorderLayout, итд. Када се иницијализује, формира се бела површина по којој се могу цртати праве линије које ће на крају бити формиране у полигон. Класа GUI има и две методе: једну која формира и рачуна површину полигона и другу која брише све нацртано на белој површини.

```
JButton calculate;
JButton erase;
Povrsina_figure table = new Povrsina_figure();
ActionListener actionListener1 = new ActionListener() {
    public void actionPerformed(ActionEvent a) {
        if (a.getSource() == calculate) {
            table.calc();
        }
    }
};

ActionListener actionListener2 = new ActionListener() {
    public void actionPerformed(ActionEvent b) {
        if (b.getSource() == erase) {
            table.clear();
        }
    }
};

public static void main(String[] args) {
    new GUI().show();
}

public void show() {

    JFrame frame = new JFrame("Површина фигуре");
    Container content = frame.getContentPane();

    content.setLayout(new BorderLayout());

    table = new Povrsina_figure();

    content.add(table, BorderLayout.CENTER);

    JPanel controls = new JPanel();

    calculate = new JButton("Израчунај");
    JPanel controls = new JPanel();

    calculate = new JButton("Израчунај");
    calculate.addActionListener(actionListener1);
    erase = new JButton("Избриши");
    erase.addActionListener(actionListener2);

    controls.add(calculate);
    controls.add(erase);

    content.add(controls, BorderLayout.NORTH);
    frame.setSize(600, 600);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
```

Слика 2: *GUI класа*

```
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
```

Слика 3: *Компоненте графичкг корисничког интерфејса*

Кликом миша на белу површину се бирају темена полигона и не мора се пазити на коју страну се црта полигон јер се ради апсолутна вредност укупне површине. Кликом на дугме „Израчунај“ повезује се последње теме са почетним уколико није повезано, боји се цео полигон у црно и креће се се са израчунавањем површине тако што метода actionPerformed(ActionEvent a) из класе GUI штампа се коначан резултат у новом прозору преко методе calc која преузима укупан резултат преко методе getSum из методе getPolygonArea у којој се и налази алгоритам за израчунавање површине задатог полигона. Када се изврши прорачун и прикаже укупна површина полигона , корисник не може доцртавати више ништа на белој површини већ може кликнути на дугме „Избриши“ које кроз методу actionPerformed(ActionEvent b) преко методе clear враћа бројач темена на нула тако да не постоји ниједно теме на белој површини а самим тим ни полигон па се може почети испочетка са цртањем.

```
public class Povrsina_figure extends JComponent {

    private static final int NUMPOINTS = 500;
    private JButton finish;
    private Polygon shape;
    private boolean isDrawn;
    private int count;
    private Color color;
    private int[] x;
    private int[] y;
    private float sum = 0;
    private float area;
    private Image image;
    private Graphics2D g2;

    public Povrsina_figure() {
        isDrawn = false;
        count = 0;
        x = new int[NUMPOINTS];
        y = new int[NUMPOINTS];
        color = Color.BLACK;
        shape = new Polygon();
        setDoubleBuffered(false);
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                if (isDrawn == false && count < NUMPOINTS) {
                    x[count] = e.getX();
                    y[count] = e.getY();
                    count++;
                    repaint();
                }
            }
        });
    }

    public void paint(Graphics g) {
        super.paint(g);
        g.drawPolyline(x, y, count);
        g.setColor(color);
        if (isDrawn) {
            g.fillPolygon(x, y, count);
        }
    }
```

```
protected void paintComponent(Graphics g) {
    if (image == null) {
        image = createImage(getSize().width, getSize().height);
        g2 = (Graphics2D) image.getGraphics();
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setPaint(Color.white);
        g2.fillRect(0, 0, getSize().width, getSize().height);
    }
    g.drawImage(image, 0, 0, null);
}

public void clear() {
    count = 0;
    isDrawn = false;
    repaint();
}

public float getPolygonArea(int[] x, int[] y, int count) {
    area = 0;
    int j = count - 1;
    for (int i = 0; i < count; i++) {
        area = area + (x[j] + x[i]) * (y[j] - y[i]);
        j = i;
    }
    sum = Math.abs(area / 2);
    return sum;
}

public float getSum() {
    isDrawn = true;
    color = Color.black;
    repaint();
    return sum;
}

public void calc() {
    getPolygonArea(x, y, count);
    JOptionPane.showMessageDialog(null, "Површина полигона је "
            + getSum());
```

Слика 4: *Klasa Povrsina_Figure*

Метода mousePressed(MouseEvent e) служи да прикупља X и Y координате и такође инкрементира бројач темена за 1 на сваки клик. Бројач је на почетку 0.

Метода paint(Graphics g) служи да прикаже нацртане линије тако што узима координате темена тренутно стање бројача. Док год је isDrawn = false, цртање је омогућено. Када се заврши цртање и кликне на дугме „Израчунај“, isDrawn постаје true и корисник не може више цртати по белој површини.

Метода paintComponent(Graphics g) служи да рашири белу површину по панелу по којој се може цртати и такође додаје anti-aliasing да би слика изгледала лепше, тј. да би линија изгледала као једна линија а не више малих изломљених линија.

Метода getPolyonArea је најзначајнија метода у класи без које би програм био као минималистичка верзија Painta. У њој је алгоритам који је написан по математичком обрасцу за израчунавање полигона:

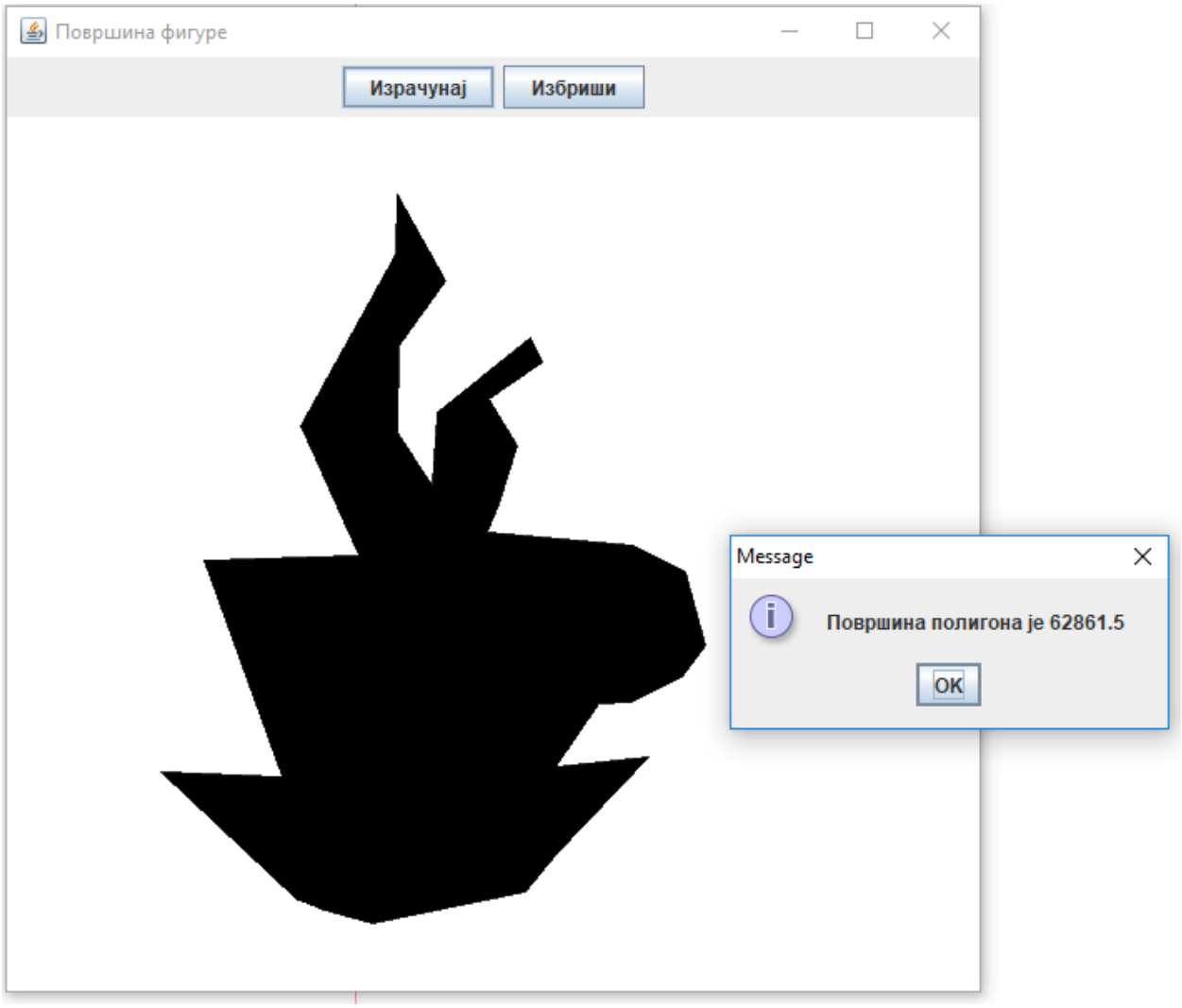
$$area = \left| \frac{(x_1\ y_2 - y_1\ x_2) + (x_2\ y_3 - y_2\ x_3) ..... + (x_n\ y_1 - y_n\ x_1)}{2} \right|$$

Алгоритам израчунава укупну површину и додељује је променљивој sum која се прослеђује даље методама и штампа коначан резулатат у новом прозору.

```
public float getPolygonArea(int[] x, int[] y, int count) {
    area = 0;
    int j = count - 1;
    for (int i = 0; i < count; i++) {
        area = area + (x[j] + x[i]) * (y[j] - y[i]);
        j = i;
    }
    sum = Math.abs(area / 2);
    return sum;
}
```

Слика 4: *Метода getPolygonArea*

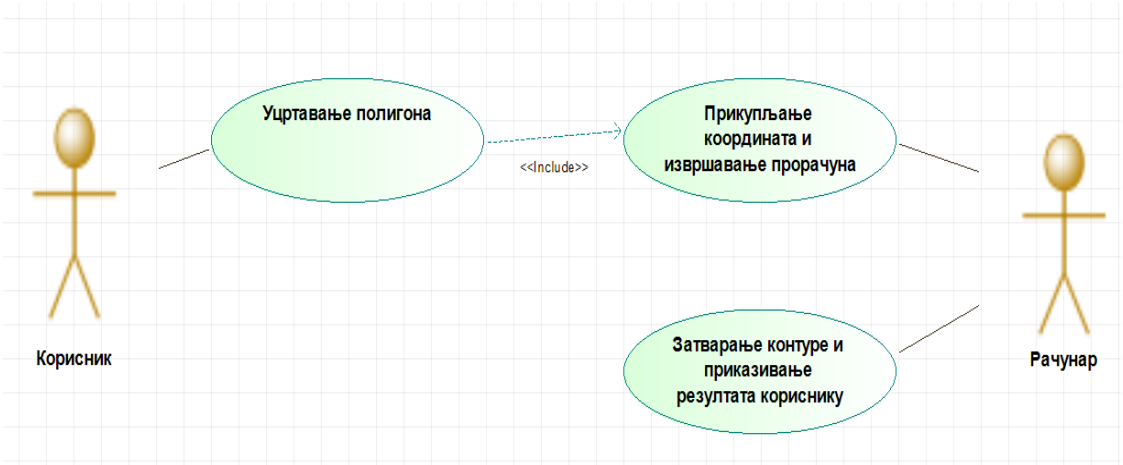
Као што је наведено горе, апликација се састоји од 2 дугмета за израчунавање и брисање, а испод ње се налази бела површина по којој је могуће цртати. Једини недостатак овог кода је што када се укрсте странице полигона, странице ће одузети, површина после укрштања ће бити негативна и одузеће се од укупне површине пре укрштања. Зато треба водити рачуна да се странице не секу.



Слика 5: *Апликација*

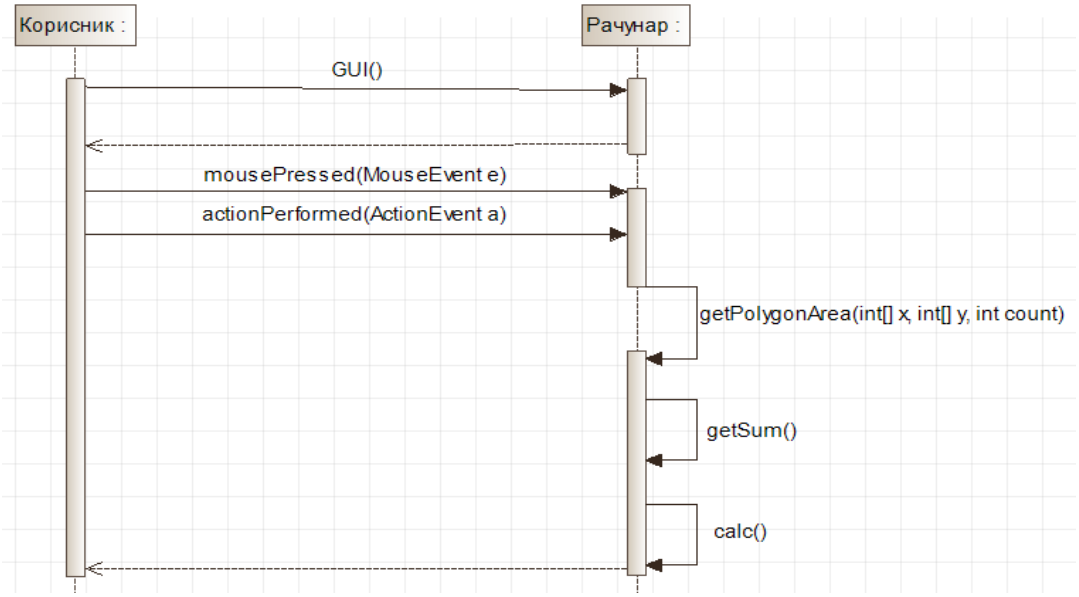
3. UML ДИЈАГРАМИ

Дијаграм случајева коришћења (use-case) приказује скуп случајева коршћења и актера. Служи за визуелизацију понашања система, подсистема или чак класе интерфејса. Ова врста дијаграма специфира шта субјекат ради.



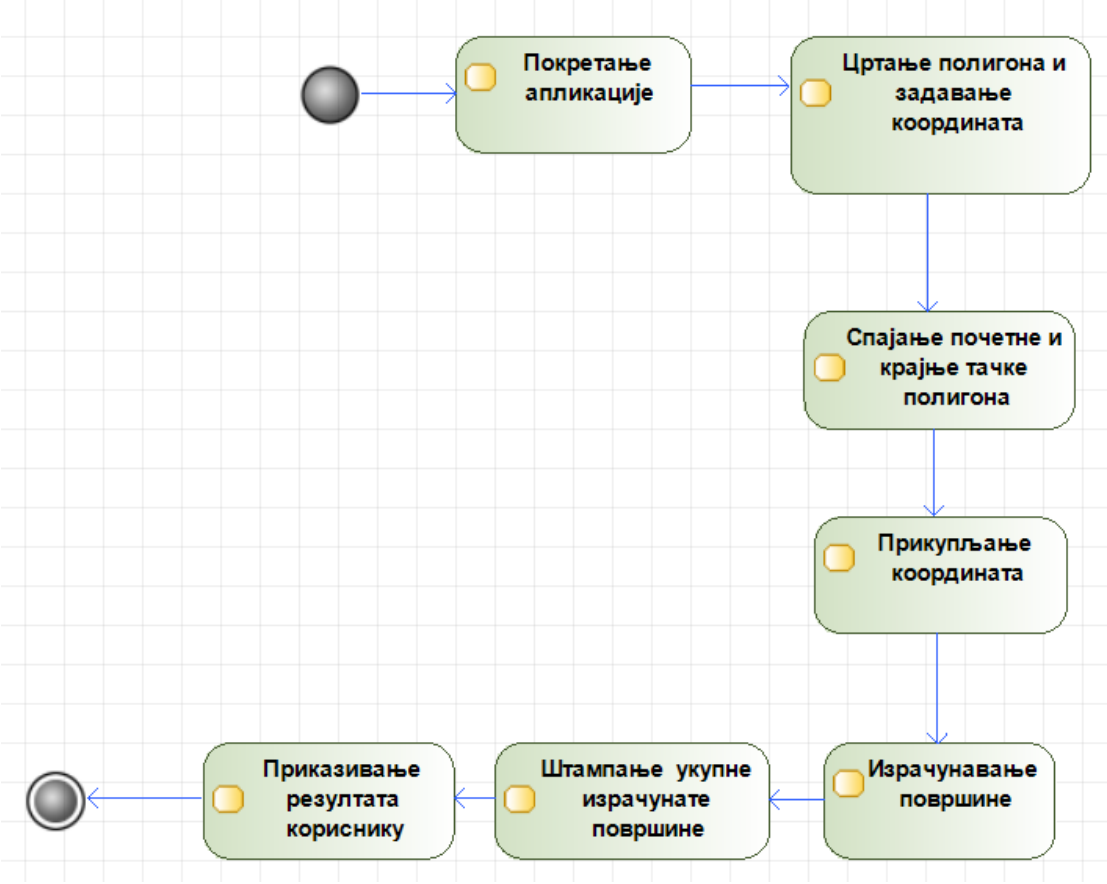
Слика 6: *Use-case дијаграм*

Дијаграм секвенци приказује комуникацију између скупа објеката, која се остварује порукама, које објекти међусобно размењују у циљу остваривања очекиваног понашања. Дијаграм описује како се операције изводе, које поруке се шаљу и када. За пројектни задатак је коришћена вертикална димензија која означава време.



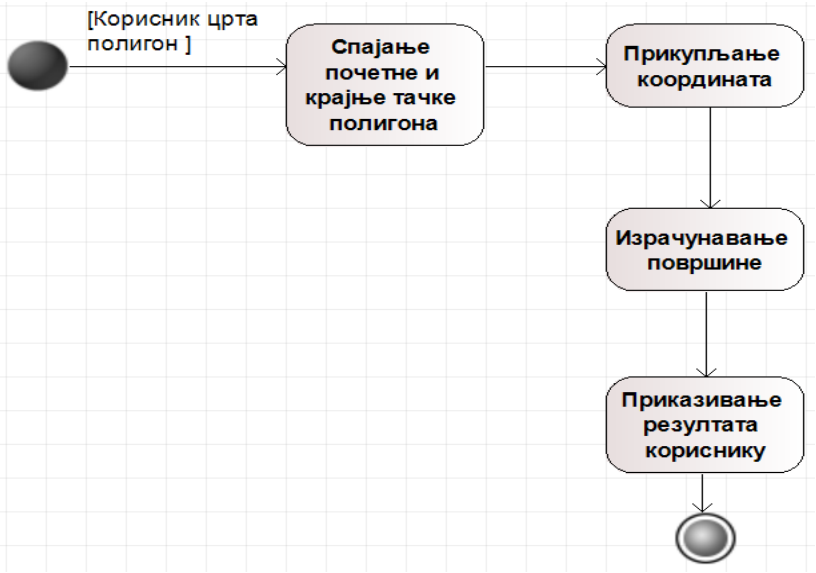
Слика 7: *Дијаграм секвенци*

Дијаграми активности су намењени за моделирање динамичких аспеката (понашања) система. Приказују ток активности коју извршавају објекти као и ток објеката између корака активности.



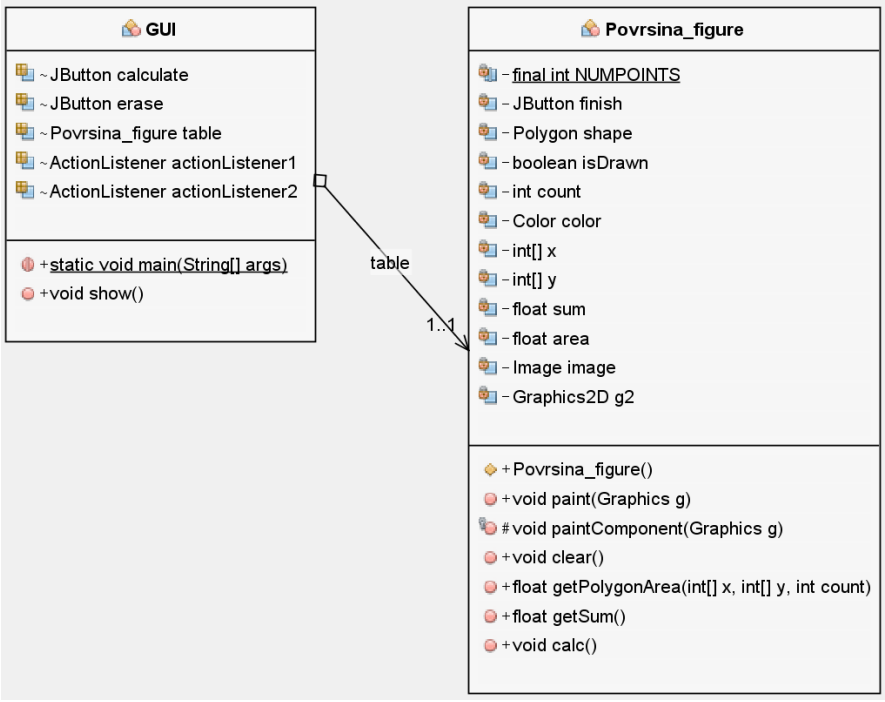
Слика 8: *Дијаграм активности*

Аутомат стања приказује понашање које специфира секвенце стања кроз која пролази. Уједно, моделира понашање неког ентитета или протокол интеракције. Ова врста дијаграма се фокусира на ток активности и динамичка понашања.



Слика 12: *Дијаграм стања*

Дијаграм класа приказује скуп класа, интерфејса и других структура, повезаним релацијама. Специфира логичке и статичке аспекте модела. Ова врста дијаграма се највише користи у објектном моделирању.



Слика 13: *Дијаграм класа*

ЛИТЕРАТУРА

1. Moodle портал http://moodle.mfkg.rs предмет Софтверски инжењеринг
2. Сајт Math Open Reference
   https://www.mathopenref.com/coordpolygonarea.html
   https://www.mathopenref.com/coordpolygonarea2.html
3. Wikipedia - https://www.wikipedia.org
4. Netbeans IDE 8.2 https://netbeans.org
5. Modelio Open Source 3.7 https://www.modelio.org
6. Web