Гимназија Крушевац

Матурски рад

Област: Информатика и рачунарство

Тема: Асинхрона комуникација између рачунара у локалној мрежи путем сокета коришћењем више програмских нити показана на примеру графичког објектно оријентисаног програмирања

Ментор:
Наталија Милићевић, проф.

Ученик:
Лазар Јелић, IV_6

Крушевац, Мај 2017. године

Садржај

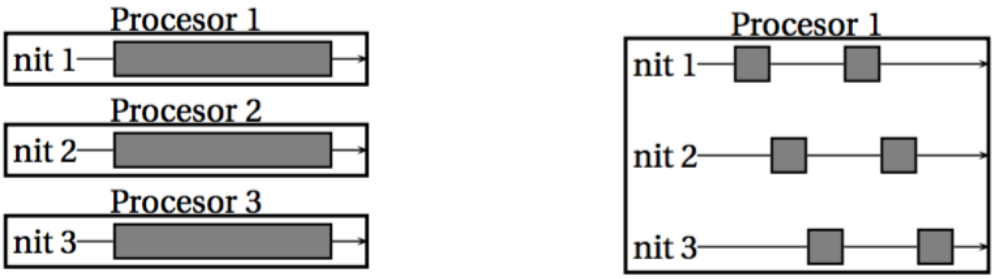
Основни појмови

Асинхрона комуникација - комуникација по режиму трансмисије може бити синхрона или асинхрона. Термин синхроно буквално значи „у исто време”, док асинхроно значи „не у исто време”. Асхинхроност програма подразумева међусобну независност процеса тако да се они могу извршавати без обзира на одсуство корисничког уноса или статуса других процеса.

Рачунарска мрежа - телекомуникациони систем за пренос података који омогућава одређеном броју независних уређаја да међусобно комуницирају. На основу величине, рачунарске мреже се деле на основу подручја које покривају. Можемо посматрати локалне мреже *LAN (Local Area Network)* и мреже на великом подручју *WAN (World Area Network)*. Постоје и поделе према праву приступа, начину преноса и топологији.

Интернет сокет - крајња тачка двосмерног међупроцесног комуникационог тока преко рачунарске мреже. Одређен је јединственом комбинацијом протокола, порта, локалне и удаљене адресе.

Коришћење више програмских нити (*multithreading*) - могућност централне процесорске јединице (*CPU*) да извршава више процеса/нити “истовремено”. Више нити се у мултипроцесорском рачунару могу извршавати истовремено, док се у једнопроцесорском рачунару може добити привид њиховог истовременог извршавања где више нити деле време јединог процесора тако што их процесор извршава мало по мало. Начин на који се више нити могу извршавати у рачунару илустрован је на слици 0.



мултипроцесорски рачунар једнопроцесорски рачунар

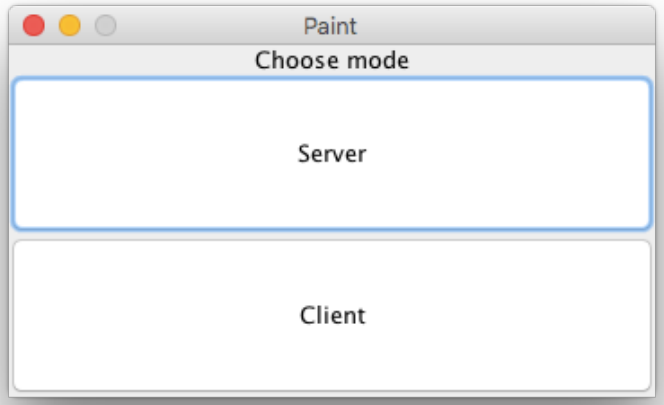
Слика 0: Типови вишенитности

Графички програми - знатно се разликују од конзолних прогрма. Највећа разлика је у томе што се конзолни програм извршава синхроно. У конзолним програмима се програмски одређује тренутак када корисник треба да унесе улазне податке и када се приказују излазни подаци. Насупрот томе, код графичких програма је корисник тај који одређује када хоће да унесе улазне податке и најчешће он сам одређује када се приказују излазни подаци програма. За графичке програме се зато каже да су вођени догађајима (*event-driven programs*). Графички програми имају много богатији кориснички интерфејс него конзолни програми.

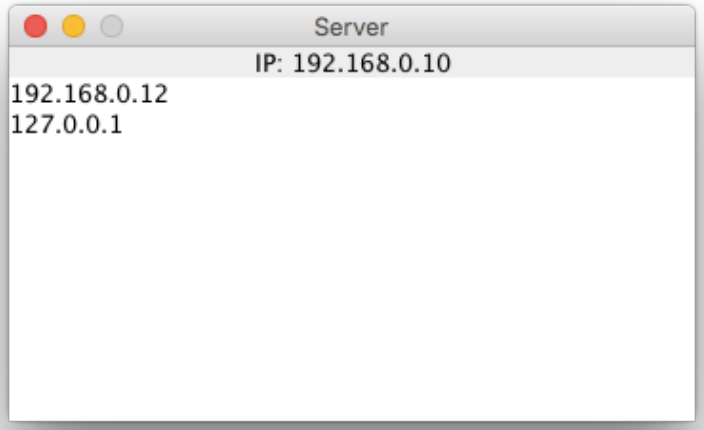
Објектно оријентисано прграмирање (ООП) - једна од програмских парадигми. Маркетиншка дефинција каже да је све на свету објекат са атрибутима и функцијама и да између тих објеката постоје везе па је тако програм реална слика стварности. Кључна разлика између класичниог процедуралног програмирања и објектно орјентисаног програмирања је непостојање јасне поделе на податке и операције које треба над подацима извршити. Основни принципи: наслеђивање, енкапсулација и полиморфизам. У циљу налажења најефикаснијег решења проблема и писања квалитетног кода, програмер треба поштовати *SOLID* принципе, користити *design patterns* и знати како треба одрадити рефакторизацију постојећег кôда.

Прозори

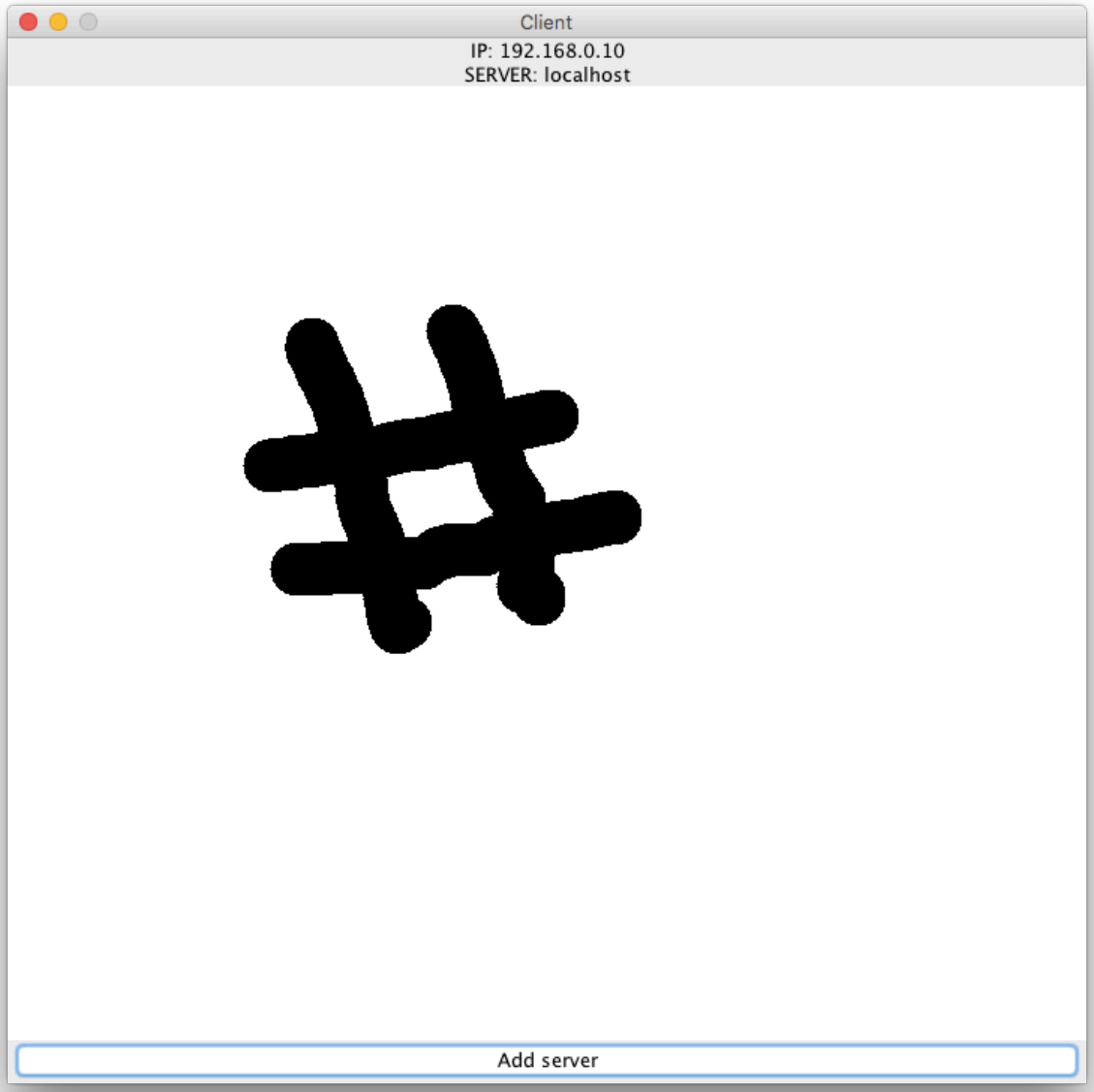
Програм се састоји од укупно 4 прозора, а то су 3 оквира (главни прозор, сервер, клијент) и 1 дијалог за додавање ИП (Интернет Протокол) адресе сервера.



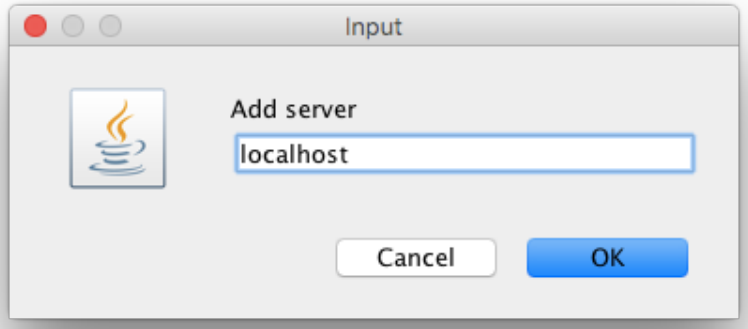
Слика 1: Главни прозор – избор начина рада



Слика 2: Сервер – приказ повезаних клијената



Слика 3: Клијент – приказ цртежа

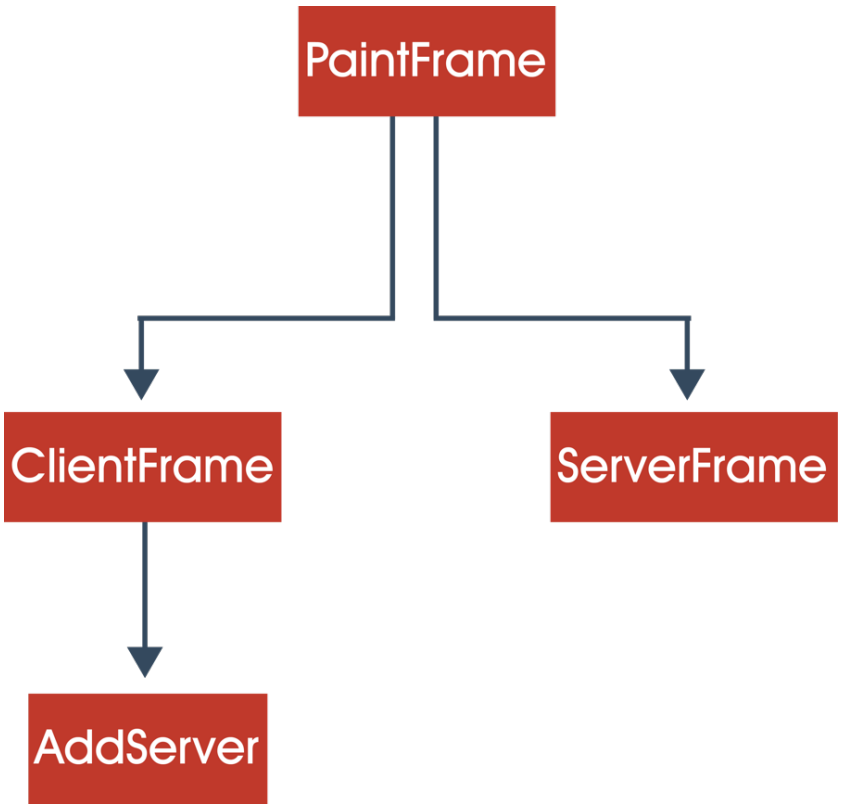


Слика 4: Повезивање клијента са сервером

Ток програма

Након избора начина рада, кориснику се може приказати прозор сервера или клијента. Корисник коме је прикзан прозор клијента има могућност повезивања са сервером. Након успешно успостављене везе, у главном прозору:

- сервера се приказује ИП адреса клијента који је иницирао везу
- клијента се омогућава цртање тачака

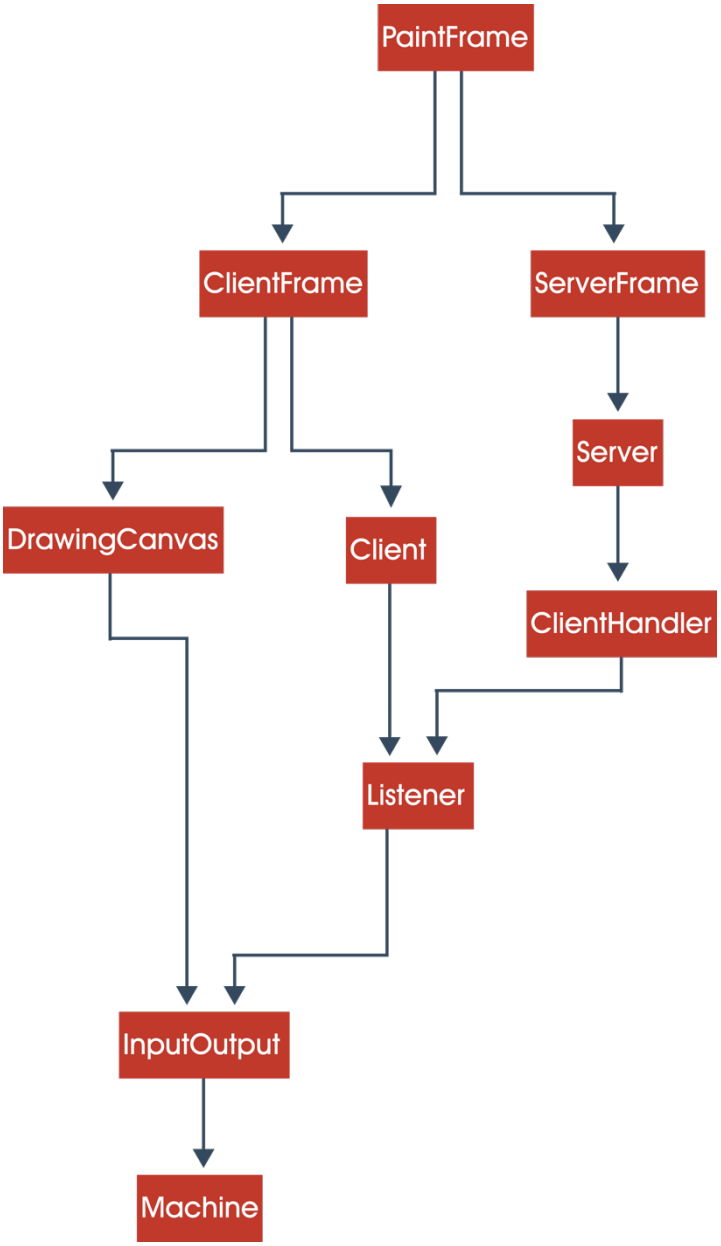


Слика 5: Графички приказ тока програма

Класе и интерфејси

Односи

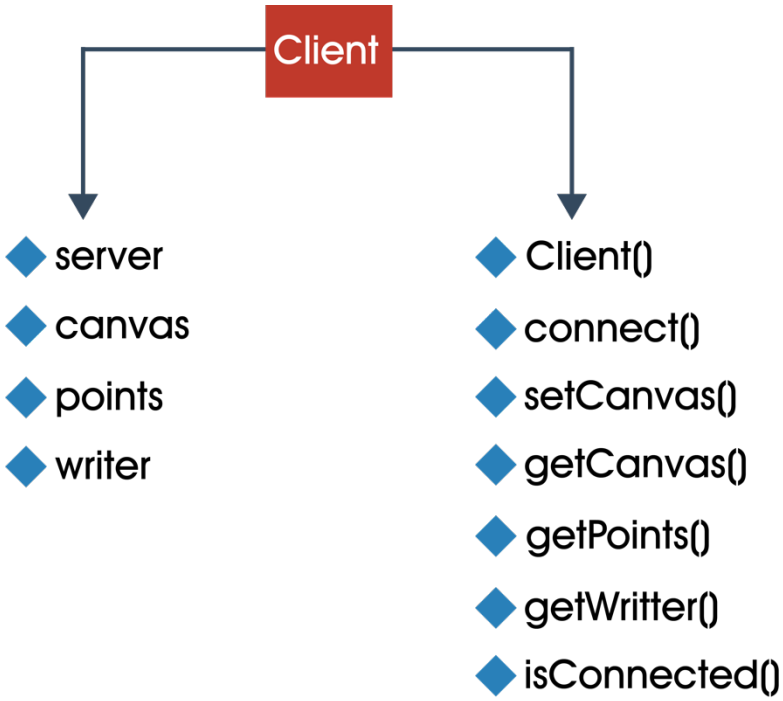
У корену хијерархијског стабла класа овог програма налази се *Machine класа* која је наслеђена у класи надлежној за целокупну комуникацију између клијента и сервера - *InputOutput*. На следећем нивоу се налазе класе за читање и писање координата тачака. Писање тих координата је омогућено у прозору клијента, а њихово читање у класи *Client* чија инстанца такође постоји у прозору клијента и у руководиоцу нових конекција који је покренут у класи *Server* чији објекат постоји у прозору сервера. Оба прозора се отварају након бирања начина рада у главном прозору програма.



Слика 6: Графички приказ односа између класа

Client.java

Класа која представља клијента. Наслеђује се *Machine* класа. Од атрибута садржи објекат сервера са којим је клијент повезан, платно за цртање и *PrintWriter* објекат који исписује нацртане тачке које се налазе у листи. Од битнијих метода ова класа поседује метод за повезивање са сервером и проверу те везе.



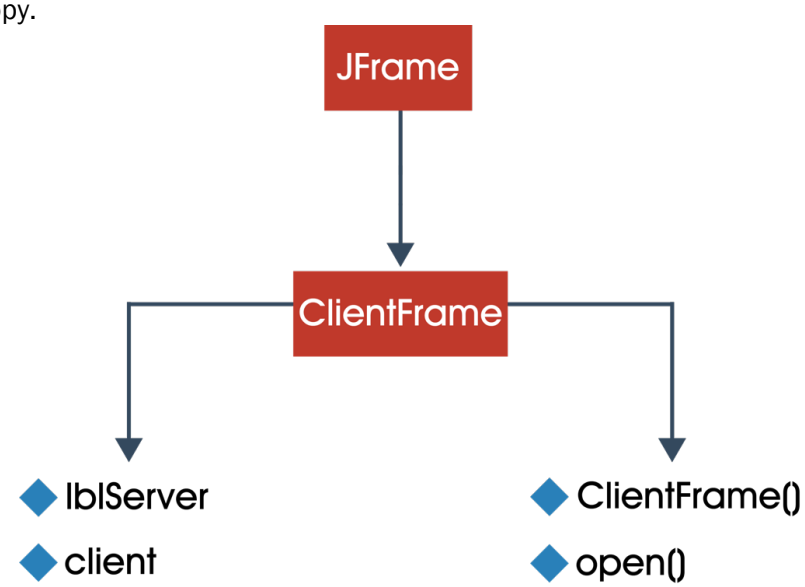
Слика 7: Дијаграм клијента

```
public class Client extends Machine {

    private Server server;
    private JComponent canvas;
    private ArrayList<Point> points;
    private PrintWriter writer;

    //odredjivanje inicijalne vrednosti niza tacaka
    public Client() {
        points = new ArrayList<Point>();
    }

    //povezivanje sa serverom
    public void connect(Server server) throws IOException {
        this.server = server;

        //kreiranje soketa na osnovu IP adrese servera i porta
        Socket socket = new Socket(server.getIP(), 2345);

        //kreiranje BufferedReader objekta na osnovu soketa
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(socket.getInputStream()));

        //kreiranje PrintWriter objekta na osnovu soketa
        writer = new PrintWriter(socket.getOutputStream());

        //praznjenje liste tacaka
        points.clear();
```

```
        //osvezivanje platna za crtanje
        canvas.repaint();

        //odredjivanje Machine objekta
        setMachine(this);

        //odredjivanje StringBuilder objekta
        setStringBuilder(new StringBuilder());

        //kreiranje programske niti koja ce citati linije BufferedReader objekta
        Thread thread = new Thread(new Listener(this, reader));

        //startovanje programske niti
        thread.start();
    }

    //odredjivanje platna za crtanje
    public void setCanvas(JComponent canvas) {
        this.canvas = canvas;
    }

    //uzimanje platna za crtanje
    public JComponent getCanvas() {

        return canvas;
    }

    //uzimanje liste tacaka
    public ArrayList<Point> getPoints() {

        return points;
    }

    //uzimanje PrintWriter objekta
    public PrintWriter getWriter() {

        return writer;
    }

    //provera konekcije izmedju klijenta i servera
    public boolean isConnected() {

        return server != null;
    }
}
```

ClientFrame.java

Класа главног прозора клијента и због тога наслеђује *JFrame* класу. *Client* објекат и лабела за ИП адресу сервера су атрибути ове класе. Садржи метод за отварање прозора док се сва подешавања налазе у конструктору.



Слика 8: Дијаграм главног прозора клијента

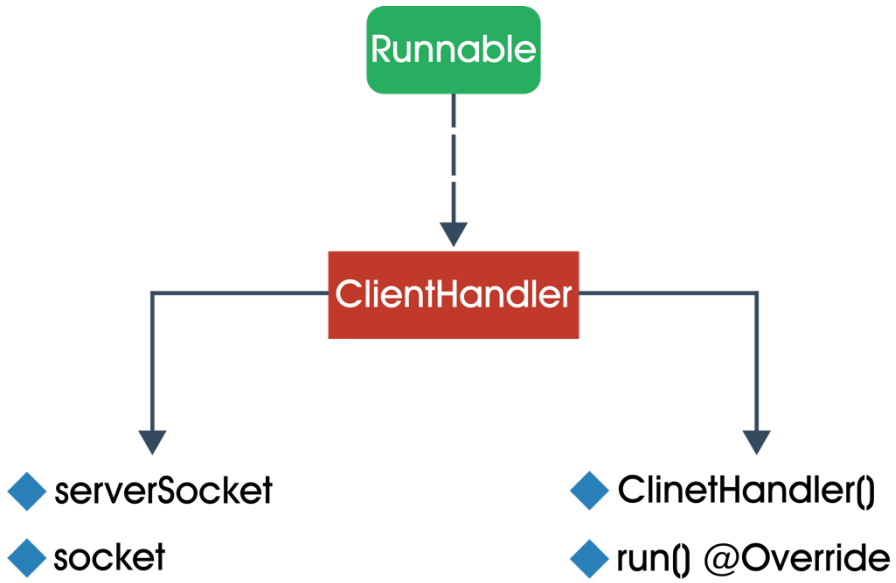
```
public class ClientFrame extends JFrame {

    private JLabel lblServer;
    private Client client;

    //konfigurisanje forme
    public ClientFrame() {
        //kreiranje novog Client objekta
        client = new Client();

        //odredjivanje naziva forme
        setTitle("Client");

        //zatvaranje forme klikom na 'X'
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        //zabrana promene velicine forme od strane korisnika
        setResizable(false);

        //uzimanje velicine ekrana
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        //deklaracija varijabli za sirinu i visinu forme
        int frameWidth = (int) (screenSize.width * 0.5);
        int frameHeight = (int) (screenSize.width * 0.5);

        //odredjivanje sirine i visine forme
        setSize(frameWidth, frameHeight);

        //odredjivanje lokacije forme na ekranu
        setLocation(screenSize.width / 2 - frameWidth / 2,
                screenSize.height / 2 - frameHeight / 2);

        //kreiranje panela
        JPanel panel = new JPanel();
```

```
        //odredjivanje tipa layouta panela
        panel.setLayout(new BorderLayout());

        //kreiranje panela za labelu
        JPanel labelPanel = new JPanel();

        //odredjivanje tipa layouta panela za labele
        labelPanel.setLayout(new BorderLayout());

        //kreiranje labele za IP adresu klijenta i odredjivanje njenog teksta
        JLabel lblIP = new JLabel("IP: " + client.getInetAddress());

        //odredjivanje poravnanja teksta u labeli za IP adresu klijenta
        lblIP.setHorizontalAlignment(SwingConstants.CENTER);

        //kreiranje labele za IP adresu servera i odredjivanje njenog teksta
        lblServer = new JLabel("SERVER: undefined");

        //odredjivanje poravnanja teksta u labeli
        lblServer.setHorizontalAlignment(SwingConstants.CENTER);

        //ubacivanje labela u panel za labele i njihovo smestanje u odredjene pozicije
        labelPanel.add(lblIP, BorderLayout.NORTH);
        labelPanel.add(lblServer, BorderLayout.SOUTH);

        //kreiranje platna za crtanje i odredjivanje njegovog klijenta, sirine i visine
        JComponent canvas = new DrawingCanvas(client, frameWidth, frameHeight);

        //kreiranje dugmeta za dodavanje servera
        JButton btnAdd = new JButton("Add server");

        //rukovodjenje dogadjajem klika
        btnAdd.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent event) {
                //kreiranje dijaloga za unos IP adrese servera i
                 odredjivanje njene inicijalne vrednosti
                String input = JOptionPane.showInputDialog(null,
                         "Add server",
                         "localhost");

                //proveravanje da li je polje za unos IP adrese servera prazno
                if(input != null && !input.isEmpty()) {
                    //Kreiranje novog Server objekta
                    Server server = new Server();

                    //odredjivanje IP adrese servera
                    server.setIP(input.trim());

                    try {
                        //povezivanje klijenta sa serverom
                        client.connect(server);
                    }catch(IOException e) {
                        //prikazivanje greske o nevazecoj IP adresi
                          servera korisniku u vidu dijaloga
                        JOptionPane.showMessageDialog(null,
                                  "Invalid address",
                                  "Error",
                                  JOptionPane.ERROR_MESSAGE);

                        //logovanje greske
                        e.printStackTrace();

                        //zavrsetak daljeg izvrsavanja metoda
                        return;
                    }

                    //odredjivanje teksta labele za IP adresu servera
                    lblServer.setText("SERVER: " + server.getIP());
```

```
                }
            }
        });

        //ubacivanje komponenti u panel i njihovo smestanje u odredjene pozicije
        panel.add(labelPanel, BorderLayout.NORTH);
        panel.add(canvas, BorderLayout.CENTER);
        panel.add(btnAdd, BorderLayout.SOUTH);

        //panel postaje glavni
        setContentPane(panel);
    }

    //otvaranje forme
    public static void open() {
        //pozivanje konstruktora i dobijanje vidljivosti forme
        new ClientFrame().setVisible(true);
    }
}
```

ClientHandler.java

Функција ове класе је да у позадини прима нове конекције преко одређеног сокета и позива метод *Server* класе за додавање клијента који је направио позив за ту конекцију.



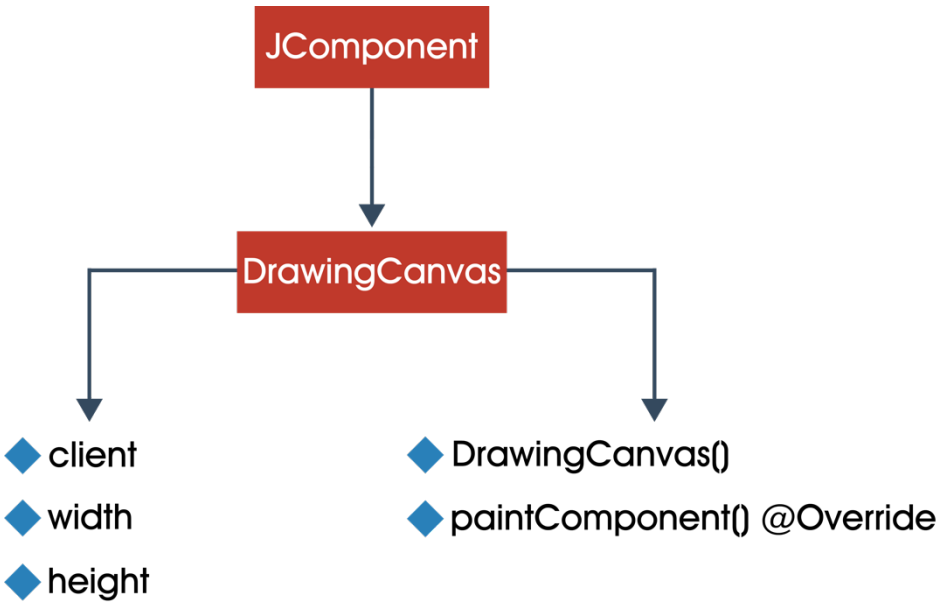
Слика 9: Дијаграм руководиоца нових конекција

```
public class ClientHandler implements Runnable {

    private ServerSocket serverSocket;
    private Server server;

    //odredjivanje ServerSocket i Server objekta
    public ClientHandler(ServerSocket serverSocket, Server server) {
        this.serverSocket = serverSocket;
        this.server = server;
    }

    //izvrsavanje u pozadini
    @Override
    public void run() {
        try {
```

```
            //konstantno izvrsavanje
            while(true) {
                //cekanje klijenta da se poveze sa serverom i kreiranje soketa
                Socket socket = serverSocket.accept();

                //uzimanje ip adrese na osnovu soketa
                InetSocketAddress inetSocketAddress = (InetSocketAddress) socket
                        .getRemoteSocketAddress();

                //kreiranje ip adrese
                String ip = inetSocketAddress.getAddress().toString().replace("/", "");

                //dodavanje klijenta
                server.addClient(ip, socket);
            }
        }catch(Exception e){
            //logovanje greske
            e.printStackTrace();
        }
    }
}
```

DrawingCanvas.java

Класа која представља платно за цртање. Наслеђује класу *JComponent*. Атрибути ове класе су висина и ширина платна, као и *Client* објекат који је конструисан у класи главног прозора клијента.



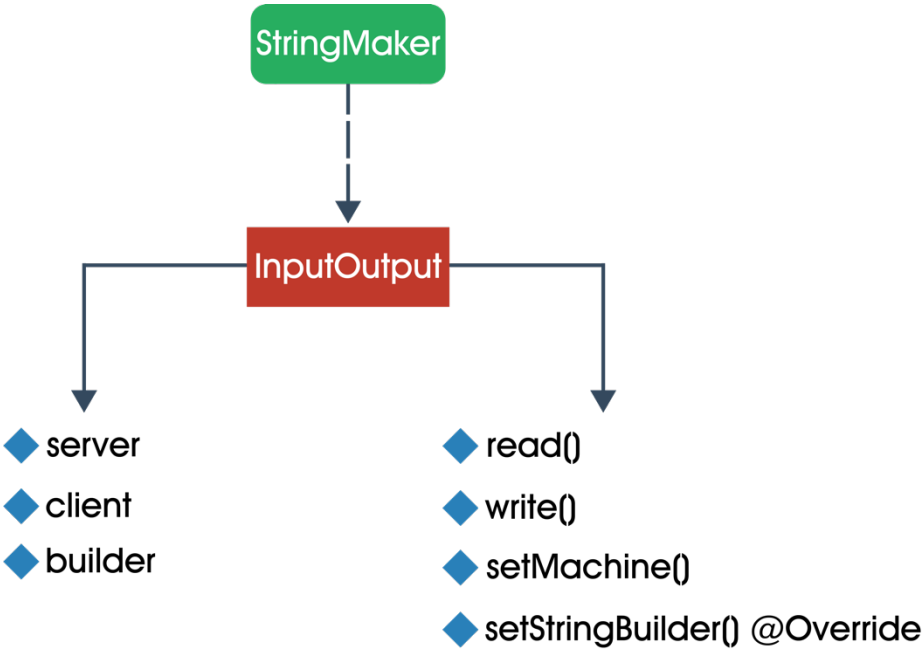
Слика 10: Дијаграм платна за цртање

```
public class DrawingCanvas extends JComponent {

    private Client client;
    private int width, height;

    //odredjivanje Client objekta i sirine i visine platna za crtanje
    public DrawingCanvas(Client client, int width, int height) {
        this.client = client;
        this.width = width;
        this.height = height;
        this.client.setCanvas(this);
```

```
        //dodavanje rukovodioca dogadjaja prevlacenja
        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                //pozivanje metoda za rukovodjenje dogadjaja nasledjene klase
                super.mouseDragged(e);

                //provera konekcije izmedju klijenta i servera
                if(client.isConnected()) {
                    //emitovanje koordinate dogadjaja prevlacenja serveru
                    client.write(client.getWriter(), e.getPoint());
                }
            }
        });
    }

    //osvezivanje platna za crtanje
    @Override
    protected void paintComponent(Graphics g) {
        //kreiranje Graphics2D objekta
        Graphics2D g2 = (Graphics2D) g;

        //odredjivanje boje platna
        g2.setColor(Color.WHITE);

        //odredjivanje oblika i velicine platna
        g2.fill(new Rectangle2D.Double(0, 0, width, height));

        //listanje kroz listu tacaka klijenta
        for(int i = 0; i < client.getPoints().size(); i++) {
            //uzimanje trenutne tacke iz liste tacaka klijenta
            Point point = client.getPoints().get(i);

            //odredjivanje boje tacke
            g2.setColor(Color.BLACK);

            //odredjivanje oblika i velicine tacke
            g2.fillOval(point.x, point.y, (int) (width * 0.05), (int) (height * 0.05));
        }
    }
}
```

InputOutput.java

Ова класа има 2 основне функције. Прва је читање података послатих од стране сервера, а друга је слање координата нацртаних тачака помоћу *PrintWriter* објекта који је конструисан на основу сокета. Овде се среће један од основних принципа ООП, а то је полиморфизам. Метод *write* се може извршити на 2 начина у зависности од прослеђених параметара. Први начин је прослеђивање *Point* објекта из кога ће координате бити извучене, формирање *String* објекта и после тога следи позивање истоименог метода, али уместо Point објекта, њему ће бити прослеђена линија која ће бити емитована помоћу *PrintWriter* објекта. У овом случају позивање истоименог метода не изазива рекурзију. Поред ова 3 метода за 2 главне функције, постоји метод за одређивање *Machine* објекта на основу којег ће се позивати један од 2 *write* метода и наслеђени метод *setStringBuilder* имплементираног интерфејса *StringMaker*.



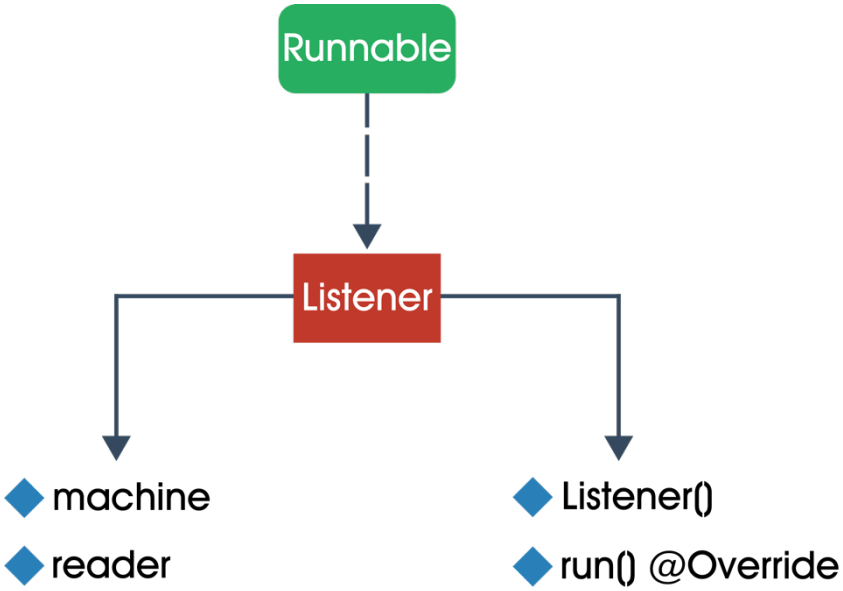
Слика 11: Дијаграм класе за читање и писање координата тачака

```
public class InputOutput implements StringMaker {

    private Server server;
    private Client client;
    private StringBuilder builder;

    //citanje linija BufferedReader objekta
    protected void read(BufferedReader reader) {
        try {
            //postavljanje inicijalne vrednosti linije na NULL
            String line;

            //provera tipa Machine objekta
            if(client != null) {
                //citanje linija sve dok one postoje
                while((line = reader.readLine()) != null) {
                    //izvlacenje koordinata tacke iz linije
                    int x = Integer.parseInt(line.substring(0, line.indexOf(".")));
                    int y = Integer.parseInt(line.substring(line.indexOf(".") + 1));

                    //kreiranje tacke i njeno dodavanje u listu tacaka klijenta
                    client.getPoints().add(new Point(x, y));

                    //osvezivanje platna za crtanje
                    client.getCanvas().repaint();
                }
            }else if(server != null) {
                //citanje linija sve dok one postoje
                while((line = reader.readLine()) != null) {
                    //emitovanje linije svim klijentima
                    server.broadcast(line);
                }
            }

            //zavrsetak citanja zatvaranjem BufferedReader objekta
            reader.close();
        }catch(IOException e) {
            //logovanje greske
            e.printStackTrace();
```

```
        }
    }

    //emitovanje tacke serveru pomocu PrintWriter objekta
    protected void write(PrintWriter writer, Point point) {
        //praznjenje StringBuilder objekta
        builder.setLength(0);
        builder.trimToSize();

        //dodavanje koordinata tacke u StringBuilder objekat
        builder.append(point.x).append(".").append(point.y);

        //kreiranje linije preko StringBuilder objekta
        String packet = builder.toString();

        //pozivanje istoimenog metoda za emitovanje linije
        write(writer, packet);
    }

    //emitovanje linije pomocu PrintWriter objekta
    protected void write(PrintWriter writer, String line) {
        //emitovanje linije
        writer.println(line);

        //zavrsetak emitovanja
        writer.flush();
    }

    //odredjivanje Machine objekta
    protected void setMachine(Machine machine) {
        //provera tipa Machine objekta
        if(machine instanceof Client) {
            //odredjivanje Client objekta
            client = (Client) machine;
        }else if(machine instanceof Server) {
            //odredjivanje Server objekta
            server = (Server) machine;
        }
    }

    //odredjivanje StringBuilder objekta
    @Override
    public void setStringBuilder(StringBuilder builder) {
        this.builder = builder;
    }
}
```

Listener.java

Класа која има функцију да у позадини извршава један од главних метода *InputOutput* класе, а то је *read* и кроз њега ће се у одређеном тренутнку позвати write метод. Атрибути ове класе су *Machine* и *BufferedReader* објекти.

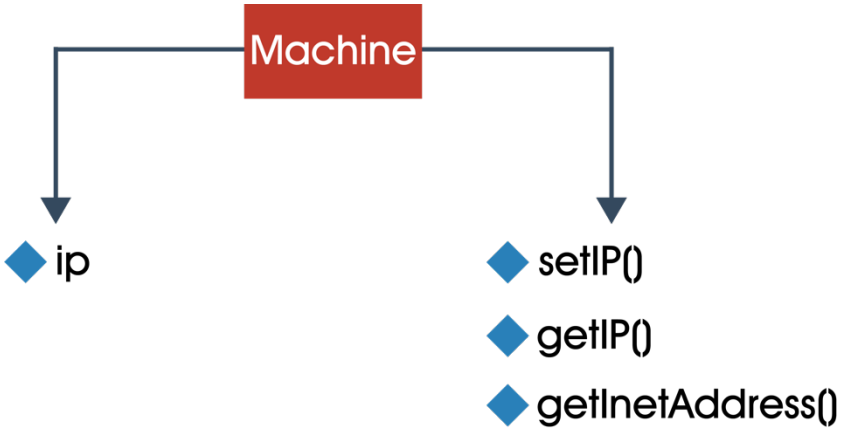


Слика 12: Дијаграм ослушчкивача сокета

```
public class Listener implements Runnable {

    protected Machine machine;
    protected BufferedReader reader;

    //odredjivanje Machine i BufferedReader objekta
    public Listener(Machine machine, BufferedReader reader) {
        this.machine = machine;
        this.reader = reader;
    }

    //izvrsavanje u pozadini
    @Override
    public void run() {
        //citanje linija BufferedReader objekta
        machine.read(reader);
    }
}
```

Machine.java

Класа која представља рачунар са својом јединственом ИП адресом у локалној мрежи. Дакле, од атрибута ова класа поседује само *String* објекат који ће да добије вредност ИП адресе рачунара у тој мрежи. Овом *String* објекту је приступ ограничем само на Machine класу модификатором private па захтева "getter" и "setter" методе. Примећујемо и *getInetAddress* метод који се позива директно у случају приказивања ИП адресе у лабелама и индиректно кроз метод *getIP* у случају када ИП адреса нема вредност. Сваки рачунар у мрежи има могућност слања и примања података па тако ова класа наслеђује *InputOutput* класу која контролише улаз и излаз података.

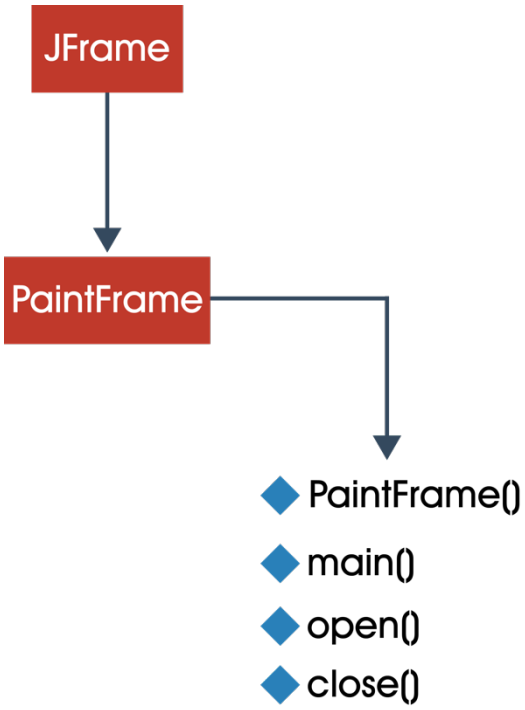


Слика 13: Дијаграм базне класе клијента и сервера

```
public class Machine extends InputOutput {

    private String ip;

    //odredjivanje IP adrese
    protected void setIP(String ip) {
        this.ip = ip;
    }

    //uzimanje IP adrese
    protected String getIP() {
        //provera da li Machine objekat poseduje IP adresu
        if(ip == null) {
            //uzimanje adrese
            getInetAddress();
        }

        //vracanje IP adrese
        return ip;
    }

    //uzimanje adrese
    protected String getInetAddress() {
        try {
            //uzimanje lokalnog hosta
            InetAddress inetAddress = InetAddress.getLocalHost();

            //kreiranje IP adrese na osnovu lokalnog hosta
            String ip = inetAddress.getHostAddress();

            //odredjivanje IP adrese
            setIP(ip);

            //vracanje IP adrese
            return ip;
        }catch(UnknownHostException e) {
            //logovanje greske
            e.printStackTrace();
        }

        //vracanje vrednosti za neuspelu konekciju
        return "undefined";
    }
}
```

PaintFrame.java

Класа главног прозора програма. Наслеђује *JFrame* класу. Атрибуте не поседује. Поред конструктора у коме се налази конфигурација прозора, ова класа поседује *open* и *close* методе који служе за отварање и затварање прозора, као и статични main метод који се извршава оног тренутка када се програм покрене.



Слика 14: Дијаграм главног прозора програма

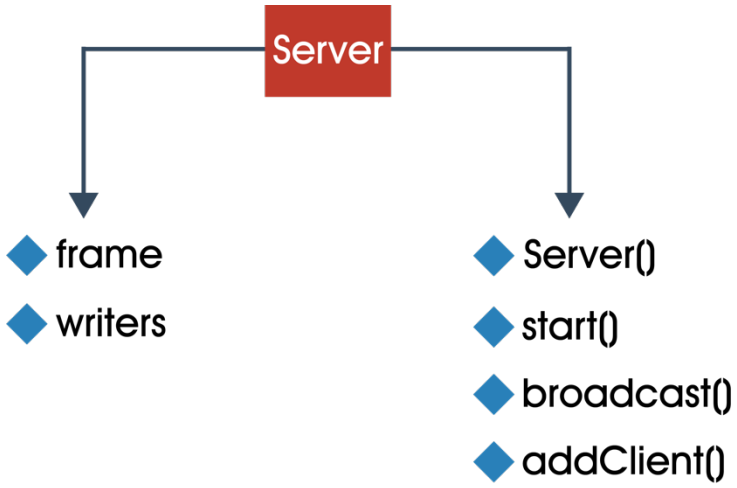
```
public class PaintFrame extends JFrame {

    //konfigurasanje forme
    public PaintFrame() {
        //odredjivanje naziva forme
        setTitle("Paint");

        //zatvaranje forme klikom na 'X'
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        //zabrana promene velicine forme od strane korisnika
        setResizable(false);

        //uzimanje velicine ekrana
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        //deklaracija varijabli za sirinu i visinu dugmica
        int frameWidth = (int) (screenSize.width * 0.25);
        int frameHeight = (int) (screenSize.width * 0.25);

        //odredjivanje sirine i visine forme
        setSize(frameWidth, frameHeight);

        //odredjivanje lokacije forme na ekranu
        setLocation(screenSize.width / 2 - frameWidth / 2,
                    screenSize.height / 2 - frameHeight / 2);

        //kreiranje panela
```

```
        JPanel panel = new JPanel();

        //odredjivanje tipa layouta panela
        panel.setLayout(new BorderLayout());

        //kreiranje labele i odredjivanje njenog teksta
        JLabel lblChoice = new JLabel("Choose mode");

        //pozicioniranje labele
        lblChoice.setHorizontalAlignment(SwingConstants.CENTER);

        //deklaracija varijabli za sirinu i visinu dugmica
        int buttonWidth = frameWidth;
        int buttonHeight = (int) (frameHeight * 0.25);

        //kreiranje dugmeta za prvi izbor
        JButton btnServer = new JButton("Server");

        //odredjivanje sirine i visine dugmica
        btnServer.setPreferredSize(new Dimension(buttonWidth, buttonHeight));

        //rukovodjenje dogadjajem klika
        btnServer.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //otvaranje forme servera i zatvaranje trenutne forme
                ServerFrame.open();
                close();
            }
        });

        //kreiranje dugmeta za drugi izbor
        JButton btnCLient = new JButton("Client");

        //odredjivanje sirine i visine dugmica
        btnCLient.setPreferredSize(new Dimension(buttonWidth, buttonHeight));

        //rukovodjenje dogadjajem klika
        btnCLient.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //otvaranje forme klijenta
                ClientFrame.open();

                //zatvaranje forme
                close();
            }
        });

        //ubacivanje komponenti u panel i njihovo smestanje u odredjene pozicije
        panel.add(lblChoice, BorderLayout.NORTH);
        panel.add(btnServer, BorderLayout.CENTER);
        panel.add(btnCLient, BorderLayout.SOUTH);

        //panel postaje glavni
        setContentPane(panel);

        //menjanje velicine forme
        pack();
    }

    //zatvaranje forme
    private void close() {
        //gubljenje vidljivosti
        setVisible(false);

        //izbacivanje forme
        dispose();
    }
```

```
    //otvaranje forme
    private static void open() {
        //pozivanje konstruktora i dobijanje vidljivosti forme
        new PaintFrame().setVisible(true);
    }

    //pokretanje programa
    public static void main(String[] args) {
        //ptvaranje forme
        open();
    }
}
```

Server.java

Ова класа је слична *Client* класи јер представља сервер на који се клијенти повезују преко својих *Client* објеката. *Machine* класа је наслеђена. Од атрибута примећујемо динамички низ, односно, листу *PrintWriter* објекта који представљају повезане клијенте па служе за слање података тим клијентима и *ServerFrame* објекат који представља прозор на рачунару на коме је покренут сервер. Постоји *start*, *broadcast* и *addClient* метод. Први метод се извршава приликом покретања сервера, други када се од једног клијента добије порука која треба бити прослеђена свим осталим клијентима, а трећи врши додавање клијента које карактерише додавање ИП адресе клијента у *TextArea* компоненту прозора *ServerFrame* класе, а затим додавање *PrintWriter* објекта конструисаног на основу сокета у листу са тим објектима и покретање посебне програмске нити која води рачуна о примању података које клијент шаље.



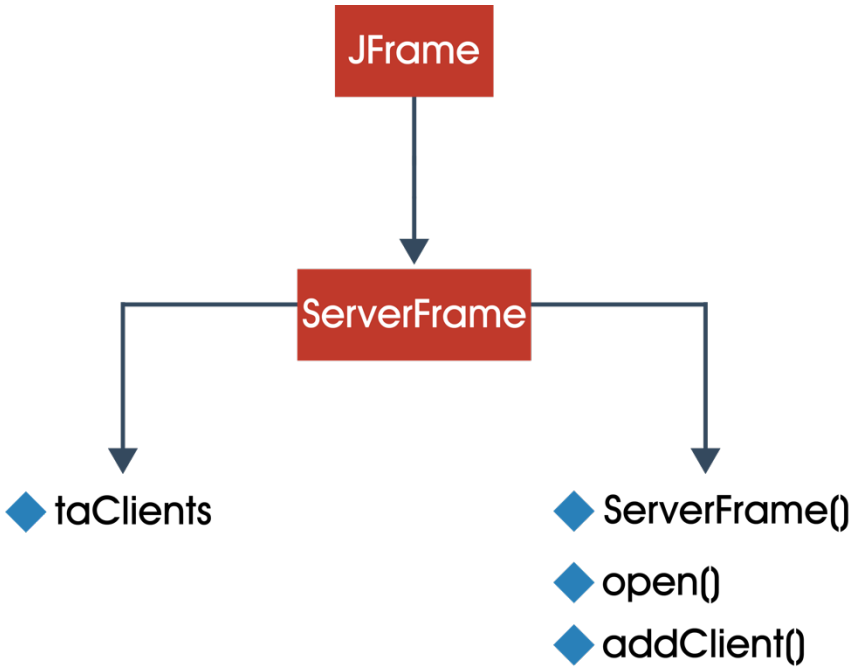
Слика 15: Дијаграм сервера

```
public class Server extends Machine {

    private ServerFrame frame;
    private ArrayList<PrintWriter> writers;

    //odredjivanje inicijalne vrednosti niza PrintWriter objekta
    public Server() {
        writers = new ArrayList<PrintWriter>();
    }

    //pokretanje servera
    public void start(ServerFrame frame) {
        this.frame = frame;

        //odredjivanje Machine objekta
```

```
        setMachine(this);

        try {
            //kreiranje ServerSocket objekta na osnovu porta
            ServerSocket serverSocket = new ServerSocket(2345);

            //kreiranje programske niti koja ce citati linije BufferedReader objekta
            Thread clientListener = new Thread(new ClientHandler(serverSocket, this));

            //startovanje programske niti
            clientListener.start();
        }catch(IOException e) {
            //logovanje greske
            e.printStackTrace();
        }
    }

    //emitovanje linije svim klijentima
    public void broadcast(String line) {
        //kreiranje Iterator objekta za listu PrintWriter objekta
        Iterator i = writers.iterator();

        //listanje kroz listu PrintWriter objekta pomocu Iterator objekta
        while(i.hasNext()) {
            //emotovanje linijje klijentu pomocu trenutnog PrintWriter objekta
            write((PrintWriter) i.next(), line);
        }
    }

    //dodavanje klijenta
    public void addClient(String ip, Socket socket) {
        //dodavanje Ip adrese klijenta
        frame.addIP(ip);

        try {
            //odredjivanje BufferedReader objekta na osnovu soketa
            BufferedReader reader = new BufferedReader(
                        new InputStreamReader(socket.getInputStream()));

            //odredjivanje PrintWriter objekta na osnovu soketa
            PrintWriter writer = new PrintWriter(socket.getOutputStream());

            //dodavanje PrintWriter objekta u listu kojoj pripada
            writers.add(writer);

            //kreiranje programske niti koja ce citati linije BufferedReader objekta
            Thread listener = new Thread(new Listener(this, reader));

            //startovanje programske niti
            listener.start();
        }catch(IOException e) {
            //logovanje greske
            e.printStackTrace();
        }
    }
}
```

ServerFrame.java

Класа која представља прозор сервера. Од атрибута поседује само *TextArea* објекат који служи да кориснику прикаже ИП адресе конектованих клијената. Поред конструктора који поседује логику за подешавање самог прозора, постоји open за отварање прозора *addClient* метод за додавање клијентове ИП адресе у *TextArea* компоненту.



Слика 16: Дијаграм главног прозора сервера

```
public class ServerFrame extends JFrame {

    private JTextArea taClients;

    //konfigurisanje forme
    public ServerFrame() {
        //kreiranje novog Server objekta
        Server server = new Server();

        //pokretanje servera
        server.start(this);

        //odredjivanje naziva forme
        setTitle("Server");

        //zatvaranje forme klikom na 'X'
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        //zabrana promene velicine forme od strane korisnika
        setResizable(false);

        //uzimanje velicine ekrana
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        //deklaracija varijabli za sirinu i visinu dugmica
        int frameWidth = (int) (screenSize.width * 0.25);
        int frameHeight = (int) (screenSize.width * 0.25);

        //odredjivanje sirine i visine forme
        setSize(frameWidth, frameHeight);
```

```
        //odredjivanje lokacije forme na ekranu
        setLocation(screenSize.width / 2 - frameWidth / 2,
                    screenSize.height / 2 - frameHeight / 2);

        //kreiranje panela
        JPanel panel = new JPanel();

        //odredjivanje tipa layouta panela
        panel.setLayout(new BorderLayout());

        //kreiranje labele i odredjivanje njenog teksta
        JLabel lblIP = new JLabel("IP: " + server.getInetAddress());

        //pozicioniranje labele
        lblIP.setHorizontalAlignment(SwingConstants.CENTER);

        //deklaracija varijable za sirinu i visinu povrsine za unos teksta
        int textAreaWidth = frameWidth;
        int textAreaHeight = (int) (frameHeight * 0.5);

        //kreiranje povrsine za unos teksta
        taClients = new JTextArea();

        //odredjivanje velicine povrsine za unos teksta
        taClients.setPreferredSize(new Dimension(textAreaWidth, textAreaHeight));

        //zabranje menjanja sadrzaja povrsine za unos teksta od strane korisnika
        taClients.setEditable(false);

        //ubacivanje komponenti u panel i njihovo smestanje u odredjene pozicije
        panel.add(lblIP, BorderLayout.NORTH);
        panel.add(taClients, BorderLayout.SOUTH);

        //panel postaje glavni
        setContentPane(panel);

        //menjanje velicine forme
        pack();
    }

    //otvaranje forme
    public static void open() {
        //pozivanje konstruktora i dobijanje vidljivosti forme
        new ServerFrame().setVisible(true);
    }

    //dodavanje klijenta
    public void addIP(String ip) {
        //ubacivanje IP adrese klijenta u povrsinu za
         unos teksta i premestanje kursora u novi red
        taClients.append(ip + "\n");
    }
}
```

StringMaker.java

Интерфејс који садржи метод за одређивање објекта *StringBuilder* за формирање *String* објекта. Имплементиран је у *InputOutput* класи.

```
public interface StringMaker {

    //odredjivanje StringBuilder objekta
    void setStringBuilder(StringBuilder builder);
}
```

Закључак

Овај пројекат је једноставан пример везе клијент-сервер која представља основу за сложеније мрежно програмирање и на којој се базира већина модерних програма јер код савременог човека повезаност игра кључну улогу у свакодневном животу.

Коришћени пакети и класе

javax.swing.*
java.awt.*
java.awt.event.*

java.io.InputStreamReader
java.io.BufferedReader
java.io.PrintWriter
java.io.IOException

java.util.ArrayList
java.util.Iterator

java.awt.geom.Rectangle2D
java.awt.event.MouseMotionAdapter
java.awt.event.MouseEvent
java.awt.event.ActionListener
java.awt.event.ActionEvent

java.net.InetAddress
java.net.InetSocketAddress
java.net.Socket
java.net.ServerSocket
java.net.UnknownHostException

Изворни кôд

Комплетан пројекат се може наћи на
https://github.com/jelic98/paint/releases

Литература

[1] Cay S. Horstmann & Gary Cornell, Core Java™ 2 Volume II - Advanced Features, Seventh Edition, Sun Microsistems, 2005.

[2] Дејан Живковић, Јава програмирање, Треће издање, Универзитет Сингидунум, Београд, 2013.

[3] http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/KomunikMreze.pdf

[4] https://sh.wikipedia.org/wiki/Objektno-orijentisano_programiranje

[5] http://www.vps.ns.ac.rs/nastavnici/Materijal/mat63.pdf

[6] Ian F. Darwin, Java Cookbook - Solutions and Examples for Java Developers, 3rd Edition, O'Reilly Media, 2014.