ДОДАТОК А
Текст програми

```
package devices;
import javax.swing.*; import java.awt.*; import java.awt.event.*;
import java.io.*; import java.util.ArrayList; import java.util.HashMap;

public class WorkPanel extends JPanel implements Serializable{
        private static final long serialVersionUID= 1L;
        JFrame jfrm; TermWind termWind; static ObjectOutputStream objOStrm; static ObjectInputStream objIStrm;
        JButton choiseButtons[]=new JButton[4], clusterChoise[]=new JButton[8], cloud, relocateLabels;
        ArrayList<JButton> unvisButtons=new ArrayList<>(), routerSet=new ArrayList<>();
        ArrayList<JPopupMenu> menus=new ArrayList<>(); ArrayList<Server> allServers=new ArrayList<>();
        ArrayList<Switch> allSwitches=new ArrayList<>(); ArrayList<ClusterPanel> allClusters=new ArrayList<>();
        ArrayList<Router> allRouters=new ArrayList<>(); JRadioButton choiseRadioButtons[]=new JRadioButton[7];
        ButtonGroup AllGroups []=new ButtonGroup[5]; ConnectLine links[];
        String RBtypes[]= {"PC","LT","MP","Server","Router","Switch","Link"}, typemode="0";
        ArrayList<JLabel> labelsOnScreen= new ArrayList<>(), textlabels= new ArrayList<>();
        int linkLocations[][], labelsCount=0, dobCount=0, lineCount=0, curElem=0, prevElem=0,
                        chsdPort[]=new int[2], iterPortPrev=0, prevPanelNmb=-1, crntPanelNmb=-1;
        JButton doubleConnecter[]= {null, null}; ArrayList<Element> DObs=new ArrayList<>(); ImageIcon packimg;
PacketTest testpkt;
        boolean linkmode=false, connected=false, popupTrig=false, mkbtn=false, termwindActive=false;
        HashMap<String, Object> srlzed = new HashMap<>(), dsrlzed = new HashMap<>();
        String[] srlzKeys= {"testpkt", "packimg" , "unvisButtons", "menus", "links", "labelsOnScreen",
"textlabels", "linkLocations", "dobCount", "lineCount",
                        "curElem", "prevElem", "chsdPort", "DObs", "typemode", "connected" };
          Object[] srlzVals= {testpkt, packimg, unvisButtons, menus, links, labelsOnScreen, textlabels,
linkLocations, dobCount, lineCount, curElem, prevElem, chsdPort, DObs, typemode, connected };

        WorkPanel(boolean mainWind){
                if (objOStrm==null) if (objIStrm==null)
                try {objOStrm  =new ObjectOutputStream (new FileOutputStream ( "D:\\java-
workspace\\Kursovoi\\serial.txt " ));
                        objIStrm =new ObjectInputStream (new FileInputStream ( "D:\\java-
workspace\\Kursovoi\\serial.txt " ) );
                        } catch (FileNotFoundException e) {e.printStackTrace();
                        } catch (IOException e1) {e1.printStackTrace();}; setLayout(null);
    jfrm=new JFrame ("NGN Network Scheme"); jfrm.setSize (1200, 720);
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    termWind=new TermWind("Terminal", this); links=new ConnectLine[200];
    JButton chooseED = choiseButtons[0] =new JButton("End Devices"), chooseSwitch = choiseButtons[1] = new
JButton("Switches"),
    chooseRouter = choiseButtons[2]= new JButton("Routers"), chooseConnect = choiseButtons[3] = new
JButton("Connections");
    packimg=new ImageIcon ( "D:\\imagesForWork\\Datagram.png"); relocateLabels=new JButton(); Route.wp=this;
    linkLocations= new int[200][]; for (int j = 0; j < linkLocations.length; j++) linkLocations[j]=new int[4];
    JRadioButton PC = choiseRadioButtons[0]= new JRadioButton ( "PC" ) ;
    JRadioButton LT = choiseRadioButtons[1]=new JRadioButton ( "Laptop" ) ;
    JRadioButton  MP = choiseRadioButtons[2]=new JRadioButton ( "MobilePhone" ) ;
    JRadioButton ServerB =  choiseRadioButtons[3]=new JRadioButton ( "Server" ) ;
    JRadioButton RouterB =  choiseRadioButtons[4]=new JRadioButton ( "Router" ) ;
    JRadioButton SwitchB =  choiseRadioButtons[5]=new JRadioButton ( "Switch" ) ;
    JRadioButton CopperStraightB =  choiseRadioButtons[6]=new JRadioButton ( "CopperStraight" ) ;
    JLabel PC_lb= new JLabel(""), LT_lb= new JLabel(""), MP_lb= new JLabel(""), Server_lb= new JLabel(""),
    Router_lb= new JLabel(""), Switch_lb= new JLabel(""), Copper_lb= new JLabel("");
    ButtonGroup ChoiseGroup = AllGroups[0]=  new ButtonGroup ( ), EDgroup  =  AllGroups[1]= new ButtonGroup () ,
    SwitchGroup  = AllGroups[2] = new ButtonGroup ( ), RouterGroup = AllGroups[3]=   new ButtonGroup ( ),
    ConnectGroup  = AllGroups[4]= new ButtonGroup ( ) ;
    ChoiseGroup.add(chooseSwitch); ChoiseGroup.add(chooseRouter); ChoiseGroup.add(chooseConnect);
ChoiseGroup.add(chooseED);
    EDgroup.add(PC); EDgroup.add(LT); EDgroup.add(MP); EDgroup.add(ServerB);
    RouterGroup.add(RouterB); SwitchGroup.add(SwitchB); ConnectGroup.add(CopperStraightB);
         setLocs(chooseED, 120, 30, 5, 613, -1);  setLocs(chooseRouter, 120, 30, 127, 580, -1);
         setLocs(chooseSwitch, 120, 30, 5, 580, -1); setLocs(chooseConnect, 120, 30, 249, 580, -1);
         forButton(PC, PersonalComputer.icon, PC_lb, 64, 25, 405, 580, 60, 50, 405, 608, -1);
         forButton(LT, Laptop.icon,  LT_lb, 78, 25, 485, 580, 60, 50, 490, 608, -1);
         forButton(MP, MobilePhone.icon, MP_lb, 98, 25, 645, 580, 60, 50, 650, 608, -1);
         forButton(ServerB, Server.icon, Server_lb, 78, 25, 565, 580, 60, 50, 570, 608, -1);
         forButton(SwitchB, Switch.icon, Switch_lb, 64, 25, 405, 580, 60, 50, 405, 608, -1);
         forButton(RouterB, Router.icon, Router_lb, 64, 25, 405, 580, 60, 50, 405, 608, -1);
         forButton(CopperStraightB, CopperStraight.icon, Copper_lb, 120, 25, 405, 580, 60, 50, 405, 608, -1);
         JComponent EDcollection[]= {PC, LT, MP, ServerB, PC_lb, LT_lb, MP_lb, Server_lb};
```

```
JComponent SwitchCollection[]= {SwitchB, Switch_lb}, RouterCollection[]= {RouterB, Router_lb};
LinkCollection[]= {CopperStraightB, Copper_lb};
         JComponent AllCollections[][]= {EDcollection, SwitchCollection, RouterCollection, LinkCollection};

         visibleOne(null,AllCollections);
        for(int i=0; i<7;i++)  createRadioListener(choiseRadioButtons[i], RBtypes[i]);
        for(int i=0; i<4;i++)  createChoiseListener(choiseButtons[i], AllCollections[i], AllCollections);

        jfrm.addMouseListener (new MouseAdapter ( ) { public void mouseReleased (MouseEvent me)  {

                if (typemode!="Link") if((me . getX ()<1000)&&(me . getY ()<570)) createDOB(me.getX(),
me.getY(), -1, 0);}}) ;
labelsCount=0; for (int i=0; i<7; i++)
{clusterChoise[i]= new JButton(((Integer)i).toString());
clusterChoise[i].setSize(50,50); clusterChoise[i].setLocation(1000,50+70*i);
int j=i; clusterChoise[i].addActionListener (new ActionListener() {
        public void actionPerformed (ActionEvent  ae) { setVisible(false);
allClusters.get(j).temp.setVisible(true);}});}
cloud=new JButton(new ImageIcon("D:\\imagesForWork\\cloud0.png")); cloud.setSize(200,120);
cloud.setLocation(475, 20);
cloud.addActionListener (new ActionListener() {  public void actionPerformed (ActionEvent  ae) {
                 if (crntPanelNmb==-1)  setVisible(false);
         else  allClusters.get(crntPanelNmb).temp.setVisible(false);}}); add(cloud);
   RoundLabel[] rlbs=new RoundLabel[3]; JLabel[] lbs=new JLabel[3];
   for(int i=0; i<3; i++)
  {rlbs[i]=new RoundLabel(String.valueOf(i+1), 0); lbs[i]=new JLabel(String.valueOf(i+1));
   rlbs[i].setSize(55, 55); lbs[i].setSize(30, 30); add( lbs[i]); add(rlbs[i]);}
   rlbs[1].setLocation(200, 80); rlbs[2].setLocation(898, 80); rlbs[0].setLocation(548, 320);
    lbs[1].setLocation(220, 90); lbs[2].setLocation(920, 90); lbs[0].setLocation(570, 330);
   JLabel r1lb=new JLabel("Router 1.0"); r1lb.setLocation(610, 250); r1lb.setSize(70, 20); add(r1lb);
   JLabel r2lb=new JLabel("Router 2.0"); r2lb.setLocation(358, 120); r2lb.setSize(70, 20); add(r2lb);
   JLabel r3lb=new JLabel("Router 3.0"); r3lb.setLocation(730, 120); r3lb.setSize(70, 20); add(r3lb);
   DiamondLabel dlb=new DiamondLabel("round", new int[] {100, 150, 200, 150, 100},
                   new int[] {200, 270, 200, 130, 200}, 5, Color.yellow, 1);
   dlb.setSize(70, 35); dlb.setLocation(80, 50);  //add(dlb);
   allClusters.add(new ClusterPanel(0, false, 0, 14, 50, 2, 7, 4, 13));
   allClusters.add(new ClusterPanel(1, false, 67, 26, 102, 2, 13, 6, 17));
   allClusters.add(new ClusterPanel(2, false, 198, 51, 205, 2, 26, 8, 26)); //131
    for (int i=0; i<3; i++) allClusters.get(i).init();  setVisible(true); jfrm.setVisible(true);
    createRouterSet(); repaint(); jfrm.add(this); repaint(); setVisible(true);}

        public static void main ( String args [ ] )  {
                SwingUtilities . invokeLater (new Runnable ( ) {
                public void run ( )  { new WorkPanel(true);}});}

        void visibleOne(JComponent jc[], JComponent jcc[][])
        {       for(JComponent[] obm:jcc) { for(JComponent ob:obm) {ob.setVisible(false);}}
        if (jc!=null) for(JComponent ob:jc) ob.setVisible(true);}

void attachLabel(int x, int y, int panelNmb)
{setLocs(textlabels.get(dobCount), 70, 40, x, y+40, panelNmb);
setLocs(labelsOnScreen.get(dobCount), 70, 70, x, y, panelNmb);
unvisButtons.get(dobCount).setLocation(x, y);}

void createDOB(int x, int y, int panelNumb, int nmbInClstr)
{if (typemode.equals("PC")) DObs.add(new PersonalComputer("PC ", panelNumb, nmbInClstr));
        else if (typemode.equals("LT")) DObs.add(new Laptop("LT ", panelNumb, nmbInClstr));
        else if (typemode.equals("MP")) DObs.add(new MobilePhone("IP-phone ", panelNumb, nmbInClstr));
        else if (typemode.equals("Router"))     {DObs.add(new Router("Router ", panelNumb, nmbInClstr));
                allRouters.add((Router) DObs.get(dobCount));}
        else if (typemode.equals("Switch")) {DObs.add(new Switch("Switch ", panelNumb, nmbInClstr));
                allSwitches.add((Switch) DObs.get(dobCount));}
        else if (typemode.equals("Server"))     {DObs.add(new Server("Server ", panelNumb, nmbInClstr));
                allServers.add((Server) DObs.get(dobCount));} else return;
        createBtn(dobCount, panelNumb, true); textlabels.add(new JLabel(DObs.get(dobCount).name));
        labelsOnScreen.add(new JLabel());
        DObs.get(dobCount).setAll(textlabels.get(dobCount), labelsOnScreen.get(dobCount),
unvisButtons.get(dobCount));
DObs.get(dobCount).IPaddress="1.1.1."+dobCount;
DObs.get(dobCount).networkIP=DObs.get(dobCount).getNetworkIP(); attachLabel(x, y, panelNumb);
DObs.get(dobCount).locX=unvisButtons.get(dobCount).getX();
DObs.get(dobCount).locY=unvisButtons.get(dobCount).getY();
if (DObs.get(dobCount).type.equals("enddevice")) termWind.N++; DObs.get(dobCount).totalNumber=dobCount++;}

void createLink(int n0, int n1, int p0, int p1, int panelNmb, int bus)
{boolean inCls=panelNmb!=-1; ArrayList<JButton> btnSet=inCls?unvisButtons:routerSet;
int[] locs=(bus==-1)? new int[] {btnSet.get(inCls?n0:n0/67).getX(),btnSet.get(inCls?n0:n0/67).getY(),
```

```
          btnSet.get(inCls?n1:n1/67).getX(),btnSet.get(inCls?n1:n1/67).getY()} :
          new int[] {btnSet.get(n0).getX(), btnSet.get(n0).getY(), btnSet.get(n0).getX(),
allClusters.get(panelNmb).buses[bus].locs[1],
          btnSet.get(n1).getX(), allClusters.get(panelNmb).buses[bus].locs[1], btnSet.get(n1).getX(),
btnSet.get(n1).getY()};
Element d0=DObs.get(n0), d1=DObs.get(n1);
d0.ports[p0].active=d0.activeports[p0]=d1.ports[p1].active=d1.activeports[p1]=true;
d0.acsdports[p0]=d1.acsdports[p1]=false;
if (panelNmb==-1) links [lineCount]=new ConnectLine(d0.ports[p0], d1.ports[p1], d0, d1, locs, bus );
else    allClusters.get(panelNmb).links[allClusters.get(panelNmb).lineCount]=new
ConnectLine(d0.ports[p0],d1.ports[p1], d0,d1,locs, bus );
if(d0.type.equals("switch")) if(d1.type.equals("enddevice"))((Switch)d0).elemsInNet.add(d1); else
((Switch)d0).hopsOverNet.add(d1);
if(d1.type.equals("switch")) if(d0.type.equals("enddevice")) ((Switch)d1).elemsInNet.add(d0); else
((Switch)d1).hopsOverNet.add(d0);
if(d0.type.equals("server")) if(d1.type.equals("enddevice")) ((Server)d0).elemsInNet.add(d1); else
((Server)d0).hopsOverNet.add(d1);
if(d1.type.equals("server")) if(d0.type.equals("enddevice")) ((Server)d1).elemsInNet.add(d0); else
((Server)d1).hopsOverNet.add(d0);
        if (panelNmb==-1) {d0.links[d0.linkcount++]=this.links[lineCount];
d1.links[d1.linkcount++]=this.links[lineCount]; linkLocations [lineCount++]=locs; repaint(); }
else {d0.links[d0.linkcount++]=allClusters.get(panelNmb).links[allClusters.get(panelNmb).lineCount];
d1.links[d1.linkcount++]=allClusters.get(panelNmb).links[allClusters.get(panelNmb).lineCount];
allClusters.get(panelNmb).linkLocations [allClusters.get(panelNmb).lineCount++]=locs;
allClusters.get(panelNmb).repaint(); }}

void uncheckAllRadio() { for(ButtonGroup bg:AllGroups) bg.clearSelection();}

void createBtn(int nmb, int panelNmb, boolean inCluster)
{JButton jb=new JButton(""); if(inCluster) unvisButtons.add(jb); else  routerSet.add(jb);
jb.setVisible(false); setLocs(jb, 50, 50, 0,0, panelNmb); final int currentNumber=nmb;
ArrayList<JMenuItem> jmi=new ArrayList<>(); JPopupMenu portMenu  =  new JPopupMenu ( ) ;
for( int iterPort=0; iterPort<(DObs.get(nmb)).portsNumber; iterPort++)
{final int iport =iterPort; jmi.add(new JMenuItem ( "port "+iterPort ));
jmi.get(iterPort). addActionListener ( new ActionListener(){
        public void actionPerformed(ActionEvent e) { curElem=currentNumber;
        if (connected) {curElem=currentNumber; chsdPort[0]=iterPortPrev; chsdPort[1]=iport;
                createLink(prevElem ,curElem, iterPortPrev, iport, panelNmb, -1);
                        switch (DObs.get(prevElem).type)
        {case "enddevice":      DObs.get(prevElem).ports[iterPortPrev].IPaddress=DObs.get(prevElem).IPaddress;
break;
        case "switch": case "router": DObs.get(prevElem).ports[iterPortPrev].IPaddress="1.1.1."+(prevElem+100);}
        switch (DObs.get(curElem).type)
        {case "enddevice": DObs.get(curElem).ports[iport].IPaddress=DObs.get(currentNumber).IPaddress;   break;
        case "switch": case "router": DObs.get(curElem).ports[iport].IPaddress="1.1.1."+(currentNumber+100);}
connected=false;}
        else {connected=true; iterPortPrev=iport; prevElem=curElem;}}} );   portMenu.add(jmi.get(iterPort));}
menus.add(portMenu); jb.addActionListener (new ActionListener() {
                public void actionPerformed (ActionEvent  ae) {
                        if (typemode=="Link")
                          {if(DObs.get(nmb).ports[chsdPort[0]].active) jmi.get(chsdPort[0]).setVisible(false);
                                if(DObs.get(nmb).ports[chsdPort[1]].active)
jmi.get(chsdPort[1]).setVisible(false);
                                portMenu.show (jfrm, jb.getX(), jb.getY()) ;}
                                else { if (!termwindActive) {termwindActive=true;
termWind.setTitle(DObs.get(nmb).name);
                                termWind.crntnumber=nmb; termWind.crntDevice=DObs.get(nmb);
termWind.setVisible(true);
                                termWind.tabProc((JPanel) termWind.tabbedPane.getSelectedComponent());}}}});
jb.addMouseMotionListener (new MouseMotionListener() { public void mouseDragged(MouseEvent e) {}
@Override public void mouseMoved(MouseEvent mm) {}});}

void createRadioListener(JRadioButton b, String type)
{b.addActionListener (new ActionListener() {
        public void actionPerformed (ActionEvent  ae) {typemode=type; }});}

void createChoiseListener(JButton b, JComponent[]group, JComponent[][] all)
{b.addActionListener (new ActionListener() {
        public void actionPerformed (ActionEvent  ae) {visibleOne(group, all); uncheckAllRadio(); typemode="0";
}});}

void forButton(JComponent rb, ImageIcon ii, JLabel jl, int setSx, int setSy, int setLx, int setLy, int setSxL,
int setSyL, int setLxL, int setLyL, int panelNmb)
{setLocs(rb, setSx, setSy, setLx, setLy,  panelNmb); jl.setIcon(ii); jl.setSize(setSxL, setSyL);
jl.setLocation(setLxL, setLyL); jl.setVisible(false); rb.setVisible(false); add(jl);}

void setLocs(JComponent rb, int setSx, int setSy, int setLx, int setLy, int panelNmb)
```

```
{rb.setLocation(setLx, setLy); rb.setSize(setSx, setSy);
if (panelNmb==-1) add(rb); else allClusters.get(panelNmb).add(rb);}

int findInMas(Object ob, Object[] mas)
{int i=0; for (Object ob1:mas) {if (ob1==ob) return i; i++;}
return -1;}

 void ping(Element e0, String ipadr)
{Element e=new Element(""), e1 = null;
if (e0.IPaddress.equals(ipadr)) System.out.println("Самоадрессация");
else {for(int i=0; i<dobCount; i++)
  { e=DObs.get(i); if (e!=null) if (e.IPaddress.equals(ipadr)) {e1=e; i=100;}}
        if (e1!=null) {ConnectLine cn=e0.findLink(e1);
        if (cn!=null) {PacketTest testpkt= new PacketTest("PacketTest ", packimg, 50, null, 0.08f, 30, 20, -1, -
1);
                add(testpkt.jl); e0.send(new Packet(50), e1, cn,testpkt, true, false);
        System.out.println("Успешный ответ от "+e1.name);} else System.out.println("Устройство не отвечает");}
        else System.out.println("Адрес не найден"); }}

  Element getByIP(String adr)
 {for (Element e0:DObs) if (e0.IPaddress.equals(adr)) return e0;
   return null;}

  void srlz(JFrame jfr) {for (int i=0; i<srlzKeys.length; i++) srlzed.put(srlzKeys[i], srlzVals[i]);
         try { objOStrm.writeObject (srlzed ) ;} catch (IOException e) { e.printStackTrace();}}

         @SuppressWarnings("unchecked")
        void dsrlz(WorkPanel wplast) { jfrm.setVisible(false); WorkPanel wp=new WorkPanel(true);

                 try {Object obj=objIStrm. readObject ( ) ; dsrlzed=(HashMap<String, Object>) obj;
for (int i=0; i<srlzKeys.length; i++) srlzVals[i]=dsrlzed.get(srlzKeys[i]);
 wp.testpkt =(PacketTest) srlzVals[0];  wp.packimg=(ImageIcon) srlzVals[1]; wp.unvisButtons=(ArrayList<JButton>)
srlzVals[2];
 wp.menus=(ArrayList<JPopupMenu>) srlzVals[3]; if(srlzVals[4]!=null) wp.links=(ConnectLine[]) srlzVals[4];
wp.labelsOnScreen=(ArrayList<JLabel>) srlzVals[5];
 wp.textlabels=(ArrayList<JLabel>) srlzVals[6]; if(srlzVals[7]!=null) wp.linkLocations=(int[][]) srlzVals[7];
wp.dobCount=(int) srlzVals[8];
 wp.lineCount=(int) srlzVals[9]; wp.curElem=(int) srlzVals[10]; wp.prevElem=(int) srlzVals[11];
wp.chsdPort=(int[]) srlzVals[12];
 wp.DObs=(ArrayList<Element>) srlzVals[13]; wp.typemode=(String) srlzVals[14]; wp.connected=(boolean)
srlzVals[15] ;
 for(JButton unv:unvisButtons)  wp.add(unv);
 for(JPopupMenu jpm:menus)   wp.add(jpm);
 wp.jfrm.setVisible(true); wplast=wp;}
                 catch (ClassNotFoundException e4) {e4.printStackTrace();}
                catch (EOFException e2) {e2.printStackTrace();}
         catch (IOException e2) { e2.printStackTrace();} }

         static void setFrame(WorkPanel wp0, JFrame jf, boolean crt) {wp0=new WorkPanel(true); }

         void createNetwork(int panelNmb, int startCnt, int pcCnt, int phoneCnt, int hrzPh)
         {      int startCount=startCnt, pcCount=pcCnt, phoneCount=phoneCnt, horizPh=hrzPh;
                 typemode="Router";     createDOB(130, 170, panelNmb, 0);
                 typemode="Switch"; createDOB(10, 170, panelNmb, 0);
                 typemode="Server"; createDOB(250, 170, panelNmb, 0);
                 createLink(startCount ,startCount+2, 0, 21, panelNmb, -1);
                 createLink(startCount ,startCount+1, 1, 18, panelNmb, -1);
                 if (panelNmb>0)
                 allClusters.get(panelNmb).buses[2].setHead(DObs.get(startCount+1));
                 allClusters.get(panelNmb).buses[1].setHead(DObs.get(startCount+1));
                 allClusters.get(panelNmb).buses[0].setHead(DObs.get(startCount+2));
        typemode="PC"; for (int i=310, j=startCount+3, y=10, n=0; j<startCount+pcCount+3; i+=100, j++, n++)
                 {createDOB(i, y, panelNmb, n); if(j==startCount+pcCount/2+3) {i=260; y=130;}
                 createLink(j, startCount+2, 0, 21, panelNmb, 0);
allClusters.get(panelNmb).buses[0].addMember(DObs.get(j)); }
                 typemode="MP"; for (int i=110, j=startCount+pcCount+3, k=1, y=280, n=0;
j<startCount+pcCount+phoneCount+3; i+=100, j++, n++ )
                          {createDOB(i, y, panelNmb, n); if(j==startCount+pcCount+horizPh*k+3)
{i=(k%2!=0?70:110); y+=((k==2)||(k==6)?150:70); k++;}
                          createLink(j, startCount+1, 0, 20, panelNmb, k<5?1:2);
                            allClusters.get(panelNmb).buses[k<5?1:2].addMember(DObs.get(j));}
                 allClusters.get(panelNmb).repaint(); typemode="0"; dynamicAddress();}

          void choosePanel(int nmb) {prevPanelNmb=crntPanelNmb;
                 if (crntPanelNmb==-1) setVisible(false); else
allClusters.get(crntPanelNmb).temp.setVisible(false);
         if (nmb==-1) setVisible(true); else allClusters.get(nmb).temp.setVisible(true);
```

```
        crntPanelNmb=nmb; relocateLabels.doClick();}

        void dynamicAddress() {int swAdr=1, edAdr=1;
                for(Switch sw:allSwitches) {for(Element ed:sw.elemsInNet)
                { ed.IPaddress="1.1."+(swAdr)+"."+edAdr++; ed.IPmask="255.255.255.0"; ed.createBinars();}
                sw.IPaddress="1.1."+(swAdr++)+".0"; sw.IPmask="255.255.255.0"; sw.createBinars(); edAdr=1;}
                for(Server sv :allServers) {for(Element ed:sv.elemsInNet)
                { ed.IPaddress="1.1."+(swAdr)+"."+edAdr++; ed.IPmask="255.255.255.0"; ed.createBinars();}
                sv.IPaddress="1.1."+(swAdr++)+".0"; sv.IPmask="255.255.255.0"; sv.createBinars(); edAdr=1;}}

        void createRouterSet() {JButton jb; typemode="Link"; int ptsX[]= {300, 800, 550}, ptsY[]= {120, 120,
250}, rtrNmbs[]= {0, 67, 198};
               for (int i=0; i<3; i++) {createBtn(rtrNmbs[i], -1, false); jb=routerSet.get(i);
jb.setLocation(ptsX[i], ptsY[i]);
                        jb.setIcon(Router.icon); jb.setVisible(true);}
                createLink(0, 198, 0, 1, -1, -1) ; createLink(67, 198, 0, 1, -1, -1) ; typemode="0";
RoundButton btn256=new RoundButton("", 1), btn64=new RoundButton("", 2), btn128=new RoundButton("", 3);
RoundButton rbs[]= { btn64, btn128, btn256};
                        for (int k=0; k<3; k++) {int ik=k; rbs[k].addActionListener (new ActionListener() {
                                public void actionPerformed (ActionEvent  ae) { setVisible(false);
choosePanel(ik);  }}); add(rbs[k]);}
                        btn256.setSize(400, 400); btn256.setLocation(370, 270); btn64.setSize(250, 250);
                        btn64.setLocation(100, 50); btn128.setSize(250, 250);btn128.setLocation(800, 50); }

        class ClusterPanel extends JPanel
        { private static final long serialVersionUID = 1L;
        int clusterNmb, linkLocations[][], lineCount=0, startNmb, pcNmb, phoneNmb, horizPh;
        ConnectLine links[]; DataBus buses[]; JButton cloud; JScrollPane jsp; JPanel temp;
ClusterPanel(int nmb, boolean iniT, int startN, int pcN, int phoneN, int vrtPC, int hrzPC, int vrtPh, int hrzPh)
{clusterNmb=nmb; links=new ConnectLine[1000];  for (int i=0; i<1000; i++) links[i]=new ConnectLine();
                buses=new DataBus[nmb==0?2:3]; startNmb=startN; pcNmb=pcN; phoneNmb=phoneN; horizPh=hrzPh;
if(iniT) init();}

                void init() {setSize(3000, 700); setLocation(20, 20); setVisible(true); setLayout(null);
          if (buses.length>2)
           buses[2]=new DataBus(2, new int[]{ 50, 820, 1200, 820}, clusterNmb);
           buses[1]=new DataBus(1, new int[]{ 50, 450, 1200, 450}, clusterNmb);
           buses[0]=new DataBus(0, new int[]{300, 100,  900, 100}, clusterNmb);
                        temp= new JPanel(); temp.setSize(1200, 700); temp.setLocation(0, 0);
temp.setVisible(false); temp.setLayout(null);
                        linkLocations= new int[1000][]; for (int j = 0; j < linkLocations.length; j++)
linkLocations[j]=new int[4];
               createNetwork(clusterNmb, startNmb, pcNmb, phoneNmb,horizPh);
               cloud=new JButton(new ImageIcon("D:\\imagesForWork\\cloud0.png"));
cloud.setSize(200,120);cloud.setLocation(10, 10);
                        cloud.addActionListener (new ActionListener() { public void actionPerformed (ActionEvent
ae) { choosePanel(-1);}});
                        add(cloud); Dimension dm=new Dimension(3000,1050); setPreferredSize(dm);
                        jsp=new JScrollPane(this); jsp.revalidate(); jsp.setVisible(true); jsp.setSize(1187,
680);jsp.setBorder(null);
                        temp.add(jsp, BorderLayout.CENTER); jsp.revalidate(); jfrm.add(temp);}

                protected void paintComponent(Graphics g) {super.paintComponent(g);
                for(int i=0; i<lineCount; i++) {if(links[i].busNmb==-1)
                         g.drawLine(linkLocations[i][0]+20, linkLocations[i][1]+20, linkLocations[i][2]+20,
linkLocations[i][3]+20);}
                for (int i=0; i<buses.length; i++) {g.drawLine(buses[i].locs[0], buses[i].locs[1],
buses[i].locs[2], buses[i].locs[3]);
                for (Element mmbr:buses[i].members)  g.drawLine(mmbr.locX+25, mmbr.locY, mmbr.locX+25,
buses[i].locs[3]);}}}

protected void paintComponent(Graphics g) {super.paintComponent(g);
for(int i=0; i<lineCount; i++) {g.drawLine(linkLocations[i][0]+20, linkLocations[i][1]+20,
linkLocations[i][2]+20, linkLocations[i][3]+20);}}}


package devices;
import java.io.Serializable; import java.util.ArrayList;

public class Route extends Thread implements Serializable {
private static final long serialVersionUID = 1L; ArrayList<Element> elemchain=new ArrayList<>();
ArrayList <ConnectLine> linkchain=new ArrayList<>(), otherWays=new ArrayList<>();
ArrayList<Route> activeRoutes; ArrayList<Element> finalElemRoute; Element astray, src, dst, nexthop;
static WorkPanel wp; Packet crntPacket; int elemcount=0, byteAmount=0, pktAmount=0, multicastcount=0;
String dstAdr, curdstAdr; CertainRoute route; Route toBeIntrptd;
boolean impasse=false, exitCycle=false, transit=false, isSuccessFinish=false;
```

```
Route(boolean init){if (init) {activeRoutes=new ArrayList<>();
        finalElemRoute=new ArrayList<>(); astray=new Element();}};

Route(Element s,  String ip1, Packet p)
{this(true);src=s; dstAdr=ip1; crntPacket=p;};

Route transmitChain(boolean multipkt, int number, boolean duplex)
{route=new CertainRoute(new ArrayList<>(), false,this);
route.orgnsr=this; route=route.transmit(src, dstAdr, crntPacket); String finalRoute="";
for (Element e:finalElemRoute) finalRoute += e.name+" "; System.out.println (finalRoute);
if (multipkt) {for(int i=0; i<number; i++) {send(crntPacket, duplex, -1);
                try {sLeep((long) (1000/NetworkSimulator.Vcadr));}
                catch (InterruptedException e) {e.printStackTrace();}}}
else System.out.println("Передача пакета завершилась "+(send(crntPacket, false, -1)?"успехом":"неудачей"));
return (Route)route;}

public class CertainRoute extends Route{
        private static final long serialVersionUID = 1L; Route orgnsr;

CertainRoute(ArrayList<Element> echain, boolean scs, Route org)
 {super(false); elemchain=echain; isSuccessFinish=scs;  orgnsr=org;};

CertainRoute transmit(Element s,  String ip1, Packet p) {
nexthop=s; dstAdr=ip1; elemchain.add(nexthop); orgnsr.elemcount++;  crntPacket=p; start();
        if (orgnsr.elemcount==1) {String finalRoute=""; for (Element e:elemchain) finalRoute += e.name+" ";
        System.out.println (finalRoute);} boolean avoid=false;
        if (deviceConsumingRmn()) { for (ConnectLine cl:otherWays){ if (orgnsr.transit) break; avoid=false;
        if (orgnsr.impasse) {elemchain=orgnsr.toBeIntrptd.elemchain; orgnsr.impasse=false;}
        for(Element e:elemchain) if (e.name.equals(cl.adjacentElem(nexthop).name)) { avoid=true;}
        if(!avoid) { CertainRoute rt=new CertainRoute(elemchain, isSuccessFinish,orgnsr);
        {orgnsr.toBeIntrptd=rt; orgnsr. astray=nexthop;}
        CertainRoute rt0=rt.transmit(cl.adjacentElem(nexthop),  dstAdr, crntPacket) ;
        try { rt.join();} catch (InterruptedException e1) {e1.printStackTrace();}
        if (orgnsr.toBeIntrptd!=null) if(orgnsr.impasse)
                {{while(orgnsr.toBeIntrptd.elemchain.get(elemchain.size()-1)!=orgnsr.astray)
                orgnsr.toBeIntrptd.elemchain.remove(elemchain.get(elemchain.size()-1));}
                if(orgnsr.toBeIntrptd.isInterrupted()) {elemchain=orgnsr.toBeIntrptd.elemchain;}}
        if (rt0!=null) {route=rt0; break;}}}} else route=this;                return route;}

boolean deviceConsumingRmn()
{if(orgnsr.transit) return false;
        if (nexthop.type.equals("switch")) {dst=((Switch) nexthop).findInThisNet(dstAdr, this);
        if (dst!=null) {new CertainRoute(elemchain, false, orgnsr).transmit(dst,  dstAdr, crntPacket); return
false;}
        else if (!(otherWays.isEmpty())) return true;else if (orgnsr.toBeIntrptd!=null)
         if (((Switch)nexthop).unfound) orgnsr.impasse=true;}
        if (nexthop.type.equals("router")) {ArrayList<Element> echain=((Router) nexthop).seekInTable(dstAdr);
                if (echain!=null) {for (Element e:echain) {elemchain.add(e); orgnsr.elemcount ++;}
                isSuccessFinish=true; return false;}
                dst=((Router) nexthop).findNetSwitchOrServer(dstAdr, this);
if (dst!=null) {new CertainRoute(elemchain,false,orgnsr).transmit(nexthop.findLink(
dst).adjacentElem(nexthop),dstAdr,crntPacket); return false;}
        else if (!(otherWays.isEmpty())) return true;
        else if (orgnsr.toBeIntrptd!=null) if (((Router)nexthop).unfound) { orgnsr.impasse=true; return true;}}
        if (nexthop.type.equals("server")){Element e=((Server) nexthop).findInThisNet(dstAdr);
        if(e!=null) {new CertainRoute(elemchain, false, orgnsr).transmit(e,  dstAdr, crntPacket); return false;}
                Router rtr=((Server) nexthop).findRouter(dstAdr);
        if(rtr!=null) { otherWays.add(nexthop.findLink(rtr)); return true;}}
         if (nexthop.type.equals("enddevice"))  {Element nextElem=nexthop.links[0].adjacentElem(nexthop);
                if
((orgnsr.elemcount==1)&&((nextElem.type.equals("router"))||(nextElem.type.equals("server")||(nextElem.type.equal
s("switch")))))
                { otherWays.add(nexthop.findLink(nextElem)); return true; }
                else if (nexthop.IPaddress.equals(dstAdr))
                                {orgnsr.finalElemRoute=elemchain;isSuccessFinish=true;orgnsr.transit=true;
return false;}
                else return false;}                                                        return false;}

public boolean routersTableMulticast(int mc)
{if (isSuccessFinish) {for (Element e:orgnsr.finalElemRoute) {multicastcount++;
                if (orgnsr.finalElemRoute.indexOf(e)>multicastcount) if (e.type.equals("router"))
                        for(ArrayList<Element> echain:((Router)e).elemchainslist)
                        if ((echain.get(0)!=orgnsr.finalElemRoute.get(0))&&(echain.get(echain.size())
!=orgnsr.finalElemRoute.get(echain.size())))
                {((Router)e).elemchainslist.add(orgnsr.finalElemRoute);
                        new CertainRoute(orgnsr.finalElemRoute, true,
orgnsr).routersTableMulticast(++multicastcount);}}
```

```
        return true;} return false;};
        public void run() {/*activeRoutes.add(this);*/ }}

public void TVhosting(Element e, Packet p, int number, int chnls)
{src=e; PacketTest require; for (int j=0; j<src.linkcount; j++) {nexthop=e.links[j].adjacentElem(e);
        if (nexthop.type.equals("switch")) for (int i=0; i<nexthop.linkcount; i++)
        {if(nexthop.links[i].adjacentElem(nexthop)instanceof Server)
dst=nexthop.links[i].adjacentElem(nexthop);}
        if (dst!=null) {int shifts=(int) (Math.random()*100); double rands[]=new double[shifts+1];
        int randsSum=0, elapsedtime=0, chsdChnl=(int) (Math.random()*chnls), shiftcount=0;
        for (int i=0; i<shifts+1; i++) randsSum+=rands[i]=Math.random()*100;
        double koef=3600000/ randsSum; int moments[]=new int[shifts+1];
        for (int i=0; i<shifts+1; i++) moments[i]=(int)(rands[i]*koef);
        require=new PacketTest(e, nexthop, 0, nexthop.panelNmb); require.start();
        try {require.join(); require=new PacketTest(nexthop, dst, 0, nexthop.panelNmb);
        require.start(); require.join(); require.autoclear(false); require=null; System.gc();
        } catch (InterruptedException e1) {e1.printStackTrace();}
        finalElemRoute.add(dst); finalElemRoute.add(nexthop); finalElemRoute.add(src);
        for(int i=0, totalCount=number*3600; i<totalCount; i++) {send(p,false, chsdChnl);
        try {sleep((long) (1000/number)); elapsedtime+=1000/number;
        if(elapsedtime>=moments[shiftcount]) {chsdChnl=(int) (Math.random()*chnls); shiftcount++;}
        } catch (InterruptedException exc) {exc.printStackTrace();}}}}}

public void serverProc(Element e, Packet p, int number) {src=e; Route rt=new Route(e, "", p);
for (int j=0; j<src.linkcount; j++) {nexthop=e.links[j].adjacentElem(e);
if (nexthop.type.equals("switch")) for (int i=0; i<nexthop.linkcount; i++)
{if(nexthop.links[i].adjacentElem(nexthop)instanceof Server ) dst=nexthop.links[i].adjacentElem(nexthop);}
if (dst!=null) {int shifts=(int) (Math.random()*10000); int randsSum=0, elapsedtime=0;
double rands[]=new double[shifts+1]; for (int i=0; i<shifts+1; i++) randsSum+=rands[i]=Math.random()*100;
double koef=3600000/ randsSum; int moments[]=new int[shifts+1]; int shiftcount=0;
for (int i=0; i<shifts+1; i++) moments[i]=(int)(rands[i]*koef); Route nativeRoute=new Route(true);
nativeRoute.finalElemRoute.add(src); nativeRoute.finalElemRoute.add(nexthop);
nativeRoute.finalElemRoute.add(dst); nativeRoute.finalElemRoute.add(nexthop);
nativeRoute.finalElemRoute.add(src); finalElemRoute=new ArrayList<>(nativeRoute.finalElemRoute);
for(int i=0, totalcount=number*3600; i<totalcount; i++) {send(p,false, 0);
try {sleep((long) (1000/number)); elapsedtime+=1000/number;
if(elapsedtime>=moments[shiftcount]) {if (wp.allServers.size()>0) if(shiftcount%2==0)
{rt.dstAdr=wp.allServers.get((int) (wp.allServers.size()*Math.random())).IPaddress;
rt.transmitChain(false, 1, false); rt.finalElemRoute=new ArrayList<>(this.finalElemRoute);
finalElemRoute=new ArrayList<>();
for (int k=rt.finalElemRoute.size()-1; k>-1; k--) finalElemRoute.add(rt.finalElemRoute.get(k));}
else finalElemRoute=new ArrayList<>(nativeRoute.finalElemRoute); shiftcount++;}}
catch (InterruptedException exc) {exc.printStackTrace();}}}}}

    public boolean send(Packet p, boolean duplex, int nmb)
{if (finalElemRoute.size()==0) {System.out.println("Цепочка элементов пуста "); return false;}
int clusterColor=finalElemRoute.get(0).panelNmb==-1?7:finalElemRoute.get(0).panelNmb;
PacketTest pkts[]=new PacketTest[finalElemRoute.size()-1];
PacketTest.GroupPacketTest gppt= new PacketTest.GroupPacketTest("", clusterColor);
 for(int i=0; i<finalElemRoute.size()-1;i++) {int callPanelNmb=finalElemRoute.get(i).type.equals("router")
                &&finalElemRoute.get(i+1).type.equals("router")?-1:finalElemRoute.get(i).panelNmb;
         pkts[i]= new PacketTest("", callPanelNmb, callPanelNmb==-1?7:callPanelNmb);
        finalElemRoute.get(i).send(p, finalElemRoute.get(i+1),
        finalElemRoute.get(i).findLink(finalElemRoute.get(i+1)), pkts[i], false, i==finalElemRoute.size()-2);
        gppt.thrdGroup.add(pkts[i]);}  gppt.start(); try {gppt.join();
             } catch (InterruptedException e) {e.printStackTrace();} gppt=null; System.gc(); return true;}}

package devices; import java.util.ArrayList;
public class NetworkSimulator extends Thread{
ArrayList <Element> deviceObs=new ArrayList<>(), elemsB=new ArrayList<>(),
elemsC=new ArrayList<>(), elemsA=deviceObs; Element src, dst; boolean finished=false;
ArrayList <NetworkSimulator> srcThreads=new ArrayList<>(),bThreads=new ArrayList<>(),cThreads=new ArrayList<>();
static float totalTime=3600000, dialogTime=120000, Vcadr=33;  Thread crntThread= new Thread("");
static int frequency=1, volumeA=160, volumeB=160, volumeC=160; int multiplicer=1, Vblock=1;

void addDevice (Element d) {deviceObs.add(d);}
void setDevices (ArrayList <Element> dobs) {deviceObs=new ArrayList<>(dobs);}

void orgnsRand() {for (Element e:deviceObs) if(e.serviceClass=='B') elemsB.add(e);
else if(e.serviceClass=='C') {elemsB.add(e); elemsC.add(e);}
for(int i=0;i<deviceObs.size();i++) {src=deviceObs.get(i); srcThreads.add(new fromOneElem(deviceObs,src,this));}
for(int i=0; i<srcThreads.size(); i++) srcThreads.get(i).start(); if (elemsB.size()>1)
{for(int i=0; i<elemsB.size(); i++) {src=elemsB.get(i); bThreads.add(new DataThread(elemsB,src,this));}
for(int i=0; i<bThreads.size(); i++) bThreads.get(i).start();}
 if (elemsC.size()>1) {for(int i=0; i<elemsC.size(); i++)
 {src=elemsC.get(i); cThreads.add(new VideoThread(elemsC,src,this));}
 for(int i=0; i<cThreads.size(); i++) cThreads.get(i).start();}}

void clear() {deviceObs=new ArrayList<>(); srcThreads=new ArrayList<>(); crntThread=null;}

protected class fromOneElem extends NetworkSimulator
{NetworkSimulator ns; int colorNmb=0,  prefnumber;
```

```
fromOneElem(ArrayList <Element> ld, Element e, NetworkSimulator n)
{this.src=e; this.elemsA=new ArrayList<Element>(ld); this.elemsA.remove(e); ns=n; e.frqRlsd=frequency;}

public void run() {System.out.println("Начался процесс случайной передачи для элемента "+src.name);
int randsSum=0, moments[]=new int[frequency+1]; float rands[]=new float[frequency+1], koef=totalTime/randsSum;
for (int i=0; i<frequency+1; i++) randsSum+=rands[i]=(float) (Math.random()*100);
for (int i=0; i<frequency+1; i++) moments[i]=(int)(rands[i]*koef); int amountTime=0;
for (int i=0; i<frequency; i++) {if(src.frqRlsd==0) break;
try { sleep(moments[i]); } catch (InterruptedException e) {e.printStackTrace();} amountTime+=moments[i];
do dst=elemsA.get((int) (Math.random()*(elemsA.size()))); while((dst==src)&&(!elemsA.isEmpty()));
System.out.println(" Стартует передача пакета между элементами "+src.name+ " и "+dst.name+" на промежутке
"+amountTime);
if (dst.ocpd==true) try {dst.crntDialog.join();} catch (InterruptedException e1) {e1.printStackTrace();}
if (src.ocpd==true) try {src.crntDialog.join();} catch (InterruptedException e1) {e1.printStackTrace();}
src.ocpd=true; dst.ocpd=true; src.crntDialog=this; dst.crntDialog=this; src.frqRlsd--; dst.frqRlsd--;
prefnumber=(int) Vcadr; while (prefnumber>99) {prefnumber/=10; multiplicer*=10;}
Route rt=new Route(src, dst.IPaddress, new Packet(volumeA, multiplicer));
rt.transmitChain(true, (int) (prefnumber*dialogTime/1000), true);
try {rt.join();} catch (InterruptedException e) {e.printStackTrace();}
rt=null; System.gc(); src.ocpd=false; dst.ocpd=false; src.crntDialog=null; dst.crntDialog=null;
System.out.println(" Произошла передача пакета между элементами "+src.name+ " и "+dst.name);}
try {sleep(moments[frequency]);} catch (InterruptedException e) {e.printStackTrace();} this.finished=true;
if (src.frqRlsd==0) for(NetworkSimulator foe:ns.srcThreads) foe.deviceObs.remove(src);
if (dst.frqRlsd==0) for(NetworkSimulator foe:ns.srcThreads) foe.elemsA.remove(dst);
for(int i=0; i<ns.srcThreads.size(); i++) if(!ns.srcThreads.get(i).finished) break;
else if (i==ns.srcThreads.size()-1){ns.finished=true;}}}

protected static class DataThread extends NetworkSimulator{ NetworkSimulator ns;
ArrayList <Element> deviceObs=new ArrayList<>(); int colorNmb=0,  prefnumber=0;
        static int pktAmount=31457, extndTime=3600000, VcadrD=pktAmount/(extndTime*1000);

DataThread(ArrayList <Element> ld, Element e, NetworkSimulator n)
{this.src=e; this.elemsB=new ArrayList<Element>(ld); this.elemsB.remove(e); ns=n; }

public void run() { float ratio=extndTime/pktAmount, mailPart=(float) Math.random(), serverPart=(1-mailPart);
int pktAmount0=(int) (pktAmount*serverPart), randsSum=0; prefnumber=pktAmount0/3600; Route rt=new Route(true);
while (prefnumber>30) {prefnumber/=10; multiplicer*=10;} pktAmount0=(int) (pktAmount*mailPart);
rt.serverProc(src,  new Packet(src.serviceClass=='C'?volumeC:volumeB, multiplicer), prefnumber);
float rands[]=new float[pktAmount+1]; for (int i=0; i<pktAmount0+1; i++) randsSum+=rands[i]=(float)
(Math.random()*100);
float koef=extndTime/ randsSum+1; int moments[]=new int[pktAmount0+1];
for (int i=0; i<pktAmount0+1; i++) moments[i]=(int)(rands[i]*koef); int amountTime=0;
for (int i=0; i<pktAmount0; i++) {try {sleep(moments[i]);} catch (InterruptedException e) {e.printStackTrace();}
amountTime+=moments[i]; dst=elemsB.get((int) (Math.random()*(elemsB.size()))); while (ratio>20) {ratio/=10;
multiplicer*=10;}
System.out.println(" Стартует передача пакета между элементами "+src.name+ " и "+dst.name+" на промежутке
"+amountTime);
rt=new Route(src, dst.IPaddress, new Packet(src.serviceClass=='C'||dst.serviceClass=='C'?volumeC:volumeB,
multiplicer));
rt.transmitChain(false, 1, false); try {rt.join();} catch (InterruptedException e) {e.printStackTrace();}
System.out.println(" Произошла передача пакета между элементами "+src.name+ " и "+dst.name); rt=null;
System.gc();}
this.finished=true; if (!ns.finished) for(int i=0; i<ns.bThreads.size(); i++) if(!ns.bThreads.get(i).finished)
break;
else if (i==ns.srcThreads.size()-1) {ns.finished=true;}}};

protected static class VideoThread extends NetworkSimulator{ ArrayList <Element> deviceObs=new ArrayList<>();
NetworkSimulator ns; int colorNmb=0, pktAmount=31457, extndTime=3600000, prefnumber; static int VcadrTV=12800;
VideoThread(ArrayList <Element> ld, Element e, NetworkSimulator n)
{this.src=e; this.elemsC=new ArrayList<Element>(ld); this.elemsC.remove(e); ns=n; }

public void run() {Route rt=new Route(true); prefnumber=VcadrTV; while (prefnumber>20) {prefnumber/=10;
multiplicer*=10;}
rt.TVhosting(src, new Packet(volumeC, multiplicer), prefnumber, 10);
try {rt.join();} catch (InterruptedException e) {e.printStackTrace();} this.finished=true;}};

public void run()
{System.out.println("Начался процесс случайной передачи в сети с частотой "+frequency);
orgnsRand(); System.out.println("Завершился процесс случайной передачи в сети с частотой "+frequency);}}

package devices; import java.awt.event.ActionEvent; import java.awt.event.ActionListener;
import java.io.Serializable; import java.util.ArrayList; import javax.swing.ImageIcon;import javax.swing.JLabel;
public class PacketTest extends Thread implements Serializable{
private static final long serialVersionUID = 1L; int  sleeptime=50, xPoints[], yPoints[],  callingPanelNmb;
static int clrNmbs[]= {2, 3, 4, 5, 12, 14, 19, 20}; float koef=0.1f;
boolean finished=false, relocated=false, graphicAccess=false; JLabel jl;

PacketTest(String name, ImageIcon icn, int st, int locs[], float k, int sx, int sy, int cpn, int clr){
super(name); init(cpn, clr); setAll(icn, st, locs, k, sx, sy);}
PacketTest(String name, int cpn, int clr) {super(name); init(cpn, clr);}
PacketTest(Element e0, Element e1, int cpn, int clr)
{init(cpn, clr); ConnectLine way=e0.findLink(e1); setLocs(way.location);}

public void init(int cpn, int clr) { jl=new JLabel(Packet.allPacketImgs[clrNmbs[clr]]);
jl.setSize(30, 30); callingPanelNmb=cpn; if (Route.wp.crntPanelNmb==callingPanelNmb) {graphicAccess=true;
if(callingPanelNmb==-1) Route.wp.add(jl); else Route.wp.allClusters.get(callingPanelNmb).add(jl);}
else {graphicAccess=false; jl.setVisible(false);}

Route.wp.relocateLabels.addActionListener( new ActionListener() {
public void actionPerformed (ActionEvent  ae) {if (!finished)
```

```
{if (Route.wp.crntPanelNmb==callingPanelNmb) {graphicAccess=true; if(callingPanelNmb==-1) Route.wp.add(jl);
else Route.wp.allClusters.get(callingPanelNmb).add(jl);} else {graphicAccess=false;
if(callingPanelNmb==-1) Route.wp.add(jl); else Route.wp.allClusters.get(callingPanelNmb).add(jl);}}}}});}

public void autoclear(boolean gc) {jl=null; xPoints=null; yPoints=null; if(gc) System.gc();}

public void run(){ equalSpeedMotion(0); finished=true;}

void setAll(ImageIcon icn, int st, int locs[], float k, int sx, int sy)
{jl.setIcon(icn); sleeptime=st; if (locs!=null) setLocs(locs); koef=k; jl.setSize(sx, sy);}

void locateAndMove(Element e0, Element e1, boolean strt) {ConnectLine cn=e0.findLink(e1);
if (cn!=null) {if (cn.location==null) {int m[]=cn.location; if((e1.locX==m[0])&&(e1.locY==m[1]))
{int x, y; x=m[2]; y=m[3]; m[2]=m[0]; m[3]=m[1]; m[0]=x; m[1]=y; cn.location=m;} setLocs(m);}
else setLocs(cn.location); if (strt) start();}}

void setLocs(int locs[]) {if (locs!=null) {     xPoints=new int[locs.length/2]; yPoints=new int[locs.length/2];
for (int i=0; i<locs.length; i+=2){xPoints[i/2]=locs[i]; yPoints[i/2]=locs[i+1];}
jl.setLocation(xPoints[0], yPoints[0]);}}

void equalSpeedMotion(int delay) {if(graphicAccess) {jl.setVisible(true); for (int i=0; i<xPoints.length-1; i++)
{int dx, dy, xleng=Math.abs(xPoints[i+1]-xPoints[i]),yleng=Math.abs(yPoints[i+1]-yPoints[i]);
if (xPoints.length==2) {dx=(int)(xleng*koef); dy=(int)(yleng*koef);} else {dx=xleng==0?0:8; dy=yleng==0?0:8;}
if (xPoints[i+1]<xPoints[i]) dx=-dx; if (yPoints[i+1]<yPoints[i]) dy=-dy;
float dhyp= (float) Math .sqrt(dx*dx+dy*dy);int delay0=(int) (dhyp*2); if (xleng==0) dx=0; if (yleng==0) dy=0;
int curx=relocated?25:0, cury=relocated?25:0;
try{for(; (Math.abs(curx)<xleng)||(Math.abs(cury)<yleng); curx+=dx, cury+=dy) {
jl.setLocation(curx+xPoints[i],cury+yPoints[i]);
        if (!jl.isVisible()) sleep(delay0);}}
catch(InterruptedException e) {System.out.println("Thread has been interrupted");}}     jl.setVisible(false);}
try {sleep(delay);} catch (InterruptedException e) {e.printStackTrace();} finished=true;}

public static class GroupPacketTest extends Thread{
ArrayList<PacketTest> thrdGroup=new ArrayList<>(); int crntColor; boolean relocated=false;
GroupPacketTest(String name) {super(name);}
GroupPacketTest(PacketTest pkts[]) {for (PacketTest p:pkts) thrdGroup.add(p);}
GroupPacketTest(PacketTest pkts[], int clrNmb) {this(pkts); crntColor=clrNmbs[clrNmb];}
GroupPacketTest(String name, int clrNmb) {this(name);crntColor=clrNmbs[clrNmb];}

public void run(){ for (PacketTest p:thrdGroup) {if(relocated) p.relocated=true; p.equalSpeedMotion(0);}
for(int i=0;i<thrdGroup.size();i++){thrdGroup.get(i).autoclear(false);thrdGroup.set(i,null);}thrdGroup=null;}

public void relocate(){ relocated=true;}}}


package devices;
import javax.swing.JLabel; import javax.swing.JPanel; import javax.swing.JTextArea; import javax.swing.JButton;
import devices.NetworkSimulator.fromOneElem; import java.io.Serializable; import java.util.ArrayList;
import java.util.regex.Matcher; import java.util.regex.Pattern; import javax.swing.ImageIcon;

public class Element extends Device implements Serializable{
private static final long serialVersionUID = 1L; ArrayList<Element> adjacent=new ArrayList<>();
String type ="undefined", status="empty", IPaddress="1.1.1.0", IPmask="255.255.255.0", networkIP="", name="",
binaddress="1.1.1.0", binmask="255.255.255.0", binnetworkIP=""; ImageIcon unitIcon; JButton unvisButton;
JTextArea terminal; JPanel confpane, calcpane, termpane; ConnectLine[] links=new ConnectLine[500]; Port ports[];
ArrayList <Packet >RcvdPackets=new ArrayList<>(); fromOneElem crntDialog=null; Route r=new Route(false);
int portsNumber=70, locX=0, locY=0, adr[]=new int[4], mask[]=new int[4], net[]=new int[4],loadedVlm=0,
crntLoad=0,
pktNmb=0, sessionPktNmb=0, linkcount=0, totalNumber=0, sessionRcv=0, panelNmb=-1; Packet packetOnReceive;
public JLabel shortcut, textlabel; char serviceClass='A'; int number=0, frqRlsd=0;
boolean activeports[]=new boolean[100], acsdports[]=new boolean[100], termwindAccess=false, ocpd=false,
                        serviceTypePhone=false, serviceTypeData=false, serviceTypeVideo=false;

Element(){ super();createDtls();}
Element(String ename){super(); name=ename; createDtls();}
Element(String ename, int nmb){this("ename"); panelNmb=nmb;}
Element(String ename, int nmb, int typeNmb) {this(); panelNmb=nmb; name=ename+(nmb+1)+"."+typeNmb;}

void createDtls() {shortcut=new JLabel(""); textlabel=new JLabel(""); unvisButton=new JButton();
ports=new Port[portsNumber]; for (int i=0; i<portsNumber; i++) {ports[i]=new Port(i);
activeports[i]=false; acsdports[i]=true;} confpane=new JPanel(); createBinars();}

void setAll(JLabel tlab, JLabel imlab, JButton unvb) {textlabel=tlab; shortcut=imlab; unvisButton=unvb;
textlabel.setText(name);unvisButton.setIcon(unitIcon);unvisButton.setText(name);unvisButton.setVisible(true);}

void receive(Packet p,boolean isDest){if (p!=null){if (packetOnReceive!=null)RcvdPackets.add(packetOnReceive);
pktNmb++; packetOnReceive=p; crntLoad=p.volume*p.multiplicer; loadedVlm+=crntLoad;
```

```
if(isDest) {sessionPktNmb+=p.multiplicer; sessionRcv+=crntLoad; setTermParams();} status="Packet received";}}

void clear() {sessionRcv=loadedVlm=crntLoad=0; packetOnReceive=null;
RcvdPackets=new ArrayList<>(); pktNmb=0; status="empty";}

void send(Packet p, Element e, ConnectLine cl, PacketTest pt, boolean strt, boolean isDest)
{p.setAll(this,e,ports[0],cl.ports[1],cl); e.receive(p,isDest); if(pt!=null)pt.locateAndMove(this,e,strt);}

String getNetworkIP() {return "";}

String[] createBinars() {String res[]=new String[3]; adr=(IPConsumer(IPaddress, res[0]=binaddress)).clone();
mask=(IPConsumer(IPmask, res[1]=binmask)).clone(); for(int i=0; i<4; i++) {net[i]=adr[i]&mask[i];
networkIP+=net[i]; if (i==3) networkIP+="." ;} res[2]=networkIP; return res;}

int[] IPConsumer(String res, String result) {result=""; int m[]=new int[4], start = 0, end = 0, i=0;
Pattern regex = Pattern.compile("[(0-9)]++"); Matcher matcher = regex.matcher(res); StringBuffer substr;
while(matcher.find()) {start=matcher.start(); end=matcher.end(); substr=new StringBuffer
(res.subSequence(start,end));result+=Integer.toBinaryString(m[i++]=Integer.parseInt(substr.toString()))+".";}
result=result.substring(0, result.length()-1); return m;}

ConnectLine findLink (Element e1) {for (ConnectLine cl:links)
if (cl!=null) if(cl.adjacentElem(this)==e1) {return cl;} return null;}

void setTermParams() {if(Route.wp.termWind.cpr!=null)
          Route.wp.termWind.cpr.setParams(crntLoad, sessionRcv, se sionRcv, sessionPktNmb,  termwindAccess);}}


package devices; import java.io.Serializable; import javax.swing.*;

public class PersonalComputer extends EndDevice implements Serializable{
        private static final long serialVersionUID = 1L; static int numberCount=0;
        static ImageIcon icon=new ImageIcon ( "D:\\imagesForWork\\PCimage.png") ;

        PersonalComputer(String ename) {super(ename); portsNumber=70;
        unitIcon=icon; number=numberCount++; if (name=="") name="PC"+number;
    serviceClass='B'; serviceTypePhone=true; serviceTypeData=true;}
        PersonalComputer(String ename, int nmb) {this(ename); panelNmb=nmb;}
    PersonalComputer (String ename, int nmb, int typeNmb)
        {super( ename,  nmb, typeNmb); serviceClass='A'; serviceTypePhone=true;
                            serviceTypeData=true; unitIcon=icon; number=numberCount++;}}

package devices; import java.io.Serializable; import java.util.ArrayList;
import javax.swing.ImageIcon; import devices.Route.CertainRoute;
public class Switch extends Element implements Serializable{
private static final long serialVersionUID = 1L; Server server; boolean unfound=false;
ArrayList<Element> elemsInNet=new ArrayList<>(), hopsOverNet=new ArrayList<>();
static ImageIcon icon=new ImageIcon ("D:\\imagesForWork\\SwitchImage.png"); static int numberCount=0;

Switch(String ename) {super(ename); super.type ="switch"; portsNumber=70;
unitIcon=icon; number=numberCount++; if (ename=="") name="Switch"+number;}
Switch(String ename, int nmb) {this(ename); panelNmb=nmb;}
Switch (String ename, int nmb, int typeNmb)
{super( ename,  nmb, typeNmb); type ="switch"; unitIcon=icon; number=numberCount++;}

Element findInThisNet(String adr, CertainRoute rt) {unfound=true; rt.otherWays.clear();
int m[]=new int[4], netDst[]=new int[4]; m=(IPConsumer(adr, new String(""))).clone();
for(int i=0; i<4; i++) netDst[i]=m[i]&mask[i]; for (int i=0;i<4;i++){if (net[i]!=netDst[i]) break;
if(i==3) for (Element e:elemsInNet) {if(e.IPaddress.equals(adr)) {unfound=false; return e;}}}
for (Element e:hopsOverNet) if(e!=rt.elemchain.get(rt.elemchain.size()-2))
                        {rt.otherWays.add(findLink(e)); unfound=false;} return null;}}


package devices; import java.util.ArrayList;
public class DataBus extends Device{
private static final long serialVersionUID = 1L; int nmb, locs[]=new int[4], panelNmb;
ArrayList<Element> members; Element head; boolean isHorizontal=true;

DataBus(int nmb, int locs[], int panelNmb) {members= new ArrayList<>();
if (locs!=null) this.locs=locs.clone(); this.panelNmb=panelNmb; this.nmb=nmb;}

void addMember(Element e) {members.add(e); if (isHorizontal) {if (e.locX<locs[0]) locs[0]=e.locX+25;
if (locs[0]<0) locs[0]=0; if (e.locX>locs[2]) locs[2]=e.locX+25;} }

void setHead(Element e) {addMember(e); head=e;}}
```