Университет ИТМО
Кафедра ВТ

Программирование
Лабораторная работа №7

Группа P3110
Вариант 311012
Нгу Фыонг Ань
Проверил:
Писмак Алексей Евгеньевич

год

Доработать программу из лабораторной работы №6 следующим образом:

1. Хранящимся в коллекции объектам добавить характеристику цвета.
2. Написать графический интерфейс для серверной части, который в центральной части окна отображает элементы коллекции с помощью `JTree`, реализует функции добавления, редактирования и удаления объекта, а также все остальные функции для управления коллекцией из предыдущей работы. Интерфейс должен удовлетворять следующим требованиям:
   - должна осуществляться авторизация пользователя с помощью пароля;
   - операции чтения и сохранения коллекции объектов должны быть реализованы как пункты меню;
   - элементы управления объектами должны располагаться в левой части окна;
   - д
3. Написать графический интерфейс для клиентской части, который отображает в окне объекты коллекции в виде кругов соответствующего размера и цвета, расположенных согласно своим координатам. Интерфейс должен удовлетворять следующим требованиям:
   - при наведении мышкой на объект должна появляться всплывающая подсказка с именем объекта;
   - должны быть реализованы фильтры для каждой характеристики объектов;
   - при реализации фильтров должны быть использованы `JComboBox`, `JTextField`, `JSlider` и другие компоненты;
   - при нажатии на кнопку "Старт" объекты, характеристики которых соответствуют текущим значениям фильтров, должны в течение 5 секунд плавно исчезать, затем в течение 4 секунд возвращаться в исходное состояние;
   - при нажатии на кнопку "Стоп" анимация должна останавливаться.
4. Графические интерфейсы реализуются с помощью библиотеки Swing. По согласованию с преподавателем библиотека может быть изменена.

Порядок выполнения работы:

б

1. Нарисовать прототипы интерфейсов приложения в инструменте прототипирования (выбирается по согласованию с преподавателем).
2. Утвердить у преподавателя нарисованные прототипы интерфейса.
3. Реализовать согласованные интерфейсы в коде.

о

Отчёт по работе должен содержать:

в

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Скриншоты интерфейса.
5. Выводы по работе.

в

Вопросы к защите лабораторной работы:

е

1. Компоненты пользовательского интерфейса. Иерархия компонентов.
2. Базовые классы `Component`, `Container`, `JComponent`.
3. Менеджеры компоновки.
4. Модель обработки событий. Класс-слушатель и класс-событие.
5. Технология JavaFX. Особенности архитектуры, отличия от AWT / Swing.
6. Технология SWT. Сходства и отличия в сравнении с Swing и JavaFX.
7. Шаблоны проектирования. GoF-паттерны.

б

х

о

д

и

м

Исходный код:
#Server

#ServerUDP7_exp

```
package serverudp;

import classes.Human;
import java.awt.EventQueue;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.time.LocalDateTime;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;
import java.util.logging.*;
import javax.swing.*;

public class ServerUDP7_exp extends Thread {

    public static void main(String[] args) throws IOException {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                LoginFrame login = new LoginFrame();
                login.loginBut.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent ev) {
                        if ((login.field2.getText().equals("password"))) {
                            login.login.dispose();
                            new Thread(new Runnable() {
                                @Override
                                public void run() {
                                    try {
                                        createList();
                                    } catch (IOException ex) {

Logger.getLogger(ServerUDP7_exp.class.getName()).log(Level.SEVERE, null, ex);
                                    }
                                }
                            }).start();
                        }
                    }
                });
            }
        });
    }

    static void createList() throws SocketException, IOException {
        DatagramSocket socket = new DatagramSocket(9999);
        Set<Human> human = Collections.newSetFromMap(new ConcurrentHashMap<Human, Boolean>());
        ArrayList<DatagramPacket> packetList = new ArrayList<DatagramPacket>();

        String fileName = "D:\\NetBeansProjects\\ServerUDP_7_Exp\\src\\serverudp\\HumanList.xml";

        try {
            XmlFile.read(fileName, human);
        } catch (FileNotFoundException ex) {
            System.out.println("FILE INPUT NOT FOUND \n");
        }

        System.out.println(LocalDateTime.now());
        System.out.print("SERVER IS ONLINE \n");

        JTreeList treeList = new JTreeList(human, packetList, socket);
        treeList.creatFrame();
        human.stream().forEach((p) -> {
            System.out.println(p);
```

```
            p.setAction("Add");
            treeList.addNode(p);
        });

        while (true) {
            human.stream().sorted();
            byte[] buf = new byte[1024];
            DatagramPacket packet = new DatagramPacket(buf, buf.length);
            socket.receive(packet);
            packetList.add(packet);
            System.out.println("a " + packet.getAddress() + " " + packet.getPort());
            //System.out.println(packetList);

            ServerThread serThread = new ServerThread(socket, packet, human, fileName);
            new Thread(serThread).start();
        }
    }
}
```

#JtreeList.java

```
package serverudp;

import classes.Human;
import java.awt.*;
import java.awt.event.*;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
import static javax.swing.GroupLayout.Alignment.*;
import javax.swing.tree.*;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class JTreeList extends JFrame{
        Set<Human> human;
        ArrayList<DatagramPacket> packetList;
        DatagramSocket socket;

        public JTreeList(Set<Human> human, ArrayList<DatagramPacket> packetList, DatagramSocket
socket){
            this.human = human;
            this.packetList = packetList;
            this.socket = socket;
        }
        private DefaultMutableTreeNode root = new DefaultMutableTreeNode("Human List");
        private JTree tree = new JTree(root);
        JFrame frame = new JFrame();
        DefaultTreeModel model = (DefaultTreeModel) tree.getModel();

        JTextField nameField = new JTextField();

        String fileName = "D:\\NetBeansProjects\\ServerUDP_7\\src\\serverudp\\HumanList.xml";


    public void creatFrame(){

        createMenuBar();
        createFunctionPanel();

        tree.setEditable(true);
        frame.add(tree);
        frame.add(new JScrollPane(tree));

        frame.setTitle("Human List");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,500);
        frame.setVisible(true);
    }
```

```
public void createFunctionPanel(){
    JPanel panel = new JPanel();
    GroupLayout layout = new GroupLayout(panel);
    panel.setLayout(layout);
    layout.setAutoCreateGaps(true);
    layout.setAutoCreateContainerGaps(true);

    JButton addBut = new JButton("Add");
    JButton delBut = new JButton("Delete");
    JButton editBut = new JButton("Edit");

    addBut.setPreferredSize(new Dimension(40, 40));

    layout.setVerticalGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(TRAILING).addComponent(addBut))
            .addGroup(layout.createParallelGroup(TRAILING).addComponent(delBut))
            .addGroup(layout.createParallelGroup(TRAILING).addComponent(editBut)));

    layout.setHorizontalGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
                    .addComponent(addBut)
                    .addComponent(delBut)
                    .addComponent(editBut)))
            ;

    delBut.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent ev){
            removeSelectedNode();
        }
    });

    addBut.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent ev){
            createAddWindow();
        }
    });

    editBut.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent ev){
            createEditWindow();
        }

    });
    frame.add(panel, BorderLayout.WEST);
}

public void createAddWindow(){
    JFrame addFrame = new JFrame("Add human");
    addFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    addFrame.setSize(200,250);
    addFrame.setVisible(true);
    JButton closeBut = new JButton("Ok");

    JLabel label1 = new JLabel("Name:");
    JTextField field1 = new JTextField(20);
    JLabel label2 = new JLabel("Color:");
    JTextField field2 = new JTextField(20);
    JLabel label3 = new JLabel("Size:");
    JTextField field3 = new JTextField(20);
    JLabel label4 = new JLabel("X:");
    JTextField field4 = new JTextField(20);
    JLabel label5 = new JLabel("Y:");
    JTextField field5 = new JTextField(20);
    JLabel label6 = new JLabel("Time:");
    JTextField field6 = new JTextField(20);
    JPanel addPan = new JPanel();
    GroupLayout aLayout = new GroupLayout(addPan);
    addPan.setLayout(aLayout);

    aLayout.setAutoCreateGaps(true);
```

```
        aLayout.setAutoCreateContainerGaps(true);
        aLayout.setHorizontalGroup(aLayout.createSequentialGroup()
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)
                .addComponent(label1)
                .addComponent(label2)
                .addComponent(label3)
                .addComponent(label4)
                .addComponent(label5)
                .addComponent(label6)
            )
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                .addComponent(field1)
                .addComponent(field2)
                .addComponent(field3)
                .addComponent(field4)
                .addComponent(field5)
                .addComponent(field6)
                .addComponent(closeBut))

        );

        aLayout.setVerticalGroup(aLayout.createSequentialGroup()
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label1)
                .addComponent(field1))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label2)
                .addComponent(field2))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label3)
                .addComponent(field3))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label4)
                .addComponent(field4))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label5)
                .addComponent(field5))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label6)
                .addComponent(field6))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(closeBut))
        );

        addFrame.add(addPan);

        closeBut.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                try {
                    Human p = new Human(field1.getText(),
                            field2.getText(),
                            Integer.parseInt(field3.getText()),
                            Integer.parseInt(field4.getText()),
                            Integer.parseInt(field5.getText()),
                            field6.getText());
                    p.setAction("Add");
                    addNode(p);
                    model.reload(root);
                    System.out.println(p);
                    human.add(p);
                    ServerUpdateThread updThread = new ServerUpdateThread(socket, packetList, p);
                    new Thread(updThread).start();
                } catch (NumberFormatException ex){
                    JOptionPane.showMessageDialog(null, "Input value is not match the given type!");
                }
                addFrame.dispose();
            }
        });
    }

    public void createEditWindow(){
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)
tree.getLastSelectedPathComponent();
```

```
    if (selectedNode != null){
        JFrame editFrame = new JFrame("Edit content");
        editFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        editFrame.setSize(200,250);
        editFrame.setVisible(true);
        JButton closeBut = new JButton("Replace");

        JLabel label1 = new JLabel("New value:");
        JTextField field1 = new JTextField(20);
        JPanel editPan = new JPanel();
        GroupLayout aLayout = new GroupLayout(editPan);
        editPan.setLayout(aLayout);

        aLayout.setAutoCreateGaps(true);
        aLayout.setAutoCreateContainerGaps(true);
        aLayout.setHorizontalGroup(aLayout.createSequentialGroup()
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)
                .addComponent(label1)
            )
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                .addComponent(field1)
                .addComponent(closeBut))
        );

        aLayout.setVerticalGroup(aLayout.createSequentialGroup()
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(label1)
                .addComponent(field1))
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
            .addGroup(aLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                .addComponent(closeBut)))
        );

        editFrame.add(editPan);

        closeBut.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                DefaultMutableTreeNode node = selectedNode;
                String oldString = selectedNode.toString();
                String newValue = field1.getText();
                if (selectedNode == root){
                    selectedNode.setUserObject(newValue);
                    editFrame.dispose();
                } else
                if (selectedNode.getParent() != root){
                    while (node.getParent() != root){
                        node = (DefaultMutableTreeNode) node.getParent();
                    }
                }
                Human p = getInfor(node);
                Human newP = p;
                Human oldP = new Human(p.name, p.color, p.size, p.PosX, p.PosY, p.time);
                oldP.setAction("Delete");
                ServerUpdateThread updThread = new ServerUpdateThread(socket, packetList, oldP);
                new Thread(updThread).start();


                try {
                    if (oldString.startsWith("COLOR: ")) {
                        newP.setColor(newValue);
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setColor(newValue));
                        selectedNode.setUserObject("COLOR: " + newValue);

                    } else
                    if (oldString.startsWith("SIZE: ")) {
                        newP.setSize(Integer.parseInt(newValue));
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setSize(Integer.parseInt(newValue)));
                        selectedNode.setUserObject("SIZE: " + newValue);
                    } else
                    if (oldString.startsWith("X: ")) {
                        newP.setPosX(Integer.parseInt(newValue));
```

```
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setPosX(Integer.parseInt(newValue)));
                        selectedNode.setUserObject("X: " + newValue);
                    } else
                    if (oldString.startsWith("Y: ")) {
                        newP.setPosY(Integer.parseInt(newValue));
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setPosY(Integer.parseInt(newValue)));
                        selectedNode.setUserObject("Y: " + newValue);
                    } else
                    if (oldString.startsWith("TIME: ")) {
                        newP.setTime(newValue);
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setTime(newValue));
                        selectedNode.setUserObject("TIME: " + newValue);
                    } else {
                        newP.setName(newValue);
                        human.stream().filter(h -> h.equal(p)).forEach((Human h) ->
h.setName(newValue));
                        selectedNode.setUserObject(newValue);
                    };
                } catch (NumberFormatException ex){
                    JOptionPane.showMessageDialog(null, "Input value is not match the given type!");
                } finally {editFrame.dispose();}
                model.reload(root);
                newP.setAction("Add");

                ServerUpdateThread updThread1 = new ServerUpdateThread(socket, packetList, newP);
                new Thread(updThread1).start();
            }
        });
        } else JOptionPane.showMessageDialog(null,"Please choose a target first!");
    }

    public void addNode(Human h){
        DefaultMutableTreeNode node = new DefaultMutableTreeNode(h.name);
        root.add(node);
        node.add(new DefaultMutableTreeNode("COLOR: " + h.color));
        node.add(new DefaultMutableTreeNode("SIZE: " + h.size));
        node.add(new DefaultMutableTreeNode("X: " + h.PosX));
        node.add(new DefaultMutableTreeNode("Y: " + h.PosY));
        node.add(new DefaultMutableTreeNode("TIME: " + h.time));
    }

    public void createMenuBar(){
        JMenuBar bar = new JMenuBar();
        JMenu menuEdit = new JMenu("File");
        JMenu menuAct = new JMenu("Action");

        JMenuItem saveItem = new JMenuItem("Save list");
        JMenuItem readItem = new JMenuItem("Read file");
        JMenuItem expandItem = new JMenuItem("Expand all");
        JMenuItem collapseItem = new JMenuItem("Collapse");

        expandItem.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                setAllNodeState(tree, root, true);
            }
        });

        collapseItem.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                setAllNodeState(tree, root, false);
            }
        });

        readItem.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                root.removeAllChildren();
                model.reload();
                human.removeAll(human);
```

```
                try {XmlFile.read(fileName, human);}
                    catch (FileNotFoundException ex) {
                        System.out.println("FILE INPUT NOT FOUND \n");
                }
                System.out.println(human);
                human.stream().forEach((p) -> {
                    System.out.println(p);
                    addNode(p);});
                model.reload(root);
            }
        });

        saveItem.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent ev){
                try {
                    XmlFile.write(fileName, human);
                } catch (UnsupportedEncodingException ex) {
                    Logger.getLogger(JTreeList.class.getName()).log(Level.SEVERE, null, ex);
                } catch (IOException ex) {
                    Logger.getLogger(JTreeList.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });

        menuEdit.add(saveItem);
        menuEdit.add(readItem);
        menuAct.add(expandItem);
        menuAct.add(collapseItem);

        bar.add(menuEdit);
        bar.add(menuAct);
        frame.setJMenuBar(bar);
    }

    public void removeSelectedNode(){
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode)
tree.getLastSelectedPathComponent();
        if (selectedNode != null){
            if (selectedNode == root){
                JOptionPane.showMessageDialog(null, "You can not delete the entire list!!");
            }
            if (selectedNode.getParent() != root){
                while (selectedNode.getParent() != root){
                    selectedNode = (DefaultMutableTreeNode) selectedNode.getParent();
                }
            }
            Human p = getInfor(selectedNode);
            model.removeNodeFromParent(selectedNode);
            p.setAction("Delete");
            ServerUpdateThread updThread = new ServerUpdateThread(socket, packetList, p);
            new Thread(updThread).start();
            human.stream().filter(h -> h.equal(p)).forEach((Human h) -> human.remove(h));
            System.out.println("Deleted " + p);
            //System.out.println(human);
        } else JOptionPane.showMessageDialog(null,"Please choose a target first!");
    }

    public Human getInfor(DefaultMutableTreeNode node){
        Human h = new Human("","",0,0,0,"");
        h.setName(node.toString());
        h.setColor(node.getChildAt(0).toString().replaceFirst("COLOR: ", ""));
        h.setSize(Integer.parseInt(node.getChildAt(1).toString().replaceFirst("SIZE: ", "")));
        h.setPosX(Integer.parseInt(node.getChildAt(2).toString().replaceFirst("X: ", "")));
        h.setPosY(Integer.parseInt(node.getChildAt(3).toString().replaceFirst("Y: ", "")));
        h.setTime(node.getChildAt(4).toString().replaceFirst("TIME: ", ""));
        return h;
    }

    public static void setAllNodeState(JTree tree, DefaultMutableTreeNode node, boolean expanded) {
      ArrayList<DefaultMutableTreeNode> list = Collections.list(node.children());
      list.forEach((treeNode) -> {
          setAllNodeState(tree, treeNode, expanded);
```

```
            });
      if (!expanded && node.isRoot()) {
          return;
      }
      TreePath path = new TreePath(node.getPath());
      if (expanded) {
          tree.expandPath(path);
      } else {
          tree.collapsePath(path);
      }
  }
}
```

#LoginFrame

```
package serverudp;

import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

class LoginFrame {

    JFrame login = new JFrame("Login to server");
    JLabel label2 = new JLabel("Password:");
    JTextField field2 = new JTextField(10);
    JButton loginBut = new JButton("Login");
    JPanel loginPan = new JPanel();

    public LoginFrame() {
        login.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        login.setSize(800, 400);
        login.setVisible(true);

        GroupLayout loginLayout = new GroupLayout(loginPan);
        loginPan.setLayout(loginLayout);

        loginLayout.setAutoCreateGaps(true);
        loginLayout.setAutoCreateContainerGaps(true);
        loginLayout.setHorizontalGroup(loginLayout.createSequentialGroup()
                .addGroup(loginLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)
                        .addComponent(label2)
                )
                .addGroup(loginLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                        .addComponent(field2)
                        .addComponent(loginBut))
        );

        loginLayout.setVerticalGroup(loginLayout.createSequentialGroup()
                .addGroup(loginLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                        .addComponent(label2)
                        .addComponent(field2))
                .addGroup(loginLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                        .addComponent(loginBut))
        );

        login.add(loginPan);

    }
}
```

#ServerThread

```
package serverudp;

import classes.Human;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
```

```
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;

class ServerThread implements Runnable {

    private DatagramSocket socket;
    private DatagramPacket packet;
    private Set<Human> human;
    private String fileName;

    public ServerThread(DatagramSocket socket, DatagramPacket packet, Set<Human> human, String
fileName) {
        this.socket = socket;
        this.packet = packet;
        this.human = human;
        this.fileName = fileName;
    }

    @Override
    public void run() {
        String s = new String(packet.getData());

        InetAddress IPAddress = packet.getAddress();
        int port = packet.getPort();
        System.out.print("CLIENT #" + port + ": " + s + "\n");

        human.stream().forEach(h -> {
            try {
                sendOb(h, IPAddress, port);
            } catch (IOException ex) {
                Logger.getLogger(ServerThread.class.getName()).log(Level.SEVERE, null, ex);
            }
        });
    }

    public void sendOb(Human ob, InetAddress IPAddress, int port) throws IOException {
        System.out.println(ob.toString());
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        ObjectOutputStream os = new ObjectOutputStream(outputStream);
        os.writeObject(ob);
        os.flush();
        byte[] data = outputStream.toByteArray();
        DatagramPacket sendPacket = new DatagramPacket(data, data.length, IPAddress, port);
        socket.send(sendPacket);
    }
}
```

#ServerUpdateThread

```
package serverudp;

import classes.Human;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;

class ServerUpdateThread implements Runnable {

    private Human h;
    private ArrayList<DatagramPacket> packetList;
    private DatagramSocket socket;
```

```
    public ServerUpdateThread(DatagramSocket socket, ArrayList<DatagramPacket> packetList, Human h)
{
        this.packetList = packetList;
        this.h = h;
        this.socket = socket;
    }

    @Override
    public void run() {
        String s;
        InetAddress IPAddress;
        //System.out.println(packetList);
        System.out.println("update " + h);
        packetList.stream().forEach( p -> {
            try {
                //System.out.println("b " +p.getAddress() + " " + p.getPort());
                sendOb(h, p.getAddress(), p.getPort());
                //System.out.println(h);
            } catch (IOException ex) {
                Logger.getLogger(ServerUpdateThread.class.getName()).log(Level.SEVERE, null, ex);
            }
        });

    }

    public void sendOb(Human ob, InetAddress IPAddress, int port) throws IOException {
        //System.out.println(ob.toString());
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        ObjectOutputStream os = new ObjectOutputStream(outputStream);
        os.writeObject(ob);
        os.flush();
        byte[] data = outputStream.toByteArray();
        DatagramPacket sendPacket = new DatagramPacket(data, data.length, IPAddress, port);
        socket.send(sendPacket);
    }
}
```

#Client

#ClientUDP7.Java

```
package clientudp;

import classes.Human;
import com.sun.pisces.Surface;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.geom.Ellipse2D;
import java.io.*;
import java.lang.reflect.Field;
import java.net.*;
import java.nio.ByteBuffer;
import java.nio.channels.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
import javax.swing.Timer;
import javax.swing.JComponent;

public class ClientUDP7 extends JPanel {

    DatagramChannel client = null;
    SocketAddress address = new InetSocketAddress((int) Math.random() * 9998);
    SocketAddress serverAdd = new InetSocketAddress("localhost", 9999);
    Set<Circle> circle = new HashSet<>();
    ExpFrame exp;

    public ClientUDP7() throws IOException {
        client = DatagramChannel.open();
        client.bind(address);
```

```
        //client.socket().setSoTimeout(5000);
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                exp = new ExpFrame();
            }

        });
    }

    public static void main(String args[]) throws SocketException, UnknownHostException,
IOException, ClassNotFoundException, SocketTimeoutException, NoSuchFieldException,
IllegalArgumentException, IllegalAccessException {
        ClientUDP7 client1 = new ClientUDP7();
        client1.StartRequest();
    }

    public void makeConnection() throws IOException {
        client.connect(serverAdd);
        client.configureBlocking(true);
    }

    public void StartRequest() throws IOException, SocketTimeoutException, ClassNotFoundException,
NoSuchFieldException, IllegalArgumentException, IllegalAccessException {
        makeConnection();
        sendRequest("Start");

        receiveResult();
    }

    public void sendRequest(String msg) throws IOException {
        //Datagram Socket Channel UDP
        ByteBuffer buffer = ByteBuffer.wrap(msg.getBytes());
        client.send(buffer, serverAdd);
    }

    public void receiveResult() throws IOException, SocketTimeoutException, ClassNotFoundException,
NoSuchFieldException, IllegalArgumentException, IllegalAccessException {

        //receive Object
        byte[] incomingData = new byte[1024];
        DatagramPacket packet = new DatagramPacket(incomingData, incomingData.length);

        while (true) {
            try {
                client.socket().receive(packet);
                byte[] data = packet.getData();
                ByteArrayInputStream in = new ByteArrayInputStream(data);
                ObjectInputStream is = new ObjectInputStream(in);
                Human h = (Human) is.readObject();
                System.out.println(h + " " + h.action);
                Circle c = new Circle(h.PosX, h.PosY, h.size,
                        (Color)
Class.forName("java.awt.Color").getField(h.color.toLowerCase()).get(null), h.action, exp);
                if (h.action.equals("Add")) circle.add(c);
                if (h.action.equals("Delete")){
                    Iterator<Circle> iter = circle.iterator();

                    while (iter.hasNext()) {
                        Circle ex = iter.next();

                        if (ex.equals(c)) {
                            System.out.println(ex);
                            iter.remove();
                        }
                }
                }
                exp.repaint();
            } catch (SocketTimeoutException e) {
                break;
            } catch (ClassNotFoundException e) {
                System.out.println("CLASS HUMAN NOT FOUND");
                break;
            } catch (PortUnreachableException ex) {
```

```
                System.out.println("SERVER IS NOT AVAILABLE AT THE MOMENT, PLEASE TRY AGAIN LATER");
            }  catch (NullPointerException e) {} catch (StreamCorruptedException e) {}
        }

    }

    public class ExpPanel extends JPanel {

        JFrame parent;

        public ExpPanel(JFrame parent) {
            this.parent = parent;
        }

        @Override
        public String getToolTipText(MouseEvent event) {
            //System.out.println(event.getX()+"  "+this.getX()+ "  "+event.getY()+"  "+this.getY());
            Point p = new Point(event.getX(), event.getY());
            for (Circle c : circle) {
                String t = tooltipForCircle(p, c);
                if (t != null) {
                    return t;
                }
            }
            this.repaint();
            return "";
            //return super.getToolTipText(event);
        }

        public String tooltipForCircle(Point p, Circle c) {
            if (c.contains(p)) {
                return "(" + c.getX() + ", " + c.getY() + ") " + c.getRadius() + " " + c.getColor()
+ ")";

            }
            return null;
        }

        @Override
        public void paint(Graphics g) {

            super.paint(g); //To change body of generated methods, choose Tools | Templates.
                        circle.forEach((c) -> { if (c.action.equals("Add"))
            c.draw(g);
            });
        }

    }

    public class ExpFrame extends JFrame {

        JComboBox comboBox = new JComboBox<>();
        ExpPanel panel = new ExpPanel(this);
        JButton button1 = new JButton();
        JButton button2 = new JButton();
        private JSlider jSlider1 = new JSlider();

        public ExpFrame() {
            initComponents();

            this.setVisible(true);
            panel.setToolTipText("");
        }

        private void initComponents() {

            int min = 0;
            int max = 100;
            int space = 50;
            JSlider jSlider1 = new JSlider(JSlider.HORIZONTAL, min, max, space);
            jSlider1.setMinorTickSpacing(5);
            jSlider1.setMajorTickSpacing(20);
            jSlider1.setPaintTicks(true);
            jSlider1.setPaintLabels(true);
```

```
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

            comboBox.setModel(new DefaultComboBoxModel<>(new String[]{"All", "Black",
                    "Blue", "Yellow", "Gray", "Green", "Orange", "Pink", "Red", "White"}));

            GroupLayout panelLayout = new GroupLayout(panel);
            panel.setLayout(panelLayout);
            panel.setBackground(Color.WHITE);
            panelLayout.setHorizontalGroup(
                    panelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addGap(0, 745, Short.MAX_VALUE)
            );
            panelLayout.setVerticalGroup(
                    panelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
                            .addGap(0, 473, Short.MAX_VALUE)
            );

            button1.setText("Start");
            button2.setText("Stop");

            GroupLayout layout = new GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(26, 26, 26)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(panel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 75, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(comboBox, javax.swing.GroupLayout.PREFERRED_SIZE, 149,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(26, 26, 26)
                        .addComponent(jSlider1, javax.swing.GroupLayout.PREFERRED_SIZE, 406,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(button2, javax.swing.GroupLayout.PREFERRED_SIZE, 92,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(button1, javax.swing.GroupLayout.PREFERRED_SIZE, 92,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(comboBox, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                        .addComponent(jSlider1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)
                            .addComponent(button2, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(button1, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(panel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
            );

            pack();


            button1.addActionListener(new ActionListener() {
```

```
                public void actionPerformed(ActionEvent e) {
                    String selectedColor = (String) comboBox.getSelectedItem();
                    int value = jSlider1.getValue();
                    Color colorx;
                    try {
                        colorx = (Color)
Class.forName("java.awt.Color").getField(selectedColor.toLowerCase()).get(null);
                    } catch (Exception ew) {
                        colorx = null; // Not defined
                    }


                    //  System.out.println(colorx);
                    for (Circle c : circle) {
                        if (selectedColor.equals("All")) {
                            if ((c.getRadius() == value)) {
                                EventQueue.invokeLater(new Runnable() {
                                    @Override
                                    public void run() {
                                        new Dissappear(ExpFrame.this.panel,c, button2);
                                    }
                                });
                            }
                            continue;
                        }
                        if ((colorx.getRGB() == c.getColor().getRGB()) && (c.getRadius() == value))
{
                            EventQueue.invokeLater(new Runnable() {
                                @Override
                                public void run() {
                                        new Dissappear(ExpFrame.this.panel,c, button2);
                                }
                            });
                        }
                    }
                }
            });
        }

        @Override
        public void paint(Graphics g) {
            super.paint(g);
//            circle.forEach((c) -> {
//                c.draw(panel.getGraphics());
//            });
        }
    }

};
```

#Circle.java

```
package clientudp;

import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.geom.Ellipse2D;


 class Circle implements Runnable {

        int x;
        int y;
        int radius;
        Color color;
        String action;
        ClientUDP7.ExpFrame frame;
        Ellipse2D ellip;
        float alpha_ellipse;
```

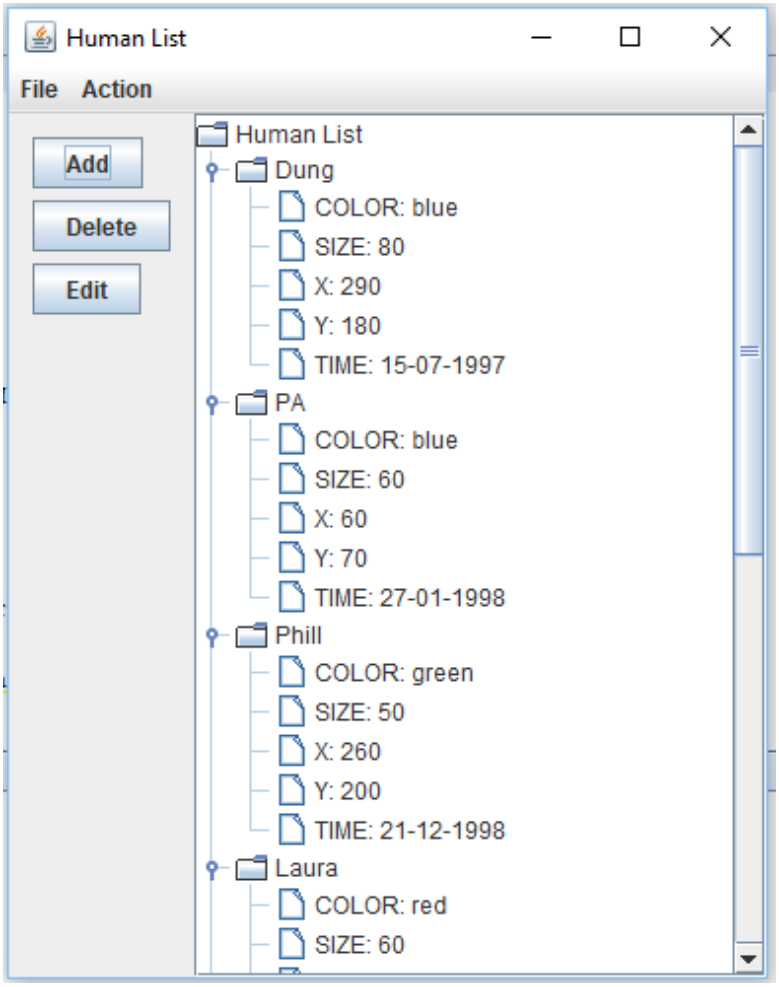
```
        public Circle(int x, int y, int radius, Color color, String action, ClientUDP7.ExpFrame
frame) {
            this.x = x;
            this.y = y;
            this.radius = radius;
            this.color = color;
            this.action = action;
            this.frame = frame;
            ellip = new Ellipse2D.Double(this.x-this.radius, this.y-this.radius, this.radius * 2,
this.radius * 2);
            alpha_ellipse = 1f;
        }

        public int getX() {
            return this.x;
        }

        public void setAction(String action){
            this.action = action;
        }

        public int getY() {
            return this.y;
        }

        public int getRadius() {
            return this.radius;
        }

        public Color getColor() {
            return this.color;
        }

        public void draw(Graphics g) {
            Graphics2D g2d = (Graphics2D) g.create();
            g2d.setColor(color);
            g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER,
                    alpha_ellipse));
            g2d.fill(this.ellip);

        }

        public boolean contains(Point p) {
            if (((p.x - this.x) * (p.x - this.x) + (p.y - this.y) * (p.y - this.y)) < radius *
radius) {
                return true;
            } else {
                return false;
            }
        }

        public void setAlpha(float alpha){
            this.alpha_ellipse = alpha;
        }

        @Override
        public void run() {
            draw(frame.panel.getGraphics());
        }

        @Override
        public boolean equals(Object obj){
             if (obj == null) {
                    return false;
            }

            if (!Circle.class.isAssignableFrom(obj.getClass())) {
                return false;
            }

            final Circle other = (Circle) obj;
            if ((this.x != other.x) || (this.y != other.y) || (this.radius != other.radius) ||
(this.color != other.color)) {
                return false;
```

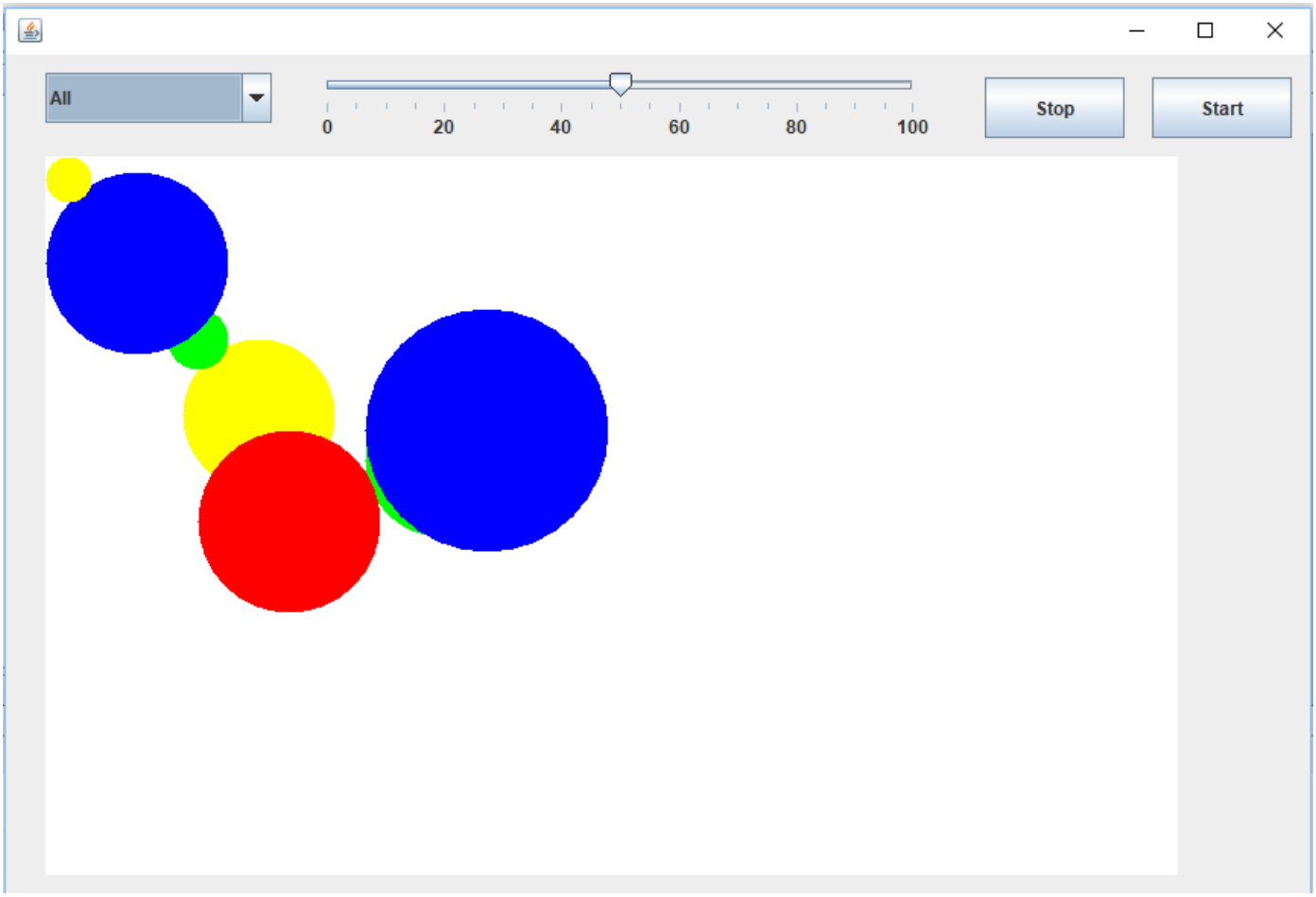
```
            }

            return true;
        }
    }
```

#Dissappear.java

```
package clientudp;

import com.sun.pisces.Surface;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JPanel;

class Dissappear implements Runnable {

        private Thread ellipseAnimator;
        private Circle c;
        private JPanel parent;
        private boolean ok;
        public Dissappear(JPanel parent,Circle c, JButton button2) {
            this.parent = parent;
            this.c = c;
            this.ok = true;
            button2.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    ok = false;
                }
            });
            ellipseAnimator = new Thread(this);
            ellipseAnimator.start();
        }

        @Override
        public void run() {
            while ((c.alpha_ellipse >= 0) && (ok)){
                parent.repaint();
                c.alpha_ellipse += -0.01f;
                if (c.alpha_ellipse <= 0) {
                    while ((c.alpha_ellipse <= 1f) && (ok)) {
                        parent.repaint();
                        c.alpha_ellipse += 0.01f;
                        if (c.alpha_ellipse > 1f) {
                            c.alpha_ellipse = 1f;
                        }
                        try {
                            Thread.sleep(40);
                        } catch (InterruptedException ex) {
                            Logger.getLogger(Surface.class.getName()).log(Level.SEVERE,
                            null, ex);
                        }
                    }
                }
                try {
                    Thread.sleep(50);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Surface.class.getName()).log(Level.SEVERE,
                            null, ex);
                }
            }
        }
    }
```

#Interface


Human List
File
Action
Add
Delete
Edit
Human List
Dung
COLOR: blue
SIZE: 80
X: 290
Y: 180
TIME: 15-07-1997
PA
COLOR: blue
SIZE: 60
X: 60
Y: 70
TIME: 27-01-1998
Phill
COLOR: green
SIZE: 50
X: 260
Y: 200
TIME: 21-12-1998
Laura
COLOR: red
SIZE: 60


All
0
20
40
60
80
100
Stop
Start