Университет ИТМО, кафедра ВТ

Лабораторная работа №5 по “Программированию интернет-приложений”

Работу выполнил

студент группы P3200

Рогов Я. С.

Преподаватель

Николаев В.В.

Санкт-Петербург, 2016

Задание: разделить приложение из лабораторной работы №4 на две составляющие - клиентскую и серверную, обменивающиеся сообщениями по заданному протоколу.

На стороне клиента осуществляются ввод и передача данных серверу, прием и отображение ответов от сервера и отрисовка области. В сообщении клиента должна содержаться вся необходимая информация для определения факта попадания/непопадания точки в область.

Сервер должен принимать сообщения клиента, обрабатывать их в соответствии с заданной областью и отправлять клиенту ответное сообщение, содержащее сведения о попадании/непопадании точки в область.

Приложение должно удовлетворять следующим требованиям:

- Для передачи сообщений необходимо использовать протокол UDP.
- Для данных в сообщении клиента должен использоваться тип float.
- Для данных в ответном сообщении сервера должен использоваться тип Integer.
- Каждое сообщение на сервере должно обрабатываться в отдельном потоке. Класс потока должен быть унаследован от класса Thread.
- Приложение должно быть локализовано на 2 языка - английский и шведский.
- Строки локализации должны храниться в текстовом файле.
- Приложение должно корректно реагировать на "потерю" и "восстановление" связи между клиентом и сервером; в случае недоступности сервера клиент должен показывать введённые пользователем точки серым цветом.

Client/Main.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
import java.awt.event.*;
import java.net.*
import java.io.*;
import java.lang.ClassNotFoundException;

import java.lang.IllegalArgumentException;
import java.lang.Thread;
import java.text.NumberFormat;
import java.util.*;
import java.util.stream.DoubleStream;

import FuncLib.*;

public class ProgramGUI extends JFrame{

  private JLabel jLabel;

  private final Dimension size = new Dimension(1100, 700);
  private final Dimension minSize = new Dimension(1100, 500);
  private final String title = "Lab4_v720";


  private final static Double[] x_values = DoubleStream.of(new double[]{
    -4.0f, -3.0f, -2.0f, 0.0f, 1.0f, 2.0f, 3.0f, 4.0f
  }).boxed().toArray(Double[]::new);

  private final static Double[] y_values = DoubleStream.of(new double[]{
    -4.0f, 2.0f, -3.0f, 4.0f, 5.0f, -1.0f, 0.0f
  }).boxed().toArray(Double[]::new);

  final private float r_SliderMin = 1.0f,
            r_SliderMax = 10.0f,
            r_SliderDefault = 3.0f,
            r_SliderDelta = 0.01f,
            r_SliderExtent = 2.0f;

  private JTextField dotCoordinates;
```

```
private JComboBox x_comboBox;
private MyJCheckBoxGroup y_checkBoxGroup;
private JButton addDotButton;
private JButton updateButton;
private ArrayList<Dot> dots;

private JButton languageButton;

private MyJSliderDouble r_Slider;
double currentR;

private Graph graph;
final private double graphBounds = 1.1 * r_SliderMax;

DrawableContourModel model;

static InetAddress serverAddress;
static int serverPort = 8822;

static DatagramSocket socket;
static int socketPort = 1994;
static int socketTimeout = 1000;
boolean serverAvailable = true;


static NumberFormat nfInstance = NumberFormat.getInstance();
ResourceBundle bundle;


static{
  nfInstance.setMaximumFractionDigits(2);

  try{
    socket = new DatagramSocket(1994);
    socket.setSoTimeout(3000);
    socketPort++;
  }
  catch(Exception e){
    System.out.println("[ERROR] Socket initialization exception: "+e);
  }

}

class Pinger extends Thread{
  public void run(){
    for(Dot d: dots)
      d.updateRadius(-1, false);
    JOptionPane.showMessageDialog(null, bundle.getString("serverUnavailable"));

    while(!serverAvailable)
      try{
        sendRequest(0.0f, 0.0f, 0.0f);
        serverAvailable = true;
      }
      catch(SocketTimeoutException e){}

    for(Dot d: dots)
      updateDot(d, currentR);
    JOptionPane.showMessageDialog(null, bundle.getString("serverAvailable"));
  }
}

public ProgramGUI(){ initGUI(); }

public void initGUI() {

  try{ serverAddress = InetAddress.getByName("localhost"); }
  catch (UnknownHostException e){
    System.out.println("[ERROR] Server domain address wasn't found");
    return;
  }

  try{bundle = ResourceBundle.getBundle("gui");}
  catch (MissingResourceException e)
  { System.out.println("[ERROR] No bundle found!");return; }

  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  this.setTitle(title);
  this.setSize(size);
  setMinimumSize(minSize);
```

```
  this.setLayout(new BorderLayout());

  dotCoordinates = new JTextField("(x; y)");
  dotCoordinates.setFont(dotCoordinates.getFont().deriveFont(20.0f));
  dotCoordinates.setEditable(false);
  this.add(dotCoordinates, BorderLayout.PAGE_START);

  JPanel controls = getControls();
  this.add(controls, BorderLayout.PAGE_END);

  model = initModel725(r_SliderDefault);
  DrawableContourView view = new DrawableContourView(2*graphBounds);

  graph = new Graph(view, dots);
  graph.setBounds(
        -graphBounds, graphBounds, 1.0d,
        -graphBounds, graphBounds, 1.0d);
  graph.setBGColor(new Color(0x63, 0x38, 0x07));
  graph.setFillColor(new Color(0xff, 0xff, 0x60));
  graph.addMouseListener(new MouseAdapter(){
    public void mouseClicked(MouseEvent e){
      addDot(  graph.translateAndScale(e.getX(), true),
          graph.translateAndScale(e.getY(), false),
          0.3);
    }
  });

  model.addObserver(graph);
  model.notifyObservers();
  this.add(graph, BorderLayout.CENTER);

  currentR = r_Slider.DgetValue();
}

private void addDot(double x, double y, double speed){
  Dot dot = new Dot(x, y, speed, graph, currentR);
  dots.add(dot);
  dot.start();
  dotCoordinates.setText(String.format("(%s; %s)", nfInstance.format(x), nfInstance.format(y)));
  updateDot(dot, currentR);
}

private JPanel getControls(){

  x_comboBox = new JComboBox<Double>(x_values);
  y_checkBoxGroup = new MyJCheckBoxGroup<Double>(y_values);

  addDotButton = new JButton(bundle.getString("addDotButton"));
  dots = new ArrayList<Dot>(15);
  addDotButton.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent e){
      Double x = (Double) x_comboBox.getSelectedItem();
      Double y = (Double) y_checkBoxGroup.getSelectedItem();
      addDot(2.2*x.doubleValue(), 1.1*y.doubleValue(), 0.3);
    }
  });

  updateButton = new JButton(bundle.getString("updateGraphButton"));
  updateButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
      currentR = r_Slider.DgetValue();
      for (Dot d: dots)
        updateDot(d, currentR);
      model.setR((float) currentR);
    }
  });

  languageButton = new JButton(bundle.getString("switchLanguageButton"));
  languageButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
      changeLanguage();
    }
  });
  r_Slider = new MyJSliderDouble(
    r_SliderMin, r_SliderMax, r_SliderDefault, r_SliderDelta);
  r_Slider.DcreateStandardLabels(r_SliderExtent);
  r_Slider.setPaintLabels(true);

  JPanel controls = new JPanel(new FlowLayout());
```

```
  controls.add(x_comboBox);
  controls.add(y_checkBoxGroup);
  controls.add(addDotButton);
  controls.add(r_Slider);
  controls.add(updateButton);
  controls.add(languageButton);
  return controls;
}

private void updateDot(Dot d, double R){
  try{
    boolean response = false;
    if (serverAvailable){
      response = sendRequest(
        (float) d.getX(),
        (float) d.getY(),
        (float) R);
      d.updateRadius(R, response);
    }
  }
  catch(SocketTimeoutException e){
    serverAvailable = false;
    new Pinger().start();
  }
}

private static boolean sendRequest(float x, float y, float R) throws SocketTimeoutException{
  ByteArrayOutputStream byteStream = new ByteArrayOutputStream(3 * 4);
  DataOutputStream dataStream = new DataOutputStream(byteStream);

  try{
    dataStream.writeFloat(x);
    dataStream.writeFloat(y);
    dataStream.writeFloat(R);

    byte[] buf = byteStream.toByteArray();
    DatagramPacket request = new DatagramPacket(buf, buf.length, serverAddress, serverPort);
    System.out.println(String.format("[INFO] Sending request (%s; %s), R=%s",
                               nfInstance.format(x), nfInstance.format(y), nfInstance.format(R)));
    socket.send(request);

    buf = new byte[81];
    DatagramPacket response = new DatagramPacket(buf, buf.length);
    ObjectInputStream dataOutput = null;

    socket.receive(response);
    dataOutput = new ObjectInputStream(new ByteArrayInputStream(buf));
    Integer value = (Integer) dataOutput.readObject();

    System.out.println(String.format("[INFO] Response for request [(%s; %s), R=%s]: %s",
                                 nfInstance.format(x),
                                 nfInstance.format(y),
                                 nfInstance.format(R),
                                 nfInstance.format(value)));
    return value==0?false:true;
  }
  catch(SocketTimeoutException e){
    throw e;
  }
  catch(IOException e){
    System.out.println(e);
  }
  catch(ClassNotFoundException e){
    System.out.println(e);
  }
  return false;
}

private void changeLanguage(){
  Locale swedish = new Locale("swe");
  if ("" == bundle.getLocale().getLanguage())
    bundle = ResourceBundle.getBundle("gui", swedish);
  else
    bundle = ResourceBundle.getBundle("gui");
  addDotButton.setText(bundle.getString("addDotButton"));
  updateButton.setText(bundle.getString("updateGraphButton"));
  languageButton.setText(bundle.getString("switchLanguageButton"));
}
```

```
  private DrawableContourModel initModel725(float defaultR){
    DrawableContourModel model = new DrawableContourModel(
      defaultR,
      null,
      (float R) -> {Path2D.Double result = new Path2D.Double();
        result.append(new Arc2D.Double(-R, -R, 2*R, 2*R, 270, -90, Arc2D.OPEN), true);
        result.append(new Line2D.Double(-R/2, 0, 0, -R), true);
        result.append(new Rectangle2D.Double(0, -R/2, R, R/2), true);
        return result;
      },
      1000.0f
    );

    return model;
  }
}
```

Client/Dot.java

```
import FuncLib.*;

import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.Color;

public class Dot extends Thread {

  private boolean isAnimated;
  private double radius=1.0;
  private Vertice vertice;
  private double speed;

  private Color color = Color.BLACK;

  private Graphics2D graphics2D;
  private Graph graph;

  private boolean isDownscaleAnim;
  private double r;

  public Dot(double x, double y, double speed, Graph graph, double R){
    vertice = new Vertice(x,y);
    this.speed = speed;
    this.graph = graph;
    isDownscaleAnim = true;
    radius = R;
    r = radius/12;
    color = Color.DARK_GRAY;
    isAnimated = false;
  }

  public void updateRadius(double d, boolean withinBounds){

    if(d>=0){
      radius = d;
      isAnimated = withinBounds;

      color = isAnimated?Color.WHITE:Color.BLACK;
    }
    else{
      isAnimated=false;
      color = Color.DARK_GRAY;
    }
    r = radius/12;
  }

  public double getX() {return vertice.x;}
  public double getY() {return vertice.y;}

  public void setGraphics2D(Graphics2D g){ graphics2D = g; }

  public Color getColor(){ return color; }

  public void updateDotSize(){
    if(isAnimated){
      if (isDownscaleAnim){
        r-=speed*radius/22;
        if(r<radius/22)
          isDownscaleAnim = false;
      }
      else{
```

```
        r+=speed*radius/22;
        if(r>radius/12)
          isDownscaleAnim = true;
      }
    }
  }

  public void paintDot(Graphics2D g, double scaleDiv){
    Rectangle rec = g.getClipBounds();

    g.fillOval(  (int)((vertice.x - r/2)*scaleDiv),
          (int)((-vertice.y - r/2)*scaleDiv),
          (int)(2*r*scaleDiv),
          (int)(2*r*scaleDiv));
  }

  public void run(){
    while(true){
      updateDotSize();
      graph.repaint();
      try{
        this.sleep(100);}
      catch(InterruptedException e){}
    }
  }

}
```

Server/Main.java

```
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.SocketAddress;
import java.net.SocketException;

import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.IOException;

public class Main {

  static DatagramSocket socket;

  public static void main(String[] args){

    try{socket = new DatagramSocket(8822);}
    catch(SocketException e){
      System.out.println("Socket initialization error!\n" + e);
    }

    System.out.println("[INFO] Server started");
    try{
      while(true){
        byte[] buf = new byte[3*4];
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        DataInputStream dataStream = new DataInputStream(new ByteArrayInputStream(buf));

        socket.receive(packet);

        float x = dataStream.readFloat();
        float y = dataStream.readFloat();
        float R = dataStream.readFloat();
        System.out.println(
          String.format("[INFO] Received packet: (%.2f; %.2f) with R=%.2f", x, y, R));
        new CheckThread(x, y, R, packet.getSocketAddress()).run();
      }
    }
    catch (IOException e){
      System.out.println("[ERROR] Exception during receiving!\n\t" + e);
    }
  }
}
```

Server/CheckThread.java

```
import FuncLib.*;

import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.io.IOException;

import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.SocketAddress;
import java.net.SocketException;


public class CheckThread extends Thread{

    DrawableContourModel model;
    Vertice v;
    SocketAddress socketAddress;

    public static void sendPacket(SocketAddress socketAddress, Integer value){
      ByteArrayOutputStream byteStream = new ByteArrayOutputStream();
      try(ObjectOutputStream objStream = new ObjectOutputStream(byteStream)){
        objStream.writeObject(value);
      }
      catch(IOException e){
        System.out.println("[ERROR] Object serializing exception!\n\t" + e);
      }

      byte[] b = byteStream.toByteArray();
      DatagramPacket packet = new DatagramPacket(b, b.length, socketAddress);
      try (DatagramSocket socket = new DatagramSocket(8823)){
        socket.send(packet);
      }
      catch(IOException e){
        System.out.println("[ERROR] Exception during sending!\n\t" + e);
      }
    }

    private static DrawableContourModel initModel725(float defaultR){
      DrawableContourModel model = new DrawableContourModel(
        defaultR,

        (float R, Vertice v) ->
          {
            double x = v.x, y = v.y;
            return x>0?
                x<=R && y>=0 && y<=R/2:
                y>0?
                  2*x+R>y:
                  x*x+y*y <= R*R;
          },null, 0.0f);
      return model;
    }

    CheckThread(float x, float y, float R, SocketAddress socketAddress){
      this.model = initModel725(R);
      v = new Vertice(x,y);
      this.socketAddress = socketAddress;
    }

    public void run(){
      boolean withinBounds = model.checkVertice(v);
      Integer response = new Integer(withinBounds? 1:0);
      System.out.println(String.format("[INFO] Response for (%.2f; %.2f): %d", v.x, v.y, response));
      sendPacket(socketAddress, response);
    }
  }
```

Вывод: в ходе выполнения данной лабораторной работы я изучил основы интернационализации, а также особенности локализации приложения в Java. Также было изучено сетевое взаимодействие посредством протокола UDP и TCP и передача сериализованных объектов при помощи сети.