Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных технологий

Кафедра информационных технологий

Контрольная работа

по курсу

Программирование сетевых приложений

Выполнил:

Студент-заочник 3 курса

Группы № 382372

ФИО Григорук

Александр Владимирович

Адрес г. Минск,

ул. Почтовая д. 3 - 64

Тел. +375 29 694 41 53

E-mail: [thanksforsource@gmail.com](mailto:thanksforsource@gmail.com)

Минск, 2016

1. ЗАДАНИЕ НА КОНТРОЛЬНУЮ РАБОТУ

1. Разработать сайт компании по оказанию образовательных услуг. К разработанному сайту составить диаграммы UML: вариантов использования, состояний, деятельности. Сайт должен содержать модуль JavaScript, позволяющий клиенту записаться в режиме online на тот или иной вид образовательной услуги. Данная запись клиента сохраняется в определенном файле.

2. Разработать GUI-приложение с организацией взаимодействия с БД, реализующий методику расчета показателей для оценки ликвидности и платежеспособности предприятия.

3. Разработать апплет, реализующий в верхней части экрана дисплея изображения тучи, а в нижней части экрана дисплея - емкость для воды. При запуске программы начинает идти дождь. При этом размер тучи уменьшается, a емкость наполняется водой.

4. Разработать в архитектуре клиент-сервер приложение для выбора типа самолета для перевозки группы грузов. Тип сетевого соединения – UDP. При выборе типа самолета исходить из того, что грузы являются неделимыми. Для перевозки может использоваться только один самолет. Груз менее 5 т способен перевозить «ТУ-134». Вес более 5 т, но менее 30 т – «ИЛ-86». Вес более 30 т, но менее 45 т – «Руслан». Исходная информация о грузоподъемности самолетов хранится в текстовом файле на сервере. Клиент посылает массу груза на сервер, назад получает результат. Клиент имеет возможность просмотра, редактирования, удаления информации на сервере.

**ЗАДАНИЕ №1**

Разработать сайт компании по оказанию образовательных услуг. К разработанному сайту составить диаграммы UML: вариантов использования, состояний, деятельности. Сайт должен содержать модуль JavaScript, позволяющий клиенту записаться в режиме online на тот или иной вид образовательной услуги. Данная запись клиента сохраняется в определенном файле.

Предметная область – учреждение дополнительного образования «Учимся вместе». Для выполнения данной работы использовались:

1. JS (JavaScript) – для создания заказа и его сохранения в файл

2. XML – для хранения информации о курсах

3. XSL – для обработки (парсинга) данных из xml и преобразование в html

4. HTML – непосредственно разметка страницы

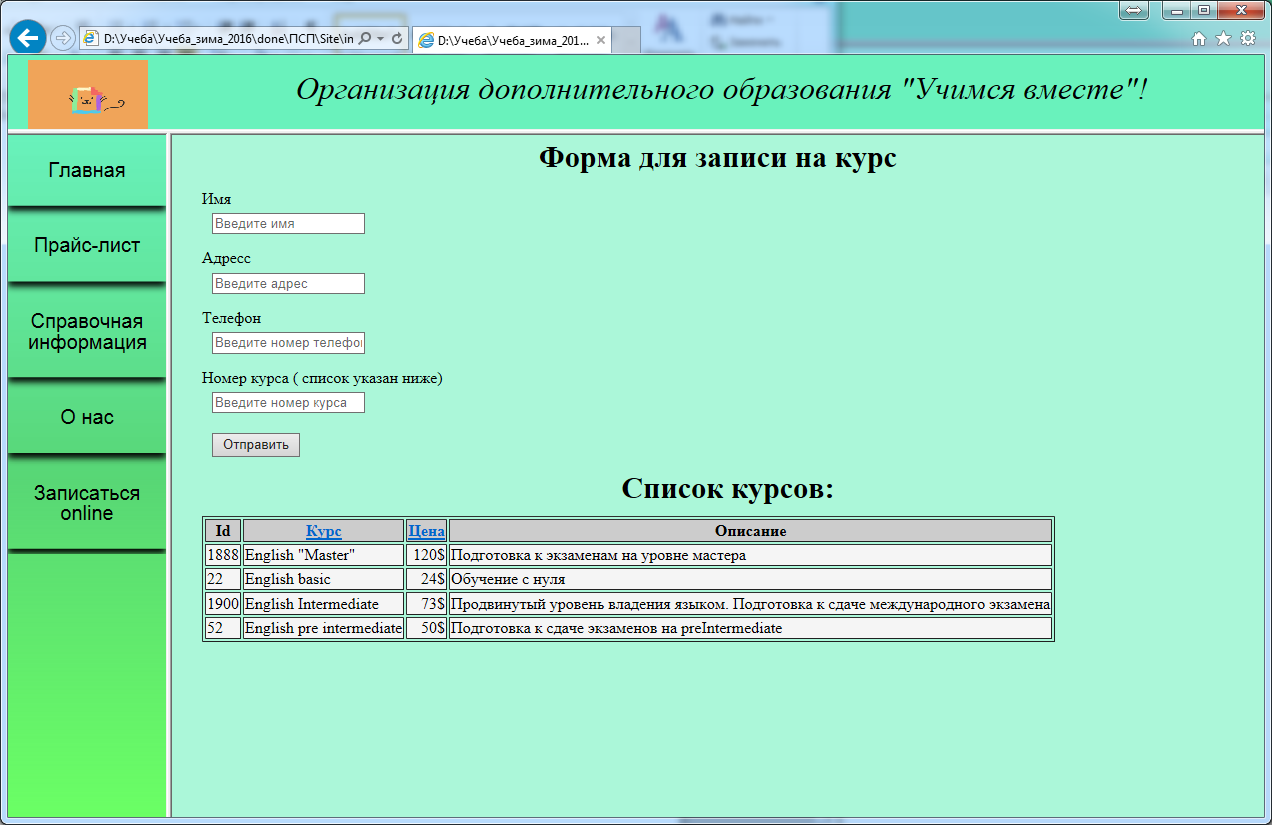


Рисунок 1.1 «Страница записи на курсы»

Диаграмма 1.1 «UML диаграмма вариантов использования»

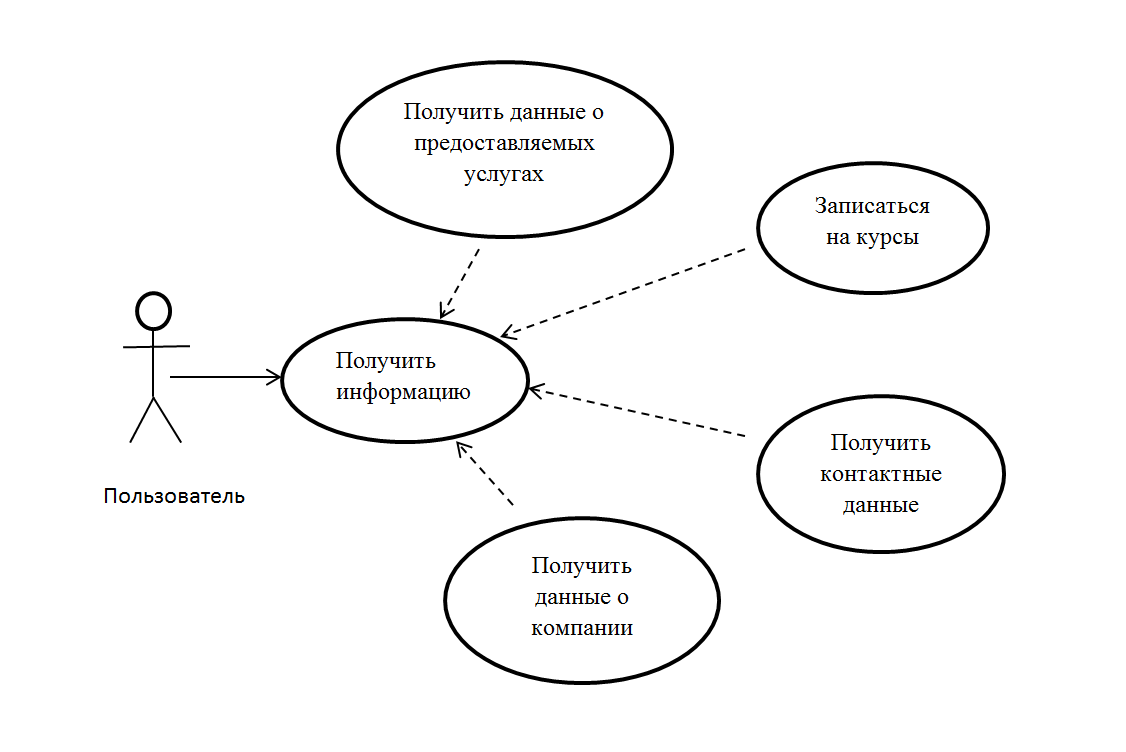
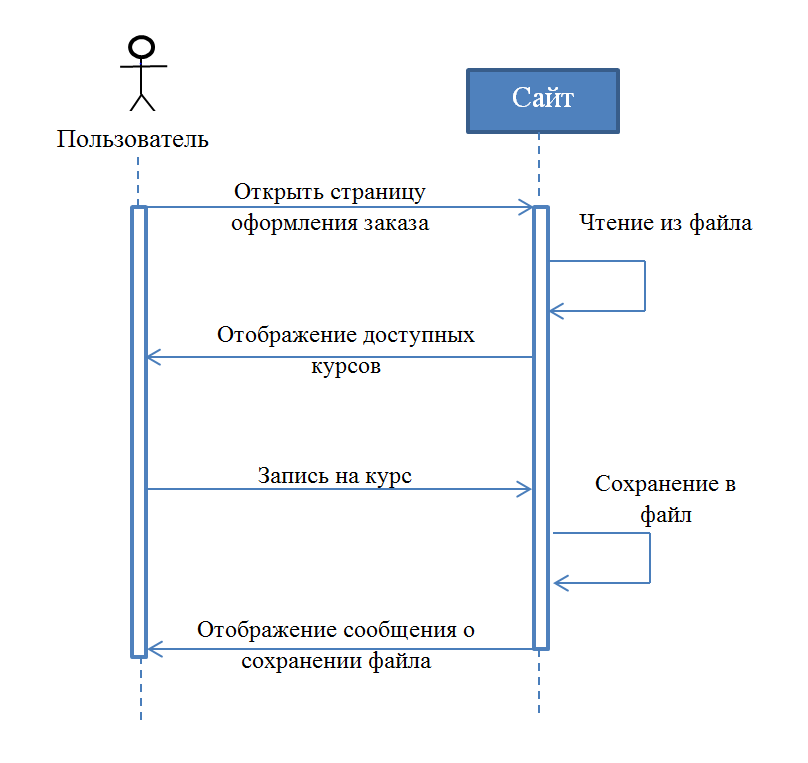


Диаграмма 1.2 «UML диаграмма последовательности»



# Пример кода из задания №1

JavaSript – обработка события по нажатию кнопки «Отправить»

$('#createOrder').click(function(event) {

var name = $("#name").val();

var address = $('#address').val();

var phone = $('#phone').val();

var kurs = $('#kurs').val();

if (count==0){if (name=="" || address=="" || phone=="" || kurs=="") {alert ("Please, fill the fileds"); }

else {var blob = new Blob(["Name : "+name +"\r\nAddress : "+address +"\r\nPhone Number: "+phone +"\r\nSelected course :"+kurs] ,{type: "text/plain;charset=utf-8"});

saveAs(blob, "Yours\_course\_ticket-"+name+".txt");

count++; }

}

else{

alert("You already have your's ticket. Choose other course"); }

});

**ЗАДАНИЕ №2**

. Разработать GUI-приложение с организацией взаимодействия с БД, реализующий методику расчета показателей для оценки ликвидности и платежеспособности предприятия.

Ликвидность Баланса — соотношение активов и пассивов, обеспечивающее своевременное покрытие краткосрочных обязательств оборотными активами. Анализ ликвидности баланса заключается в сравнении средств по активу, сгруппированных по степени их ликвидности, с обязательствами по пассиву, сгруппированными по срокам их погашения.

Активы делятся на четыре группы:

* А1 — наиболее ликвидные активы;
* А2 — быстрореализуемые активы;
* А3 — медленно реализуемые активы;
* А4 — труднореализуемые активы.

Пассивы группируются по степени срочности их оплаты:

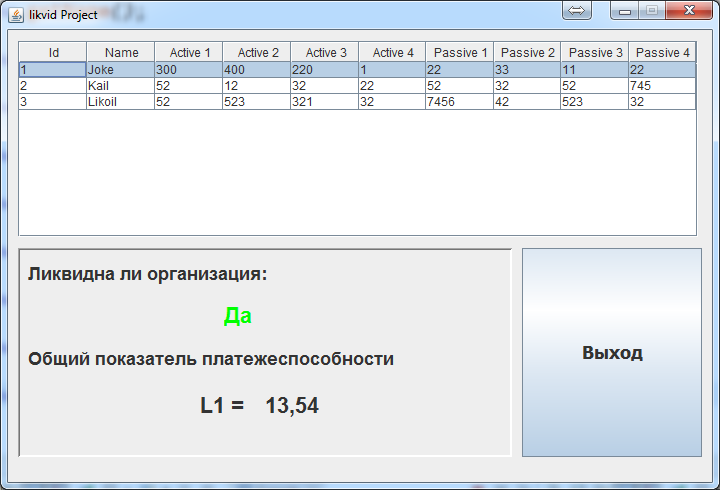
* П1 — наиболее срочные обязательства;
* П2 — краткосрочные пассивы;
* П3 — долгосрочные пассивы;
* П4 — постоянные пассивы.

Для определения ликвидности Баланса следует сопоставить итоги соответствующих групп активов и пассивов. Баланс считается абсолютно ликвидным, если одновременно выполняются следующие условия:

* A1>=П1,
* A2>=П2,
* A3>=П3,
* (!) A4<=П4

Для комплексной оценки платежеспособности используют общий показатель платежеспособности, исчисляемый по формуле: L1 = ( A1 + 0.5 A2 + 0.3 A3 ) / (П1 + 0.5 П2 + 0.3 П3).

Внешний вид программы представлен на рисунке 2.1.

  
Рисунок 2.1 «Главное окно программы»

В основе работы данной программы лежит взаимодействие с БД. Мы используем JDBC драйвер и базу данных MySQL . Для данной программы мы создали структуру из 3ех таблиц. Результат запроса к ним можно увидеть в главном окне программы. Опишем данные таблицы и их поля:

1. Organization
   1. Id (int, PK,NotNull);
   2. Name (varchar());
2. Actives
   1. Id(int,PK,NotNull);
   2. OrganiationId(int, FK,NotNull);
   3. A1
   4. A2
   5. A3
   6. A4
3. Passives
   1. Id(int,PK,NotNull);
   2. OrganiationId(int, FK,NotNull);
   3. P1
   4. P2
   5. P3
   6. P4

Листинг программы:

public class MainForm {

private JFrame frame;

private JTable table;

private List<TableClass> tableDataList = new ArrayList<TableClass>();

private final Action action = new SwingAction();

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

DataBaseProvider.GetInstance();

MainForm window = new MainForm();

} catch (Exception e) { e.printStackTrace();}}});}

public MainForm() {

ShowTableFromDb();

initialize();

}

private void initialize() {

frame = new JFrame();

frame.setTitle("likvid Project");

frame.setPreferredSize(new Dimension(710, 480));

frame.setBounds(100, 100, 710, 485);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

TableModel model = new MyTableModel(tableDataList);

table = new JTable(model);

table.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

table.setFillsViewportHeight(true);

table.setBounds(10, 11, 680, 196);

JScrollPane scrollPane = new JScrollPane(table);

scrollPane.setBounds(10, 11, 680, 196);

frame.getContentPane().add(scrollPane);

JPanel panel = new JPanel();

panel.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null, null));

panel.setBounds(10, 218, 494, 209);

frame.getContentPane().add(panel);

panel.setLayout(null);

JLabel label = new JLabel(

"\u041B\u0438\u043A\u0432\u0438\u0434\u043D\u0430 \u043B\u0438 \u043E\u0440\u0433\u0430\u043D\u0438\u0437\u0430\u0446\u0438\u044F:");

label.setFont(new Font("Arial", Font.BOLD, 18));

label.setBounds(10, 11, 330, 30);

panel.add(label);

JLabel PayableValueLabel = new JLabel("0.0");

PayableValueLabel.setFont(new Font("Arial", Font.BOLD, 22));

PayableValueLabel.setBounds(247, 131, 108, 52);

panel.add(PayableValueLabel);

JLabel label\_1 = new JLabel(

"\u041E\u0431\u0449\u0438\u0439 \u043F\u043E\u043A\u0430\u0437\u0430\u0442\u0435\u043B\u044C \u043F\u043B\u0430\u0442\u0435\u0436\u0435\u0441\u043F\u043E\u0441\u043E\u0431\u043D\u043E\u0441\u0442\u0438");

label\_1.setFont(new Font("Arial", Font.BOLD, 18));

label\_1.setBounds(10, 93, 410, 36);

panel.add(label\_1);

JLabel IsLikvidLabel = new JLabel("None");

IsLikvidLabel.setFont(new Font("Arial", Font.BOLD, 22));

IsLikvidLabel.setBounds(206, 41, 55, 52);

panel.add(IsLikvidLabel);

JLabel label\_3 = new JLabel("L1 = ");

label\_3.setFont(new Font("Arial", Font.BOLD, 22));

label\_3.setBounds(182, 131, 55, 52);

panel.add(label\_3);

JButton exitButton = new JButton("\u0412\u044B\u0445\u043E\u0434");

exitButton.setFont(new Font("Tahoma", Font.BOLD, 18));

exitButton.setBounds(514, 218, 180, 209);

exitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

frame.setVisible(false);

System.exit(0);

}

});

ListSelectionModel selModel = table.getSelectionModel();

selModel.addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent e) {

String result = "";

int[] selectedRows = table.getSelectedRows();

for (int i = 0; i < selectedRows.length; i++) {

int selIndex = selectedRows[i];

TableModel model = table.getModel();

Object value = model.getValueAt(selIndex, 0);

if (CheckLikvid(model, selIndex)) {

IsLikvidLabel.setForeground(Color.GREEN);

IsLikvidLabel.setVisible(true);

IsLikvidLabel.setText("\u0414\u0430");

} else {

IsLikvidLabel.setForeground(Color.RED);

IsLikvidLabel.setVisible(true);

IsLikvidLabel.setText("\u041D\u0435\u0442");

}

double res = CalculatePayability(model, selIndex);

PayableValueLabel.setText(String.format("%.2f", res));

}

}

});

frame.getContentPane().add(exitButton);

frame.setLocationRelativeTo(null);

frame.pack();

frame.setResizable(false);

frame.setVisible(true);

}

public double CalculatePayability(TableModel model, int selIndex) {

int a1 = (int) model.getValueAt(selIndex, 2);

int a2 = (int) model.getValueAt(selIndex, 3);

int a3 = (int) model.getValueAt(selIndex, 4);

int p1 = (int) model.getValueAt(selIndex, 6);

int p2 = (int) model.getValueAt(selIndex, 7);

int p3 = (int) model.getValueAt(selIndex, 8);

double tmp = a1 + 0.5 \* a2 + 0.3 \* a3;

double tmp2 = p1 + 0.5 \* p2 + 0.3 \* p3;

return tmp / tmp2;

}

public boolean CheckLikvid(TableModel model, int selIndex) {

int a1 = (int) model.getValueAt(selIndex, 2);

int a2 = (int) model.getValueAt(selIndex, 3);

int a3 = (int) model.getValueAt(selIndex, 4);

int a4 = (int) model.getValueAt(selIndex, 5);

int p1 = (int) model.getValueAt(selIndex, 6);

int p2 = (int) model.getValueAt(selIndex, 7);

int p3 = (int) model.getValueAt(selIndex, 8);

int p4 = (int) model.getValueAt(selIndex, 9);

if (a1 >= p1 && a2 >= p2 && a3 >= p3 && a4 <= p4) {

return true;

}

return false;

}

public void ShowTableFromDb() {

try {

Statement s = DataBaseProvider.GetNewStatement();

String sql = "Select org.\*,act.a1,act.a2,act.a3,act.a4," + "pas.p1,pas.p2,pas.p3,pas.p4 "

+ "from likvid.organization as org "

+ "INNER JOIN likvid.actives as act ON act.OrganizationId = org.Id "

+ "INNER JOIN likvid.passives as pas ON pas.OrganizationId = org.Id ";

ResultSet rs = s.executeQuery(sql);

ResultSetMetaData md = rs.getMetaData();

// get counts of the columns

int columns = md.getColumnCount();

if (tableDataList != null)

tableDataList.clear();

// read every row with .next

while (rs.next()) {

TableClass p = new TableClass(rs);

tableDataList.add(p);

}

rs.close();

s.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

public class MyTableModel extends AbstractTableModel {

private List<TableClass> rows;

@Override

public int getRowCount() {

return rows.size(); }

@Override

public int getColumnCount() {

return 10;

}

public Class<?> getColumnClass(int columnIndex) {

return String.class;

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

@Override

public Object getValueAt(int rowIndex, int colIndex) {

TableClass row = rows.get(rowIndex);

switch (colIndex) {

case 0: return row.getId();

case 1: return row.getName();

case 2: return row.getActive1();

case 3: return row.getActive2();

case 4: return row.getActive3();

case 5: return row.getActive4();

case 6: return row.getPassive1();

case 7: return row.getPassive2();

case 8: return row.getPassive3();

case 9: return row.getPassive4();

} return "";

}

public MyTableModel(List<TableClass> data) {

this.rows = data;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0: return "Id";

case 1: return "Name";

case 2: return "Active 1";

case 3: return "Active 2";

case 4: return "Active 3";

case 5: return "Active 4";

case 6: return "Passive 1";

case 7: return "Passive 2";

case 8: return "Passive 3";

case 9: return "Passive 4";

}

return "";

}

}

**ЗАДАНИЕ №3**

Разработать апплет, реализующий в верхней части экрана дисплея изображения тучи, а в нижней части экрана дисплея - емкость для воды. При запуске программы начинает идти дождь. При этом размер тучи уменьшается, a емкость наполняется водой.

При выполнении задания № 3 использовалась технология javaapplet. Java-апплет — прикладная программа, чаще всего написанная на языке программирования Java в форме байт-кода. Java-апплеты выполняются в веб-браузере с использованием виртуальной Java машины (JVM), или в Sun's AppletViewer, автономном средстве для испытания апплетов. Java-апплеты были внедрены в первой версии языка Java в 1995 году. Java-апплеты обычно пишутся на языке программирования Java, но могут быть написаны и на других языках, которые компилируются в байт-код Java, таких, как Jython.

Апплеты используются для предоставления интерактивных возможностей веб-приложений, которые не могут быть предоставлены HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться с помощью плагинов браузерами многих платформ, включая Microsoft Windows, UNIX, Apple Mac OS и др.

В данной задаче использовались библиотеки для рисования графики (использовались классы JApplet, Graphic, Graphic2D).

Рисование сложных фигур (облака и ведра) использовались примитивы, доступные в библиотеках графики java.awt и java.swing. Это рисования эллипсов, прямоугольников и линий. На рисунке 3.1. отображен кадр из работы данного апплета.

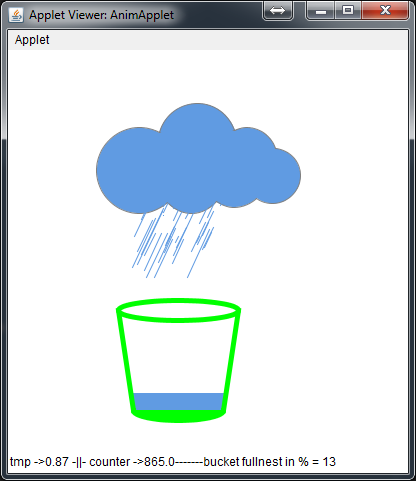


Рисунок 2.1 «Апплет дождя»

Для анимации необходимо постоянно обновлять сцену и менять правила рисования элементов. Задержка между кадрами осуществляется с помощью функции sleep , которая останавливает поток выполнения на указанное кол-во миллисекунд. В качестве основного элемента используется счетчик, состояние которого и является качественным показателем для изменения правил рисования кадра.

Капли дождя представлены классом RainDrop, объект которого реализует каплю. У нее есть стартовая позиция, конечная позиция (т.е. максимальная длинна пути) и время жизни.

Т.к. капли не могут быть постоянно на холсте , они со временем исчезают. Именно для этого и предназначено время жизни капель. Фактически это поле, которое может быть в 2ух состояниях – капля еще «жива» или капля уже завершила свой путь. Благодаря такому механизму, в процессе рисования нового кадра, в цикле , который рисует капли, присутствуют только те капли, которые еще падают. Это позволяет освободить память от ненужных капель, а также не заполняет холст излишними элементами.

С течением времени изменяется размер облака и заполняется водой ведро. Их анимация построена на том же принципе, что и капли дождя.

**Листинг программы: AnimApplet.java**

public class AnimApplet extends JApplet {

Random r = new Random();

public AnimApplet() {

System.out.println("Constructor");

}

public class RainDrop{

public int startPointX,startPointY,endPointX,endPointY;

double angle;

public int tmpLength,length;

boolean endOfLife =false;

Color color;

public RainDrop(int x1,int y1,int x2,int y2,int l,double a)

{

// for rain zone - (x1,y1) to (x2,y2)

startPointX = x1+r.nextInt(Math.abs(x2-x1));

startPointY = y1+r.nextInt(Math.abs(y2-y1));

angle = a;

length = l;

}

public void slide()

{

endPointX = (int) (startPointX + Math.cos(Math.PI/180\*angle)\*tmpLength++);

endPointY = (int) (startPointY - Math.sin(Math.PI/180\*angle)\*tmpLength++);

if (tmpLength>=length)

endOfLife = true;

}

}

ArrayList<RainDrop> rain;

ArrayList<RainDrop> removeRainList;

Image offimage;

Graphics offg;

Area area;

Area area2;

double counter = 1000;

public void init() {

rain = new ArrayList<RainDrop>() ;

new Thread() {

@Override

public void run() {

while (true) {

repaint();

try {

sleep(50);

if (counter==0) {

showStatus("Bucket is full");

sleep(10000);

counter=1000;

}

} catch (InterruptedException e) {

e.printStackTrace(); //To change body of catch statement use File | Settings | File Templates.

}

}

}

}.start();

}

public void paint(Graphics g) {

// заного рисуем облако ( оно исчезнет как фон, если этого не сделать)

area = drawSkyBallon(90,50);

update(g);

}

public void update(Graphics g) {

double tmpProcent = BigDecimal.valueOf(counter/1000).setScale(2, BigDecimal.ROUND\_HALF\_DOWN).doubleValue();

if (offimage == null) {

offimage = createImage(getWidth(), getHeight());

offg = offimage.getGraphics();

}

if (rain.size()>0)

{

// очищаем экран от капель, которые уже были очищены

offg.clearRect(0,0,getWidth(),getHeight());

}

// берем "холст" который получился с прошлой итерации, как основу

Graphics2D g2d = (Graphics2D)offg;

// сглаживаем углы, чтобы рисуемые детали двигались более "гладко"

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING,

RenderingHints.VALUE\_ANTIALIAS\_ON);

// переопределяем заного т.к. мы уже очистили нужные капли

removeRainList = new ArrayList<RainDrop>() ;

// трансформируем облако - уменьшая его зум - возьмем 1.4 от настоящего и постепенно уменьшаем

if (tmpProcent>0.70) {

AffineTransform at = new AffineTransform();

at.scale(tmpProcent \* 1.4, tmpProcent \* 1.4);

at.translate(70 - 100\*(tmpProcent), 20 - tmpProcent \* 30);

area = area.createTransformedArea(at);

}

// зарисуем облако - сначала заполнение

g2d.setPaint (new Color(96, 155, 226));

g2d.fill ( area );

// затем - края полученной формы ( т.к. это area все круги объединены как 1 объект union)

g2d.setPaint ( Color.GRAY );

g2d.draw ( area );

g2d.setColor(new Color(96, 155, 226));

rain.add(new RainDrop(140,100,220,190,40,-115));

rain.add(new RainDrop(140,120,220,190,60,-115));

for (RainDrop drop : rain){

drop.slide();

g2d.drawLine(drop.startPointX,drop.startPointY,drop.endPointX,drop.endPointY);

if (drop.endOfLife)

// если жизненный цикл капли окончен - добавляем ее на удаление

removeRainList.add(drop);

}

// убираем из цикла те капли, которые уже "закончили свой путь"

rain.removeAll(removeRainList);

// рисуем картинку , которая получилась в итоге

if (counter>0)

counter--;

showStatus("tmp ->"+tmpProcent+" -||- counter ->"+counter+"-------bucket fullnest in % = "+ (int)((1-tmpProcent)\*100));

// draw GeneralPath (polygon)

drawWater(112,40,40,100,(1-tmpProcent));

drawBucket(110,40,new Color(0,255,0));

g.drawImage(offimage, 0, 0, this);

}

private Area drawSkyBallon(double startPosX,double startPosY)

{ // use Graphic2d

Area area = new Area();

//левый крайний

area.add ( new Area ( new Ellipse2D.Double (startPosX+0,startPosY+20,70,70) ) );

// средний верхний

area.add ( new Area ( new Ellipse2D.Double (startPosX+50,startPosY+0,65,65) ) );

// средний нижний (сразу после большого левого)

area.add ( new Area ( new Ellipse2D.Double (startPosX+50,startPosY+35,55,55) ) );

// средний правый (следующий после предыдущего)

area.add ( new Area ( new Ellipse2D.Double (startPosX+85,startPosY+30,55,55) ) );

// третий (правый) верхний ( после большого среднего)

area.add ( new Area ( new Ellipse2D.Double (startPosX+98,startPosY+20,50,50) ) );

// маленький крайний правый

area.add ( new Area ( new Ellipse2D.Double (startPosX+122,startPosY+37,45,45) ) );

return area;

}

private void drawBucket(int startPosX,int startPosY,Color color){

Graphics2D g = (Graphics2D)offg;

//нарисуем стакан (ведро)

Color tmp = g.getColor();

g.setColor(color);

BasicStroke pen1 = new BasicStroke(5); //толщина линии 5

g.setStroke(pen1);

// верхушка

g.drawOval(startPosX+0,startPosY+210,120,20);

//левая линия

g.drawLine(startPosX+0,startPosY+220,startPosX+15,startPosY+320);

//правая линия

g.drawLine(startPosX+120,startPosY+220,startPosX+105,startPosY+320);

//линия дна

//g.drawLine(startPosX+15,320,startPosX+105,320);

g.drawArc(startPosX+15, startPosY+310, 90, 20, 0, -180);

g.fillArc(startPosX+15, startPosY+310, 90, 20, 0, -180);

pen1 = new BasicStroke(1); //толщина линии 1

g.setStroke(pen1);

g.setColor(tmp);

}

public void drawWater(int startX,int startY,int w, int h,double proc)

{

int x1Points[] = {};

int y1Points[] = {};

if (proc >= 0.1 && proc <= 0.4)

{

// 20%

x1Points = new int[] {startX+9, startX+104, startX+102,startX+15 };

y1Points = new int[] {startY+303, startY+303, startY+320,startY+320};

}

if (proc >0.4 && proc <= 0.6)

{

// 40%

x1Points = new int[] {startX+8, startX+107, startX+102,startX+15 };

y1Points = new int[] {startY+283, startY+283, startY+320,startY+320};

}

if (proc > 0.6 && proc <= 0.8)

{

// 60%

x1Points = new int[] {startX+6, startX+110, startX+102,startX+15 };

y1Points = new int[] {startY+263, startY+263, startY+320,startY+320};

}

if (proc >0.8 && proc <= 0.9)

{

// 80%

x1Points = new int[] {startX+4, startX+115, startX+102,startX+15 };

y1Points = new int[] {startY+243, startY+243, startY+320,startY+320};

}

if(proc>0.9) {

// 100%

x1Points = new int[] {startX+0, startX+115, startX+102,startX+15 };

y1Points = new int[] {startY+223, startY+223, startY+320,startY+320};

}

Graphics2D g = (Graphics2D)offg;

if (proc > 0.1) { // if 0 % - nothing to paint

GeneralPath polygon =

new GeneralPath(GeneralPath.WIND\_EVEN\_ODD,

x1Points.length);

polygon.moveTo(x1Points[0], y1Points[0]);

for (int index = 1; index < x1Points.length; index++) {

polygon.lineTo(x1Points[index], y1Points[index]);

}

polygon.closePath();

Area area4 = new Area(polygon);

g.fill(polygon);

}

}

}

**ЗАДАНИЕ №4**

Разработать в архитектуре клиент-сервер приложение для выбора типа самолета для перевозки группы грузов. Тип сетевого соединения – UDP. При выборе типа самолета исходить из того, что грузы являются неделимыми. Для перевозки может использоваться только один самолет. Груз менее 5 т способен перевозить «ТУ-134». Вес более 5 т, но менее 30 т – «ИЛ-86». Вес более 30 т, но менее 45 т – «Руслан». Исходная информация о грузоподъемности самолетов хранится в текстовом файле на сервере. Клиент посылает массу груза на сервер, назад получает результат. Клиент имеет возможность просмотра, редактирования, удаления информации на сервере.

## Серверная сторона.

На стороне сервера сокрыта вся основная логика по обработке данных и обработке файла. Файл, в котором содержится информация о самолетах имеет вид:

TY-134

0

5

#

IL-86

5

30

#

Ruslan

30

45

#

Его внутрення структура обусловлена способом обработки данных, поэтому любое внешнее изменение данного файла может повредить работу данной программы.

При запуске AeroPlanServer , мы увидим вывод вида:

Name:TY-134 Min: 0 Max: 5

Name:IL-86 Min: 5 Max: 30

Name:Ruslan Min: 30 Max: 45

Hi - its a AeroPlane Server!

Первые четыре строки отображают доступные самолеты и их грузоподьемность min и max. На этом взаимодействие пользователя с серверной частью заканчивается, т.к. команды сервер не принимает из консоли - только из клиентской части.

## Клиентская часть.

Клиентская часть реализована также в консольном интерфейсе.

При запуске AeroPlanClient появится следующее:

Start Client - print help to get all avaiiable commands

Enter command or weight:

## Команды для управления

1) Простой ввод цифр - пользователь может ввести цифры, которые обозначают вес груза. Сервер же вернет название самолета, который сможет этот груз поднять. 2) help - выведет текст следующего вида:

In console, you can type some kind of command what listed below:

-> Digits (0 - 99) - Server will the name of the plane, which can handle that weight

-> status - Server will return all awailable planes

-> change - start to change plane's information.

Enter a number (chek that in status), a name, min weigth and max weight

Example: 2 'press Enter' Boeng2002 'press Enter' 0 'press Enter' 40

Вкратце обясняет как пользоваться программой. P.S. Написано на английском т.к. есть проблемы с кодировкой при переходе на разные ОС

3) status - отправляет запрос на сервер для того, чтобы узнать текущие самолеты в наличии. Вернется статус вида:

Name:TY-134 Min: 0 Max: 5

Name:IL-86 Min: 5 Max: 30

Name:Ruslan Min: 30 Max: 45

4) change - эта команда необходима для изменения данных на сервере.

* После ввода команды, программа предложит ввести номер самолета по (int), который необходимо заменить (список можно посмотреть через команду status).
* Новое имя самолета (символьная строка)
* Минимальная грузоподьемность - целое число (int)
* Максимальная грузоподьемность - целое число (int) После ввода, клиент отправит на сервер запрос на изменение. Если операция прошла успешно, от сервера вернется

"Data changed!";

иначе, будет сообщение об ошибке. При успехе операции также изменятся выводимые командой status самолеты.

Листинг программы:

**Client.java**

public class UDPAeroClient {

DatagramSocket clientSocket = null;

// choose ip - its a localhost or (127.0.0.1)

InetAddress IPAddress = InetAddress.getByName("localhost");

// create variable for the data send and receive

byte[] sendData = new byte[1024];

byte[] receiveData = new byte[1024];

public UDPAeroClient() throws IOException {

System.out.println("Start Client - print help to get all avaiiable commands");

initClient();

}

private void SendToServer(String sentence) {

try {

// create new connect for the UDP connect

if (clientSocket == null)

clientSocket = new DatagramSocket();

sendData = new byte[1024];

receiveData = new byte[1024];

sendData = sentence.getBytes();

// choose 5022 socket because we have launched server here

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 5022);

// send

clientSocket.send(sendPacket);

// receive

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence = new String(receivePacket.getData());

// out

System.out.println("FROM SERVER:\n" + modifiedSentence);

} catch (IOException e) {

e.printStackTrace();

if (clientSocket != null)

clientSocket.close();

}

}

public void initClient() {

String sentence;

try {

while (true) {

// read console's input

System.out.println("Enter command or weight:");

BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));

String message = inFromUser.readLine().trim();

if (message.contentEquals("change")) {

message = "/ch ";

System.out.println("Enter number a plane");

inFromUser = new BufferedReader(new InputStreamReader(System.in));

String tmpStr = inFromUser.readLine();

message += tmpStr + "\n";

System.out.println("Enter new name of the plane");

inFromUser = new BufferedReader(new InputStreamReader(System.in));

tmpStr = inFromUser.readLine();

message += tmpStr + "\n";

System.out.println("Enter min weight of the plane");

inFromUser = new BufferedReader(new InputStreamReader(System.in));

tmpStr = inFromUser.readLine();

message += tmpStr + "\n";

System.out.println("Enter max weight of the plane");

inFromUser = new BufferedReader(new InputStreamReader(System.in));

tmpStr = inFromUser.readLine();

message += tmpStr + "\n";

}

if (message.contentEquals("help")) {

System.out.println("AeroPlaneServer v0.1. Created by AlexOAnder");

System.out.println("In console, you can type some kind of command what listed below:");

System.out.println("-> Digits (0 - 99) - Server will the name of the plane, which can handle that weight");

System.out.println("-> status - Server will return all awailable planes");

System.out.println("-> change - start to change plane's information. \n "+"Enter a number (chek that in status), a name, min weigth and max weight");System.out.println("Example: 2 'press Enter' Boeng2002 'press Enter' 0 'press Enter' 40"); } else { SendToServer(message);}}

} catch (Exception e) {

e.printStackTrace();

} finally {

if (clientSocket != null)

clientSocket.close();

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

new UDPAeroClient();

}}

**Server.java**

public class UDPAeroServer {

DatagramSocket serverSocket = null;

String delimiter = "#";

public List<Plane> planes = new ArrayList<Plane>();

public UDPAeroServer() {

try {

LoadPlaneData();

initUDP();

}

catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private void initUDP() {

try {

serverSocket = new DatagramSocket(5022);

byte[] receiveData = new byte[1024];

byte[] sendData = new byte[1024];

System.out.println("Hi - its a AeroPlane Server!");

while (true) {

receiveData = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receivePacket);

String sentence = new String(receivePacket.getData());

sentence = sentence.trim();

System.out.println("RECEIVED: " + sentence);

String answer = "Something Wrong";

int weight = 0;

try {

weight = Integer.parseInt(sentence.trim());

answer = FindPlane(weight);

} catch (NumberFormatException ex) {

answer = "Wrong int - try to send int, please";

}

if (sentence.contains("/ch")) {

if (ChangePlanesFile(sentence) == 0)

answer = "Data changed!";

LoadPlaneData(); // after data change, we need to reload our planesList

}

if (sentence.contentEquals("status")) {

answer = ShowStatus();

}

System.out.println("Answer->" + answer);

InetAddress IPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = answer.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);

serverSocket.send(sendPacket);

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

if (serverSocket != null)

serverSocket.close();

}

}

public static void main(String args[]) {

new UDPAeroServer();

}

private int WriteToFile(Plane pl, int number) {

BufferedReader reader;

try {

reader = new BufferedReader(new FileReader("src/PlaneData.txt"));

String line;

List<String> lines = new ArrayList<String>();

while ((line = reader.readLine()) != null) {

lines.add(line);

}

reader.close();

int iterator = 0;

int countOfDelimiters = 0;

int startErasePos = 0;

if (number == 1) {

startErasePos = 0;

} else {

for (String tmp : lines) {

if (delimiter.equals(tmp)) {

countOfDelimiters++;

}

if (countOfDelimiters == number-1)

{

startErasePos = iterator+1;

break;

}

iterator++;

}

}

lines.remove(startErasePos);

lines.remove(startErasePos);

lines.remove(startErasePos);

// change lines

lines.add(startErasePos, pl.Name);

lines.add(startErasePos + 1, Integer.toString(pl.MinWeight));

lines.add(startErasePos + 2, Integer.toString(pl.MaxWeight));

// open file for the change

BufferedWriter writer = new BufferedWriter(new FileWriter("src/PlaneData.txt"));

writer.flush();

for (String tmp : lines) {

writer.write(tmp);

writer.write("\n");

}

writer.close();

return 0;

} catch (IOException e) {

e.printStackTrace();

return -1; // -1 mean error;

}

}

private int ChangePlanesFile(String line) {

Plane chgPlane = new Plane();

try {

int number = 0;

int i = line.indexOf('\n');

String s = line.substring(4, i);

line = line.substring(i + 1);

System.out.println("Num->" + s);

// find a number

number = Integer.parseInt(s);

if (number < 0 || number > 3)

return -1;

int j = 0;

while (true) {

j++;

int i1 = line.indexOf('\n');

if (i1 < 0) {

i1 = line.length();

String s2 = line.substring(0, i1);

chgPlane.MaxWeight = Integer.parseInt(s2.trim());

System.out.println("s" + j + "->" + s2);

break;

}

String s1 = line.substring(0, i1);

if (j == 1)

chgPlane.Name = s1.trim();

if (j == 2)

chgPlane.MinWeight = Integer.parseInt(s1.trim());

line = line.substring(i1 + 1);

System.out.println("s" + j + "->" + s1);

if (line.length() <= 1)

break;

}

// 0 mean success, -1 -> error

return WriteToFile(chgPlane, number);

} catch (NumberFormatException ex) {

return -1;

}

}

private String FindPlane(int weight) {

try {

if (weight == 0) {

return "Nothing to carry - zero weight";

}

if (!planes.isEmpty())

for (Plane pl : planes) {

if (pl.MinWeight <= weight && pl.MaxWeight > weight) {

return pl.Name;

}

}

return "NotAvailable";

} catch (Exception e) {

e.printStackTrace();

return "Server Error!";

}

}

private String ShowStatus() {

String res = "";

for (Plane p : planes) {

res += "Name:" + p.Name + " Min: " + p.MinWeight + " Max: " + p.MaxWeight + " \n ";

}

return res;

}

private void LoadPlaneData() throws IOException {

BufferedReader reader = new BufferedReader(new FileReader("src/PlaneData.txt"));

String line;

List<String> lines = new ArrayList<String>();

while ((line = reader.readLine()) != null) {

lines.add(line);

}

reader.close();

int counter = 0;

List<Plane> tmpPlanes = new ArrayList<Plane>();

Plane pl = new Plane();

for (String tmp : lines) {

if (!delimiter.equals(tmp)) {

counter++;

switch (counter) {

case 1: {

pl.Name = tmp;

break;

}

case 2: {

pl.MinWeight = Integer.parseInt(tmp);

break;

}

case 3: {

pl.MaxWeight = Integer.parseInt(tmp);

break;

}

}

} else {

counter = 0;

tmpPlanes.add(pl);

pl = new Plane();

}

}

for (Plane p : tmpPlanes) {

System.out.println("Name:" + p.Name + " Min: " + p.MinWeight + " Max: " + p.MaxWeight + " ");

}

planes = tmpPlanes;

}

}

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Java. Объектно-ориентированное программирование: А. Н. Васильев — Москва, Питер, 2011 г.- 400 с.
2. Java: Карманный справочник: Питер Дж. ДеПаскуале — Москва, КУДИЦ-Образ, 2005 г.- 128 с.
3. WEB-программирование на Java и JavaScript: Андрей Гарнаев, Сергей Гарнаев — Москва, БХВ-Петербург, 2005 г.- 1040 с.
4. Язык программирования Java и среда NetBeans: В. Монахов — Санкт-Петербург, БХВ-Петербург, 2011 г.- 704 с.