

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики
Кафедра информационных систем управления

Никончик Даниил Викторович

ОТЧЕТ
по учебной практике
студента 2 курса 13 группы

Старший преподаватель
Гутников Сергей Евгеньевич

Минск 2021

Оглавление

Задание №1	3
Задание №2	4
Задание №3	5
Задание №4	6
Задание №5	8
Задание №6	10
Задание №7	12
Задание №8	14
Задание №9	16
Задание №10.....	18
Задание №11.....	19
Задание №12.....	21
Список литературы	24

Задание №1

Постановка задачи:

Изобразить приближающийся издали шар, удаляющийся шар. Шар должен двигаться с постоянной скоростью. Для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape (можно взять базовым библиотечный класс, реализующий Shape) - выполнить указанные в задании перемещения указанной фигуры с помощью аффинного преобразования координат - выполнить рисунок в окне фрейма с выбранной толщиной границы фигуры, цветом границы и цветом внутренней области (вводить толщину и цвет в качестве аргументов ваших программ).

Особенности реализации:

Я использовал стандартный класс эллипса `Ellipse2D` реализующий `Shape`.

Для вращения использовал аффинное преобразование.

Результат работы программы:

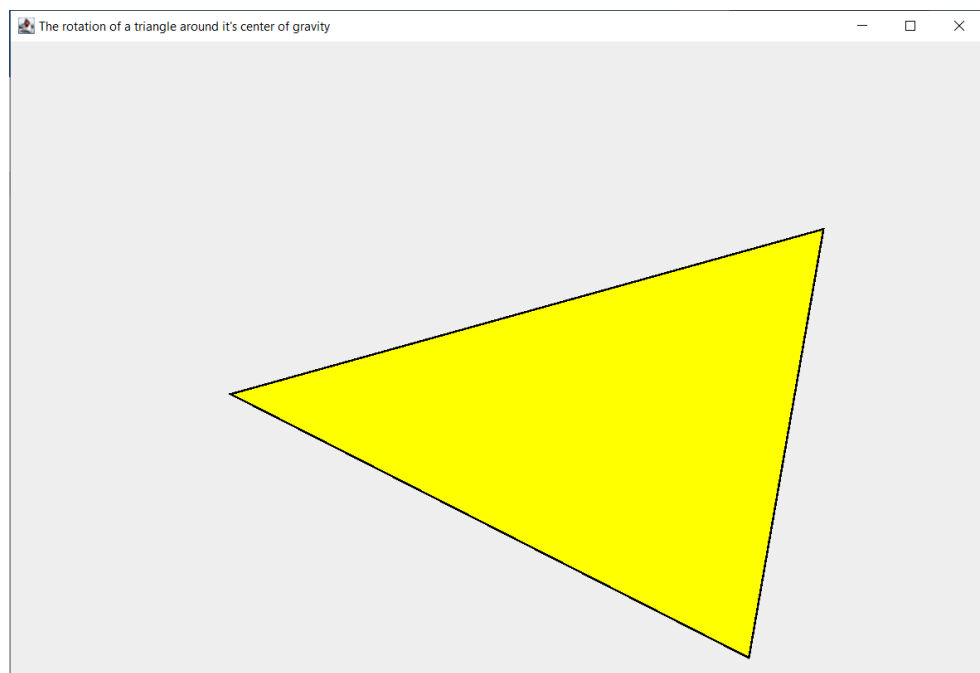


Рисунок 1 – Скриншот окна приложения

Задание №2

Постановка задачи:

В следующих заданиях создайте тестовое приложение (Frame/JFrame) для демонстрации вашего решения, при этом: - для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape; - создайте указанный фильтр изображения; при тестировании выведите фигуру без фильтра и с фильтром (аналогично фильтрам из примеров); - моделируйте освещение и тень от объекта при помощи альфа-канала и/или механизма обработки изображения; - при рисовании используйте сглаживание, внеэкранный буфер и преобразования координат.

Фигура (дорожный знак): вопросительный знак в квадрате, цвет квадрата и символа – жёлтый, цвет фона – серый с градиентной заливкой слева-направо

Фильтр: Blur

Особенности реализации:

Рисуем фигуру и применяем к ней необходимый фильтр:

```
public class GraphSamplePane extends JComponent {
    GraphSample example;
    Dimension size;

    public GraphSamplePane(GraphSample example) {
        this.example = example;
        size = new Dimension(example.getWidth(), example.getHeight());
        setMaximumSize( size );
    }

    public void paintComponent(Graphics g) {
        g.setColor(Color.white);
        g.fillRect(0, 0, size.width, size.height);
        example.draw((Graphics2D) g, this);
    }

    public Dimension getPreferredSize() { return size; }
    public Dimension getMinimumSize() { return size; }
}
```

Рисунок 2.1 - Скриншот кода приложения

Результат работы программы:

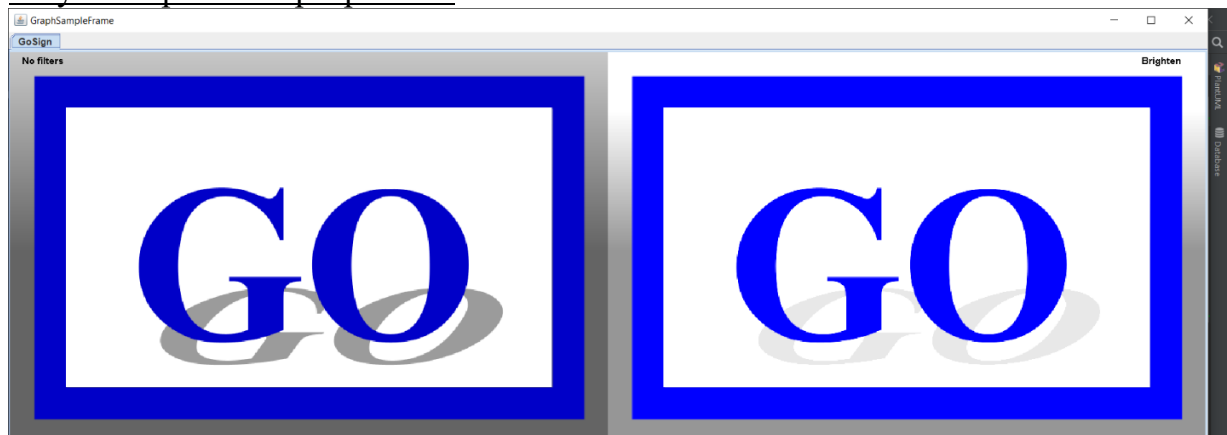


Рисунок 2.2 - Скриншот окна приложения

Задание №3

Постановка задачи:

Разработайте пользовательский класс Shape реализующий рисование улитки паскаля. Разработайте пользовательский класс Stroke для отображения указанного контура, используя в качестве исходных точек результаты класса Shape, созданного на шаге 1)

Создайте приложение (Frame/JFrame) для тестирования и демонстрации разработанных классов.

Особенности реализации:

1. Линия: Циссоида

$$y^2 = x^3 / (2a - x)$$



Контур:

Высчитываю координаты.

```
public int currentSegment(double[] coordinate) {  
    if (start) {  
        start = false;  
  
        return SEG_MOVETO;  
    }  
    if (t >= Math.PI / 2 - h) {  
        done = true;  
        return SEG_CLOSE;  
    }  
    coordinate[0] = (2 * a * Math.pow(Math.tan(t), 2) / (1 +  
Math.pow(Math.tan(t), 2))) + centerX;  
    coordinate[1] = (2 * a * Math.pow(Math.tan(t), 3) / Math.pow(Math.tan(t),  
2)) + centerY;  
    return SEG_LINETO;  
}
```

Рисунок 3.1 – Скриншот кода приложения

Результат работы программы:

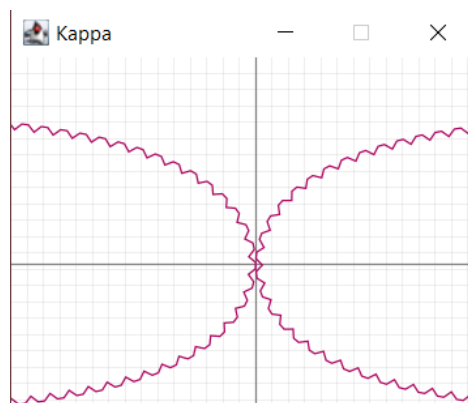


Рисунок 3.2 – Скриншот окна приложения

Задание №4

Постановка задачи:

Для выполнения задания используется решение задания №3.

- Модифицировать программу следующим образом. В демонстрационное приложение добавить возможность печати небольшого отчёта о решении задания №3. Отчёт должен содержать следующее:
 - рисунок с подписью алгебраической линии
 - исходный текст класса Shape, реализующий рисование указанной алгебраической линии
- При печати использовать режим альбомной ориентации страницы и двустороннюю печать. Рисунок должен занимать не более половины страницы, при печати выровнять его по горизонтали.

Особенности реализации:

Используем классы CustomShape и MyStroke, созданные в прошлой программе, описанные подробнее в прошлом отчёте.

Для печати был использован класс HardCopyWriter, который реализует интерфейс Writer. Приведен метод используемый для печати:

```
public static void printFile(FileReader in, HardcopyWriter out) {
    try {
        char [] buffer = new char [4096];
        int numchars;
        //out.setFontStyle(Font.ITALIC);
        while((numchars = in.read(buffer)) != -1)
            out.write(buffer, 0, numchars);
        in.close();
        out.close();
    }
    catch (Exception e) {
        System.err.println(e);
        System.err.println("Usage: " + "java HardcopyWriter$PrintFile
<filename>");
        System.exit(1);
    }
    System.exit(0);
}
```

Рисунок 4.1 – Скриншот кода приложения

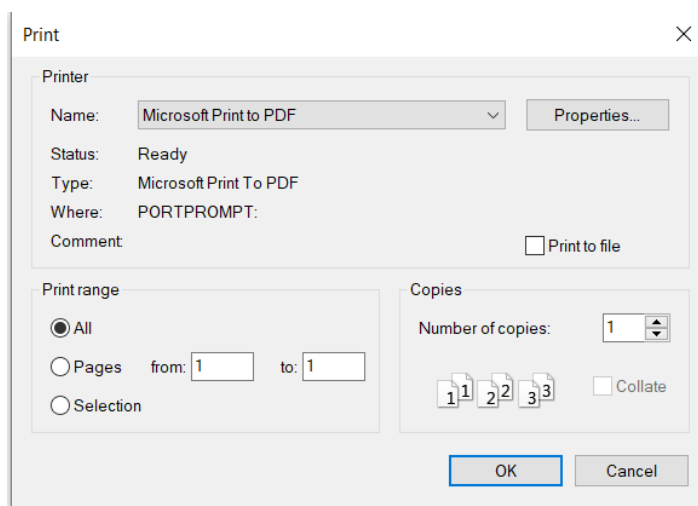


Рисунок 4.2 – Скриншот окна приложения

Результат работы программы:

На рисунке 4 изображено окно приложения с настройками печати.

На рисунке 5 изображен отчёт по заданию, в том виде, в каком он был бы напечатан.

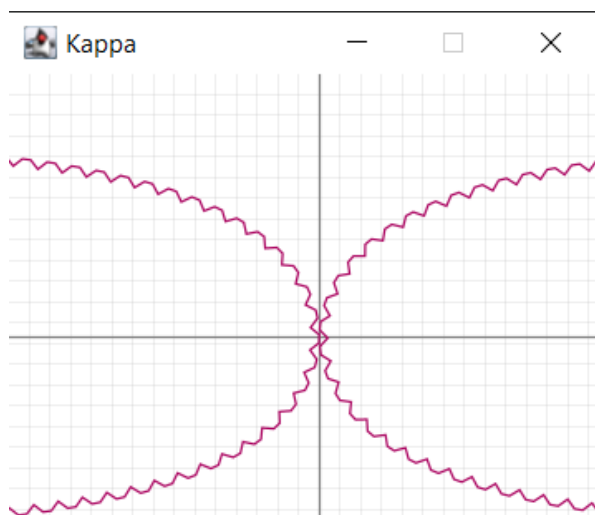


Рисунок 4.3 – Скриншот распечатанного файла

Задание №5

Постановка задачи:

- Разработайте систему классов/интерфейсов для предметной области Вашего варианта задания. Данные необходимо упорядочить по атрибутам/свойствам товаров, предметов и т.п. в виде дерева.
- Разработайте графическое приложение для ввода/отображения данных Вашего варианта задания. При отображении структуры данных в виде дерева реализуйте интерфейс `javax.swing.Tree.TreeModel`. Листья дерева отображайте в виде таблицы, для этого реализуйте интерфейс `javax.swing.table.TableModel`. (пример похожего приложения – Проводник Windows)
- Организуйте создание/загрузку/сохранение данных вашего варианта задания в файл

Вариант: Расписание автовокзала.

Особенности релизации:

Ниже приведен код реализации добавления нового препарата на аптечный склад.

```
void addNewItem() {
    DragTreeNode where, insert, root = treeModel.getRoot();

    if (addResult != null) {
        try {
            insert = new DragTreeNode(addResult.getName(), addResult);
            if ((where = findNode(root, addResult.getType())) != null) {
                treeModel.insertNodeInto(insert, where,
                    where.getChildCount(), false);
            } else {
                treeModel.insertNodeInto(new
                    DragTreeNode(addResult.getType()),
                    root,
                    root.getChildCount(),
                    false);
                where = findNode(root, addResult.getType());
                treeModel.insertNodeInto(insert, where,
                    where.getChildCount(), false);
            }
        } catch (Exception e) {
            addResult = null;
            return;
        }
    }
    addResult = null;
}
```

Рисунок 5.1 – Скриншот кода приложения

Результат работы программы:

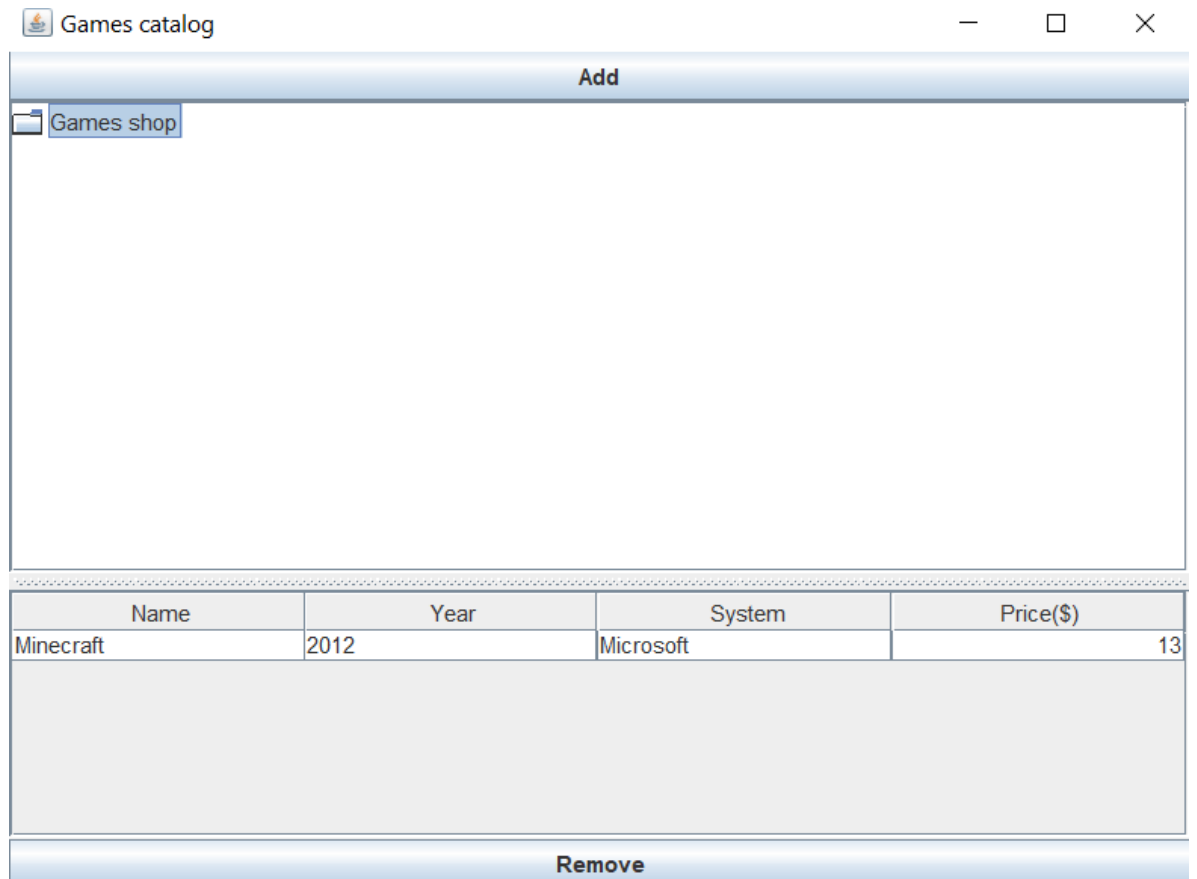


Рисунок 5.2 – Скриншот окна приложения

Задание №6

Постановка задачи:

Для выполнения задания используется ваш вариант решения задания No3. Модифицируйте вашу программу следующим образом.

Создайте тестовое приложение, добавьте в ваш класс рисования алгебраической линии возможность «перетаскивание» (drag-and-drop). Реализуйте необходимые интерфейсы в классе и в приложении для демонстрации «перетаскивания» алгебраической линии между несколькими копиями тестового приложения. При реализации интерфейса тестового приложения следуйте рекомендациям стандарта CUI (Common User Interface).

Особенности релизации:

```
public void drop(DropTargetDropEvent e) {
    this.setBorder(NORMAL_BORDER);
    if (e.isDataFlavorSupported(CustomShape.decDataFlavor) ||
        e.isDataFlavorSupported(DataFlavor.stringFlavor)) {
        e.acceptDrop(DnDConstants.ACTION_COPY_OR_MOVE);
    } else {
        e.rejectDrop();
        return;
    }

    Transferable t = e.getTransferable();
    CustomShape droppedScribble;
    try {
        droppedScribble = (CustomShape)
            t.getTransferData(CustomShape.decDataFlavor);
    } catch (Exception ex) {
        try {
            String s = (String) t.getTransferData(DataFlavor.stringFlavor);
            droppedScribble = CustomShape.getFromString(s);
        } catch (Exception ex2) {
            e.dropComplete(false);
            return;
        }
    }

    Point p = e.getLocation();
    droppedScribble.translate((int) p.getX(), (int) p.getY());
    witches.add(droppedScribble);
    repaint();
    e.dropComplete(true);
}
```

Рисунок 6.1 – Скриншот кода приложения

Результат работы программы:

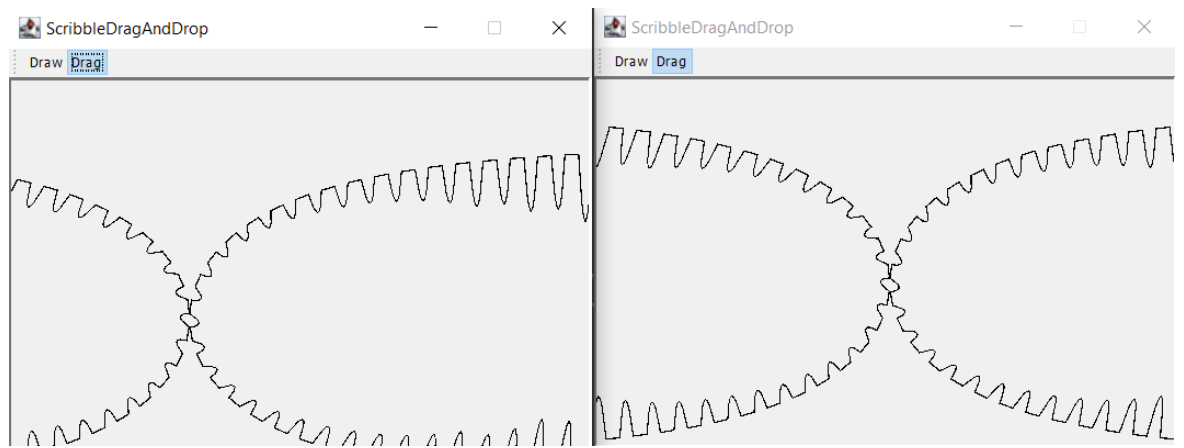


Рисунок 6.2 – Скриншот окна приложения

Задание №7

Постановка задачи:

Исследовать предложенную предметную область, спроектировать структуру базы данных объектов выбранной предметной области (из не менее чем 2-х таблиц объектов). Разработайте графическое приложение для создания/ввода/отображения БД Вашего варианта задания. Содержимое БД отображайте в виде таблиц. При реализации интерфейса следуйте рекомендациям стандарта CUI (Common User Interface).

Особенности релизации:

Создал класс PharmacyWarehouse, который отвечает за подключение к БД. Ниже его метод, который создает необходимые по заданию таблицы:

```
public void create() {
    System.setProperty("derby.system.home", "C:\\\\Derby");
    try {
        Class.forName(driver);
        Connection conn = DriverManager.getConnection(connect);
        Statement statement = conn.createStatement();
        statement.executeUpdate("CREATE TABLE PharmacyWarehouses " +
            "(" +
            "id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1) PRIMARY KEY, " +
            "name VARCHAR(32)" +
            ")");

        statement.executeUpdate("CREATE TABLE DrugTypes " +
            "(" +
            "id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1) PRIMARY KEY, " +
            "name VARCHAR(64), " +
            "pharmacyWarehouseId INTEGER REFERENCES PharmacyWarehouses(id)" +
            ")");

        statement.executeUpdate("CREATE TABLE Drugs " +
            "(" +
            "id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1) PRIMARY KEY, " +
            "name VARCHAR(64), " +
            "producer VARCHAR(64), " +
            "number VARCHAR(128), " +
            "drugTypeId INTEGER REFERENCES DrugTypes(id)" +
            ")");
    } catch (Exception e) {
        System.err.println("Run-time error: ");
        e.printStackTrace();
    }
    pendingQueries.clear();
}
```

Рисунок 7.1 – Скриншот кода приложения

Результат работы программы:



The screenshot shows a window with a title bar containing a small icon on the left and standard window controls (minimize, maximize, close) on the right. The window contains a table with three columns: 'Name', 'Genre', and 'Price'. The first row of data shows 'Survival' in the Name column, 'Minecraft' in the Genre column, and '15.0' in the Price column. The rest of the table area is empty.

Name	Genre	Price
Survival	Minecraft	15.0

Рисунок 7.2 – Скриншот окна приложения

Задание №8

Постановка задачи:

Изучите материал примера по быстрому введению в среду разработки NetBeans и компоненты JavaBeans по адресу: <http://docs.oracle.com/javase/tutorial/javabeans/quick/index.html> 2) Разработайте простой компонент вашего варианта задания на базе класса Canvas. Создайте файл манифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет: bsu.fpmi.educational_practice 3) Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Компонент: Плоская вертикальная линия. Свойства: высота и цвет

Особенности релизации:

Методы отрисовки и установки параметров компонента Java Beans.

```
public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;

    GradientPaint redtowhite = new GradientPaint((int)this.x, (int)this.y +
(int)this.height / 2, this.leftColor, (int) this.width, (int)this.height,
        this.rightColor);
    Shape Line = new Line2D.Double(this.x, this.y, this.x + this.width,
this.y + this.height);
    g2.setPaint(redtowhite);
    g2.draw(Line);
    g2.fill(Line);
}

public void setWidth(double width) {
    this.width = width;
}

public void setHeight(double height) {
    this.height = height;
}

public void setRightColor(Color rightColor) {
    this.rightColor = rightColor;
}

public void setLeftColor(Color leftColor) {
    this.leftColor = leftColor;
}
```

Рисунок 8.1 – скриншот кода приложения

Результат работы тестовой программы:

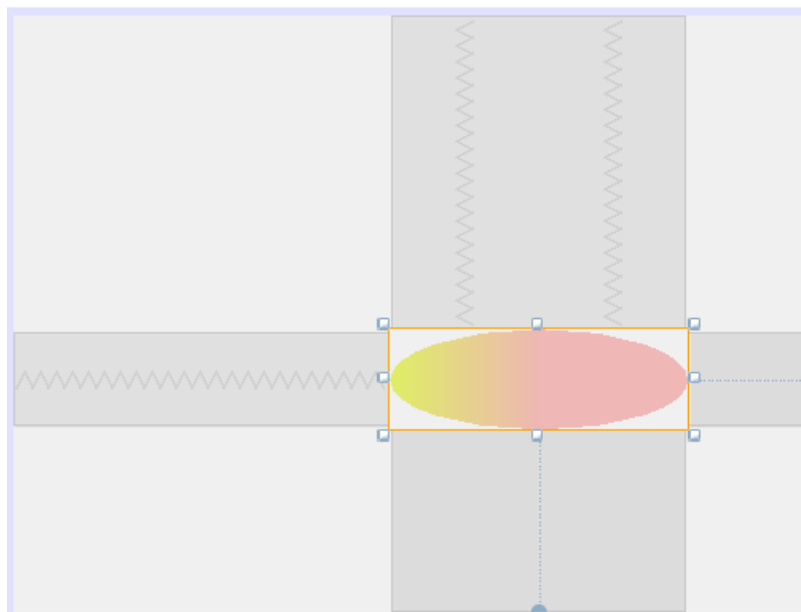


Рисунок 8.2 – Скриншот окна приложения

Задание №9

Постановка задачи:

Разработайте компонент вашего варианта задания. Создайте файл манифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет: `bsu.fpmi.educational_practice` 2) Компонент должен реализовать класс `BeanInfo` с информацией о компоненте. 3) Создайте тестовое приложение в NetBeans с использованием вашего компонента.

Интерфейсные компоненты с реализацией собственного события `AcceptEvent`. Определить также интерфейс слушателя вашего события `AcceptListener`. Передавать слушателю события информацию о том, в результате чего произошло событие, если событие может генерироваться от нескольких действий.

Одноточный статический текст, строка ввода и кнопка. Свойства: текст, текст кнопки. Событие генерируется при нажатии на кнопку. Событие передаёт ещё и введенный текст.

Особенности релизации:

Компонент Java Beans

Манифест:

Manifest-Version:

1.0

Name:

MessageJPanel

Java-Bean: true

Сама кнопка

```
package bsu.fpmi.educational_practice;

enum ButtonValues { button, button1, button2 }
enum LabelValues { label, label1, label2 }

public class MessageJPanel extends javax.swing.JPanel {

    private char key = 13;

    public MessageJPanel() { initComponents(); }

    @SuppressWarnings("unchecked")
    private void initComponents() {

        textJTextField = new javax.swing.JTextField();
        okJButton = new javax.swing.JButton();
        textJLabel = new javax.swing.JLabel();

        textJTextField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                textJTextFieldKeyTyped(evt);
            }
        });
    }
}
```

Рисунок 9.1 – скриншот кода приложения

Результат работы тестовой программы:

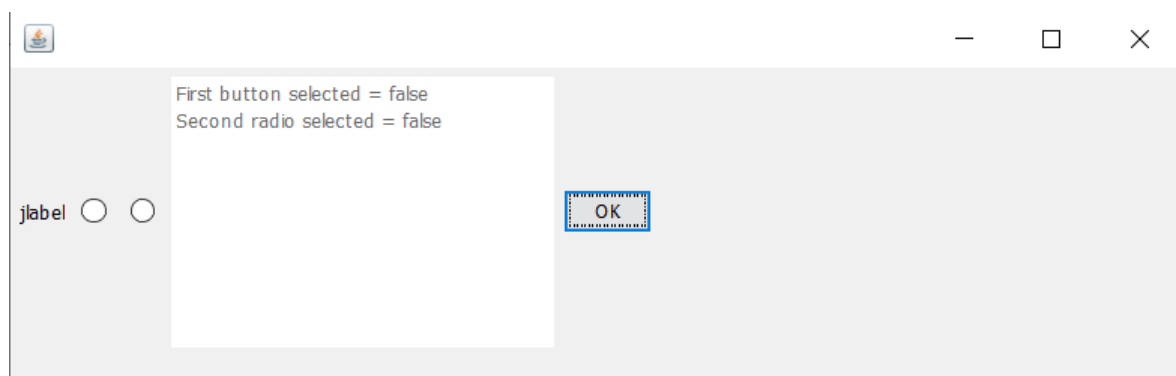


Рисунок 9.2 – Скриншот окна приложения

Задание №10

Постановка задачи:

Для решения задания №10 используем решённый вариант задания №9. Номера заданий сохраняются. Модифицируем тип свойства компонента так, как указано ниже:

Основная задача:

Создаём собственный редактор для каждого свойства компонента. Каждый редактор ограничивает возможные значения свойства, предоставляя выбор из списка трёх – пяти допустимых значений (т. е. определяем методы `getTags()`). Регистрируем редакторы в классе `BeanInfo` компонента.

Дополнительно:

Попытайтесь создать настройщик компонента, который позволит изменять списки допустимых значений для свойств вашего компонента.

Особенности реализации:

Для выполнения задания был создан класс `MessageTextEditor`.

```
public class MessageTextEditor extends PropertyEditorSupport {
    public String[] getTags() {
        return new String[] { "Текст", "Ваш текст", "Text" };
    }

    public void setAsText(String s) {
        setValue(s);
    }

    public String getJavaInitializationString() {
        Object o = getValue();
        if (o.equals("Текст")) return "Текст";
        else if (o.equals("Ваш текст")) return "Ваш текст";
        else if (o.equals("Text")) return "Text";
        return null;
    }
}
```

Рисунок 10.1 – скриншот кода приложения

Результат выполнения программы:

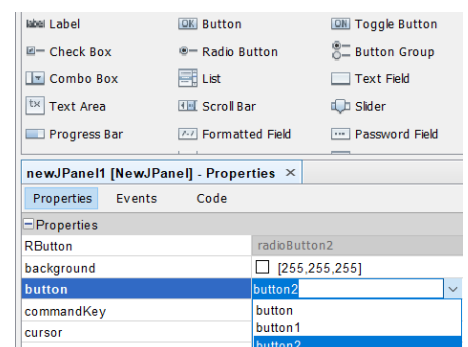
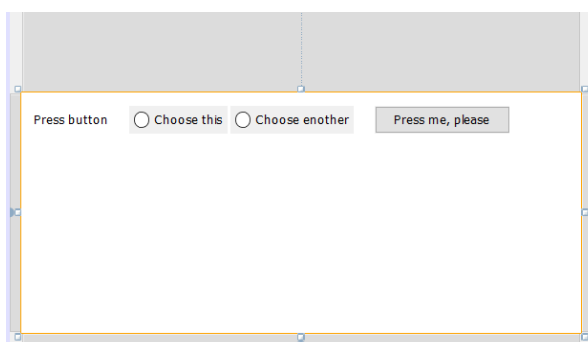


Рисунок 10.2 – Скриншот окна приложения

Задание №11

Задание: создать сервлет и взаимодействующие с ним пакеты Java-классов и HTML-документов, выполняющие действия для решения задания. Представить решение в виде web-приложения(как в примере).

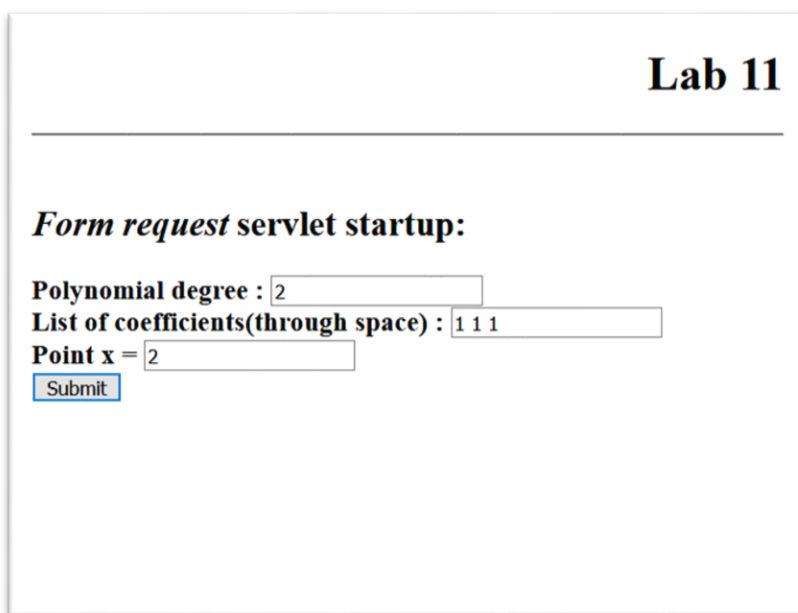
Условие, вариант 2: Вычисление тригонометрических функций в градусах и радианах с указанной точностью. Выбор функций должен осуществляться через выпадающий список.

Особенности реализации: Имеется кнопка, по нажатию на которую происходит сортировка массива и открывается страница с новым массивом.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/result', methods = ['POST', 'GET'])
def result():
    if request.method == 'POST':
        result = request.form
        numbers = [int(value) for value in result['numbers'].split(',')]
        numbers.sort()
        return render_template("result.html", numbers = numbers)
```

Рисунок 11.1 – Скриншот кода приложения



Lab 11

Form request servlet startup:

Polynomial degree :

List of coefficients(through space) :

Point x =

Рисунок 11.2 – Страница задания данных для полинома

Polynomial value at point: $P(2.0) = 7.0$

Рисунок 12.3 – Успешное вычисление полинома

Задание №12

Постановка задачи:

Изучить пример 2

Проанализировать вариант задания. Можно ли его реализовать как часть MUD системы (например, в одной из комнат MudPlace), требуется ли для этого внести изменения в парадигму MUD? Какие изменения потребует реализация клиента MUD, другие классы примера? Оформить эти размышления в отчёте в качестве анализа предметной области. При реализации, по возможности, использовать парадигму MUD и классы примера 2 при реализации варианта задания.

Создайте на основе технологии RMI клиент/серверное приложение:

Почта – обмен текстовыми сообщениями и файлами. Сервер ведёт список зарегистрированных клиентов. Каждый клиент при регистрации получает почтовый ящик (ограниченного размера) для хранения поступающих сообщений. Клиент может послать сообщение другому зарегистрированному на сервере клиенту. После успешной доставки сообщения клиенту, оно автоматически удаляется из его почтового ящика

Особенности реализации:

Важнейший шаг в написании приложения с помощью технологии RMI – определиться с интерфейсом, который будет реализовать сервер. Для заданного приложения в таком интерфейсе достаточно двух методов:

1. List<Messages> getMessages () – метод, получающий все сообщения, известные серверу
2. List< Messages > getMessages s(Date) – метод, получающий сообщения за определённую дату

На стороне сервера также реализован класс NewsServiceClient, который выполняет всю логику приложения. В частности, он умеет добавлять сообщения в хранилище, сериализовать хранилище в файл и доставать из файла при последующих запусках. Также при перезапуске сервера происходит фильтрация новостей, а именно, новости, старше одного месяца, удаляются из хранилища (и из файла). Хранилищем является файл Messages.txt.

Непосредственное создание сервера:

```
ServerInterface stub = (ServerInterface)
UnicastRemoteObject.exportObject(worker, 0);
Registry reg = LocateRegistry.createRegistry(12345);
reg.bind("MyServer", stub);
```

Клиент может запросить у сервера все сообщения, либо сообщения за указанный период. Подключение клиента:

```
Registry reg = LocateRegistry.getRegistry("127.0.0.1", 12345);  
stub = (ServerInterface) reg.lookup("MyServer");
```

Сначала запускается Server, затем вводится текущая дата, лишь после этого можно запустить Client(столько раз, сколько необходимо подключить клиентов) для трансляции добавленных в сервере новостей, а также запросов клиента.

Также в этом задании предлагалось провести рассуждение, подходит ли концепция MUD для реализации данного приложения.

Если говорить коротко, парадигма MUD не подходит для реализации данного задания. К этому есть следующие причины:

MUD само по себе расшифровывается как Multi user domain. Ключевое здесь то, что MUD ориентировано на наличие большего числа взаимодействующих между собой пользователей. В данном задании взаимодействие осуществляется только между сервером и клиентом

Парадигма MUD хоть и представляет большую гибкость при создании виртуального мира, однако эта гибкость несёт с собой дополнительные расходы. Кажется неразумным поддерживать интерфейс класс MudPlace, если пользоваться будут только методами получения новостей

Единственный вариант внедрения приложения в парадигму MUD – создание отдельной комнаты, в которой можно узнавать новости. Т.е. сервер из задания будет реализацией некоего MudPlace. Однако для этого необходимо немного менять интерфейс MUD-парадигмы, а именно теперь описание комнаты должно быть динамически изменяемым, а также зависеть от параметра, заданного пользователем (день, за который хочется узнать новости).

Результат работы программы:

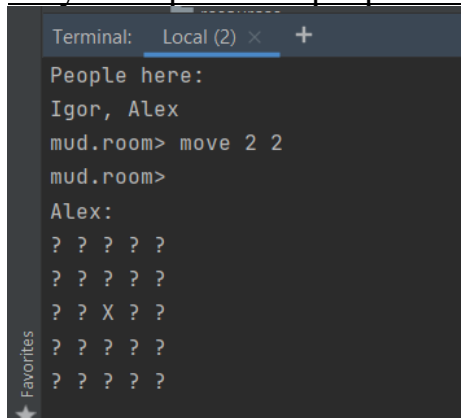


Рисунок 12 – Скриншот окна приложения

На скриншоте изображен клиент, а также продемонстрированы разные действия клиента.

Список литературы

1. Г. Шилдт. Java . Полное руководство, 8-е издание, М.: Вильямс, 2012.
2. Хабибуллин И. Ш. Java 7. - СПб.: БХВ-Питербург, 2012.
3. Кей С. Хорстман. Java2 Основы. Том 1. С.-Питербург. 2007.
4. Кей С. Хорстман. Java2 Тонкости программирования. Том 2. С.-Питербург. 2007.