

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра информационных систем управления

Крагель Алина Олеговна

Отчет
по учебной вычислительной практике
студента 2 курса 10а группы

Преподаватель
Гутников Сергей Евгеньевич
Старший преподаватель кафедры ИСУ

Минск, 2021

Оглавление

1. Лабораторная работа №1	3
2. Лабораторная работа 2.....	4
3. Лабораторная работа №3	5
4. Лабораторная работа №4.....	7
5. Лабораторная работа №5	9
6. Лабораторная работа №6.....	11
7. Лабораторная работа №7.....	13
8. Лабораторная работа №8.....	16
9. Лабораторная работа №9.....	17
10. Лабораторная работа №10.....	19
11. Лабораторная работа №11	21
12. Лабораторная работа №12.....	23
Заключение	27
Список использованной литературы.....	28

1. Лабораторная работа №1

Постановка задачи:

Для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape.

- Выполнить указанные в задании перемещения указанной в задании 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 программы).

Вариант 4.

Изобразить отрезок, вращающийся в плоскости экрана вокруг точки, движущейся по отрезку.

Особенности реализации:

Изменяемые пользователем параметры – толщина линии, цвет отрезка и цвет точки (задаются шестнадцатеричной кодировкой цвета), движущейся по отрезку. Положение точки на отрезке изменятся благодаря методу *pointMove*, описанного в качестве метода класса *linearSegment*, включающий в себя отрезок и точку:

```
void pointMove(double delta) {  
    linePoint[1] = new Ellipse2D.Double(194 + delta, 294, 12, 12);  
}
```

Для изменения положения прямой задействовано аффинное преобразование, где в качестве “якоря” используется точка с уже измененными координатами. Изменение положения происходит благодаря инкрементированию переменной *ang* при каждой отрисовке фрейма.

Результаты выполнения:

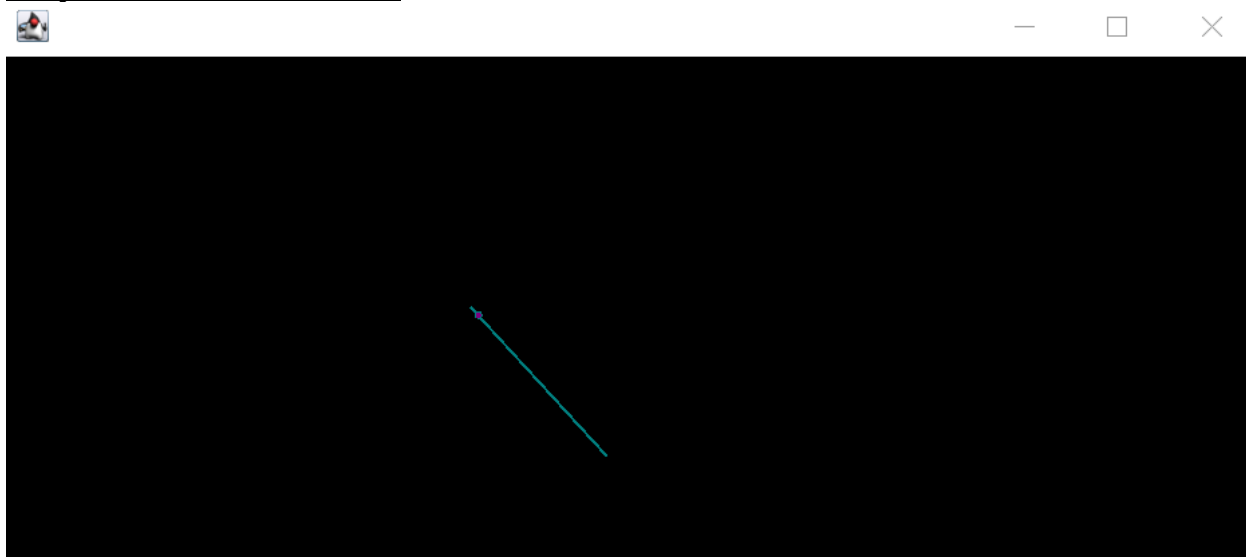


Рисунок 1.1 - Линия и точка в разные моменты времени и положения относительно друг друга.

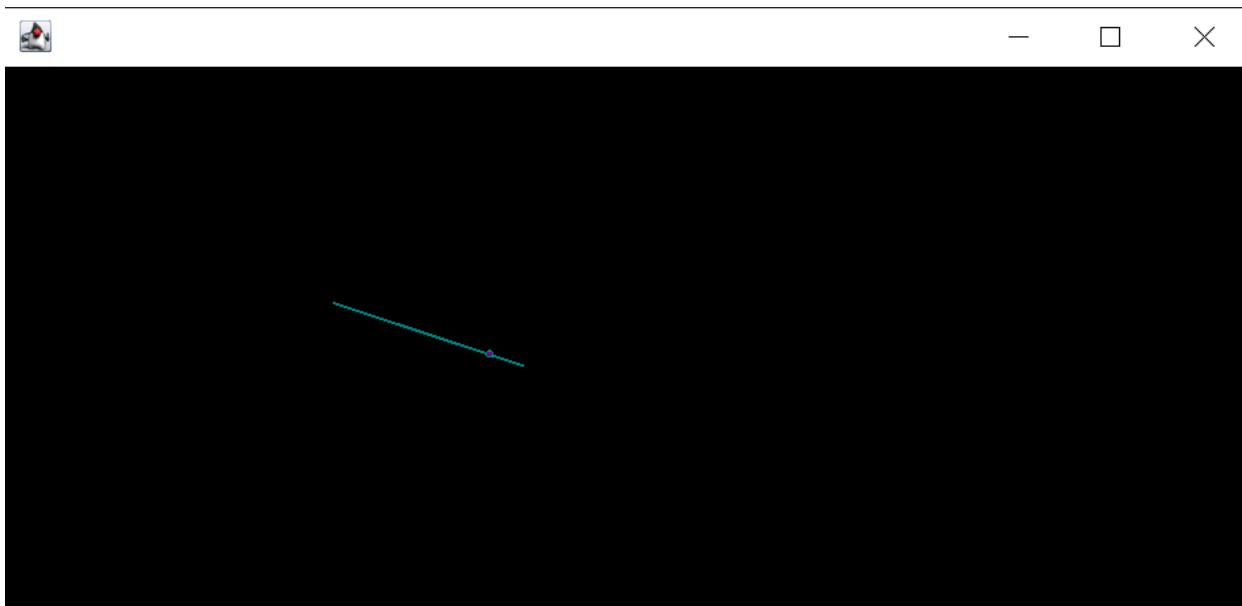


Рисунок 1.2 - Линия и точка в разные моменты времени и положения относительно друг друга.

2. Лабораторная работа 2

Постановка задачи:

В следующих заданиях создайте тестовое приложение (Frame/JFrame) демонстрации вашего приложения, 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.
- Моделируйте освещение и тень от объекта при помощи альфа-канала и/или механизма обработки изображения.
- При рисовании используйте сглаживание, внеэкранный буфер и преобразования координат

Вариант 13.

Фигура (дорожный знак): надпись STOP в прямоугольнике, цвет прямоугольника и надписи – красный, цвет фона – серый с градиентной заливкой снизу-вверх. Фильтр: Rotate CW degrees (поворот по часовой стрелке на 45).

Особенности реализации:

Для реализации использован интерфейс Shape, в методе paint() которого осуществляется требуемая заливка:

```
gf.setPaint(new GradientPaint((int)rectangle.getX(),(int)rectangle.getY() +
height,new Color(50,50,50),
(int)rectangle.getX(),(int)rectangle.getY(),new Color(150,150,
150)));
gf.fillRect((int)rectangle.getX(),(int)rectangle.getY(),200,150);
```

В методе main() задаются основные действия над фигурами, например, прорисовка тени изображения с помощью аффинных преобразований до применения фильтра:

```

g2d.setPaint(new Color(0, 0, 0, 75));
g2d.translate(75, 75);
AffineTransform shadow = AffineTransform.getShearInstance(-1.0, 0.0);
shadow.scale(1.0, 0.5);
g2d.fill(shadow.createTransformedShape(new Rectangle2D.Double(X+20, Y+20,
width, height)));
g2d.setTransform(save);
roadsign.paint(g2d);

```

И после:

```

img.setPaint(new Color(0, 0, 0, 75));
img.translate(75, 75);
shadow = AffineTransform.getShearInstance(-1.0, 0.0);
shadow.scale(1.0, 0.5);
img.fill(shadow.createTransformedShape(new Rectangle2D.Double(X, Y, width,
height)));

```

Выполнение поворота изображения:

```

AffineTransform rotate = AffineTransform.getRotateInstance(Math.PI /
4, 100, 250);
AffineTransformOp filter = new AffineTransformOp(rotate,
AffineTransformOp.TYPE_BILINEAR);

```

Использование *BufferedImage*:

```

BufferedImage bimage = new BufferedImage(width + X + 70, height + Y,
BufferedImage.TYPE_INT_ARGB);
Graphics2D img = bimage.createGraphics();

```

Для обработки изображения *BufferedImage()* передаются метода *filter()*.

Результаты выполнения:



Рисунок 2 – Знак до и после применения фильтров

3. Лабораторная работа №3

Постановка задачи:

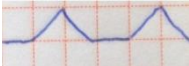
- 1) Разработайте пользовательский класс *Shape*, реализующий рисование указанной алгебраической линии.
- 2) Разработайте пользовательский класс *Stroke* для отображения указанного контура, используя в качестве исходных точек результаты класса *Shape*, созданного на шаге 1).

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

Вариант №13.

Линия: Локон Аньези.

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

Контур: 

В силу задаваемой линии формулой, в классе *WitchOfAgnesi* используется *final* переменная *param*, которая передается в конструктор наравне с переменными центрирования фигуры *centerX* и *centerY*. Класс наследует интерфейс *Shape*. В добавок ко всему в классе перегружен метод *getPathIterator()*, реализованный с помощью вспомогательного внутреннего класса *PlotIterator*, реализующий интерфейс *PathIterator*. В методе этого класса *currentSegment()* вычисляются текущие точки фигуры, осуществляется их запись в массив *coords*, переданный в качестве параметра метода. Класс контура *MyStroke* создан, как реализация интерфейса *Stroke*. Рисование сегмента заданной фигуры осуществляется в методе *createStrokedShape()* с помощью объекта класса *GeneralPath* и *FlatteningPathIterator*. рисование реализовано рекуррентно через хранение координат предыдущей точки: вычисляется расстояние между предыдущей и текущей точками, делится пополам, после чего линия направляется под углом 45 градусов на расстояние, равное половине расстояния между точками умноженное на корень из двух. Аналогично прорисовывается оставшаяся часть линии с поправкой на угол в -45 градусов.

Результаты выполнения:

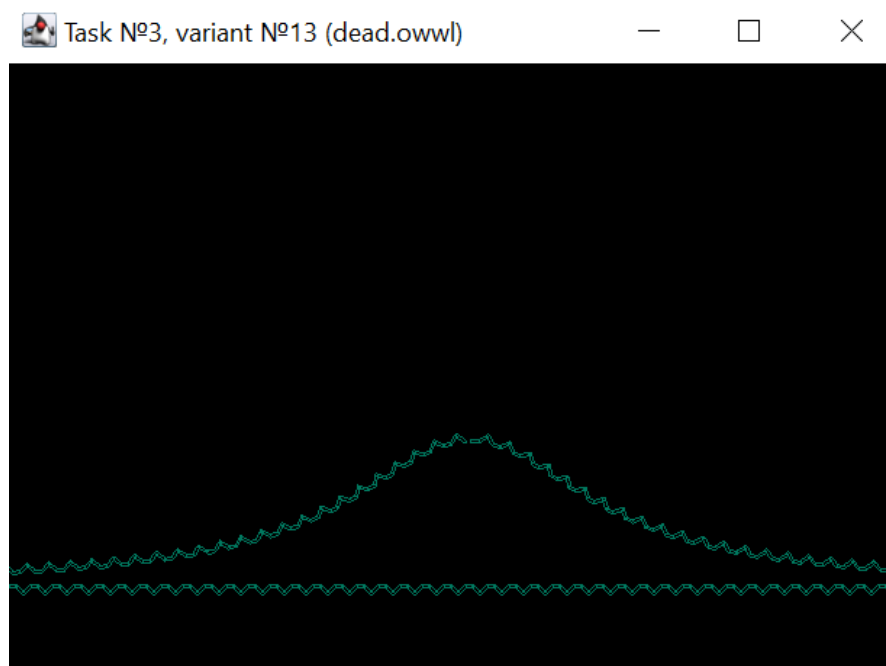


Рисунок 3 – Локон Аньези, прорисованный заданным контуром.

4. Лабораторная работа №4

Постановка задачи:

Для выполнения задания используется вариант решения задания №3. Модифицируйте вашу программу следующим образом. В демонстрационное приложения добавьте возможность печати небольшого отчёта о решении задания №3. Отчёт должен содержать следующее:

- Рисунок с подписью (по стандарту) алгебраической линии вашего задания.
- Исходный текст класса Shape, реализующий рисование указанной алгебраической линии. Для длинных строк, выходящих за границы области печати, организуйте перенос текста на новую строку с разрывом по пробельным символам.

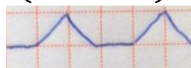
При печати используйте режим альбомной ориентации станицы и двустороннюю печать. Рисунок должен занимать не более половины страницы, при печати выравнивать его по горизонтали.

Вариант №13.

Линия: Локон Аньези.

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

Контур:



Особенности реализации:

Фрейм приложения Лабораторной работы №3 дополнился функционалом в виде меню с возможностью печати, для реализации которой использовался класс HardcopyWriter. Метод printDemoPage() печатает фигуру как экземпляр класса BufferedImage. Текст класса же в свою очередь приводится к однострочному виду (то есть конкатенируется в одну строку), а затем посимвольно обрабатывается, и, при встрече символа '\n', печать переводится на новую строку при помощи метода newline(), где ширина и высота строки вычисляются заранее. Во время печати храним уже напечатанное число строк. Если эта величина превышает некоторую заданную величину, то с помощью метода newpage() печать переводится на следующую страницу.

Результат работы программы:

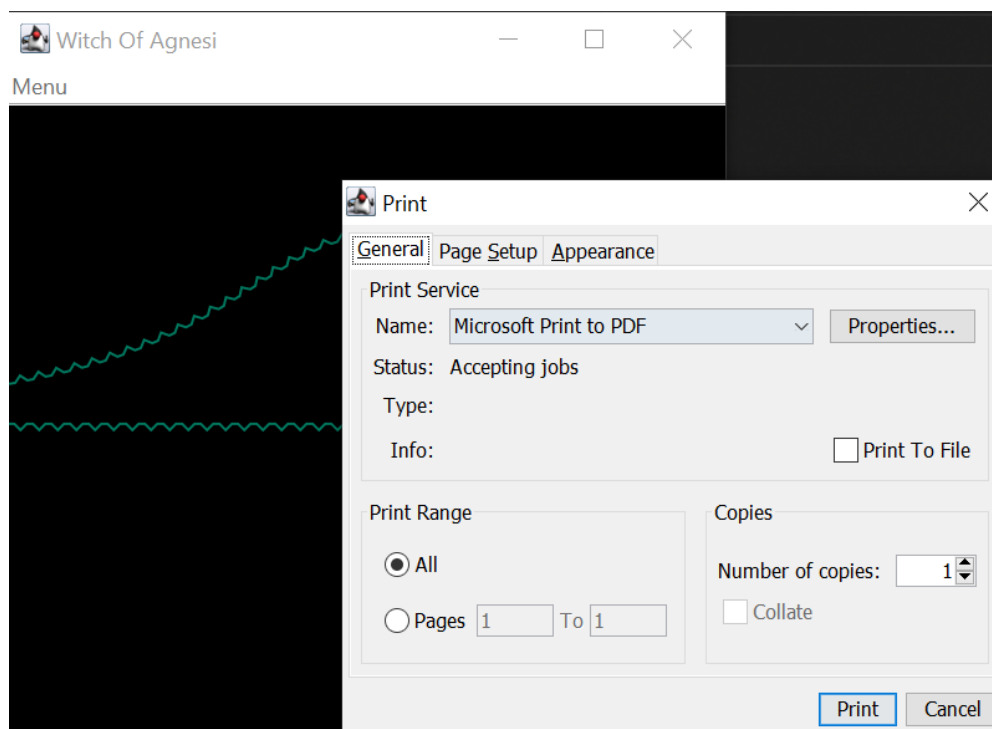
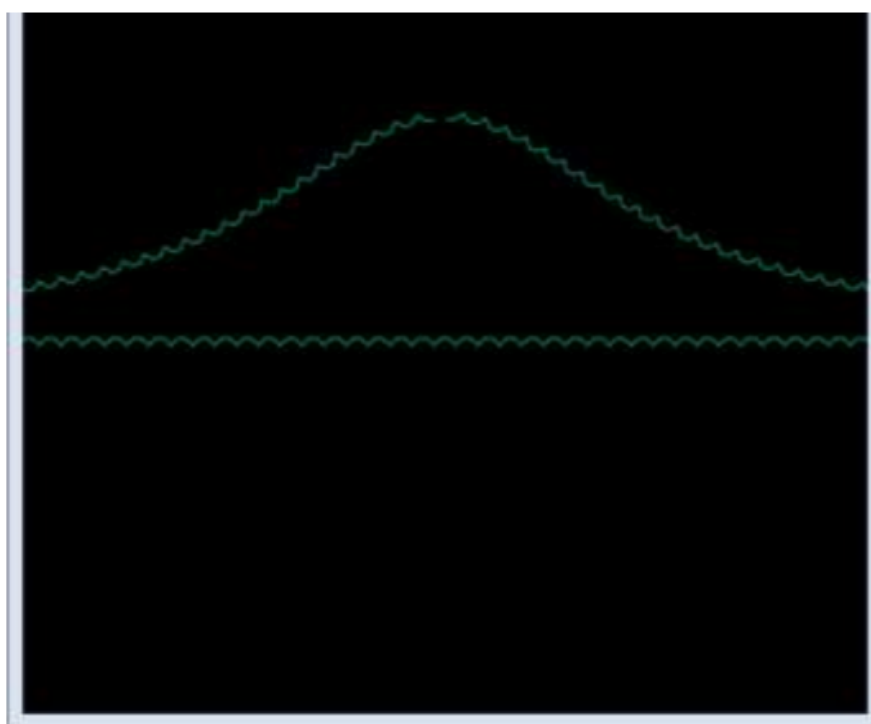


Рисунок 3.1 – Print в Меню приложения.



Witch of Agnesi

Рисунок 3.2 – Локон Аньези распечатанный в pdf.


```
import java.awt.*;
import java.awt.geom.AffineTransform;
import java.awt.geom.PathIterator;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

public class WitchOfAgnesi implements Shape {
    private final int param;
    private final double startAngle = 0.01;
    private final double endAngle = startAngle + Math.PI - 2 * 0.01;
    private final float centerX;
    private float centerY;

    public WitchOfAgnesi(int param, int centerX, int centerY) {
        this.param = param;
        this.centerX = centerX;
        this.centerY = centerY;
    }

    @Override
    public Rectangle getBounds() {
        return null;
    }

    @Override
    public Rectangle2D getBounds2D() {
        return null;
    }

    @Override
    public boolean contains(double x, double y) {
        return false;
    }

    @Override
    public boolean contains(Point2D p) {
        return false;
    }

    @Override
    public boolean intersects(double x, double y, double w, double h) {
        return false;
    }

    @Override
    public boolean intersects(Rectangle2D r) {
        return false;
    }

    @Override
    public boolean contains(double x, double y, double w, double h) {
        return false;
    }

    @Override
    public boolean contains(Rectangle2D r) {

```

Рисунок 3.3 – Код класса, распечатанный в pdf.

5. Лабораторная работа №5

Постановка задачи:

- 1) Разработать систему классов/интерфейсов для предметной области Вашего варианта задания. Данные необходимо упорядочить по атрибутам/свойствам товаров, предметов и т.п. в виде дерева.
- 2) Разработайте графическое приложение для ввода/отображения данных 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`.
- 3) Организуйте создание/загрузку/сохранение данных вашего варианта задания в файл.

Вариант №13.

Каталог исторических памятников.

Особенности реализации:

Данные о каждом историческом монументе хранятся в объектах класса Node. Сами эти объекты находятся внутри TModel в виде ArrayList. TModel является моделью как для дерева, так и для таблицы. На главной форме представлено дерево, в котором реализовано объединение монументов по году создания, и таблица, в которой отображаются записи с полями по названию, году, скульпторам и адресу по выбранному объекту Node. Реализовано сохранение созданного дерева в виде txt-файла и загрузка данных из файла.

Результат работы программы:

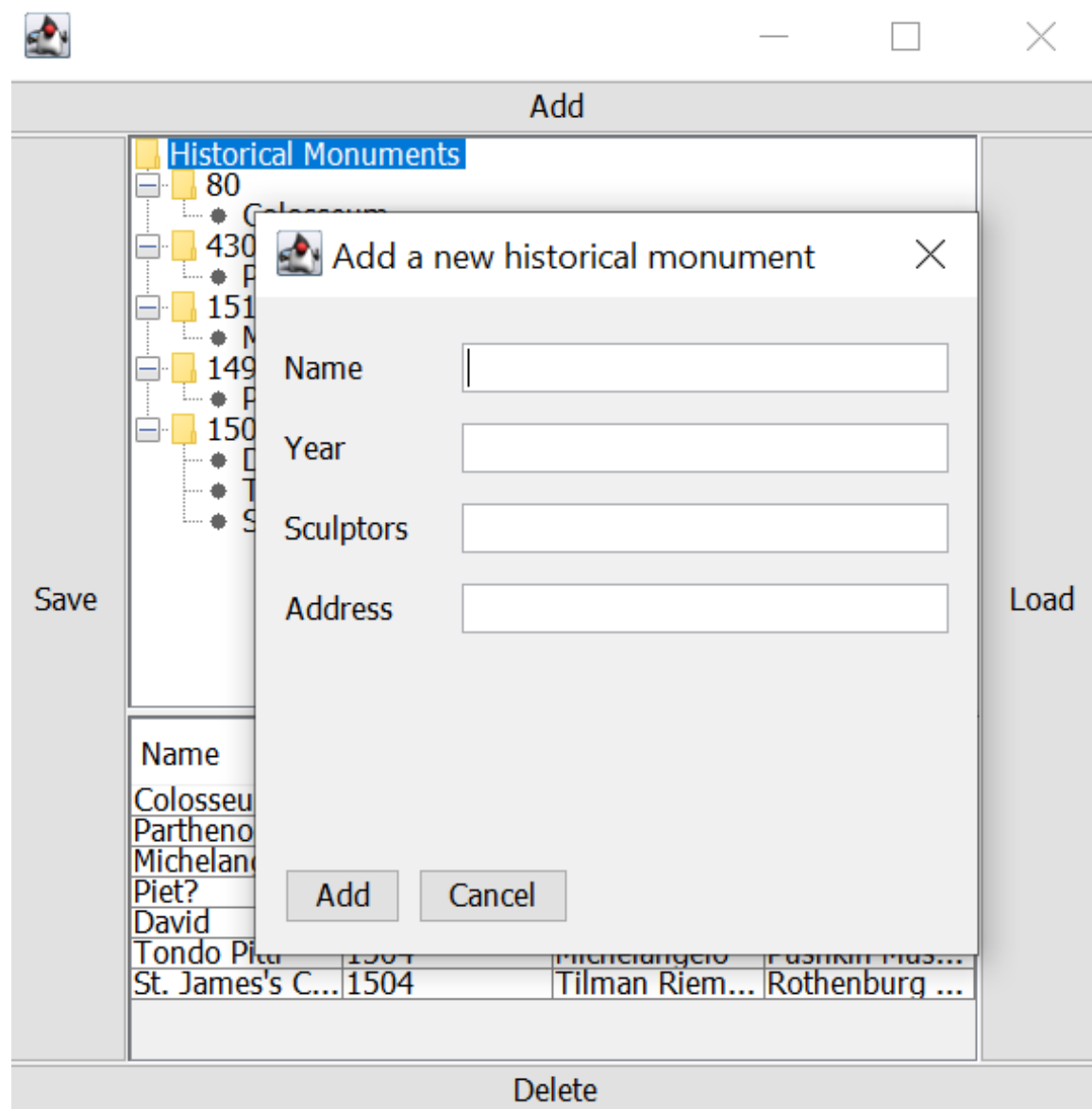


Рисунок 4.1 – Добавление нового элемента.

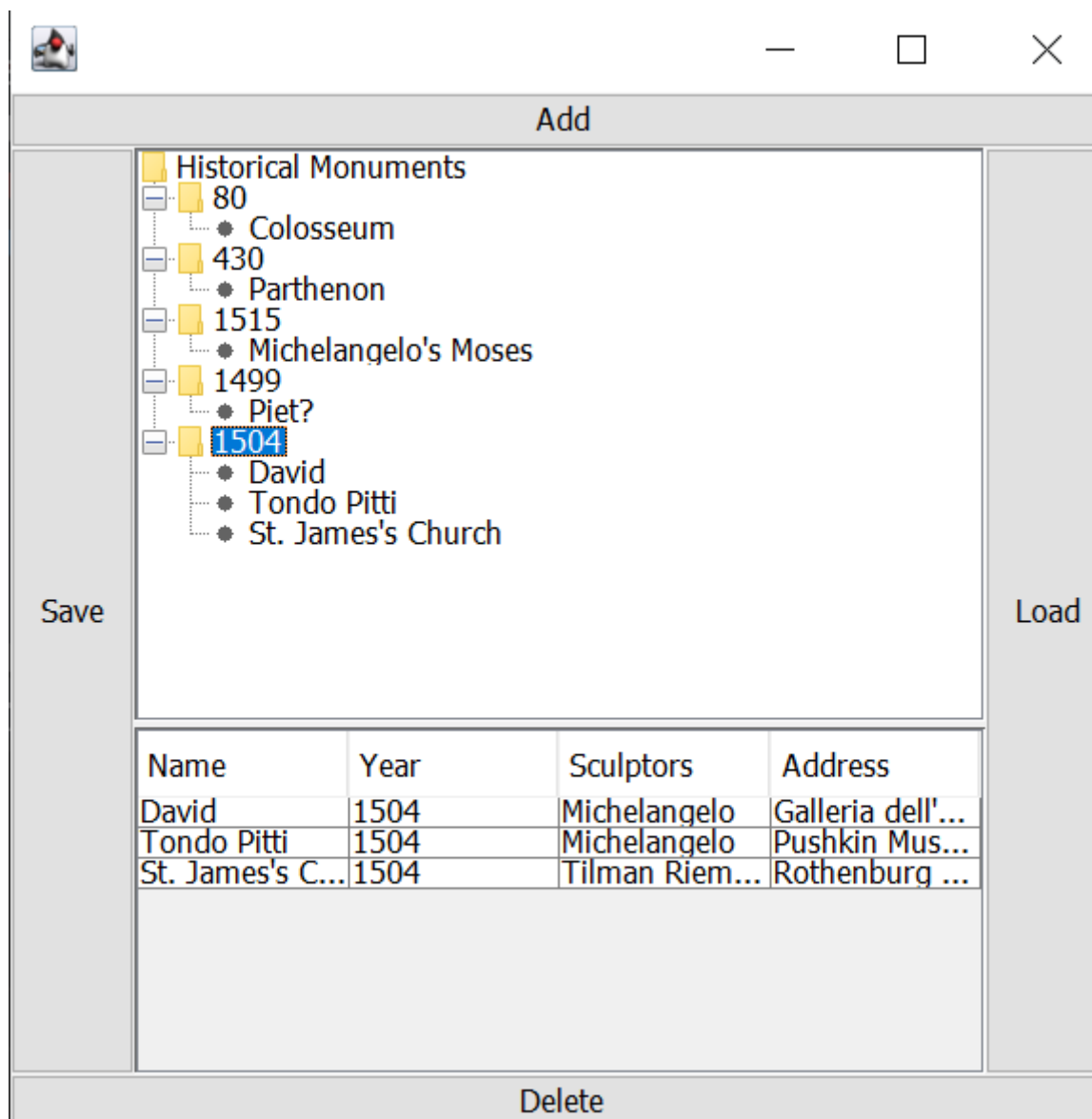


Рисунок 4.2 – Содержание выбранного Node.

6. Лабораторная работа №6

Постановка задачи:

Для выполнения задания используется вариант решения задания №3. Модифицировать вашу программу следующим образом.

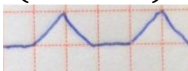
Создать тестовое приложение, добавить в класс рисования алгебраической линии возможность «перетаскивание». Реализовать необходимые интерфейсы в классе и в приложении для демонстрации «перетаскивания» алгебраической линии между несколькими копиями тестового приложения.

При реализации интерфейса тестового приложения следовать рекомендациям CUI.

Вариант №13.

Линия: Локон Аньези.

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

Контур: 

Особенности реализации:

Для реализации задания используются DragGestureListener, DragSourceListener, DropTargetListener, MouseListener, MouseMotionListener. Пакет java.awt.datatransfer предоставляет возможность передачи данных между приложениями и поддерживает метод обмена данными типа «вырезание и вставка» (cut-and-paste).

Пакет java.awt.dnd поддерживает метод передачи данных типа «перетаскивание» (drag-and-drop).

Класс java.awt.datatransfer.DataFlavor является центральным в процессе передачи данных; он представляет тип данных, подлежащих передаче. Каждый формат данных содержит удобочитаемое имя, объект Class, указывающий тип передаваемых данных, и тип MIME, определяющий кодировку, используемую при передаче данных.

Результат работы программы:

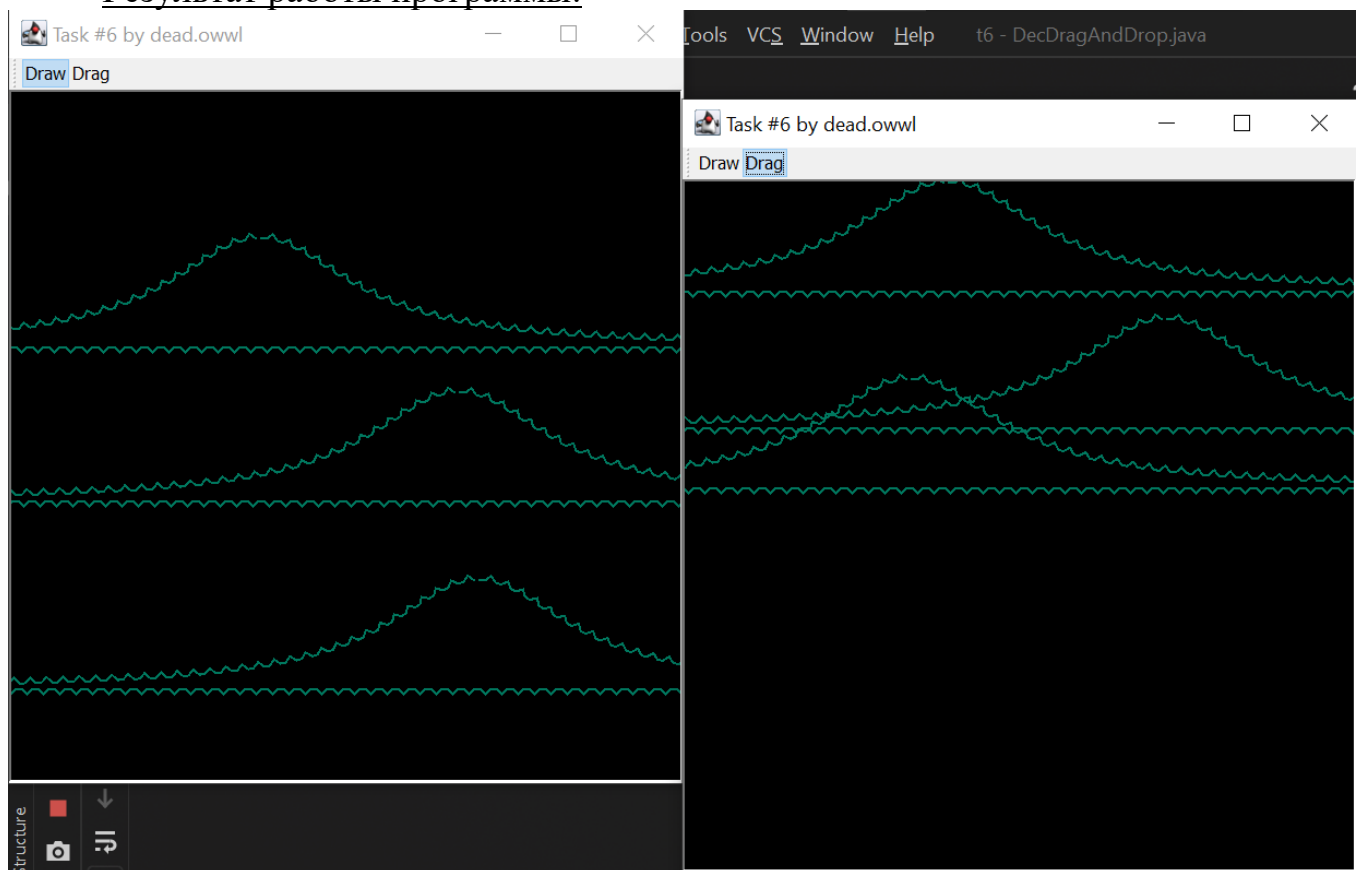


Рисунок 6.1 – Запуск двух экземпляров приложения с уже нарисованными Линиями.

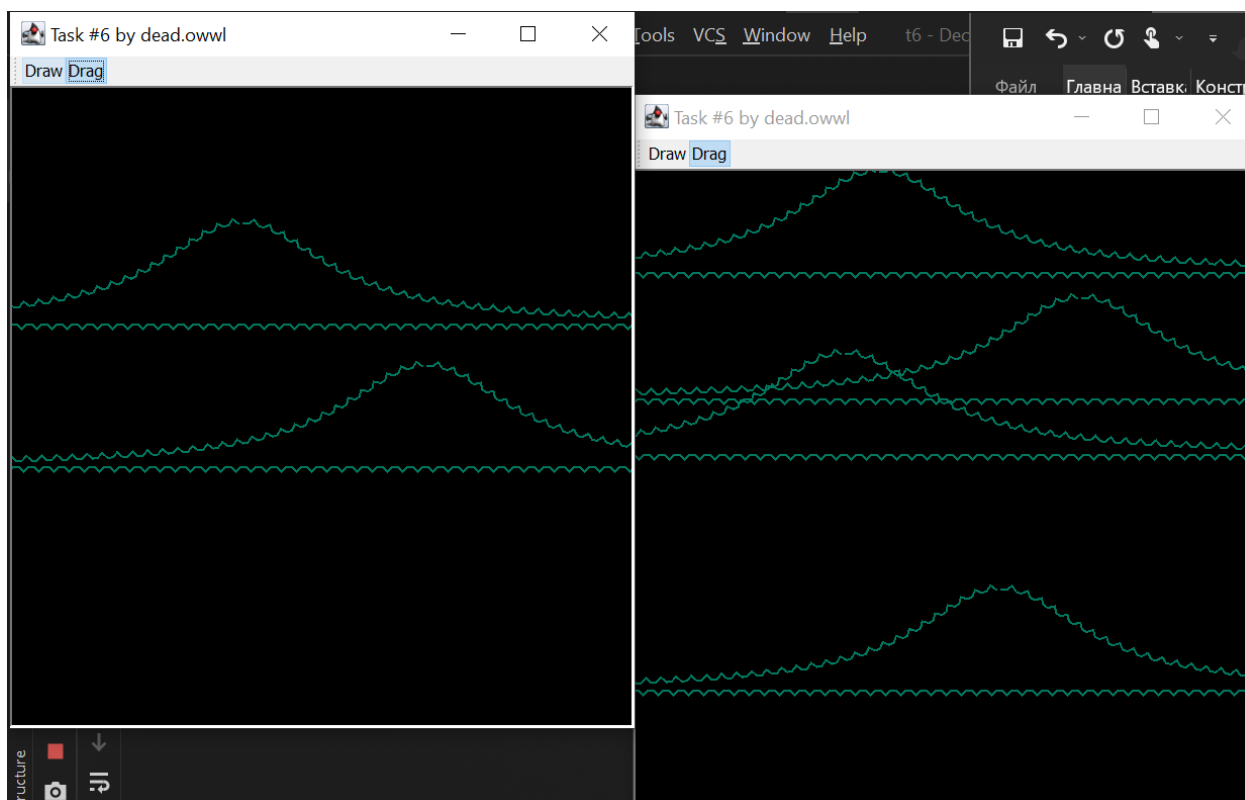


Рисунок 6.2 – Перетаскивание нижней Линии левого окна на правое окно.

7. Лабораторная работа №7

Постановка задачи:

- Исследовать предложенную предметную область, спроектировать структуру базы данных объектов выбранной предметной области (из не менее чем 2-х таблиц объектов). Согласуйте проект БД с преподавателем. Обязательно работаем с Derby!
- 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).

Вариант №13.

Каталог исторических памятников.

Особенности реализации:

Структура базы данных объектов представляет три таблицы, реализованные соответственно для MonumentItem, VisitorItem и VisitorBasketItem. Главная таблица – CatalogTableMonumentModel, содержит поля “Название исторического памятника - Тип исторического памятника - Возраст – Цена посещения/просмотра”, ключевое значение – название памятника. Таблица VisitorItem содержит имя посетителя, его пол и дату рождения. Таблица VisitorBasketItem представляет собой информацию о посещении конкретным туристом исторического памятника

и состояние оплаты. Таким образом таблица VisitorBasketItem зависит от поля с названием памятника в CatalogTableMonumentModel и поля с именем туриста в VisitorItem. Осуществлено добавление нового элемента в каждую таблицу и создание базы данных, заполнение ее готовыми значениями.

Результат работы программы:

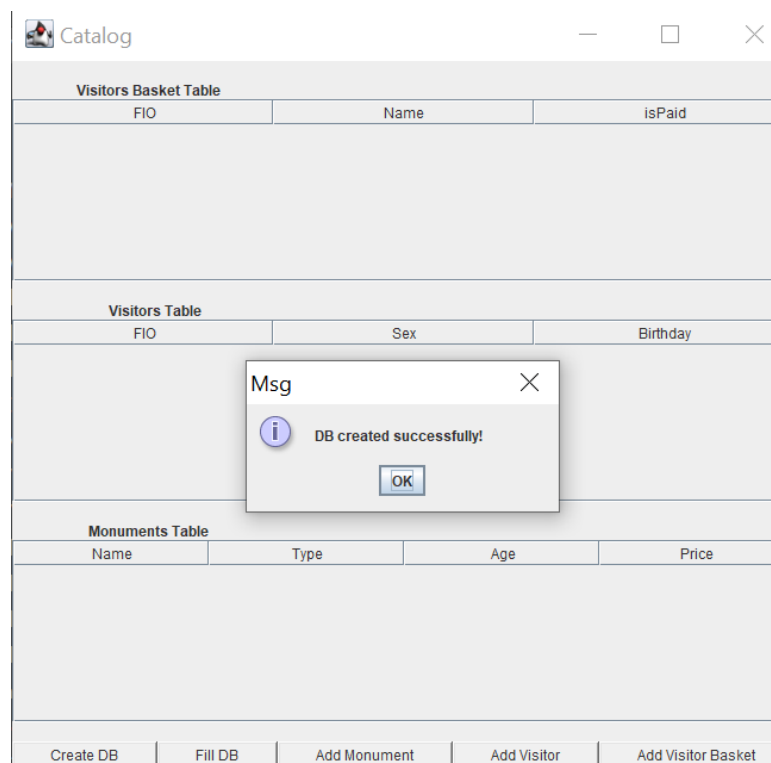



Рисунок 7.1 – Создание базы данных.


Catalog

Visitors Basket Table		
FIO	Name	isPaid
Angela Schruder	Lincoln Memorial	<input checked="" type="checkbox"/>
Benoit Charpentier	Parthenon	<input type="checkbox"/>
David Cranz	Lincoln Memorial	<input checked="" type="checkbox"/>
Frieder Fromm	St. Peter's Basilica	<input checked="" type="checkbox"/>
Michaela Holst	Lincoln Memorial	<input type="checkbox"/>
Nadine Durand	St. Peter's Basilica	<input checked="" type="checkbox"/>
Nadine Durand	Parthenon	<input checked="" type="checkbox"/>
Wolfgang Schultz	St. Peter's Basilica	<input checked="" type="checkbox"/>

Visitors Table		
FIO	Sex	Birthday
Andre Petit		1/27.02.1989
Angela Schruder		0/11.10.1983
Benoit Charpentier		1/21.06.1979
David Cranz		1/01.02.1980
Frieder Fromm		1/06.05.1977
Michaela Holst		0/07.03.1991
Nadine Durand		0/31.08.1988
Wolfgang Schultz		1/10.01.1990

Monuments Table			
Name	Type	Age	Price
Colosseum	Memorial	80	10
Eiffel Tower	Tower	1889	10
Forbidden City	Palace Complex	1406–1420	10
Lincoln Memorial	Memorial	1922	10
Palace of Versailles	Palace Complex	1682	10
Parthenon	Mausoleum	1653	10
St. Peter's Basilica	Church	18 November 1626	11
Taj Mahal	Mausoleum	1653	10

Create DB

Fill DB

Add Monument

Add Visitor

Add Visitor Basket

Рисунок 7.2 – Заполнение базы данных.

Catalog

Visitors Basket Table		
FIO	Name	isPaid
Angela Schruder	Lincoln Memorial	<input checked="" type="checkbox"/>
Benoit Charpentier	Parthenon	<input type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>

Add Monument

Price:
 Name:
 Age:
 Type:

Add Close

Monuments Table	
Name	Type
Colosseum	Memorial
Eiffel Tower	Tower
Forbidden City	Palace Complex
Lincoln Memorial	Memorial
Palace of Versailles	Palace Complex
Parthenon	Mausoleum
St. Peter's Basilica	Church
Taj Mahal	Mausoleum

Birthdays	
FIO	Age
1	27.02.1989
0	11.10.1983
1	21.06.1979
1	01.02.1980
1	06.05.1977
0	07.03.1991
0	31.08.1988
1	10.01.1990

Monuments Table			
Name	Type	Age	Price
Colosseum	Memorial	80	10
Eiffel Tower	Tower	1889	10
Forbidden City	Palace Complex	1406–1420	10
Lincoln Memorial	Memorial	1922	10
Palace of Versailles	Palace Complex	1682	10
Parthenon	Mausoleum	1653	10
St. Peter's Basilica	Church	18 November 1626	11
Taj Mahal	Mausoleum	1653	10

Create DB Fill DB Add Monument Add Visitor Add Visitor Basket

Рисунок 7.3 – Добавление нового памятника.

Catalog

Visitors Basket Table		
FIO	Name	isPaid
Angela Schruder	Lincoln Memorial	<input checked="" type="checkbox"/>
Benoit Charpentier	Parthenon	<input type="checkbox"/>
David Cranz	Lincoln Memorial	<input checked="" type="checkbox"/>
Frieder Fromm	St. Peter's Basilica	<input checked="" type="checkbox"/>
Michaela Holst	Lincoln Memorial	<input type="checkbox"/>
Nadine Durand	St. Peter's Basilica	<input checked="" type="checkbox"/>
Nadine Durand	Parthenon	<input checked="" type="checkbox"/>
Wolfgang Schultz	St. Peter's Basilica	<input checked="" type="checkbox"/>

Visitors Table		
FIO	Sex	Birthdate
Andre Petit		
Angela Schruder		
Benoit Charpentier		
David Cranz		
Frieder Fromm		
Michaela Holst		
Nadine Durand		
Wolfgang Schultz		

Add Visitor

FIO:
 Sex:
 Birthdate:

Add Close

Monuments Table	
Name	Type
Colosseum	Memorial
Eiffel Tower	Tower
Forbidden City	Palace Co
Lincoln Memorial	Memorial
Palace of Versailles	Palace Co
Parthenon	Mausoleu
St. Peter's Basilica	Church
Taj Mahal	Mausoleum

Birthdays	
FIO	Age
1	27.02.1989
0	11.10.1983
1	21.06.1979
1	01.02.1980
1	06.05.1977
0	07.03.1991
0	31.08.1988
1	10.01.1990

Monuments Table			
Name	Type	Age	Price
Colosseum	Memorial	80	10
Eiffel Tower	Tower	1889	10
Forbidden City	Palace Co	1406–1420	10
Lincoln Memorial	Memorial	1922	10
Palace of Versailles	Palace Co	1682	10
Parthenon	Mausoleu	1653	10
St. Peter's Basilica	Church	18 November 1626	11
Taj Mahal	Mausoleum	1653	10

Create DB Fill DB Add Monument Add Visitor Add Visitor Basket

Рисунок 7.3 – Добавление нового посетителя.

Catalog

Visitors Basket Table		
FIO	Name	isPaid
Angela Schruder	Lincoln Memorial	<input checked="" type="checkbox"/>
Benoit Charpentier	Parthenon	<input type="checkbox"/>
David Cranz	Lincoln Memorial	<input checked="" type="checkbox"/>
Frieder Fromm	St. Peter's Basilica	<input checked="" type="checkbox"/>
Michaela Holst	Lincoln Memorial	<input type="checkbox"/>
Nadine Durand	St. Peter's Basilica	<input checked="" type="checkbox"/>
Nadine Durand	Parthenon	<input checked="" type="checkbox"/>
Wolfgang Schultz	St. Peter's Basilica	<input checked="" type="checkbox"/>

Add Visitor Basket

FIO:
 Name:
 isPaid:

Birthday	
1	27.02.1989
0	11.10.1983
1	21.06.1979
1	01.02.1980
1	06.05.1977
0	07.03.1991
0	31.08.1988
1	10.01.1990

Age	Price
80	10
1889	10
1406–1420	10
1922	10
1682	10
1653	10
18 November 1626	11
1653	10

Lincoln memorial	memorial	
Palace of Versailles	Palace Complex	
Parthenon	Mausoleum	
St. Peter's Basilica	Church	
Taj Mahal	Mausoleum	

Рисунок 7.4 – Добавление посетителя конкретного памятника.

8. Лабораторная работа №8

Постановка задачи:

- Изучить материал примера по быстрому введению в среду разработки NetBeans и компоненты JavaBeans по адресу <http://docs.oracle.com/javase/tutorial/javabeans/quick/index.html>.

- Разработать простой компонент варианта задания на базе класса

Canvas. Создать файл манифеста и упаковать компонент вместе с исходным кодом разработанных классов. При разработке поместить классы в пакет `bsu.fpmi.edupract`.

- Создать тестовое приложение в NetBeans с использованием разработанного компонента.

Вариант №4.

3D вертикальная линия. Свойства – высота.

Особенности реализации:

Класс `Line 3D` наследует класс `Canvas` и определяет свойства `width` (константное) и `height` (высота), для высоты создаются методы `get` и `set`.

Класс содержит конструктор по умолчанию, в которой высоте присваивается значение 200.

Результат работы программы:

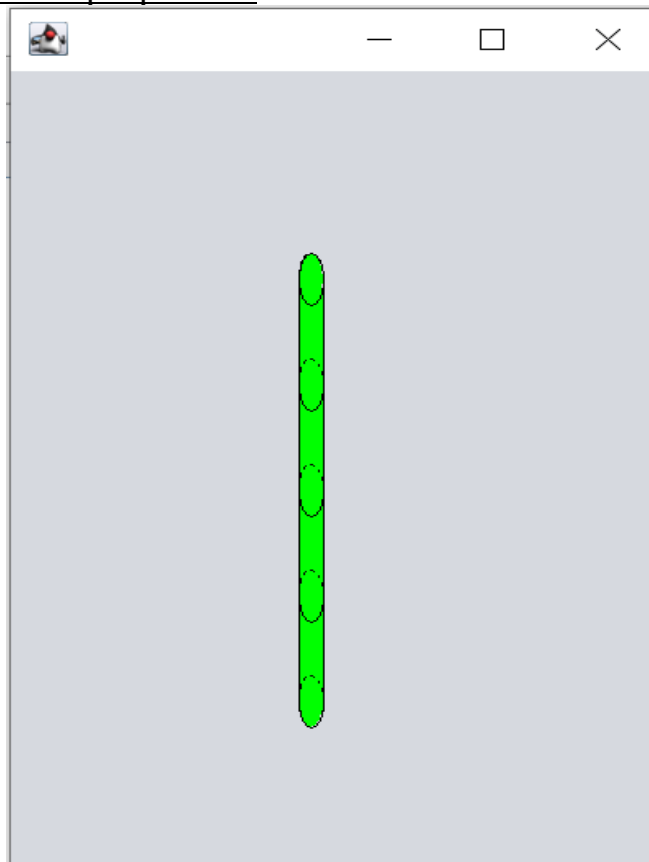


Рисунок 8.1 – Рисование 3D вертикальной линии.

9. Лабораторная работа №9

Постановка задачи:

1) Разработать компонент варианта задания. Создать файл манифеста и упаковать компонент вместе с исходным кодом разработанных классов. При разработке поместить все классы в пакет `bsu.fpmi.educpract`.

2) Компонент должен реализовать класс `BeanInfo` с информацией о компоненте.

3) Создать тестовое приложение в NetBeans с использованием разработанного компонента.

Вариант №4.

Одноточный статический текст, две независимые радио-кнопки и обычная кнопка. Свойства: текст, текст кнопки, текст радио-кнопок. Событие генерируется при нажатии на обычную кнопку. Событие передает состояние радио-кнопок.

Особенности реализации:

Для кнопки реализован `ActionListener actionPerformed`, передающий состояние независимых радио-кнопок и состояние самой кнопки.

Реализован класс `MyJPanelBeanInfo`, работающий со свойствами подкомпонентов, позволяющий работать с `Properties`.

Результат работы программы:

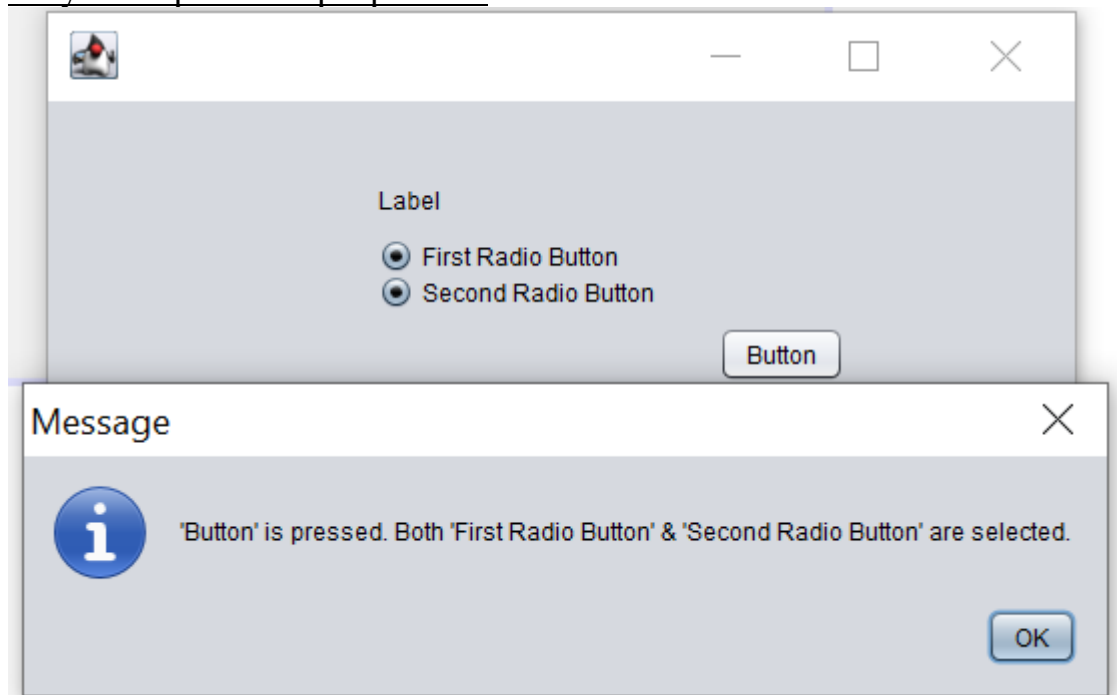


Рисунок 9.1 – Демонстрация обработанного события нажатия кнопки с передачей параметров радио-кнопок.

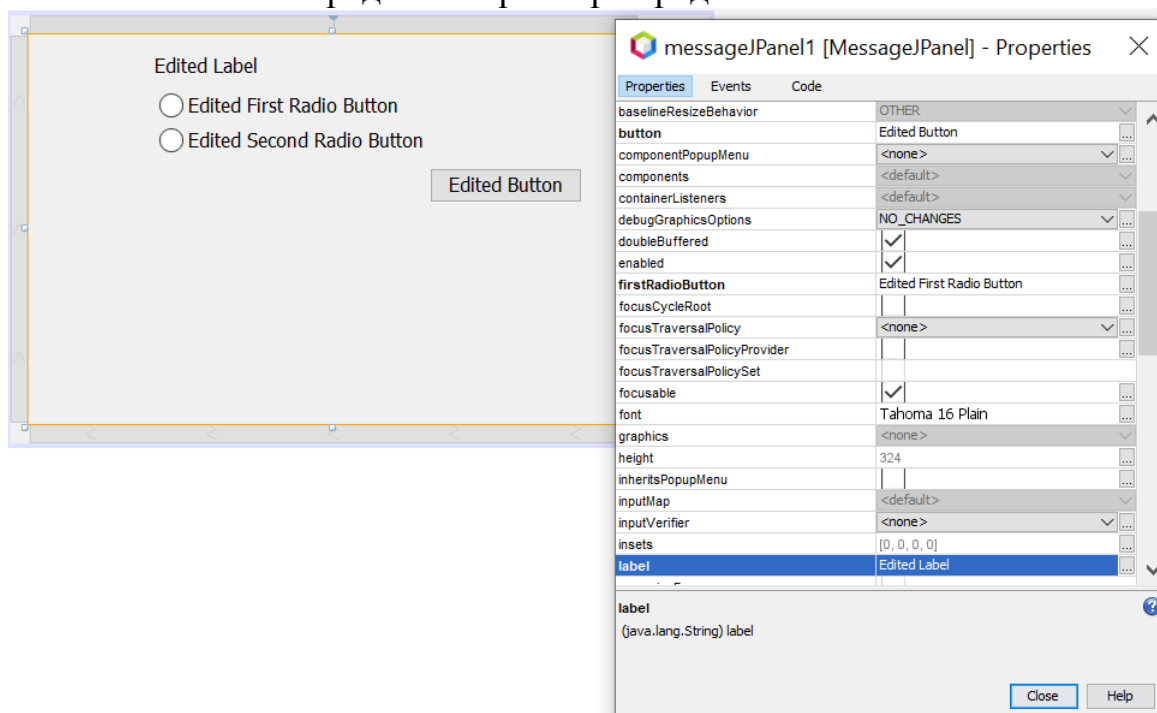


Рисунок 9.2 – Изменение свойств подкомпонент.

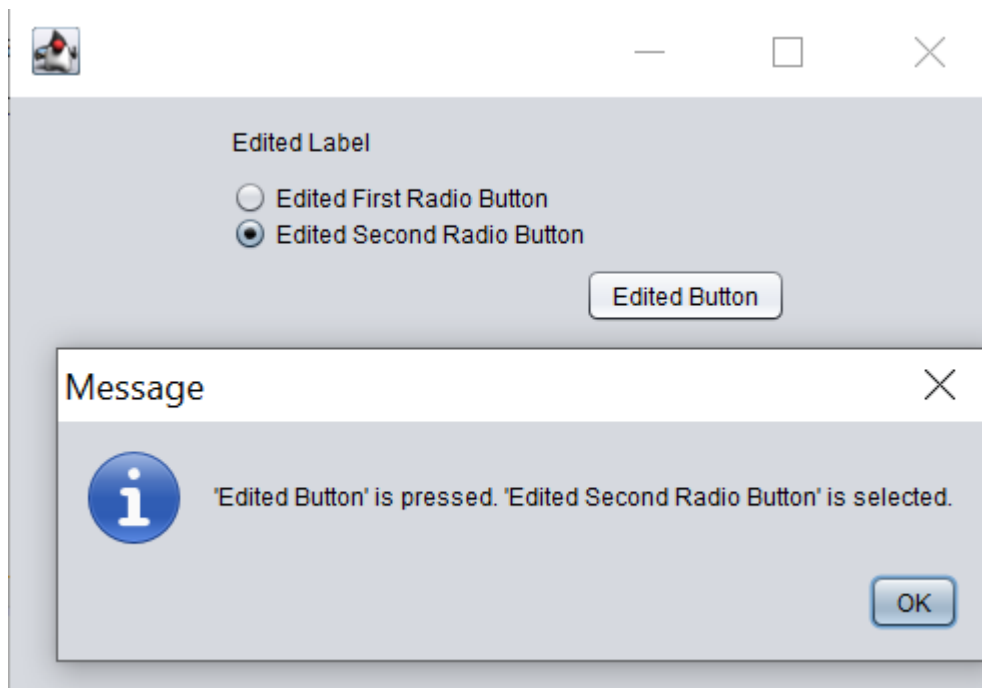


Рисунок 9.3 – Демонстрация обработанного события нажатии кнопки с передачей параметров радио-кнопок после изменения свойств подкомпонентов.

10. Лабораторная работа №10

Постановка задачи:

Для решения задания №10 использовать решенный вариант задания №9. Номера вариантов сохраняются. Модифицируем тип свойства компонента так, как указано ниже:

Основная задача:

Создать собственный редактор для каждого свойства компонента. Каждый редактор ограничивает возможные значения свойства, предоставляя выбор из списка трёх – пяти допустимых значений. Регистрируем редакторы в классе BeanInfo компонента.

Дополнительно:

Создать настройщик компонента, который позволит изменять списки допустимых значений для свойств компонента.

Вариант №4.

Однотрочный статический текст, две независимые радио-кнопки и обычная кнопка. Свойства: текст, текст кнопки, текст радио-кнопок. Событие генерируется при нажатии на обычную кнопку. Событие передает состояние радио-кнопок.

Особенности реализации:

Для каждого свойства компонента реализованы вспомогательные классы MyLabelEditor, FirstRadioButtonEditor, SecondRadioButtonEditor и MyButtonEditor, которые обрабатывают изменения свойств компонента для

каждого отдельного подкомпонента. Изменения зафиксированы в MyJPanelInfo для установки новых свойств необходимым компонентам.

Результат работы программы:

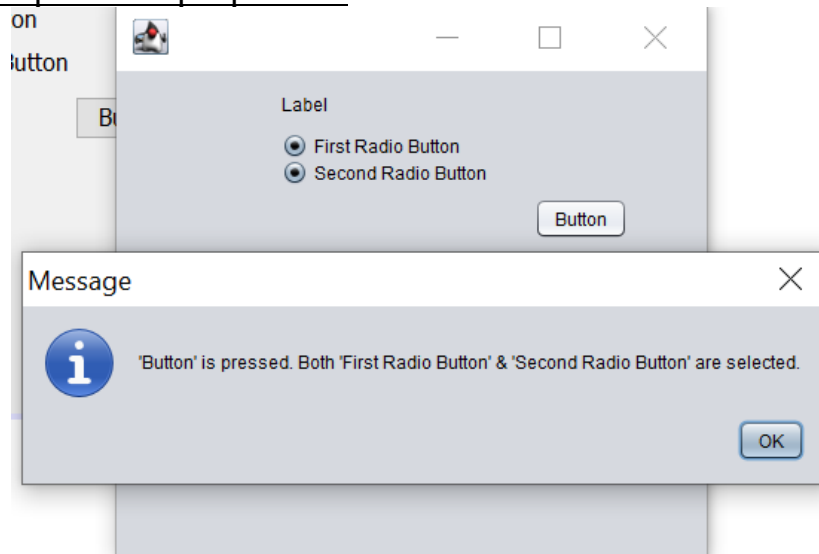


Рисунок 10.1 – Демонстрация обработанного события нажатия кнопки с передачей параметров радио-кнопок

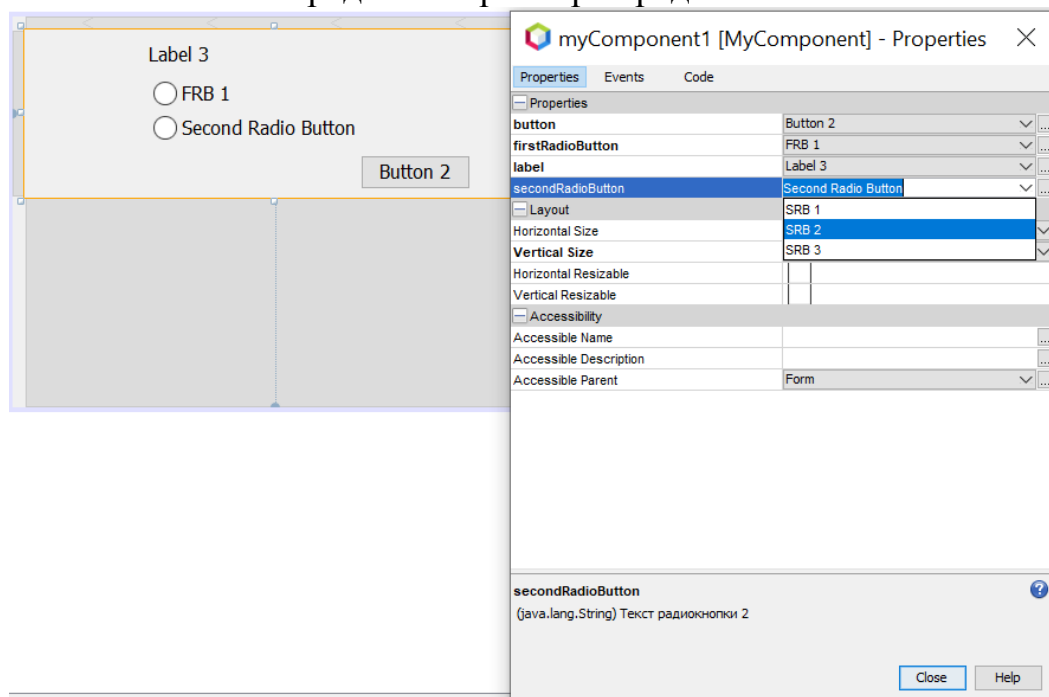


Рисунок 10.2 – Изменения свойств подкомпонент через реализованные выпадающие списки в Properties

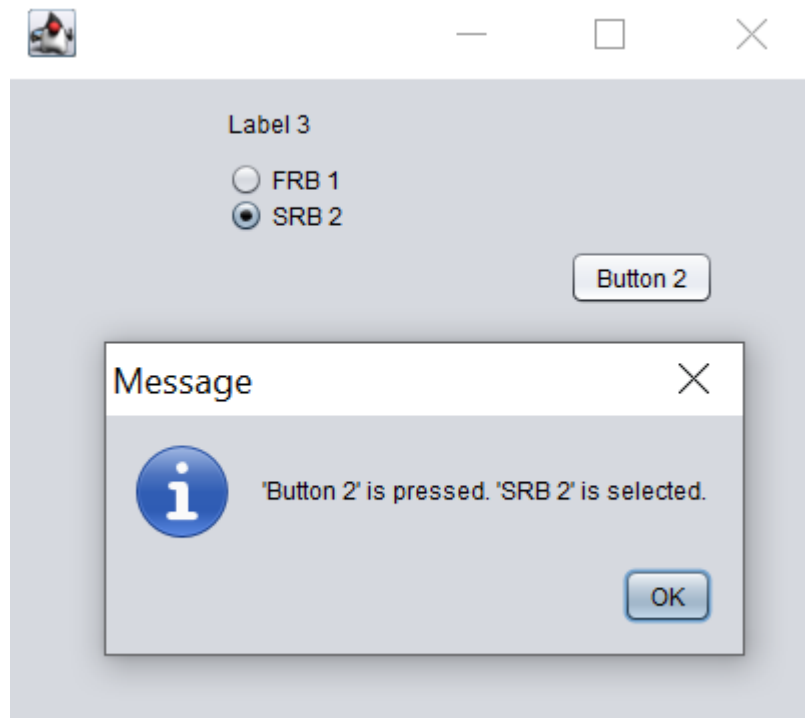


Рисунок 10.3 – Демонстрация обработанного события нажатия кнопки с передачей параметров радио-кнопок после изменения свойств подкомпонентов.

11. Лабораторная работа №11

Постановка задачи:

Создать сервлет и взаимодействующие с ним пакеты Java-классов и HTML документов, выполняющие действия для решения вашего варианта задания. Представить решение в виде web-приложения.

Вариант 14.

Base64 кодирование/декодирование строки текста.

Особенности реализации:

Для решения задания был создан проект по типу Dynamic Web Project, где был создан класс реализации сервлета EncodeServlet. Класс сервлета наследуется от класса HttpServlet. Перед определением класса прописана аннотация.WebServlet, которая указывает, с какой конечной точкой будет сопоставляться данный сервлет. То есть данный сервлет будет обрабатывать запросы по адресу. Для обработки GET-запросов (например, при обращении к сервлету из адресной строки браузера) сервлет должен переопределить метод doGet. То есть, к примеру, в данном случае запрос по адресу /encode-servlet будет обрабатываться методом doGet. Этот метод принимает два параметра. Параметр типа HttpServletRequest инкапсулирует всю информацию о запросе. А параметр типа HttpServletResponse позволяет управлять ответом. Для запуска сервлета воспользуемся опять же контейнером сервлетов Apache Tomcat. В каталоге Tomcat в папке web создаётся каталог для нового приложения, который носит название

helloapp. В проекте создана страница index.html для отправки форм. Это самый распространённый способ отправки данных.

Результат работы программы:

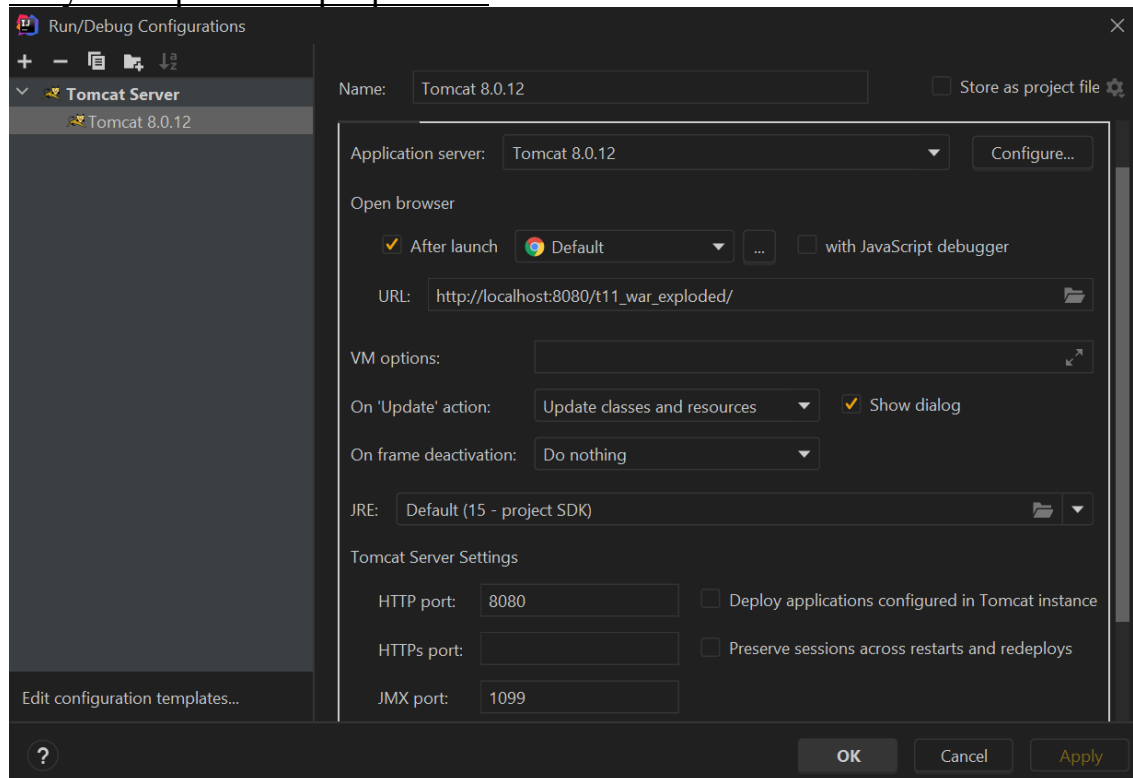
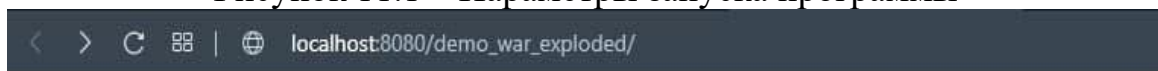


Рисунок 11.1 – Параметры запуска программы



Base64

☒ Encode
☐ Decode

Рисунок 11.2 – Кодирование входной строки

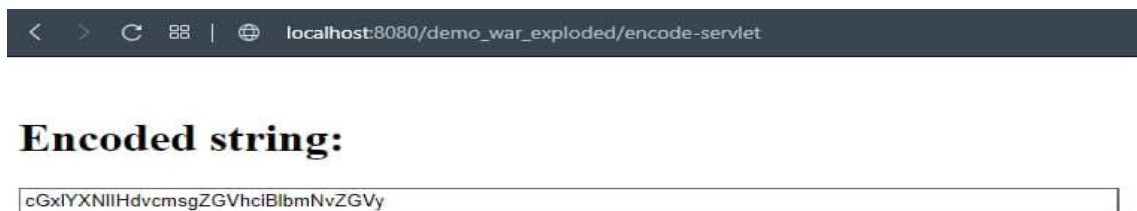


Рисунок 11.3 – Результат кодирования входной строки

Base64

A screenshot of a web browser window. The address bar shows 'localhost:8080/demo_war_exploded/decode-servlet'. The page content displays 'Base64' in bold. Below this, a text input field contains the Base64 encoded string 'cGxIYXNlIHdvcmsgZGVhciBlbmNvZGVy'. There are two radio buttons: 'Encode' (unselected) and 'Decode' (selected). Below the radio buttons is a 'Submit' button.

Рисунок 11.4 – Декодирование входной строки

Decoded string:

please work dear encoder

Рисунок 11.5 – Результат декодирования входной строки

12. Лабораторная работа №12

Постановка задачи:

Проанализируйте ваш 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 33 MUD, другие классы примера? Оформите эти размышления в вашем отчёте в качестве анализа предметной области. При реализации, по возможности, используйте парадигму MUD и классы примера 2 при реализации вашего варианта задания. 3) Создайте на основе технологии RMI клиент/серверное приложение:

Вариант 5.

Заказ еды по сети. Сервер предоставляет подключившимся клиентам меню со списком блюд и ценами. Клиент сообщает серверу свой заказ и адрес доставки.

Особенности реализации:

Для создания приложения на базе RMI был создан интерфейс, расширяющий `java.rmi.Remote` `Mud`. В этом интерфейсе определены экспортируемые методы, реализуемые удалёнными объектами и объявленные как генерирующие исключение `java.rmi.RemoteException`. Для сервера это методы `getPerson(String name)`, `delClient(MailClientInterface current)`, `addPerson(MailClientInterface current)`, `printMes(String str)`. Для клиента это `showMessage()`, `getName()`, `addOrder(String letter)`. Был определен класс, производный от `java.rmi.server.UnicastRemoteObject`, реализующий удаленный интерфейс, `MudClient`, создающий экземпляр удаленного объекта и экспортирован объект, доступный для использования клиентами путем регистрации имени объекта в службе реестра. При использовании RMI клиент и сервер не взаимодействуют непосредственно. На стороне клиента ссылка на удаленный объект реализуется в виде экземпляра класса-заглушки. Когда клиент вызывает удаленный метод, в действительности вызывается метод объекта-заглушки. Заглушки генерируются автоматически утилитой `rmic`.

Результат работы программы:

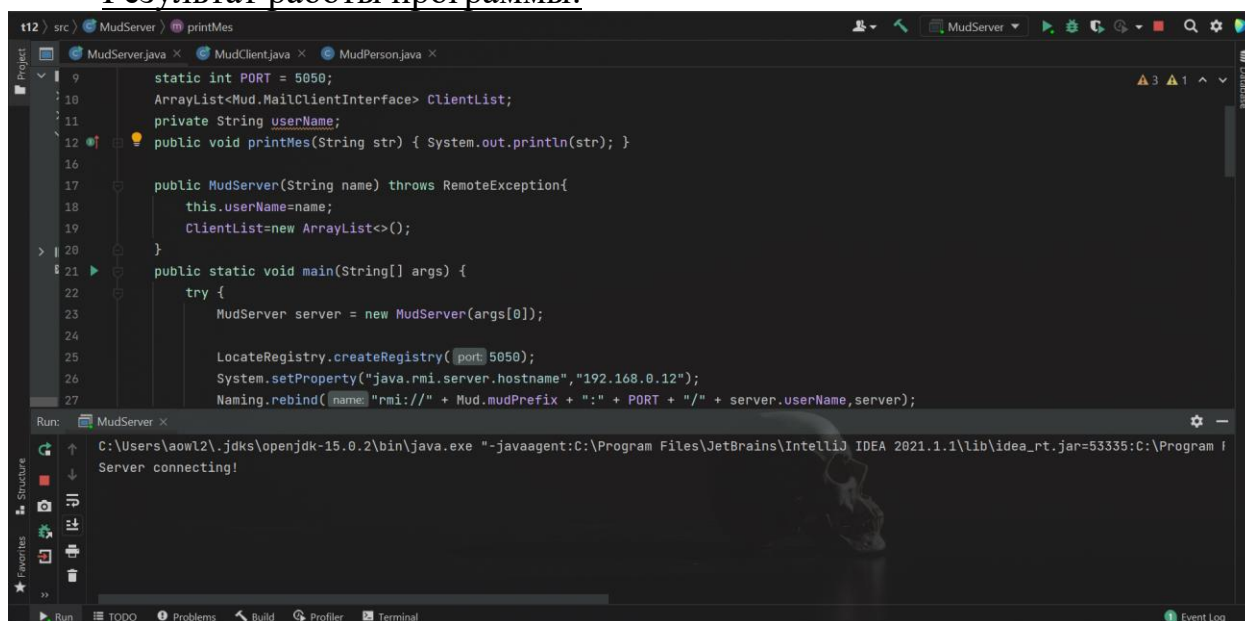


Рисунок 12.1 – Запуск сервера и его подключение


```
Run: MudServer x MudClient x
C:\Users\ao\l2\jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.1\lib\idea_rt.jar=53353:C:\Program I
You are connected to MailServerInterface!
[MyServer]:Do you want to register or login?
  r : register
  l : login
>> r
>> Input your name: name
>> You are successful registered as "name"
1) Драники  Цена: 3.76 руб.
2) Салат Цезарь.  Цена: 2.24 руб.
3) Яйца.  Цена: 0.87 руб.
4) Каша вязкая.  Цена: 0.79 руб.
5) Суп.  Цена: 1.01 руб.
6) Кока-Кола.  Цена: 0.95 руб.

>> |
```

Рисунок 12.2 – Запуск первого клиента, регистрация нового пользователя и вывод предлагаемого меню

```
Run: MudServer x MudClient x
2) Салат Цезарь.  Цена: 2.24 руб.
3) Яйца.  Цена: 0.87 руб.
4) Каша вязкая.  Цена: 0.79 руб.
5) Суп.  Цена: 1.01 руб.
6) Кока-Кола.  Цена: 0.95 руб.

>> s
>> Input your address
>> my home
>> input your order

>> 2 5
>> sh
name's orders:
Name: name
Addresss: my home
Order: 2 5

>>
```

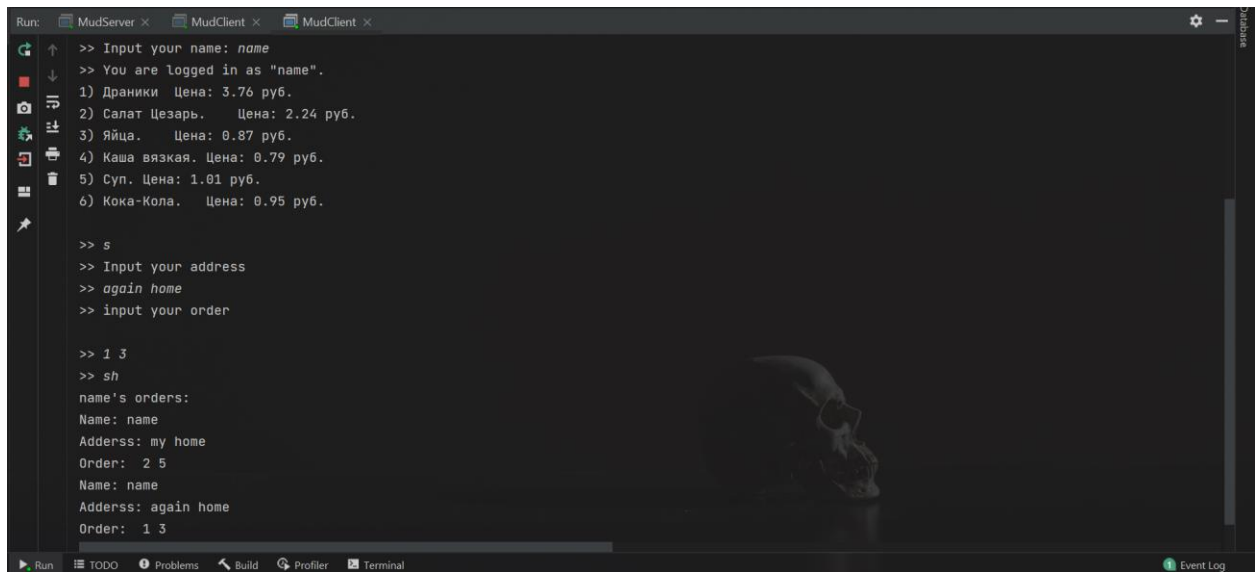
Рисунок 12.3 – Заказ первого клиента и вывод заказа клиенту

```
Run: MudServer x MudClient x
C:\Users\ao\l2\jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.1\lib\idea_rt.jar=53335:C:\Program I
Server connecting!
Name: name
Addresss: my home
Order: 2 5
|
```

Рисунок 12.4 – Вывод заказов на сервер

```
Run: MudServer x MudClient x MudClient x
C:\Users\ao\l2\jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.1\lib\idea_rt.jar=53497:C:\Program I
You are connected to MailServerInterface!
[MyServer]:Do you want to register or login?
  r : register
  l : login
>> r
>> Input your name: name
>> Name "name" is already used.
[MyServer]:Do you want to register or login?
  r : register
  l : login
>> |
```

Рисунок 12.5 – Запуск второго клиента и попытка регистрации пользователя с существующим именем

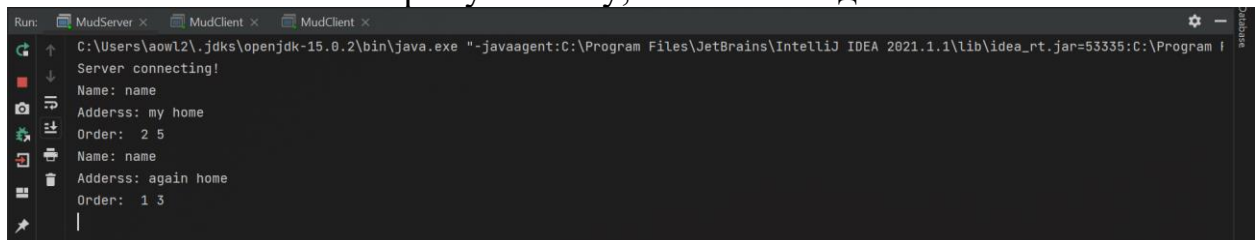


```
Run: MudServer x MudClient x MudClient x
>> Input your name: name
>> You are logged in as "name".
1) Драники Цена: 3.76 руб.
2) Салат Цезарь. Цена: 2.24 руб.
3) Яйца. Цена: 0.87 руб.
4) Каша вязкая. Цена: 0.79 руб.
5) Суп. Цена: 1.01 руб.
6) Кока-Кола. Цена: 0.95 руб.

>> s
>> Input your address
>> again home
>> input your order

>> 1 3
>> sh
name's orders:
Name: name
Addresss: my home
Order: 2 5
Name: name
Addresss: again home
Order: 1 3
```

Рисунок 12.6 – Вход с существующим именем со второго клиента, вывод меню второму клиенту, заказ и вывод заказа



```
Run: MudServer x MudClient x MudClient x
C:\Users\awl2\.jdk\openjdk-15.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.1.1\lib\idea_rt.jar=53335:C:\Program I
Server connecting!
Name: name
Addresss: my home
Order: 2 5
Name: name
Addresss: again home
Order: 1 3
|
```

Рисунок 12.7 – Вывод заказов на сервер

Заключение

По итогу работы с приложениями Java на занятиях по учебной практике приобретен богатый опыт, были изучены новые методы работы.

Был изучен и проработан принцип работы с 2D-изображениями в Java 2D API, с графикой и пользовательским интерфейсом Java, базами данных средствами JDBC, изучена новая среда разработки NetBeans с интерфейсом JavaBeans и сервлеты. Были усвоены знания по вызову метода удалённого объекта, или по-другому RMI.

Список использованной литературы

1. Г. Шилдт. Java . Полное руководство – 8-е издание – М.: ООО И.Д. Вильямс, 2012
2. Документация по Apache Tomcat [Электронный ресурс]. – Режим доступа: <http://tomcat.apache.org/tomcat-8.0-doc/>
3. Derby [Электронный ресурс]. – Режим доступа: <https://db.derby.org>
4. Хорстманн К., Корнелл Г. - "Java. Библиотека профессионала. Том 1 и 2". 9-е издание (2014)