**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра информационных систем управления**

**Отчёт**

**по учебной вычислительной практике**

Возовикова Никиты Александровича

студента 2 курса группы 10

специальности «Компьютерная Безопасность»

дневной формы получения

высшего образования

Научный руководитель:

старший преподаватель

Гутников Сергей Евгеньевич

Минск, 2021

Оглавление

[Лабораторная работа №1 3](#_Toc72826559)

[Лабораторная работа №2 5](#_Toc72826560)

[Лабораторная работа № 3 7](#_Toc72826561)

[Лабораторная работа №4 9](#_Toc72826562)

[Лабораторная работа № 5 12](#_Toc72826563)

[Лабораторная работа № 6 15](#_Toc72826564)

[Лабораторная работа №7 18](#_Toc72826565)

[Лабораторная работа №8 20](#_Toc72826566)

[Лабораторная работа №9 22](#_Toc72826567)

[Лабораторная работа №10 25](#_Toc72826568)

[Лабораторная работа №11 28](#_Toc72826569)

[Лабораторная работа №12 30](#_Toc72826570)

[Заключение 33](#_Toc72826571)

[Список использованной литературы 33](#_Toc72826572)

# Лабораторная работа №1

## **Тема:**

**«**Java 2D API»

## **Цель**:

Научиться использовать средства двумерной графики в Java из пакета *java.awt*, в том числе научится создавать собственный класс фигур, реализующих интерфейс Shape, а также использовать класс *AffineTransform* для их трансформирования.

## Постановка задачи:

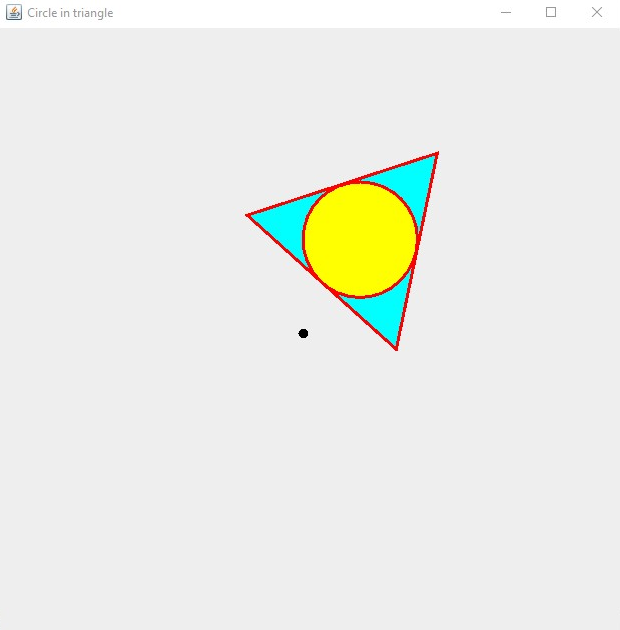
* Для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape (можно взять базовый библиотечный класс, реализующий Shape)
* Выполнить указанное в задании перемещение указанной фигуры с помощью аффинного преобразования координат
* Выполнить рисунок в окне фрейма с выбранной толщиной границы фигуры, цветом границы и цветом внутренней области (вводить толщину и цвет в качестве аргумента ваших программ).

Изобразить круг, вписанный в равносторонний треугольник. Задать вращение фигуры вокруг выбранной точки против часовой стрелки.

## Особенности реализации:

Для создания сложной фигуры использовался класс Area из java.awt.geom.Area. Для преобразования координат используются аффинные преобразования координт(AfineTransform). Обновление кадров происходит с помощью timer из javax.swing.Timer.

## Результат работы программы:



*Рисунок 1.1*

## Выводы:

* Класс Graphics2D, в отличие от своего предка Graphics, позволяет использовать вещественные координаты.
* Для удобного способа задания параметров отрисовки контуров можно использовать класс BasicStroke, который реализует интерфейс Stroke.
* Для создания своего собственного класса, представляющего собой фигуру, достаточно реализовать интерфейс Shape. При правильной реализации, Graphics2D сможет корректно отобразить нашу фигуру.
* Аффинные преобразования при использовании графики в Java не обязательно делать вручную, можно использовать класс AffineTransform.

# Лабораторная работа №2

## Тема:

**«**Java 2D API (продолжение)**»**

## Цель:

Научиться методам обработки изображений в Java 2D API, к которым относятся сглаживание, применение фильтров, использование внеэкранных буферов, текстурные и градиентные заливки, работа с альфа-каналами.

## Постановка задачи:

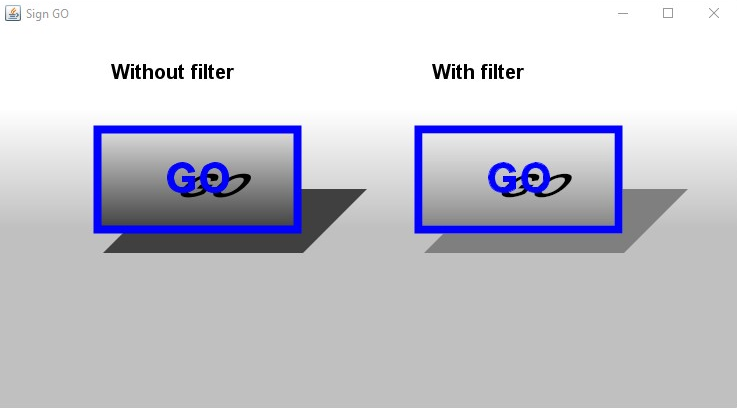
* для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape;
* создать указанный фильтр изображения; при тестировании вывести фигуру без фильтра и с фильтром (аналогично фильтрам из примеров);
* моделировать освещение и тень от объекта при помощи альфа-канала и/или механизма обработки изображения;
* при рисовании использовать сглаживание, внеэкранный буфер и преобразования координат.

Фигура (дорожный знак): надпись GO в прямоугольнике, цвет прямоугольника и надписи – синий, цвет фона – серый с градиентной заливкой снизу-вверх. Фильтр: Brighten

## Особенности реализации:

Градиент реализуется при помощи GradientPaint(). Cглаживание - setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON). Тень добавляется при помощи методов getShearInstance()и scale() из AffineTransform.

## Результат работы программы:



*Рисунок 2.1*

**Выводы:**

* Работа с внеэкранными буферами незначительно отличается от работы напрямую. Необходимо только воспользоваться методом BufferedImage.getGraphics(), после чего работать с привычным классом Graphics.
* Для установки сглаживания необходимо установить ключи ANTIALIASING и TEXT\_ANTIALIASING в объекте класса RenderHints.
* Множество фильтров обработки изображений уже реализованы в Java и они достаточно просты в использовании.

# Лабораторная работа № 3

## Тема:

**«**Java 2D API (продолжение)**»**

## Цель:

Научиться создавать свои произвольные фигуры и контуры с использованием интерфейсов Shape и Stroke из Java 2D API.

## Постановка задачи:

* Разработать пользовательский класс Shape, реализующий рисование указанной алгебраической линии.
* Разработать пользовательский класс Stroke для отображения указанного контура, используя в качестве исходных точек результаты класса Shape, созданного на шаге 1
* Создать приложение для тестирования и демонстрации разработанных классов.

Линия: Каппа

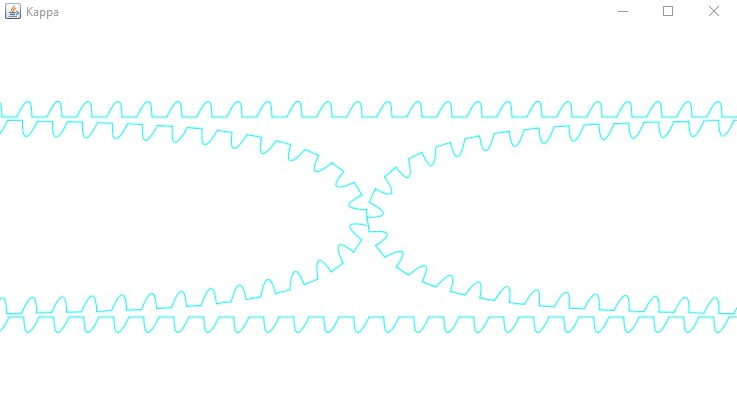
Контур: 

## Особенности реализации:

Каждая точка линии высчитывается по параметрической формуле. Для представления линии используется класс MyShape, реализующий интерфейс Shape. Метод getPathIterator() возвращает объект внутреннего класса ListIterator, который содержит методы, необходимые для задания сегментов для отрисовки.

Контур линии задается классом MyStroke, реализующим интерфейс Stroke. Рисунок выполняется во фрейме методом draw() объекта класса Graphics2D.

Результаты работы программы:



*Рисунок 3.1*

**Выводы:**

* Основной задачей при создании своей фигуры является созданием для нее класса, реализующего интерфейс PathIterator, т. к. он отвечает за вид фигуры;
* При создании своей фигуры можно рисовать не только прямыми линиями, но также и кривыми Безье (SEG\_CUBICTO, SEG\_QUADTO);
* Процесс создания своей кисти не сильно отличается от создания PathIterator’а для фигуры, за исключением того, что нужно ориентироваться на существующую фигуру.

# Лабораторная работа №4

## Тема:

**«**Печать с помощью API Java**»**

## Цель:

Научиться пользоваться системой печати с использованием интерфейсов Printable и Pageable из Java API.

## Постановка задачи:

Для выполнения задания используется ваш вариант решения задания №3.

Модифицируйте вашу программу следующим образом. В демонстрационное приложение добавьте возможность печати небольшого отчёта о решении задания №3. Отчёт должен содержать следующее:

- рисунок с подписью (!) 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 страницы и двустороннюю печать. Рисунок должен занимать не более половины страницы, при печати выровнять его по горизонтали

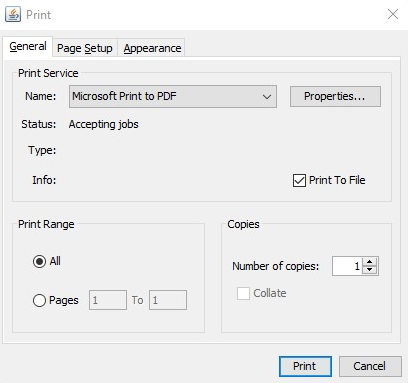
Линия: Каппа

Контур: 

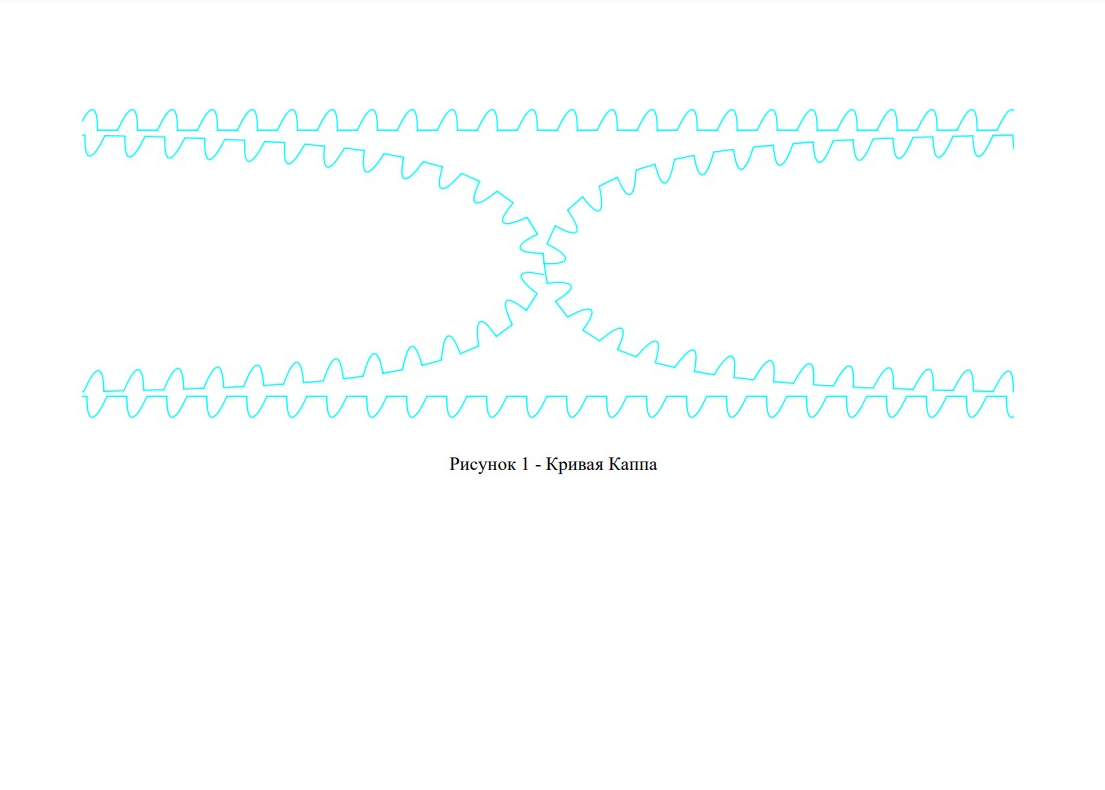
## Особенности реализации:

Для печати текста и графика использую класс HardcopyWriter, расширяемый класс Writer. Для печати кадра и подгонки его по размерам использую BufferedImage и метод getScaledInstance(newWidth, newHeight, Image. SCALE\_SMOOTH);

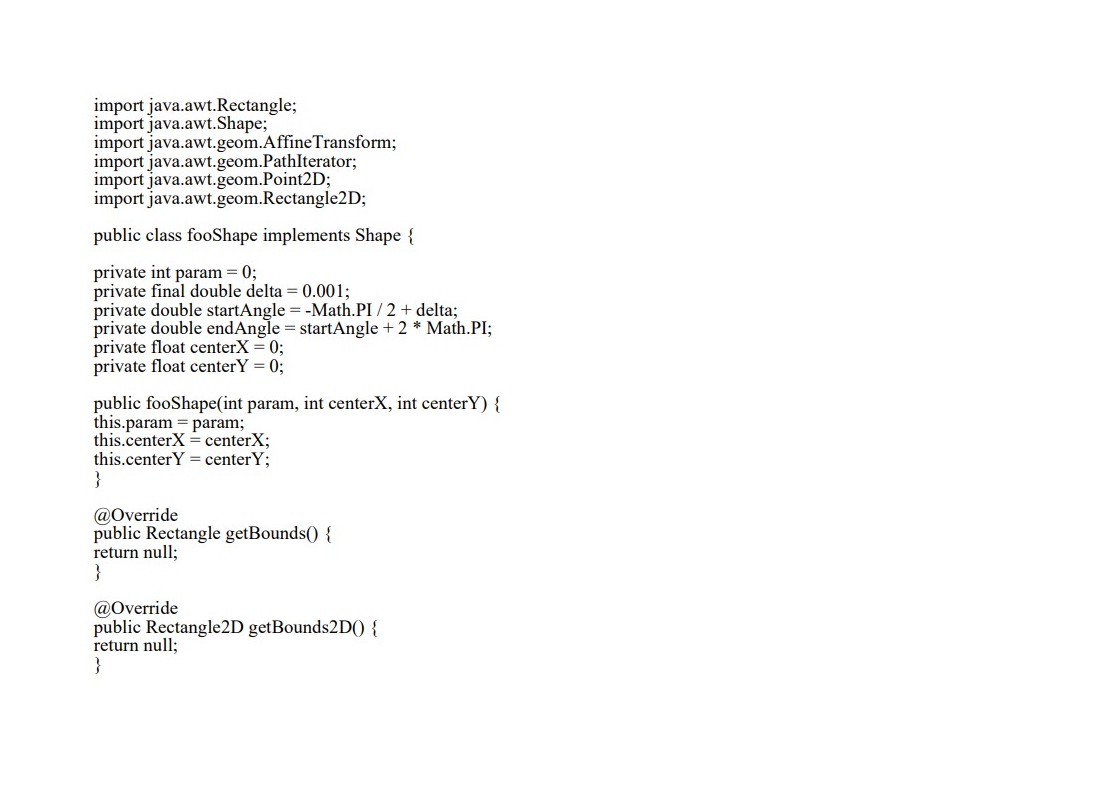
## Результаты работы программы:



*Рисунок 4.1*



*Рисунок 4.2*

**

*Рисунок 4.3*

**Выводы:**

* Работа с принтерами обеспечивается классом *PrinterJob*, в котором уже есть готовый графический интерфейс для настройки печати.
* Для того, чтобы узнать, сколько текст будет занимать места на странице можно использовать класс *FontMetrics*.
* Если текст не помещается на одну страницу, то разбивку на страницы необходимо делать самостоятельно.

# Лабораторная работа № 5

## Тема:

**«**Графика и пользовательский интерфейс Java**»**

## Цель:

Научиться использовать компоненты Swing javax.swing.Tree.TreeModel и javax.swing.table.TableModel.

## Постановка задачи:

1) Разработайте 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 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)

3) Организуйте создание/загрузку/сохранение данных вашего варианта задания в файл (Вариант тестовых данных сдаётся вместе с исходным кодом задачи).

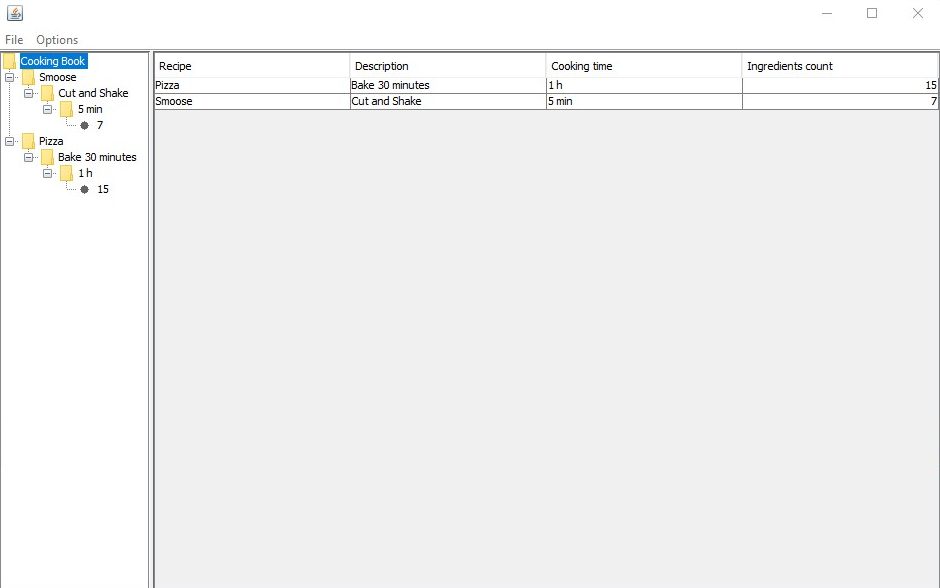
Вариант: Кулинарный справочник.

## Особенности реализации:

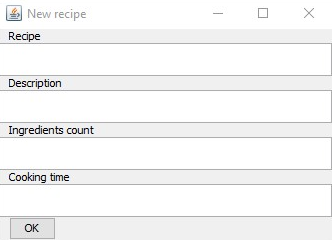
Класс *JTable* отображает табличные данные. Он особенно легко применяется, когда данные организованы в виде массива массивов. Если же это не так, то необходимо создать модель для таблицы в виде класса, реализующего интерфейс *javax.swing.table.TableModel*. Эта модель будет служить переводчиком между данными и компонентом *JTable*.

Компонент *JTree* применяется для отображения данных, имеющих структуру дерева. Если данные имеют форму вложенных массивов, векторов или хеш-таблиц, можно передать корневой узел структуры данных конструктору *JTrее*, и он их отобразит. Данные, имеющие древовидную структуру, обычно имеют иную форму, чем перечисленные выше. Отобразить такие данные можно, реализовав интерфейс *javax.swing.Tree.TreeModel*, чтобы проинтерпретировать данные способом, пригодным для использования компонентом *JTree.*

## Результаты работы программы:



*Рисунок 5.1*

**

*Рисунок 5.2*

**Выводы:**

* Для того чтобы JTree или JTable отрисовали изменения в таблице, необходимо в модели сохранять слушателей и во время изменения данных их оповещать. При установке модели, компоненты сами подписываются в ней.
* Сортировку в таблице можно не реализовывать самому, а использовать автоматически сгенерированную, указав setAutoCreateRowSorter(true).
* Дерево и таблица могут использовать один и тот же объект как модель, если класс этой модели реализует оба интерфейса TableModel и TreeModel.

# Лабораторная работа № 6

## Тема:

**«**Графика и пользовательский интерфейс Java»

## Цель:

Научиться обмену объектами в компонентах Swing средствами пакетов java.awt.datatransfer и java.awt.dnd в виде drag-and-drop.

## Постановка задачи:

Для выполнения задания используется ваш вариант решения задания №3. 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).

Линия: Каппа

Контур: 

## Особенности реализации:

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  редставляет тип данных, подлежащих передаче. Каждый формат данных (data flavor) содержит удобочитаемое имя, объект Class, указывающий тип передаваемых данных, и тип MIME, определяющий  одировку, используемую при передаче данных.

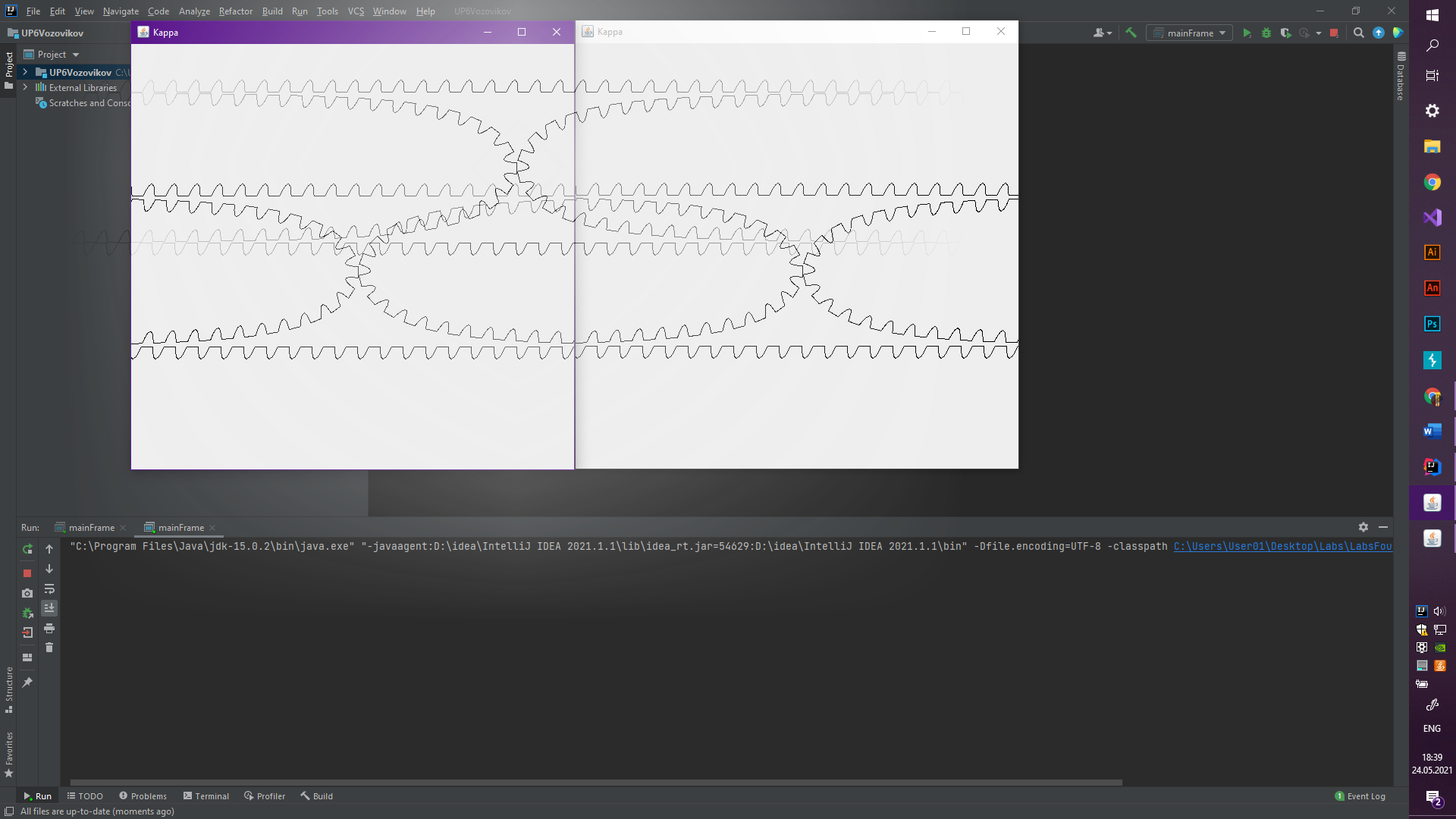
В классе DataFlavor предопределена пара наиболее часто используемых форматов для передачи строк и списков объектов File. Кроме этого, в нем определено несколько типов MIME, используемых этими форматами. Интерфейс java.awt.datatransfer.Transferable является важной частью механизма передачи данных. Этот интерфейс задает три метода, которые должны быть реализованы каждым объектом, желающим сделать свои данные доступными для передачи:

* getTransferDataFlavor() – возвращает массив всех типов DataFlavor, которые он может использовать для передачи своих данных;
* isDataFlavorSupported() – проверяет, поддерживает ли объект Transferable данный формат;
* getTransferData() – возвращает данные в формате, соответствующем запрошенному DataFlavor.

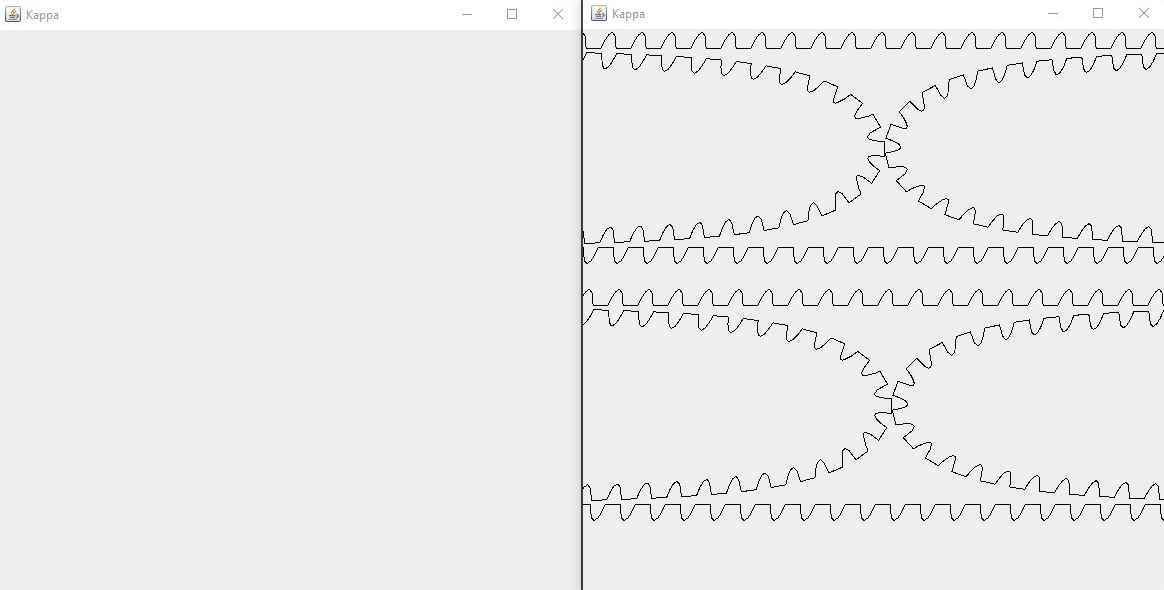
Архитектура передачи данных основывается на механизме сериализации объектов как на одном из средств передачи данных между приложениями.

Начиная с Java 1.2, была введена поддержка обмена данными методом «перетаскивание» (drag-and-drop). Программный интерфейс для этого механизма находится в пакете java.awt.dnd и основан на той же архитектуре DataFlavor и Transferable, что и механизм вырезания и вставки

## Результаты работы программы:



*Рисунок 6.1*



*Рисунок 6.2*

**Выводы:**

* Для того, чтобы наш созданный класс можно было использовать в передаче, необходимо, чтобы он реализовывал интерфейс Transferable. Его реализация значительно упрощается, если мы можем использовать механизм сериализации через реализацию интерфейса Serializable;
* Основная задача в реализации механизма drag-and-drop состоит в реализации интерфейсов DragGestureListener, DragSourceListener и DropTargetListener;
* Drag-and-drop может работать как в пределах одного приложения, так и между несколькими.

# Лабораторная работа №7

## Тема:

«Работа с базами данных средствами JDBC»

## Цель:

Научиться работать с базами данных посредством технологии JDBC.

## 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).

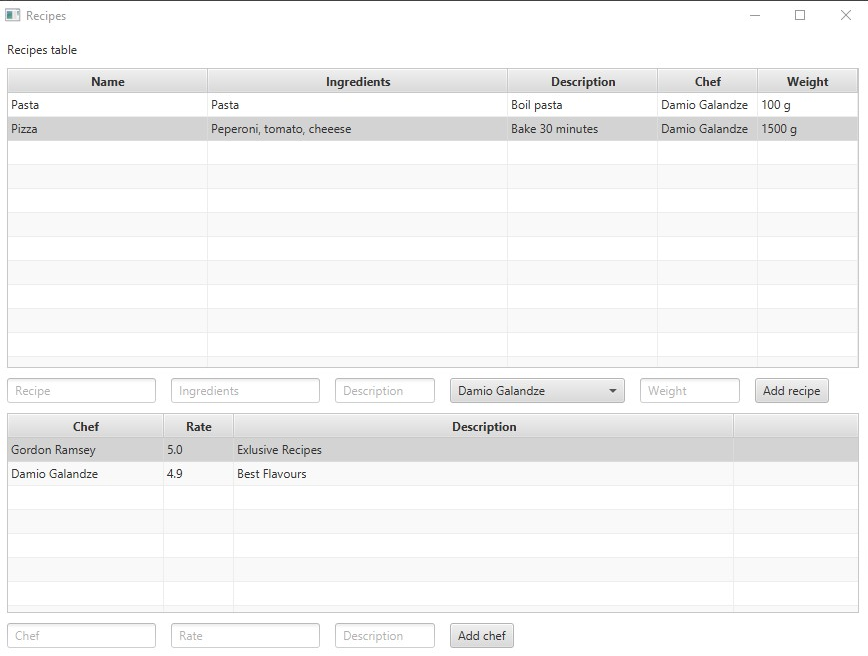
Вариант: Кулинарный справочник.

## Особенности реализации:

В данном задании у каждого лекарства имеются параметры Recipe, Ingredients, Description, Weight и Chef. Chef в свою очередь имеет поля Chef, Rate и Description

Для обработки данных в базе данных используются SQL запросы.

Проведение транзакций происходит с помощью объектов классов Statement и PreparedStatement. Результат выборки из базы помещается в объект класса ResultSet, который хранит ответ от базы в удобном виде.

Результат работы программы:

*Рисунок 7.1*

**Выводы:**

Java Database Connectivity является удобным методом работы с информационными системами, которые работают с большими объёмами сложно структурированной информации.

* Для добавления/извлечения данных из базы используется язык запросов SQL;
* Проведение транзакций происходит с помощью объектов классов Statement и PreparedStatement;
* Результат выборки из базы помещается в объект класса ResultSet, который хранит ответ от базы в удобном виде.

# Лабораторная работа №8

## Тема:

«Изучение среды разработки NetBeans с интерфейсом JavaBeans»

## Цель:

Научиться создавать собственные java-bean компоненты, а также устанавливать их.

## Постановка задачи:

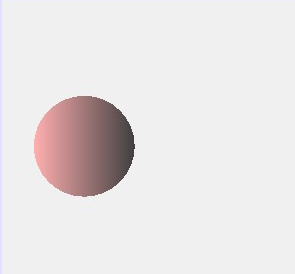
* Изучить материал примера по быстрому введению в среду разработки NetBeans
* Разработать простой компонент варианта задания на базе класса Canvas
* Создать приложение в среде NetBeans с использованием компонента

Эллиптическая градиентная (слева направо) заливка. Свойства: ширина, высота и цвет слева (начало градиента), цвет справа (завершение градиента).

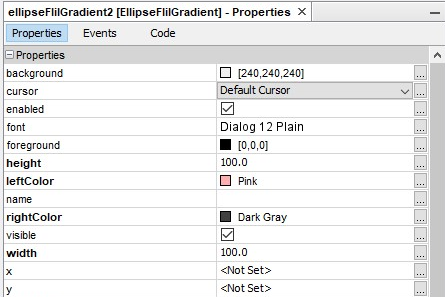
## Особенности реализации:

Из-за того, что Java разрешает загружать классы динамически, контейнерные программы могут загружать неизвестные им классы. Контейнерные средства определяют поддерживаемые компонентом свойства, события и методы при помощи механизма интроспекции (introspection). Интроспекция основана на механизме отражения (reflection) java.lang.reflect, предназначенном для получения информации о членах класса.

Результат работы программы:

**

*Рисунок 8.1*

**

*Рисунок 8.2*

**Выводы:**

Все Swing- и Awt-компоненты могут функционировать как компоненты JavaBeans.

Для того, чтобы класс соответствовал спецификации *JavaBeans*, он должен реализовывать интерфейс *Serializable*, иметь публичный конструктор по умолчанию и для свойств иметь соответствующие методы *get* и *set*.

Реализация компонента как *JavaBeans*-компонента позволяет легко использовать в визуальных редакторах интерфейсов, например, в *NetBeans*.

# Лабораторная работа №9

## Тема:

«Компоненты JavaBeans»

## Цель:

Научиться разработке своих собственных компонентов JavaBeans с собственными событиями и свойствами.

## Постановка задачи:

* Разработайте компонент вашего варианта задания. Создайте файл ма-

нифеста и упакуйте компонент вместе с исходным кодом разработанных классов. При разработке поместите все ваши классы в пакет: bsu.fpmi.edupract

* Компонент должен реализовать класс BeanInfo с информацией о ком-

поненте.

* Создайте тестовое приложение в NetBeans с использованием вашего

компонента. Интерфейсные компоненты с реализацией собственного события AcсeptEvent.

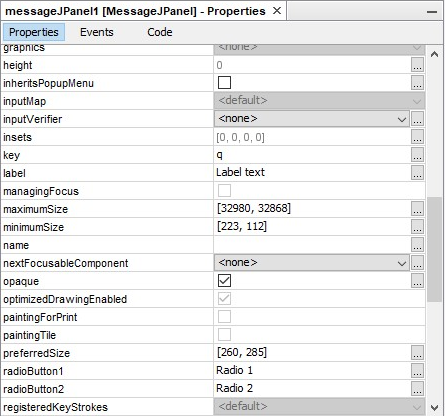
Определить также интерфейс слушателя вашего события AcсeptListener: Передавать слушателю события информацию о том, в результате чего произошло событие, если событие может генерироваться от нескольких действий пользователя.

Вариант: Однострочный статический текст, две независимых радио-кнопки и обычная кнопка. Свойства: текст, текст кнопки, текст радио-кнопок, символ подтверждения. Событие генерируется при нажатии на обычную кнопку или вводе символа. Событие передаёт ещё и состояние радио-кнопок.

## Особенности реализации:

В качестве компонента JavaBeans выступает класс AcceptPanel. В нем присутствуют свойства set/getButton для установки текста кнопки, set/geLabelдля установки текста надписи,set/getFlag для установки текста флага и set/getKey для установки символа подтвержения. Класс унаследован от класса Panel. Компонент включает в себя событие AcceptEvent. Событие возбуждается, когдапроисходит нажатие на кнопку. Событие распознается средой NetBeans. Для того, чтобы было описание создано разработанных свойств и событий, был создан класс AcceptPanelBeanInfo. Большая его часть была автоматически сгенерирована средой NetBeans при непосредственном его создании.

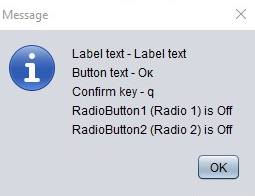
## Результат работы программы:

**

*Рисунок 9.1*



*Рисунок 9.2*



*Рисунок 9.3*

##### **Выводы:**

Для того чтобы созданное событие было распознано в IDE, разработанный слушатель должен наследоваться от класса *EventListener*, а само событие должно наследоваться от *EventObject*.

Для работы с событиями должны быть реализованы *add* и *remove* методы в компоненте. Описания событий и свойств можно определить в классе, реализующем интерфейс

*BeanInfo*.

# Лабораторная работа №10

## Тема:

«Компоненты JavaBeans»

## Цель:

Научиться разрабатывать собственные редакторы свойств и настройщики компонентов JavaBeans.

## 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:

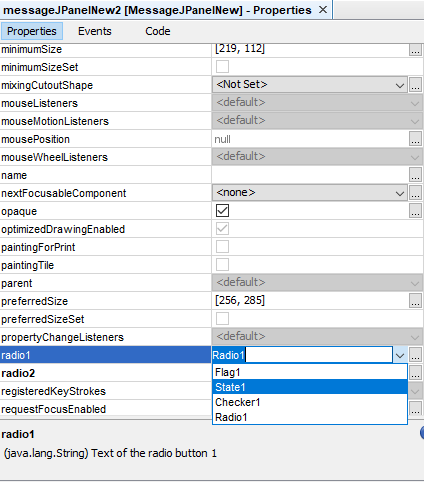
Попытайтесь создать настройщик компонента, который позволит изменять списки допустимых значений для свойств вашего компонента.

## Особенности реализации:

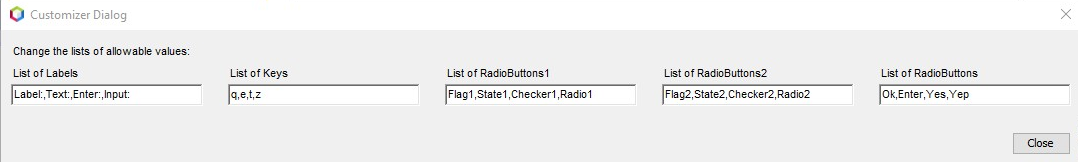
В качестве редакторов для компонентов выступают классы FlagTextEditor, KeyConfirmEditor, LabelTextEditor. В каждом таком классе определены методы getTags(), который возвращает массив тегов; setAsText() для конвертирования имен в конкретные значения; getJavaInitializationString() - для корректной инициализации тегов. Инициализируем редакторы в классе MessageJPanelBeanInfo с помощью метода setPropertyEditorClass().

В качестве настройщика компоненты выступает класс MessageJPanel-Customizer, в котором имплементирован TextListener. При изменении полей в Сustomizer слушатель считывает поля и изменяет теги на те, что ввел пользователь.

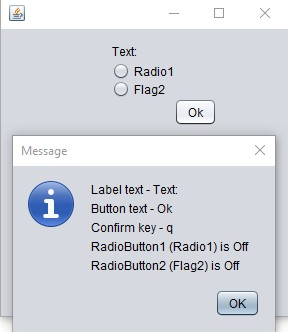
## Результат работы программы:



*Рисунок 10.1*



*Рисунок 10.2*



*Рисунок 10.3*

**Выводы:**

Настройку свойств собственных компонентов можно контролировать, если для каждого свойства создать свой класс, реализующий *PropertyEditor*. Он будет отвечать за редактор настроек привязанного к нему свойства. Для настройки своих свойств можно создать свой класс-наследник *Component*, в котором сделать удобный для редактирования GUI.

# Лабораторная работа №11

## **Тема:**

«Сервлеты и JSP»

## Цель:

Научиться создавать сервлеты, изучить технологию JSP. Разобраться в принципах работы с ApacheTomcat.

## 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.

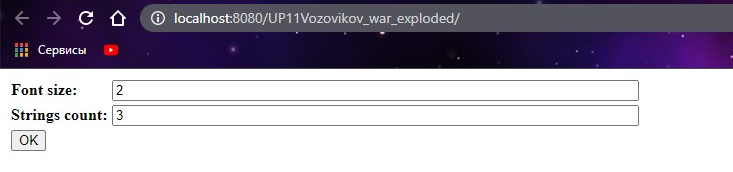
Вариант: Вывод фрагментов текстов шрифтами различного размера. Размер шрифта и количество строк задается на стороне клиента.

## Особенности реализации:

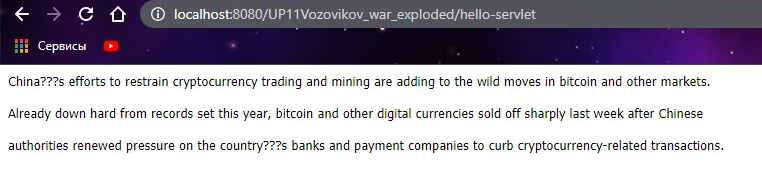
Сервлеты (servlets) выполняются под управлением Web-сервера. Для слежения за работой сервлетов и управления ими создается специальный программный модуль, называемый контейнером сервлетов. Он загружает сервлеты, инициализирует их, передаёт им запросы клиентов, принимает ответы. Чтобы сервлет мог работать, он должен быть зарегистрирован в контейнере, установлен (deploy) в него. Установка (deployment) сервлета в контейнер включает получение уникального имени и определение начальных параметров сервлета, запись их в конфигурационные файлы, создание каталогов для хранения всех файлов сервлета и другие операции. Контейнеры сервлетов создаются как часть Web-сервера или как встраиваемый в

него модуль. Большую популярность получили встраиваемые контейнеры Tomcat, разработанные сообществом Apache Software Foundition

## Результат работы программы:



*Рисунок 11.1*



*Рисунок 11.2*

**Выводы:**

Как правило, сервлет не выполняется один. Он работает в составе Web-приложения.

Web-приложения составляют все ресурсы, написанные для обслуживания запросов клиента: сервлеты, JSP, HTML-страницы, XML-документы, другие документы, изображения и чертежи, музыкальные и видеофайлы. Спецификация Java Servlet Specification описывает структуру каталогов, содержащих все эти ресурсы.

В качестве веб-сервера для сервлетов можно использовать бесплатный Apache Tomcat.

При помощи Java можно создавать сервлеты, при помощи которых можно создавать динамические веб-сайты.

При разработке веб-приложений можно использовать как сервлеты, так и технологию JavaServer Pages.

# Лабораторная работа №12

## Тема:

«Вызов удаленных методов (RMI)»

## Цель:

Научиться использовать концепцию MUD. Технология RMI.

## Постановка задачи:

* Изучите пример 2
* 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 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.

* Создайте на основе технологии RMI клиент/серверное приложение:

Удалённая консоль Windows. Организация удалённого запуска и работы с cmd.exe

## Особенности реализации:

Конкретно данный вариант задания не целесообразно реализовать как часть Mud системы, так как командная строка не предоставляет возможность одновременно работать сразу многим пользователям, т.е. терминал не может быть многопользовательским. Каждый из пользователей не может выполнять множество различных вычислительных процессов, которые будут использовать ресурсы именно одного компьютера.

В данной модели сервер определяет объекты, которые могут использоваться удаленными клиентами. Клиенты вызывают методы удаленных объектов так же, как если бы они были локальными объектами, выполняющимися внутри той же виртуальной машины, что и клиент. Технология RMI скрывает лежащий в ее основе механизм транспортировки параметров методов и возвращаемых значений через сеть.

Для того, чтобы создать приложение на базе RMI, нужно:

* Создать интерфейс, расширяющий java.rmi.Remote. В этом интерфейсе

определены экспортируемые методы, реализуемые удалёнными объектами (то есть методы, реализуемые сервером и вызываемые удаленным клиентом). Каждый метод этого интерфейса должен быть объявлен как генерирующий исключение java.rmi.RemoteException.

* Определить класс, производный от java.rmi.server.UnicastRemoteObject

(или от потомка), реализующий удаленный интерфейс. Этот класс представляет удаленный, или серверный, объект.

* Написать программу (сервер), создающую экземпляр удаленного объ-

екта. Экспортировать объект, сделав его доступным для использования клиентами путем регистрации имени объекта в службе реестра.

* При использовании RMI клиент и сервер не взаимодействуют непо-

средственно. 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.

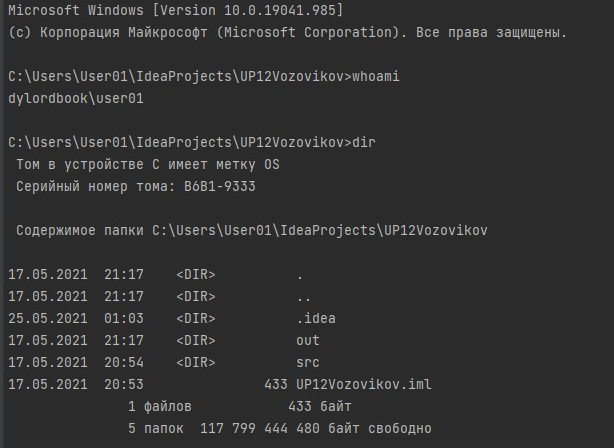
* Написать клиентскую программу, использующую экспортированный

сервером удаленный объект.

## Результат работы программы:



*Рисунок 12.1*



*Рисунок 12.2*

**Выводы:**

* RMI позволяет разрабатывать сетевые Java приложения, не используя низкоуровневые сетевые интерфейсы;
* Для регистрации классов RMI на сервере должен быть запущен *rmiregistry*:
* В случае проблем с сетью, удаленным вызовом и т.п. методы выбрасывают исключение

*RemoteException*;

* Классы для использования в RMI должны наследоваться от *UnicastRemoteObject*.

# Заключение

По итогу работы с приложениями Java на занятиях по учебной практике получен достаточный опыт на уровне, были изучены многие вещи, приобретены полезные навыки, которые несомненно пригодятся в дальнейшей работе.

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, также сервлеты.

Среди всего прочего, достаточно хорошо были усвоены знания по вызову метода удалённого объекта, или по-другому RMI. Стало более понятно клиент-серверное взаимодействие. Не секрет, что клиент-серверные приложения являются распространенными и в то же время самыми сложными в разработке. Проблемы возникают на любом этапе, будь то выбор средств для выполнения запросов или методы кэширования результата

# Список использованной литературы

1. С. Хорстман. Java2 Основы. Том 1. С.-Питербург., 2007
2. Кей С. Хорстман. Java2 Тонкости программирования. Том 2. С.-Питербург., 2007
3. Хорстманн К., Корнелл Г. - "Java. Библиотека профессионала. Том 1 и 2". 9-е издание (2014)
4. Г. Шилдт. Java . Полное руководство – 8-е издание – М.: ООО И.Д. Вильямс, 2012
5. Документация по Apache Tomcat [Электронный ресурс]. – Режим доступа: http://tomcat.apache.org/tomcat-8.0-doc/
6. Derby [Электронный ресурс]. – Режим доступа: https://db.derby.org