Задание 1 (вариант 1)

**Тема:** Java 2D API. Рисование и заливка фигур. Трансформации. Задание стилей линий при помощи класса BasicStroke. Рисование линий. Заливка фигур при помощи классов Раint.

**Постановка задачи:**

Задать движение окружности в окне так, чтобы при касании границы окружность отражалась от нее с эффектом упругого сжатия. Для изображения окружности:

1. создать класс, реализующий интерфейс Shape (движение окружности выполнить с помощью аффинного преобразования);
2. задать толщину и цвет границы фигуры в качестве параметров апплета или фрейма.

**Особенности реализации:**

Окружность – частный случай эллипса, поэтому класс Circle реализуется аналогично классу Ellipse2D.Double с небольшими дополнениями: помимо методов интерфейса Shape добавлены функции Point2D.Double getRightBounds() и Point2D.Double getLeftBounds() для того чтобы следить за приближением окружности к границам фрейма. Также был добавлен класс CircleIterator, наследуемый от PathIterator.

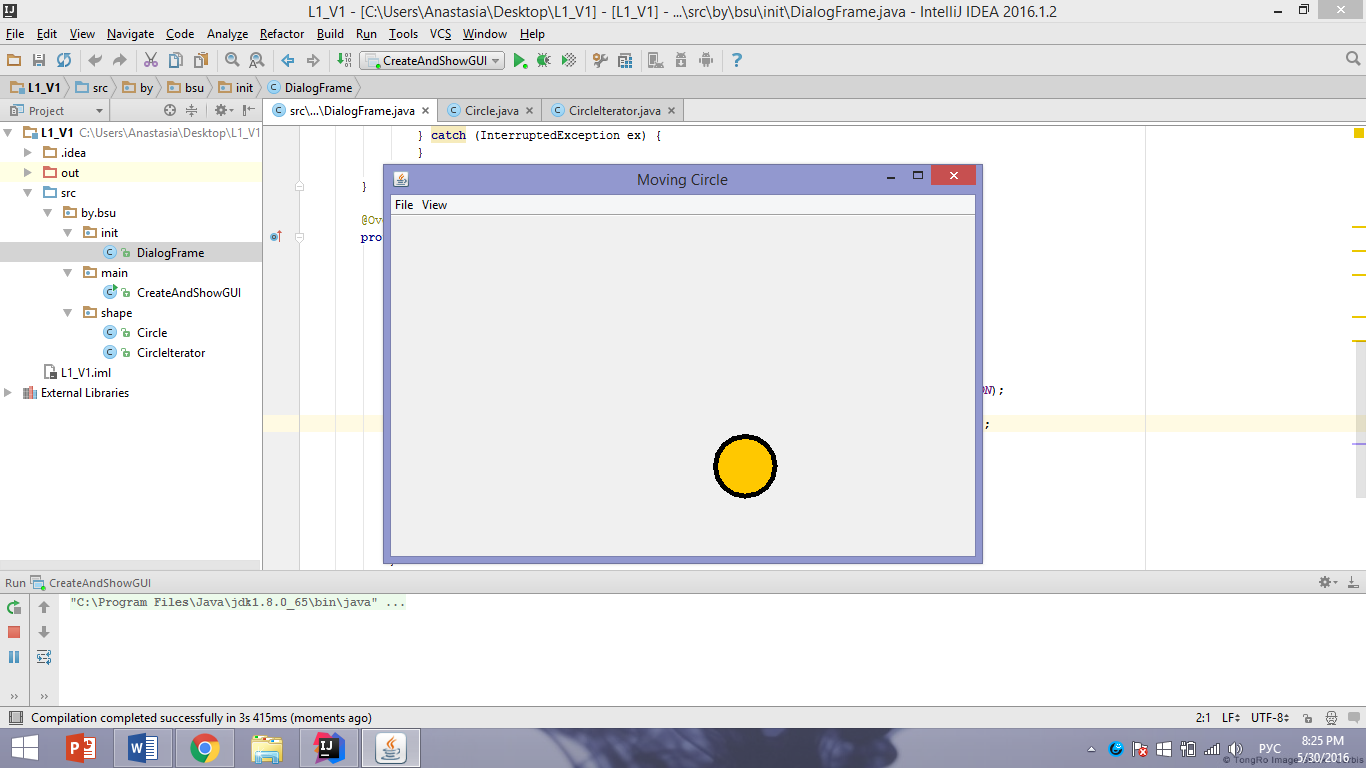
Движение окружности было реализовано с помощью аффинных преобразований:

AffineTransform.getTranslateInstance(double tx, double ty),

из класса AffineTransform, которая перемещает заданный объект покоординатно. С помощью параметров можно регулировать скорость движения окружность.

В качестве дополнения можно менять цвет заливки с помощью пункта меню.

**Результат работы программы:**



Рисунок

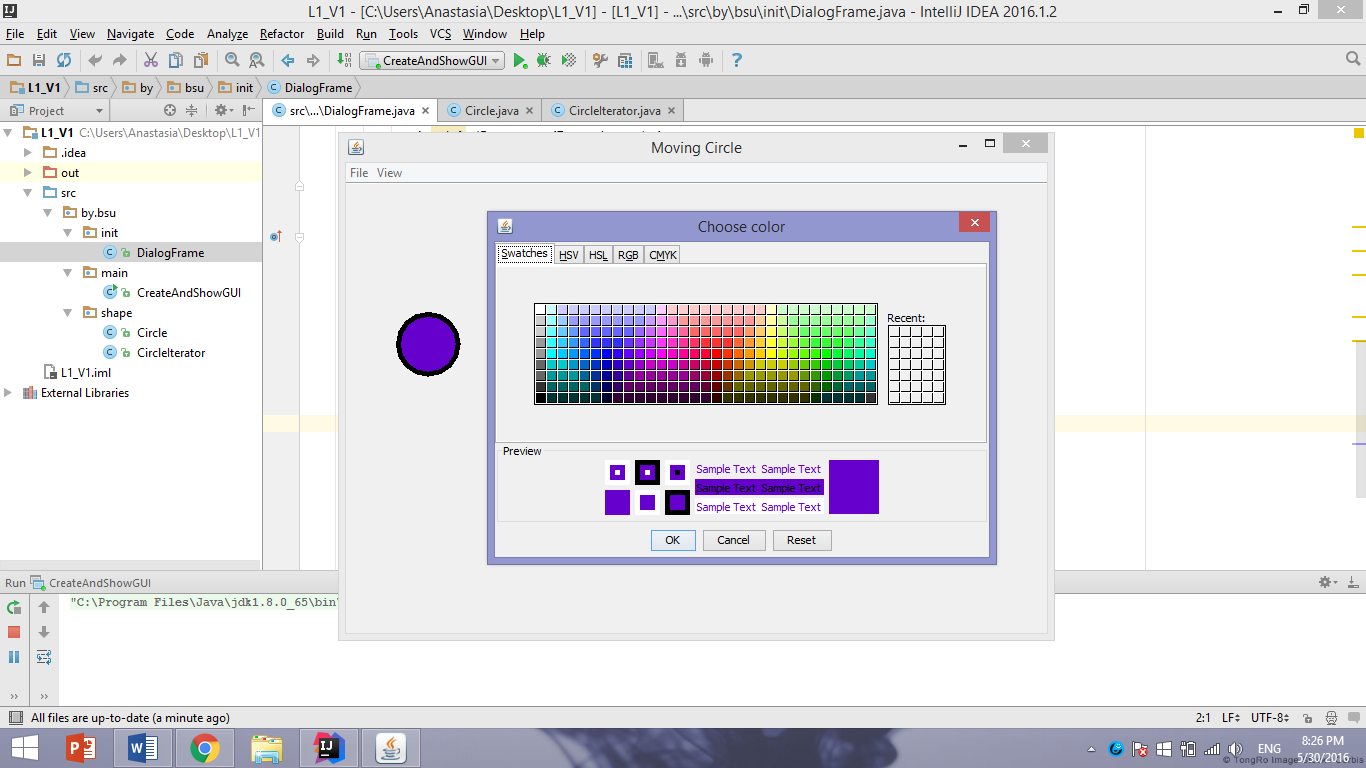


Рисунок 2

Задание 2 (вариант 10)

**Тема:** Java 2D API. Сглаживание. Комбинирование цветов при помощи AlphaComposite. Обработка изображений.

**Постановка задачи:**

Создать тестовое приложение или апплет для демонстрации фигуры, при этом:

1. для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape;
2. создать указанный фильтр изображения; при тестировании выведите фигуру без фильтра и с фильтром (аналогично фильтрам из примеров);
3. смоделировать освещение и тень от объекта при помощи альфа-канала и/или механизма обработки изображения;
4. при рисовании использовать сглаживание, вне экранный 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:**

Прозрачность реализуется с помощью альфа-канала и класса AlphaComposite. Тень – с помощью композиции фигур.

Существует 12 правил композиции. Для прозрачности используется AlphaComposite.**DST\_OVER**(исходные пиксели не оказывают никакого влияния на целевые пиксели и перезаписывают пустые пиксели).

Также важно сглаживать изображение, использую функцию из класса Graghic2D: setRenderingHint(RenderingHints.Key, Object value).

**Результат работы программы:**



Рисунок 3

Задание 3 (вариант 11)

**Тема:** Java 2D API. Пользовательские фигуры. Пользовательские классы Stroke.

**Постановка задачи:**

Создать тестовое приложение или апплет для демонстрации:

1. пользовательского класса Shape, реализующий рисование квадратрисы:
2. пользовательского класс Stroke для отображения контура-синусоиды.

**Особенности реализации:**

Для реализации квадратисы были созданы классы Quadratrix implements Shape и QuadratrixIterator implements PathIterator. В параметрах конструктора задаются начальное и конечное значения координат , а также параметр .

Для реализации контура в виде синусоиды был создан класс SineStroke implements Stroke. График разбивается на несколько частей, каждая из которых рисуется с помощью кривой третьего порядка, за которую отвечает функция curveTo(double x1, double y1, double x2, double y2, double x3, double y3) из класса Path2D.

**Результат работы программы:**

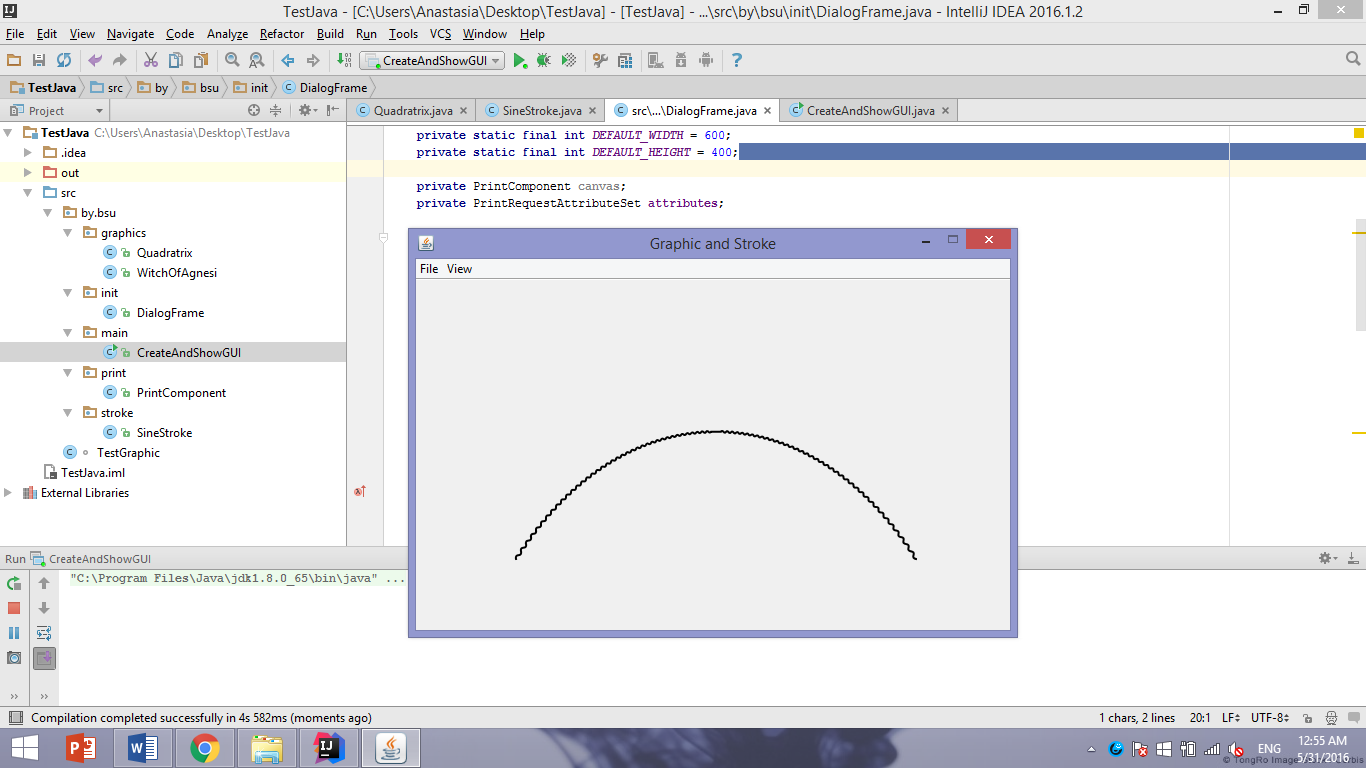


Рисунок 4

Задание 4 (вариант 11)

**Тема:** Java 2D API. Печать с помощью API Java. Печать многостраничных текстовых   
документов. Печать Swing-документов.

**Постановка задачи:**

Печать графика с подписью из лабораторной работы №3 и исходного текста класса Shape, реализующего рисование этого графика.

Печать должна удовлетворять следующим требованиям:

1. режим альбомной ориентации и двусторонняя печать;
2. график должен занимать не более половины страницы;
3. выравнивание графика по горизонтали.

**Особенности реализации:**

Задание 5 (вариант 11)

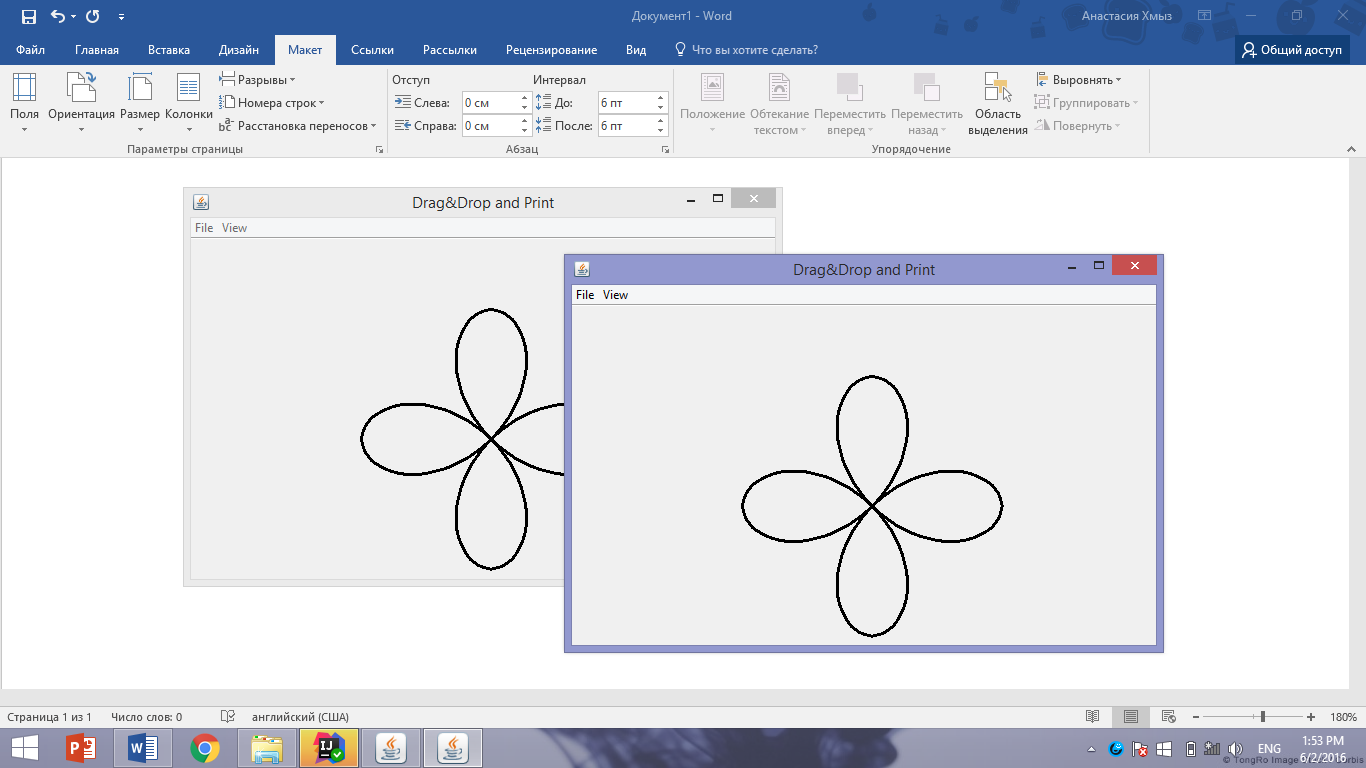
**Тема:** Передача данных в GUI. Архитектура передачи данных. Простое копирование и вставка. Тип данных Transferable. Вырезание и вставка рисунков. Перетаскивание рисунков.

**Постановка задачи:**

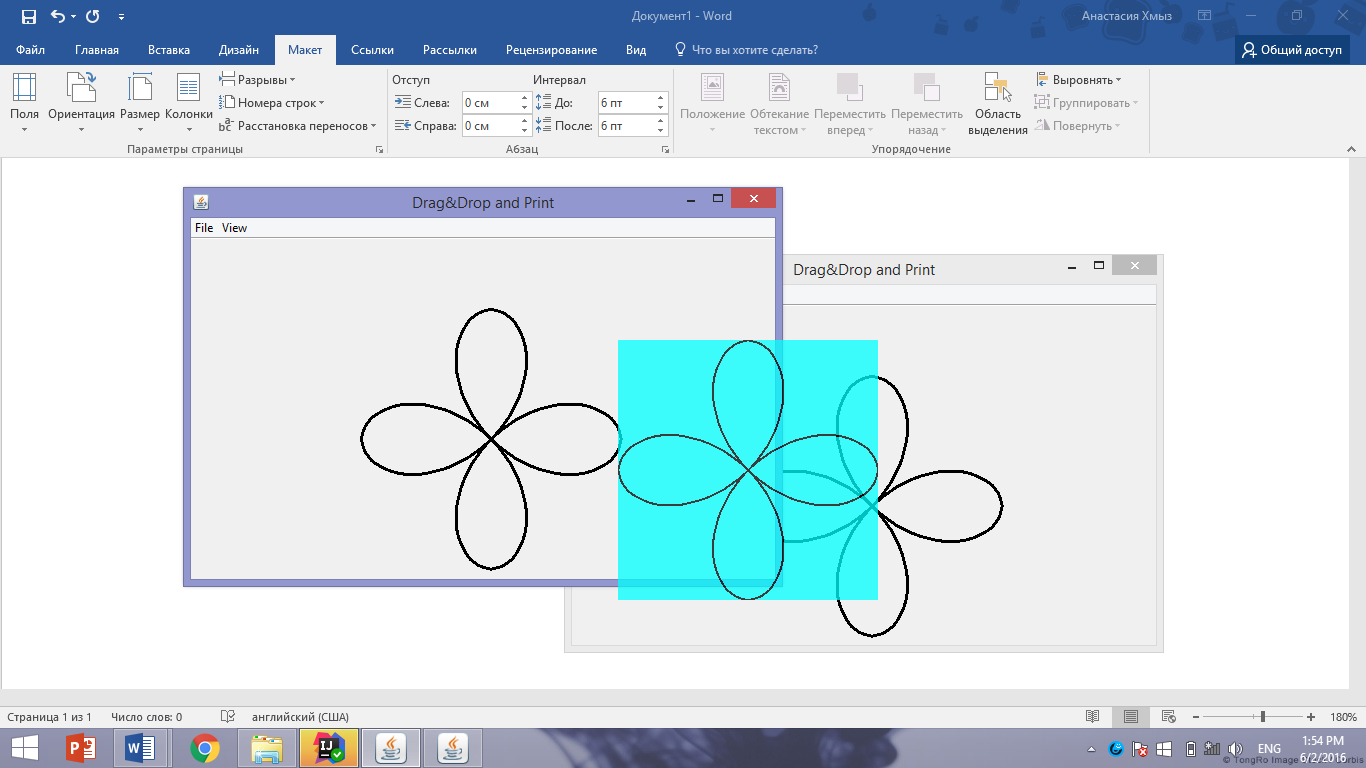
Добавить в класс рисования алгебраической линии из лабораторной №3 возможность «перетаскивание» (drag-and-drop). Реализовать необходимые интерфейсы в классе и в приложении для демонстрации «перетаскивания» алгебраической линии между несколькими копиями тестового приложения.

**Особенности реализации:**

**Результат работы программы:**



Рисунок



Рисунок

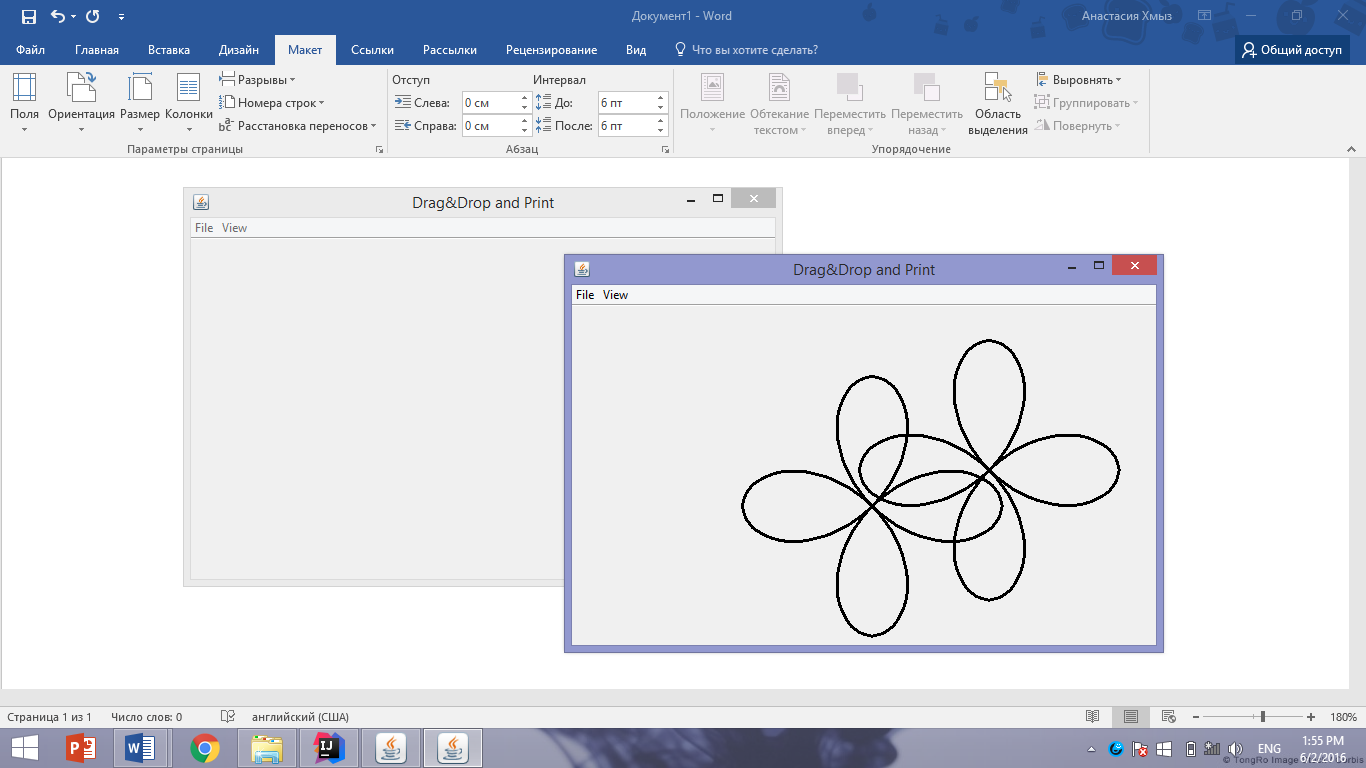


Рисунок 7

Задание 10 (вариант 2)

**Тема:** Компоненты, контейнеры. Управление компоновкой. Обработка событий. Действия и отражение. Создание собственных диалоговых окон.

**Постановка задачи:**

Разработать приложение для выполнения следующих действий с текстовым файлом:

1. прочитать текст из выбранного файла;
2. найти в тексте слова, начинающиеся с заглавной буквы.

В клиентской области создать две панели, разделённые по вертикали, в левой части отобразить содержимое файла, в правой части отобразить найденные слова, для каждого слова использовать отдельный компонент JButton, при нажатии на кнопку изменять заголовок окна приложения на текст кнопки.

Интерфейс должен содержать следующие команды (в формате Подменю/Команда):

1. File/Quit – завершение приложения после подтверждения пользователя
2. View/Font – выбор шрифта для компонентов интерфейса
3. View/ Look and Feel – выбор стиля интерфейса (как в примерах)
4. Help/About – вывод диалогового окна которое печатает условие задачи и информацию о разработчике (ФИО студента, группа, курс, факультет, ВУЗ)
5. (команды) – команды необходимые для решения задачи, например, File/Open для выбора файла и т.п.

**Особенности реализации:**

Один из главных этапов создания GUI на основе Java – размещение компонентов в контейнеры, которые предназначены для гибкости оформления. Реализации фрейма и поставленной задачи должны быть взаимно независимы.

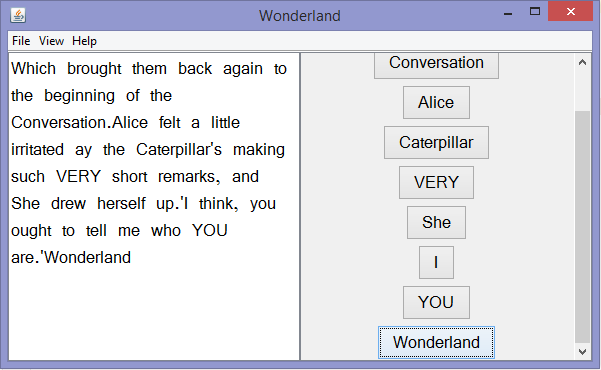
Для изменения Look and feel можно запросить систему об установленных темах с помощью UIManager.*getInstalledLookAndFeels*() для последующего изменения. Шрифт изменяется с помощью setFontFont font). Доступные шрифты можно получить с помощью getAvailableFontFamilyNames().

Для создания собственного окна About был создан класс, наследуемый от JDialog. Окно About должно открываться единожды и, пока данное окно открыто, управление главным окном не может быть возможным.

aboutItem.addActionListener(event -> {  
 **if** (**dialog** == **null**) {  
 **dialog** = **new** AboutDialog(DialogFrame.**this**);  
 **dialog**.setLocationRelativeTo(**null**);  
 }  
 **dialog**.setVisible(**true**);  
});

Чтобы название окна изменялось вместе с нажатием кнопки, необходимо добавление слушателей событий, которые адекватно отвечают на это событие. Для каждой из кнопок слушатель выглядел следующим образом: jButtons[i].addActionListener(event -> setTitle(title)).

**Результат работы программы:**



Рисунок

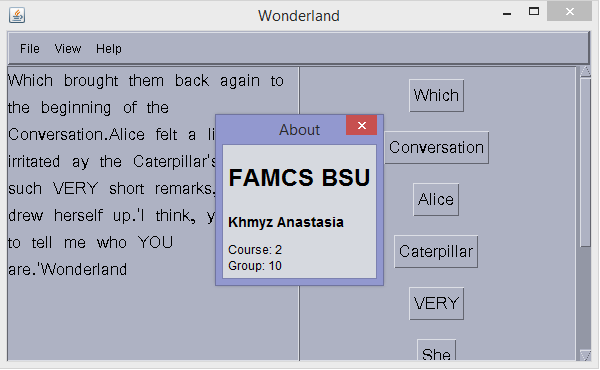


Рисунок 9

Задание 12 (вариант 2)

**Тема:**

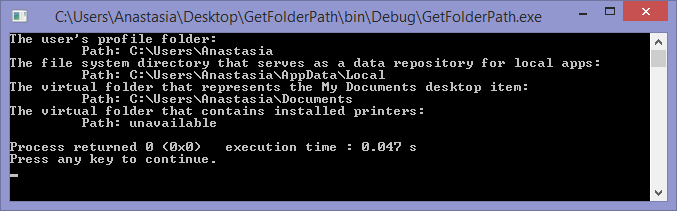
**Постановка задачи:**

Создать тестовое приложение для демонстрации.

Разработать класс Java для получения путей к специальным каталогам Windows (Win32 API: SHGetSpecialFolderPath) с помощью native методов. Создать тестовое приложение для демонстрации.

**Особенности реализации:**

**Результат работы программы:**



Литература

Блинов

Хорстманн