Белорусский государственный университет

Факультет прикладной математики и информатики

Кафедра информационных систем управления

Кротов Денис Александрович

**Отчет**

**по учебной практике**

студента 2 курса 10 группы

**Преподаватель**

Клименкова Наталья Юрьевна,

ассистент кафедры ИСУ

Минск, 2016

**Оглавление**

[Задание №1.4 3](#_Toc452485257)

[Задание №2.6 4](#_Toc452485258)

[Задание №3.6 5](#_Toc452485259)

[Задание №4.6 6](#_Toc452485260)

[Задание №5.6 7](#_Toc452485261)

[Задание №6.4 8](#_Toc452485262)

[Задание №7.4 9](#_Toc452485263)

[Задание №8.8 10](#_Toc452485264)

[Задание №9.8 11](#_Toc452485265)

[Задание №10.10 12](#_Toc452485266)

[Задание №11.7 14](#_Toc452485267)

[Задание №12.11 15](#_Toc452485268)

## Задание №1.4

Постановка задачи:

Общее условие для всех вариантов: - для изображения указанной в задании фигуры создать класс, реализующий интерфейс Shape - выполнить указанные в задании перемещения указанной фигуры с помощью аффинного преобразования координат - выполнить рисунок в окне апплета или фрейма с выбранной толщиной границы фигуры, цветом границы и цветом внутренней области (вводить толщину и цвет в качестве аргументов ваших программ или параметров апплета).

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

Особенности реализации:

Для вращения фигуры использовалось аффинное преобразование координат и метод update(), для обратного вращения использовалась дополнительная переменная shift, которая домножалась на -1(т.е. поворачивала в противоположную сторону) в случае, если круг достиг половины длины отрезка.

Аффинное преобразование - AffineTransform at = AffineTransform. getRotateInstance (Math.toRadians(angle \* (++x)), rotationAxis + RAD / 2 + shift,center.y);

angle \* (++x)-угол поворота умноженный на коэффициент поворота.

rotationAxis + RAD / 2 + shift – изменение координат по x.

center.y - изменение координат по y.

Переменная х изменялась по таймеру.

Результат работы программы:

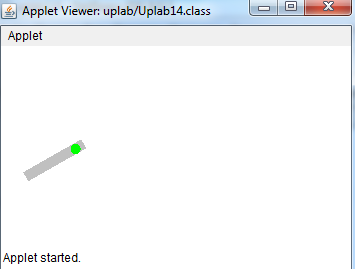
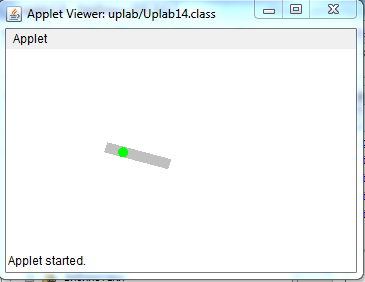


Рисунок 1‑ Скриншот работы приложения Рисунок 2‑ Скриншот работы приложения

через некоторое время

## Задание №2.6

Постановка задачи:

В следующих заданиях создайте тестовое приложение или апплет 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.

6. Фигура (дорожный знак): надпись STOP в прямоугольнике, цвет прямоугольника и надписи – красный, цвет фона – серый с градиентной заливкой сверху-вниз Фильтр: Rotate CСW 45 degrees (поворот на 45 градусов)

Особенности реализации:

Для написания слова STOP использовалось 4 объекта shape, по 1 на каждую букву, для создания тени и градиента использовались готовые функции: Paint shadowPaint,AffineTransform.getShearInstance() и setPaint(new GradientPaint()).

Код фильтра(поворот на 45 градусов):

AffineTransformOp op=new AffineTransformOp(AffineTransform.getRotateInstance(Math.PI/4,0, 190),AffineTransformOp.TYPE\_NEAREST\_NEIGHBOR);

Результат работы программы:



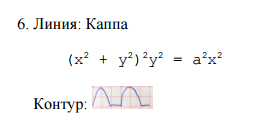
Рисунок 3 ‑ Скриншот работы приложения

## Задание №3.6

Постановка задачи:

* - Разработать пользовательский класс Shape реализующий рисование указанной алгебраической линии.
* - Разработать пользовательский класс Stroke для отображения указанного контура.
* - Создать приложение или апплет для тестирования и демонстрации разработанных классов.

Линия и контур:



Особенности реализации:

Каждая точка линии высчитывается по параметрической формуле. Для того, чтобы изменить стандартный контур, при передаче координат следующей точки линии необходимо соединить её с предыдущей не прямым отрезком, а кривой:

double x1 = prevCoord[0]; double y1 = prevCoord[1];double x2 = coords[0];

double y2 = coords[1];double dx = x2 - x1;double dy = y2 - y1;

newshape.curveTo(x1, y1, x1 + 2 \* dy, y1 - 2 \* dx, x2 - dx / 2, y1 + dy / 2);

newshape.lineTo(x2, y2);Здесь обозначены координаты настоящей и последующей точки

Результат работы программы:

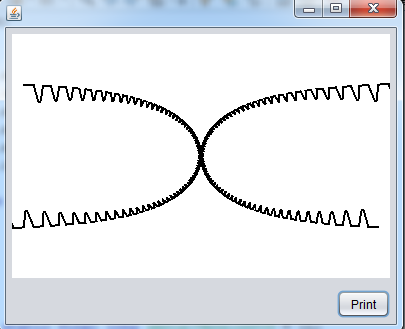


Рисунок 4‑ Скриншот работы приложения

## Задание №4.6

Постановка задачи:

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.

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

Особенности реализации:

Добавляем кнопку, с помощью которой и будем печатать и создаем дополнительный класс наследуемый от Writer, в котором перегружаем метод write(), а также создаем конструктор, содержащий в себе параметры печатаемой страницы, размер фрейма для печати.

Результат работы программы:

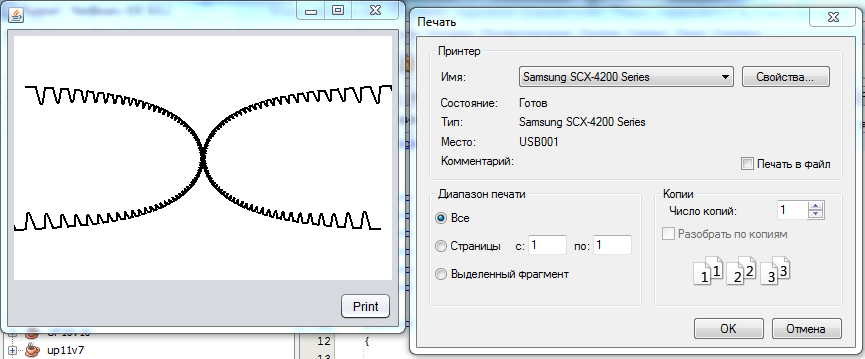


Рисунок 5‑ Скриншот приложения при нажатии кнопки Print

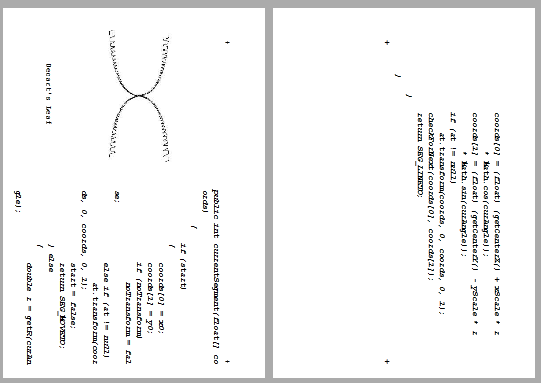


Рисунок 6‑ Скриншот печатаемого документа

## Задание №5.6

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).

Особенности реализации:

Создается дополнительный класс, в котором создаются функции, необходимые для Dnd:

dragGestureRecognized(DragGestureEvent e), dragDropEnd(DragSourceDropEvent e), dragEnter(DropTargetDragEvent e), drop(DropTargetDropEvent e).

dragGestureRecognized() выделяет прямоугольник, в котором содержится наша фигура и перемещает его пока кнопка мышки зажата.

drop() проверяет, в какую точку была перемещена фигура и в случае перемещения на другой фрейм получаем dropComplete(true).

Также имплементировались следующие интерфейсы: Shape, Transferable(возможность перетаскивания), Serializable(возможность записи состояния), Cloneable(возможность открывать несколько окон приложения).

Результат работы программы:

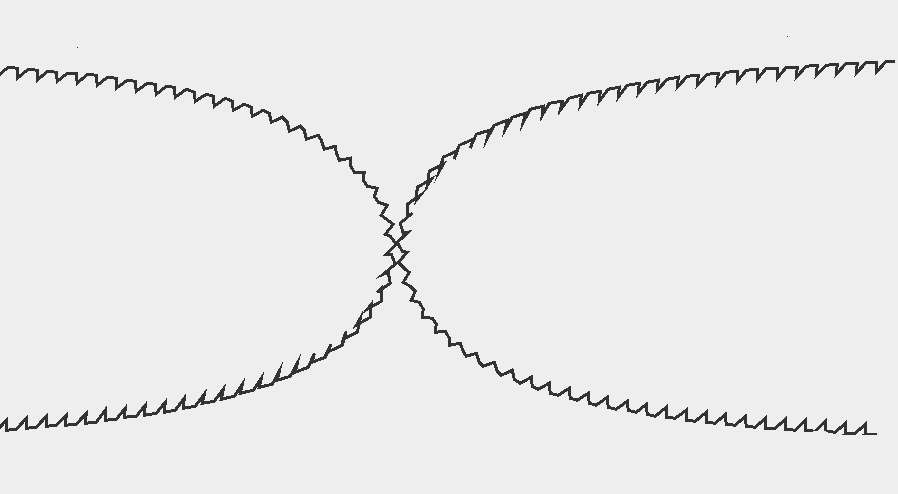


Рисунок 7‑ Скриншот работы приложения

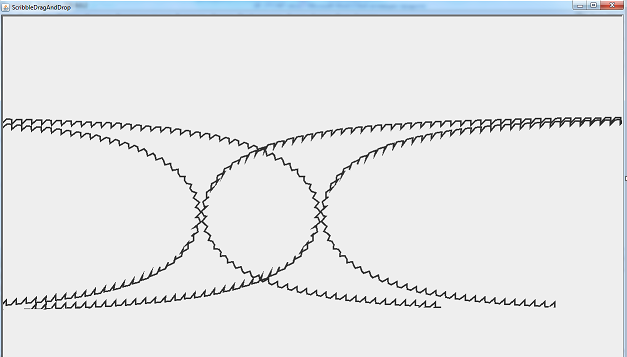


Рисунок 8‑ Скриншот работы 2 приложений,

из одного окна перетянута фигура на другое

## Задание №6.4

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. - При реализации интерфейса следуйте рекомендациям стандарта CUI (Common User Interface).

4) Аудио - магазин (CD, виниловые диски, аудио-плееры, акустика и т.д.)

Особенности реализации:

Используем следующий метод valueChanged(TreeSelectionEvent e), который организован так: метод проверяет, является ли место, куда помещается информация, листом и если нет, то таблица не отображается, а если же является, то информация добавляется в фрейм и происходит перерисовка всех объектов, к которым информация имеет отношение (подкатегория, таблица).

Результат работы программы:

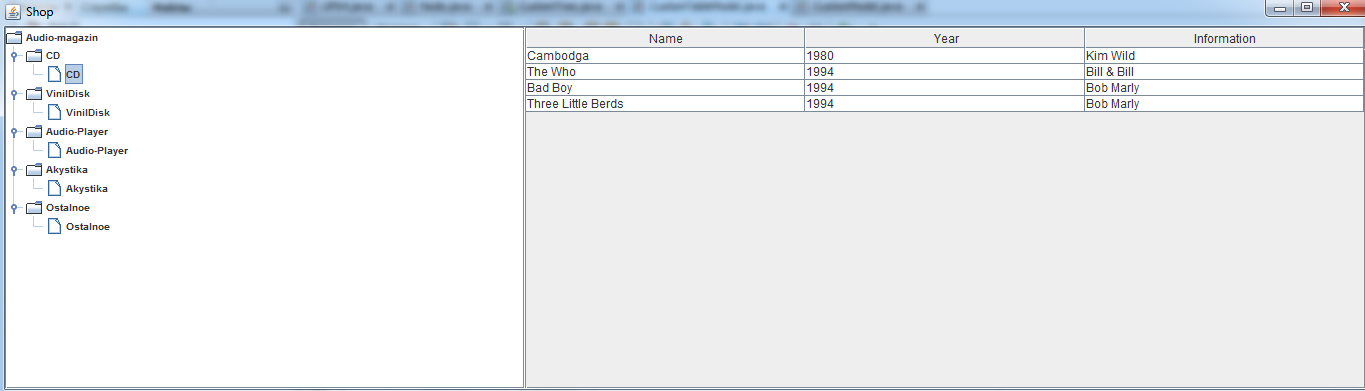


Рисунок 9‑ демонстрация работы приложения

## Задание №7.4

Постановка задачи:

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.2016 Создайте тестовое приложение в NetBeans с использованием вашего компонента.

4. 3D вертикальная линия. Свойства: высота

Особенности реализации:

Создаем класс LineVertical3D, наследуемый от Canvas.Поля класса: private int height;int width,lengh;private Shape rect;. Для свойств прописывались get/set-методы. В set-методах после изменения значения определённого свойства вызывался метод repaint() – что визуально меняло данный компонент. Далее создавался manifest-файл – он необходим для регистрации нашего компонента в среде NetBeans и полноценного использования наряду со стандартными компонентами.

Создаем трехмерную сетку координат, далее преобразуя 2D сетку в 3D рисуем по линиям 3D прямую.

Результат работы программы:

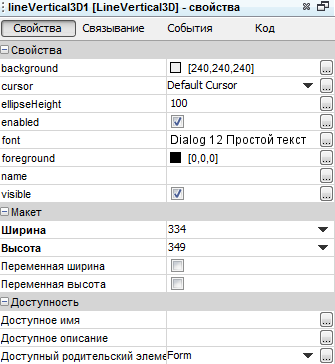
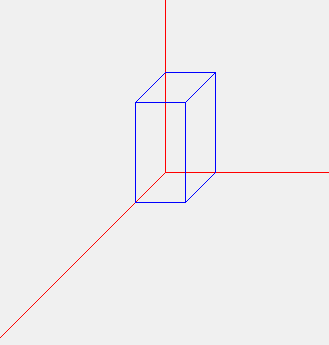


Рисунок 10‑ Форма Рисунок 11‑ свойства формы

## Задание №8.8

Постановка задачи:

1) 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\_practice2016

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

3) Создайте тестовое приложение в NetBeans с использованием вашего компонента.

8. Однострочный статический текст, строка ввода и кнопка. Свойства: текст, текст кнопки, символ подтверждения. Событие генерируется при нажатии на кнопку или вводе символа. Событие передаёт ещё и введенный текст

Особенности реализации:

Создается панель на основе JPanel, а также поле для текста на основе FieldText(поле, куда будет выводиться введенный текст) и кнопка(по нажатию введенный текст отображается в fieldText),вместе с текстом передается полное имя панели .

Результат работы программы:

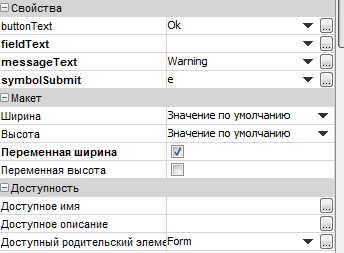


Рисунок 12‑ свойства формы



Рисунок 13‑ форма

## 

Рисунок 14-демонстрация работы приложения

## Задание №9.8

Постановка задачи:

Для решения задания №9 используем решённый вариант задания №8.

Создаём 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.

Особенности реализации:

Добавлена еще 1 панель с другим названием**(**названия на выбор **staticlist = { "Info", "Warning", "Error" })**, аналогично на выбор предоставляем несколько вариантов для других редакторов.

Результат работы программы:

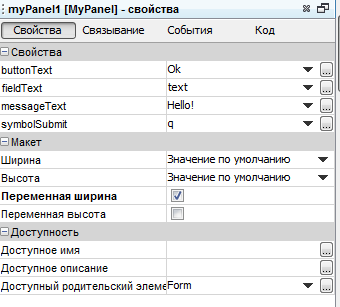
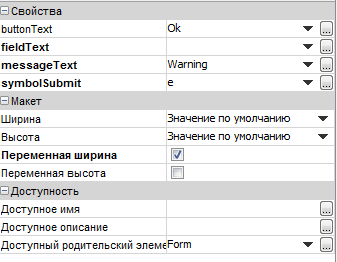
 

Рисунок 15‑ свойства второй панели Рисунок 16‑ свойства второй панели

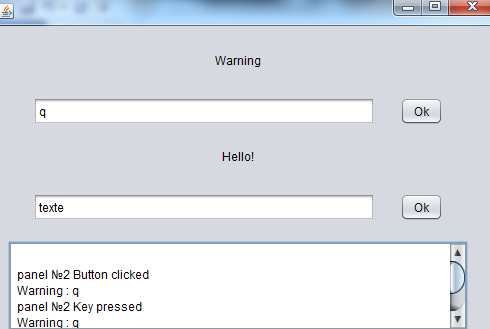


Рисунок 17‑ демонстрация работы приложения

## Задание №10.10

Постановка задачи:

Разработайте приложение для выполнения несложных действий с текстовым файлом указанных в вашем варианте задания. - При реализации интерфейса следуйте рекомендациям стандарта CUI (Common User Interface). Главное меню приложения обязательно должно содержать следующие команды (в формате Подменю/Команда):

File/Quit – завершение приложения после подтверждения пользователя

View/Font – выбор шрифта для компонентов интерфейса

View/ Look and Feel – выбор стиля интерфейса (как в примерах)

Help/About – вывод диалогового окна которое печатает условие задачи и информацию о разработчике (ФИО студента, группа, курс, факультет, ВУЗ)

(команды) – команды необходимые для решения вашего варианта задачи, например, File/Open для выбора файла и т.п. - Приложение должно корректно обрабатывать команды системного меню (изменение размеров окна, команды «Закрыть», «Свернуть» и т.д.)

10. Читает текст из выбранного файла, разбивает его на предложения. В клиентской области создать две панели, разделённые по вертикали, в левой части отобразить содержимое файла, в правой части отобразить все предложения, переводя все слова восклицательных предложений в верхний регистр, для каждого предложения использовать отдельный компонент JLabel

Особенности реализации:

Создаем класс FontChooser, в нем создаем выбор шрифта:

String family; // The name of the font family

int style; // The font style

int size; // The font size

Font selectedFont; // The Font they correspond to

С помощью первых 3 полей определяем шрифт и подставляем его в 4 поле.

Далее создаем форму MyFrame, в которой создаем 2 панели, которые делят окно на 2 части(левую и правую), в левой будет открываться сам файл, а в правой будет уже обработанный файл.

Текст считывался в BufferReader, далее отработка текста:

String[] all=text.split("[\\?\\.\\!]\\s+"); //расщепляем текст на предложения

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

{

text=all[i];//в другую строку добавляем предложение

if(text.matches(".\*\\!$"))// если в тексте есть ! то меняем регистр

text = smena\_rigistr(text);

Результат работы программы:

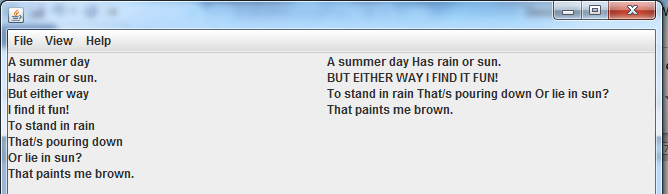


Рисунок 18‑ Стандартная работа

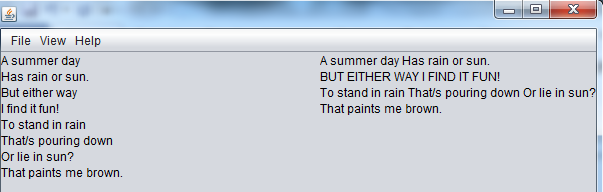


Рисунок 19‑ Работа приложения с интерфейсом Numbus

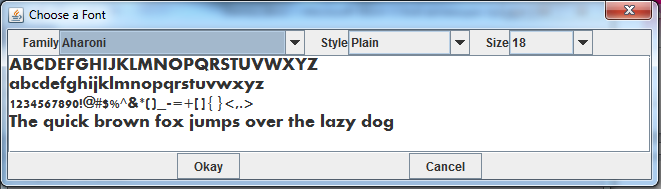


Рисунок 20‑ Выбор шрифта

## Задание №11.7

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 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:

7) Учёт рабочего времени. Сервер ведёт учёт времени работы клиентов, данные сохраняются в файле. Клиент при запуске связывается с сервером и сообщает данные клиента. Сервер каждые 5 мин запрашивает подтверждение у клиента, что он ещё подключен. Если клиент не отвечает, он закончил работу.

Особенности реализации:

Создается класс Server, внутри которого лежит класс Info, содержащий имя юзера и время, когда он подключился. Далее создаем функции addUser(String name) и exitUser(String name) для добавления и выхода юзера. Использование парадигмы MUD возможно, однако она не использовалась, так как, по моему мнению, в этом варианте это не является необходимым, потому что клиенты не взаимодействуют между собой, в то же время MUD(многопользовательская область) могла использоваться в качестве взаимодействия между сервером и юзером.

Результат работы программы:

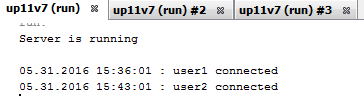
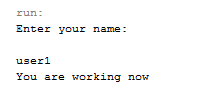


Рисунок 21‑ Работа с 1 юзером(run юзера) Рисунок 22‑ Работа с 2 юзерами(run сервера)

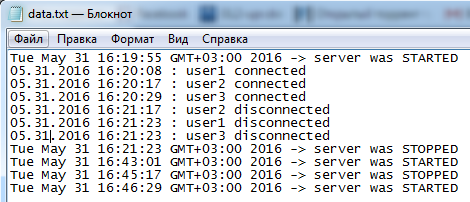


Рисунок 23‑ Запись в файл учёт рабочего времени

## Задание №12.11

Постановка задачи:

- Разработайте класс Java для решения вашего варианта задания с помощью native методов. - Создайте тестовое приложение (одно или несколько) для демонстрации всех возможностей вашей разработки - Используйте MS Visual Studio С/С++ и Win32 API для разработки кода, зависящего от платформы, при необходимости можно использовать встроенный ассемблер.

11. Реализуйте работу с Windows Clipboard (см. функции Win32 API: OpenClipboard, CloseClipboard, EmptyClipboard, SetClipboardData, GetClipboardData и т.п.). Для тестирования создайте два Java-приложения – первое помещает фрагмент текста в Clipboard, второе – извлекает текст и печатает.

Особенности реализации:

Создаем 2 фрейма:1 для извлечения из буфера обмена,1 для добавления в буфер обмена

Создается h файл на основе

Алгоритм выполнения данной задачи следующий:   
Функция прописывается в Cpp – файле, который необходимо скомпилировать. После компиляции создаётся библиотека dll. Далее в java – файле подключается созданная библиотека следующим образом:

static {

try {

System.load(“d:/MyDocuments/up12v11/Den/up12v11/build/classes/up12v11/ ConsoleApplication3.dll”);

} catch (UnsatisfiedLinkError e) {

e.printStackTrace();

}

}

6) Создаем файл для демонстрации работы приложения.

Результат работы программы:

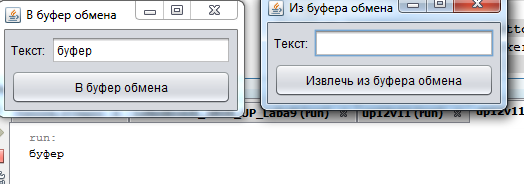


Рисунок 24.1‑ демонстрация работы

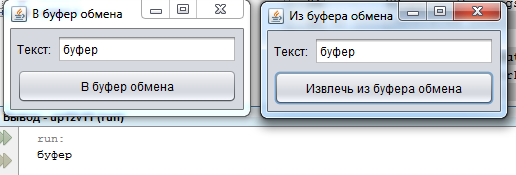


Рисунок 24.2-демонстрация работы