**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ**

**им. проф. М.А. Бонч-Бруевича»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций**

**Отчет о выполнении  
практического занятия №14-15**

Выполнил: Обучающиеся 4 курса, 581 группы,

Лепов Алексей Валерьевич

Проверил: преподаватель,  
Баталов Дмитрий Иннокентьевич

Санкт-Петербург

2021 год

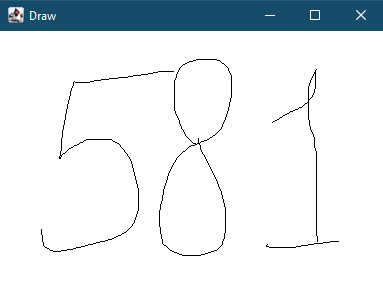
1. **Постановка задач.**

* Изучить работу с графикой.
* Посмотреть на результаты компиляции.
* Также нужно проверить свои навыки загрузки кода программ на хостинге проектов программного обеспечения github.com

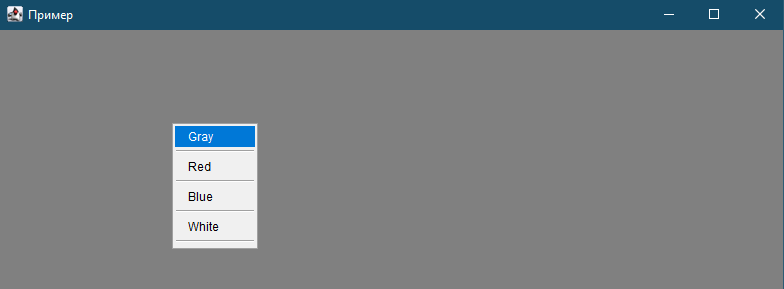
1. **Ход выполнения работ.**

* Выполнено Задание 1 практической работы. Запущена на выполнение программа DrawCanvas.java примитивного графического редактора, приведенный в конспекте лекционного занятия «Занятие № 12. Программирование графики и обработка событий»;
* Выполнено Задание 2. практической работы. Выполнен код программы данный в Примере 1. «Выпадающее меню»;
* Даны ответы на контрольные вопросы.

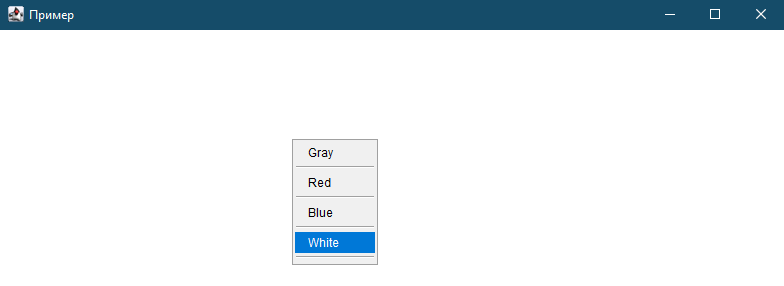
1. **Результаты компиляции программного кода.**



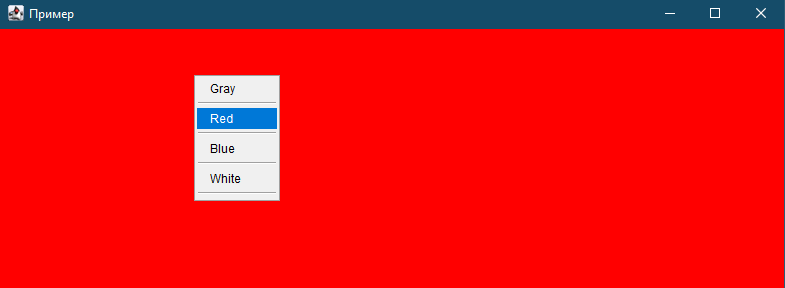
1. *Результат работы программы.* *Задание 1.*



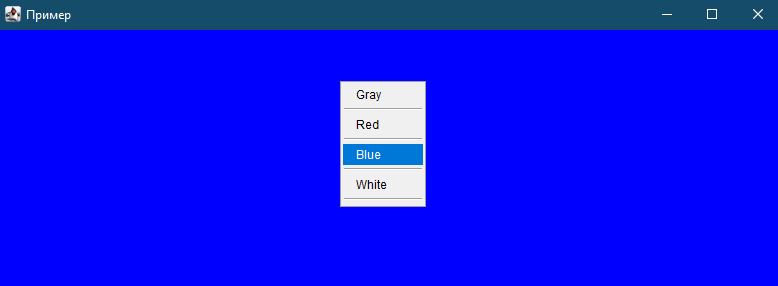
1. *Результат работы программы.* *Задание 2.*



1. *Результат работы программы.* *Задание 2.*



1. *Результат работы программы.* *Задание 2.*



1. *Результат работы программы.* *Задание 2.*
2. **Исходный код проектов.**

Выгрузка проекта была реализована с помощью онлайн инструментария GitHub: (<https://github.com/AlexeyLepov/SPbCK_Lepov_PZ_14-15.git>).

1. **Вопросы для защиты.**
2. Что такое GUI?

GUI (Graphical User Interface, графический интерфейс пользователя) — это форма пользовательского интерфейса, которая позволяет пользователям взаимодействовать с устройством с помощью графических элементов (значки, курсоры, кнопки). Действия в графическом интерфейсе обычно выполняются путём прямого манипулирования графическими элементами.

1. Какие два типа компонентов GUI существует?

* Компоненты AWT. Это просто «Java-оболочка» для элементов управления той операционной системы, на которой они работают. Все запросы к этим компонентам перенаправляются к операционной системе, которая и выполняет всю работу. Использованные ресурсы AWT старается освобождать автоматически. Это немного усложняет архитектуру и влияет на производительность.
* Компоненты Swing Важнейшим отличием Swing от AWT является то, что компоненты Swing вообще не связаны с операционной системой и поэтому гораздо более стабильны и быстры. Такие компоненты в Java называются легковесными (lightweight), и понимание основных принципов их работы во многом объяснит работу Swing.

1. Что такое элементы управления и что такое контейнеры?

Это специальные компоненты библиотеки Swing для создания графического интерфейса приложения. Они представляют собой окна операционной системы, в которых размещаются компоненты пользовательского интерфейса.

1. Какие классы элементов управления существуют?

* JFrame
* JWindow
* JDialog
* JApplet

1. Что необходимо сделать, чтобы ввести компонент в контейнер?

Конструктор JFrame() без параметров создает пустое окно. Конструктор JFrame(String title) создает пустое окно с заголовком title. Чтобы создать простейшую программу с пустым окном необходимо использовать следующие методы:

* setSize(int width, int height) - определение размеров окна;
* setDefaultCloseOperation(int operation) - определение действия при завершении программы;
* setVisible(boolean visible) - сделать окно видимым.

1. Как можно перехватить и обработать события, пришедшие от компонентов?

Всякий раз при вызове метода потока, который потенциально может привести к возникновению исключительной ситуации, необходимо обеспечить и способ обработки исключительных ситуаций для того, чтобы перехватывать любые возбуждаемые ситуации, например:

try

{ // здесь вызываются методы для потоков

}

catch(InterruptedException e)

{ // потоку был послан метод, которым он

// не может оперировать в данном состоянии,

// можно остановить поток его методом stop()

}

1. Какие типы переключателей существуют?

Язык Java поддерживает два типа флажков-переключателей (класс **Checkbox**): исключающие и неисключающие.

**Неисключающие** флажки (*переключатели с независимой фиксацией)* могут быть выбраны независимо от состояния других флажков. Такие флажки являются самостоятельными объектами класса Checkbox, для их создания используются конструкторы **Checkbox()** и **Checkbox(String label)**.

**Исключающие** флажки (*переключатели с зависимой фиксацией*) позволяют одновременно выбрать только один элемент в группе флажков. Если выбирается другой элемент в группе, то выбор предыдущего флажка отменяется, а новый выбор выделяется. Для создания такой группы флажков сначала создается объект класса **CheckboxGroup**, а затем создаются объекты класса Checkboх, входящие в эту группу, для чего используется конструктор **Checkbox(String label, CheckboxGroup group, boolean state).**

1. Как несколько переключателей объединить в группу?

Класс CheckboxGroup содержит методы для определения текущего выбора и установки нового выбора для флажков группы. Метод getCurrent() возвращает ссылку на выбранный флажок группы, а метод setCurrent() переносит выбор на задаваемый флажок группы.

При помощи методов класса getLabel() и getState() класса Checkbox можно получить метку флажка и текущее состояние переключателя, а при помощи методов setLabel() и setState() изменить метку флажка установить состояние флажка.

Для проверки того, какой группе флажков принадлежит флажок, класса Checkbox содержит метод getCheckboxGroup(), а для того, чтобы поместить ранее созданный независимый флажок в группу, или для изменения группы, в которую теперь будет входить флажок, используется метод setCheckboxGroup().

1. Чем отличаются выпадающие и раскрывающиеся списки? Как осуществляется в них выбор элементов?

Класс **CheckboxGroup** содержит методы для определения текущего выбора и установки нового выбора для флажков группы. Метод **getCurrent()** возвращает ссылку на выбранный флажок группы, а метод **setCurrent()** переносит выбор на задаваемый флажок группы.

А списки особенно полезны, если необходимо объединить функциональные возможности меню с возможностью множественного выбора элементов.

1. При помощи какого метода обрабатываются сообщения от линейки прокрутки?

Полосы прокрутки к спискам, меню выбора и полям редактирования при необходимости добавляются автоматически. Но можно использовать их и как независимые, отдельные компоненты для обеспечения прокрутки необходимой информации. Создание линейки прокрутки (класс Scrollbar) и добавление ее в контейнер осуществляется, например, следующим образом:

*Scrollbar scrlbar=new Scrollbar.HORIZONTAL,25,15,1,100);*

*add(scrlbar);*

1. Порождают ли события компоненты класса Label и для чего используются эти компоненты? В чем преимущества их использования перед обычным выводом текста методами класса Graphics?

В классе Graphics определено множество методов для отрисовки простейших графических элементов – линий, прямоугольников и многоугольников, окружностей и овалов, текста, картинок и т.д.

Label - этот компонент отображает надпись. Соответственно, и его основной конструктор принимает один аргумент типа String – текст надписи. С помощью стандартных свойств класса Component – шрифт, цвет, фоновый цвет – можно менять вид надписи. Текст можно сменить и после создания Label с помощью метода setText.

1. Что такое текстовые поля и текстовые области? Чем они отличаются?

Текстовые поля поддерживают ввод маскируемых символов, т.е. символов, ввод которых на экране отображается каким-либо одним символом (эхо-символом), а не фактически вводимыми символами. Для установки такого эхо-символа используется метод setEchoCharacter(), а для того, чтобы определить, какой символ используется в качестве эхо-символа, - метод getEchoChar(). Для проверки того, имеет ли поле эхо-символ, применяется метод echoCharIsSet().

1. Что такое контейнеры? Какие основные виды контейнеров существует?

Контейнеры - это объекты (компоненты), позволяющие помещать в себя другие различные компоненты.

Класс контейнеров **Сontainer** - подкласс класса Component. Существует два вида основных вида контейнеров: **панели** (класс Panel, подклассом которого является класс Applet) и **окна** (класс Window, подклассами которого являются Frame и Dialog.

Контейнеры имеют следующие деревья наследования:

Object -> Component -> Сontainer -> **Panel**

Object -> Component -> Сontainer -> Panel -> **Applet**

Object -> Component -> Container -> **Window**

Object -> Component -> Container -> Window -> **Frame**

Object -> Component -> Container -> Window -> **Dialog**

Object -> Component -> Container -> Window -> Dialog -> **FileDialog**

1. Для чего чаще всего используются панели?

**Когда интерфейс состоит из большого количества элементов, почти всегда есть смысл объединить группы связанных по смыслу элементов с помощью панелей.**

1. Как можно рисовать в окне панели?

Так как окно класса Frame произошло от класса Container, то в него можно добавлять различные компоненты и панели, так же как в делается для апплетов и панелей

1. В чем основное отличие окон и панелей?

Диалоговые окна удобны для приглашения пользователя к выполнению какого-либо действия и для подтверждения того, что пользователь ознакомился с сообщением программы. Диалоговые окна в отличие от фреймов не реализуют панель меню автоматически. Поскольку диалоговые окна обычно меньше по размерам, чем фреймы, они бывают удобны для быстрого диалога с пользователем.

1. Что является обязательным параметром конструктора при создании экземпляра класса окон?

Менеджер GridBagLayout является самым сложным менеджером компоновки. В нем используется наиболее совершенный алгоритм реагирования на изменение размеров контейнера, и позволяет реализовывать сложный интерфейс, в котором контейнер содержит много компонентов различных размеров, некоторые из которых должны находиться в одном и том же заданном положении относительно других. Когда используется этот менеджер, необходимо задавать параметры расположения для каждого компонента с помощью метода setConstraints(). Удобнее всего создать для каждого компонента экземпляр класса GridBagConstraints, что позволит изменять расположение этого компонента независимо от других.

1. Каковы отличительные особенности имеют фреймы?

Фрейм (класс Frame) - это объект, который может существовать без всякой связи с окном WWW-браузера. С помощью класса Frame можно организовать интерфейс независимого апплета.

Окно, созданное на базе класса Frame, больше всего похоже на главное окно обычного приложения Windows, и может пользоваться большим количеством методов класса Frame. Оно автоматически может иметь главное меню (так как класс Frame реализует интерфейс MenuContainer), для него можно устанавливать форму курсора и пиктограмму. Внутри такого окна можно рисовать. Так как окно класса Frame произошло от класса Container, то в него можно добавлять различные компоненты и панели, так же как в делается для апплетов и панелей.

1. Какие методы должны быть переопределены в новом подклассе фреймов?

Кроме окон и фреймов есть особый подкласс окон, называемый диалоговыми окнами или просто диалоги. В отличие от фреймов или обычных окон диалог предоставляет пользователю окно, в котором нужно выполнить те или иные действия, лишь по завершении которых пользователь сможет продолжить работу с программой.

Диалоговые окна (класс Dialog) используются в основном для одноразового запроса информации у пользователя или для вывода небольших порций информации на экран.

1. Какой класс контейнеров автоматически поддерживает работу с меню? Почему он это делает автоматически?

Диалоговые окна в отличие от фреймов не реализуют панель меню автоматически.

1. Как добавить меню в контейнер?

Для того, что создать и добавить меню в контейнер, необходимо сначала создать панель меню (объект класса MenuBar), затем добавить в нее меню (объект класса Menu). В выпадающие меню класса Menu также можно добавлять элементы (строки или объекты класса MenuItem). Подготовленную панель меню теперь можно ввести в контейнер его методом add().

1. Как создать новое меню и добавить в него элементы?

Для формирования главного меню создается объект класса MenuBar, а затем в него добавляются отдельные меню:

MenuBar mainMenu=new MenuBar(); // создание панели меню

Menu fileMenu=new Menu("File"); // создание меню "File"

Menu helpMenu=new Menu("Help"); // создание меню "Help"

После создания отдельных меню в них можно добавить элементы. Для этого нужно вызвать метод add() класса Menu:

fileMenu.add(new MenuItem("New")); // строка "New" меню "File"

fileMenu.addSeparator(); // горизонтальный разделитель

fileMenu.add(new MenuItem("Exit")); // строка "Exit" меню "File"

helpMenu.add(new MenuItem("Content")); // строка "Content" меню "Help"

helpMenu.add(addSeparator()); // горизонтальный разделитель

helpMenu.add(new MenuItem("About")); // строка "Content" меню " About "

1. Какие существуют классы элементов меню?

Класс MenuItem определяет поведение элементов меню. Пользуясь методами этого класса, можно блокировать или деблокировать отдельные элементы - это нужно делать, например, если в данный момент функция, соответствующая строке меню, недоступна или не определена. Можно также изменять текстовые строки, соответствующие элементам меню, что может пригодиться для переопределения их значения.

Некоторые методы класса MenuItem:

* disable - Блокирование элемента меню
* enable - Разблокирование элемента меню. Другая версия этого метода осуществляет блокировку и разблокировку по условию
* getLabel - Получение текстовой строки элемента меню
* isEnable - Проверка того, является ли элемент заблокированным
* setLabel - Установка текстовой строки элемента меню

1. Какими методами обрабатываются события меню?

Когда пользователь выбирает одну из команд меню, происходит генерация события класса ActionEvent. Для его обработки используется интерфейс ActionListener с одним методом actionPerformed.

1. Для чего в основном используются окна диалогов?

Диалоговые окна удобны для приглашения пользователя к выполнению какого-либо действия и для подтверждения того, что пользователь ознакомился с сообщением программы. Диалоговые окна в отличие от фреймов не реализуют панель меню автоматически. Поскольку диалоговые окна обычно меньше по размерам, чем фреймы, они бывают удобны для быстрого диалога с пользователем.

1. Каковы важные отличия окон диалогов от фреймов?

В отличие от фреймов или обычных окон диалог предоставляет пользователю окно, в котором нужно выполнить те или иные действия, лишь по завершении которых пользователь сможет продолжить работу с программой. Они используются в основном для одноразового запроса информации у пользователя или для вывода небольших порций информации на экран.

1. Объект какого класса должен обязательно быть родителем диалогового окна?

При работе с диалоговыми окнами важно помнить одно правило: каждое диалоговое окно обязательно должно иметь фрейм в качестве родителя. Это значит, что диалоговое окно нельзя открыть непосредственно из апплета. Чтобы создать диалоговое окно, необходимо сначала завести фрейм, даже если его единственным значением будет служить родителем для диалогового окна. Только если апплет уже использует фреймы, можно обойтись без этой подготовительной стадии.

1. Как создать диалог своего класса?

Чтобы создать диалоговое окно, необходимо сначала завести фрейм, даже если его единственным значением будет служить родителем для диалогового окна. Только если апплет уже использует фреймы, можно обойтись без этой подготовительной стадии.

1. Для чего предназначены менеджеры компоновки? Какие существуют режимы размещения?

Менеджеры компоновки – библиотеки, вычисляющие и устанавливающие размер и местоположение компонентов в контейнере.

Это класс, который имеет четыре открытых поля – top, right, bottom, left, значения которых описывают отступы со всех четырех сторон, которые необходимо сделать, чтобы получить область, доступную для расположения компонент.