МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения

# ОТЧЁТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
## по дисциплине
## «Программирование на языке Java»

## Тема: Обработка событий в Java программах с графическим интерфейсом пользователя.

Выполнил студент группы ИКБО-16-20 Пак С.А.

Принял ассистент кафедры ИиППО Русляков А.А.

Практические работы выполнены «_____» 2021г.

«Зачтено» «_____» 2021г

Москва 2021

# СОДЕРЖАНИЕ

# ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## 1. Механизм обработки событий библиотеки Swing

В контексте графического интерфейса пользователя наблюдаемыми объектами являются элементы управления. Они могут сообщить своим наблюдателям об определенных событиях, как элементарных, так и о высокоуровневых.

Наблюдателями должны являться объекты классов, поддерживающих специальные интерфейсы. Такие классы в терминологии Swing называются слушателями.

## 2. Интерфейс MouseListener и обработка событий от мыши

Слушатель событий от мыши должен реализовать интерфейс MouseListener. В этом интерфейсе перечислены следующие методы:

- public void mouseClicked(MouseEvent event) — выполнен щелчок мышкой на наблюдаемом объекте;
- public void mouseEntered(MouseEvent event) — курсор мыши вошел в область наблюдаемого объекта;
- public void mouseExited(MouseEvent event) — курсор мыши вышел из области наблюдаемого объекта;
- public void mousePressed(MouseEvent event) — кнопка мыши нажата в момент, когда курсор находится над наблюдаемым объектом;
- public void mouseReleased(MouseEvent event) — кнопка мыши отпущена в момент, когда курсор находится над наблюдаемым объектом.

Чтобы обработать нажатие на кнопку, требуется описать класс, реализующий интерфейс MouseListener. Далее необходимо создать объект этого класса и зарегистрировать его как слушателя интересующей нас кнопки. Для регистрации слушателя используется метод addMouseListener().

## 3. Класс MouseAdapter

Класс MouseAdapter реализует интерфейс MouseListener, определяя пустые реализации для каждого из его методов. Можно унаследовать своего слушателя от этого класса и переопределить те методы, которые нам нужны.

## 4. Общая структура слушателей

Кроме слушателей MouseListener визуальные компоненты Swing поддерживают целый ряд других слушателей.

Каждый слушатель должен реализовывать интерфейс ***Listener, где *** — тип слушателя. Практически каждому из этих интерфейсов (за исключением тех, в которых всего один метод) соответствует пустой класс-заглушка

***Adapter. Каждый метод интерфейса слушателя принимает один параметр типа ***Event, в котором собрана вся информация, относящаяся к событию.

Чтобы привязать слушателя к объекту (который поддерживает 45 соответствующий тип слушателей) используется метод add***Listener(***Listener listener).

## 5. Слушатель фокуса FocusListener

Слушатель FocusListener отслеживает моменты, когда объект получает фокус (то есть становится активным) или теряет его.

Интерфейс FocusListener имеет два метода:

- public void focusGained(FocusEvent event) — вызывается, когда наблюдаемый объект получает фокус;
- public void focusLost(FocusEvent event) — вызывается, когда наблюдаемый объект теряет фокус.

## 6. Слушатель колёсика мышки MouseWheelListener

Слушатель MouseWheelListener оповещается при вращении колесика мыши в тот момент, когда данный компонент находится в фокусе. Этот интерфейс содержит всего один метод:

public void mouseWheelMoved(MouseWheelEvent event).

## 7. Слушатель клавиатуры KeyListener

Слушатель KeyListener оповещается, когда пользователь работает с клавиатурой в тот момент, когда данный компонент находится в фокусе. В интерфейсе определены методы:

- public void mouseKeyTyped(KeyEvent event) — вызывается, когда с клавиатуры вводится символ;
- public void mouseKeyPressed(KeyEvent event) — вызывается, когда нажата клавиша клавиатуры;
- public void mouseKeyReleased(KeyEvent event) — вызывается, когда отпущена клавиша клавиатуры.

## 8. Слушатель изменения состояния ChangeListener

Слушатель ChangeListener реагирует на изменение состояния объекта. В интерфейсе определен всего один метод:

public void stateChanged(ChangeEvent event).

## 9. Слушатель событий окна WindowListener

- public void windowOpened(WindowEvent event) — окно открылось;
- public void windowClosing(WindowEvent event) — попытка закрытия окна (например, пользователя нажал на крестик). Слово «попытка» означает,

что данный метод вызовется до того, как окно будет закрыто и может воспрепятствовать этому (например, вывести диалог типа «Вы уверены?» и отменить закрытие окна, если пользователь выберет «Нет»);

- public void windowClosed(WindowEvent event) — окно закрылось;
- public void windowIconified(WindowEvent event) — окно свернуто;
- public void windowDeiconified(WindowEvent event) — окно развернуто;
- public void windowActivated(WindowEvent event) — окно стало активным;
- public void windowDeactivated(WindowEvent event) — окно стало неактивным.

## 10. Слушатель событий компонента ComponentListener

Слушатель ComponentListener оповещается, когда наблюдаемый визуальный компонент изменяет свое положение, размеры или видимость. В интерфейсе четыре метода:

- public void componentMoved(ComponentEvent event) — вызывается, когда наблюдаемый компонент перемещается (в результате вызова команды setLocation(), работы менеджера размещения или еще по какой-то причине);
- public void componentResized(ComponentEvent event) — вызывается, когда изменяются размеры наблюдаемого компонента;
- public void componentHidden(ComponentEvent event) — вызывается, когда компонент становится невидимым;
- public void componentShown(ComponentEvent event) — вызывается, когда компонент становится видимым.

## 11. Слушатель выбора элемента ItemListener

Слушатель ItemListener реагирует на изменение состояния одного из элементов, входящих в состав наблюдаемого компонента. Слушатель обладает одним методом:

public void itemStateChanged(ItemEvent event).

## 12. Универсальный слушатель ActionListener

Среди многочисленных событий, на которые реагирует каждый элемент управления (и о которых он оповещает соответствующих слушателей, если они к нему присоединены), есть одно основное, вытекающее из самой сути компонента и обрабатываемое значительно чаще, чем другие. Например, для кнопки это щелчок на ней, а для выпадающего списка — выбор нового элемента.

Для отслеживания и обработки такого события может быть использован особый слушатель ActionListener, имеющий один метод:

public void actionPerformed(ActionEvent event).

# РЕШЕНИЕ ЗАДАЧИ

## 1. Постановка задачи

**Вариант:** 3

**Задание:** Реализация программу на Java с JTextArea и двумя меню: Цвет: который имеет возможность выбора из три возможных : синий, красный и черный Шрифт: тривида: “Times New Roman”, “MS Sans Serif”, “Courier New”.Вы должны написать прогамму, которая с помощью меню, может изменять шрифт и цвет текста, написанного в JTextArea

## 2. Программный код

Файл App.java:

```
package ru.mirea;

import java.awt.Font;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JMenu;
import javax.swing.JFrame;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;

public class App extends JFrame {
  private static final int WINDOW_WIDTH = 1000;
  private static final int WINDOW_HEIGHT = 600;

  /**
   * Создаёт окно приложения
   */
  public App() {
    super("Super Text Editor");

    setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    setLayout(null);

    JMenuBar menuBar = new JMenuBar();

    /* Пункты меню */
    JMenu fontOption = new JMenu("Family");
    JMenu fontColor = new JMenu("Color");
```

```
    /* Шрифты, которые можно выбрать */
    JMenuItem[] families = {
      new JMenuItem("Free Mono"),
      new JMenuItem("Ubuntu Mono"),
      new JMenuItem("Liberation Serif")
    };

    /* Различные цвета текста */
    JMenuItem[] colors = {
      new JMenuItem("Red"),
      new JMenuItem("Blue"),
      new JMenuItem("Black")
    };

    /* Текстовое поле */
    JTextArea textArea = new JTextArea();

    /* Полоса прокрутки для текстового поля */
    JScrollPane scrollBar = new JScrollPane(textArea);

    /* Расположение меню и его размеры */
    menuBar.setBounds(0, 0, WINDOW_WIDTH, 28);

    /* Выбор шрифта для пунктов меню */
    fontOption.setFont(new Font("Free Mono", Font.BOLD, 15));
    fontColor.setFont(new Font("Free Mono", Font.BOLD, 15));

    /* Установка шрифта, переноса слов */
    textArea.setFont(new Font("Free Mono", Font.PLAIN, 18));
    textArea.setLineWrap(true);
    textArea.setWrapStyleWord(true);

    /* Расположение и размеры текстового поля */
    scrollBar.setBounds(0, menuBar.getHeight(), WINDOW_WIDTH,
WINDOW_HEIGHT - menuBar.getHeight());

    /* Возможность сменить шрифт */
    for (JMenuItem item : families) {
      /* Выполнение действия при нажатии на пункт меню */
      item.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          String fontName = item.getText();       // название
шрифта
          textArea.setFont(new Font(fontName, Font.PLAIN, 18));
        }
```

```
      });

      fontOption.add(item);
    }

    /* Возможность сменить цвет текста */
    for (JMenuItem item : colors) {
      /* Выполнение действия при нажатии на пункт меню */
      item.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          String fontName = item.getText();       // название
шрифта
          textArea.setFont(new Font(fontName, Font.PLAIN, 18));
        }
      });

      fontOption.add(item);
    }

    /* Возможность сменить цвет текста */
    for (JMenuItem item : colors) {
      item.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
          String color = item.getText();
          Color colorToSet;

          if (color.equals("Red")) {
            colorToSet = Color.RED;
          }
          else if (color.equals("Blue")) {
            colorToSet = Color.BLUE;
          }
          else {
            colorToSet = Color.BLACK;
          }

          textArea.setForeground(colorToSet);
        }
      });

      fontColor.add(item);
    }

    menuBar.add(fontOption);
    menuBar.add(fontColor);
```

```
    add(menuBar);
    add(scrollBar);
  }

  /**
   * Запускает приложение
   * @param args      аргументы командной строки
   */
  public static void main(String[] args) {
    App window = new App();

    window.setVisible(true);
    window.setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
}
```

## 3. Вывод программы

На рис.1, 2 показана смена шрифта.

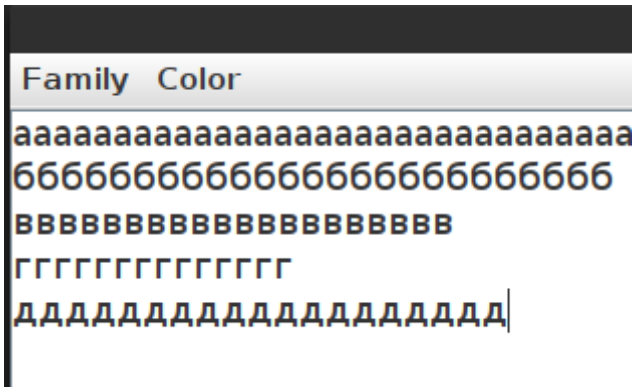


Рис.1 Смена шрифта ч.1

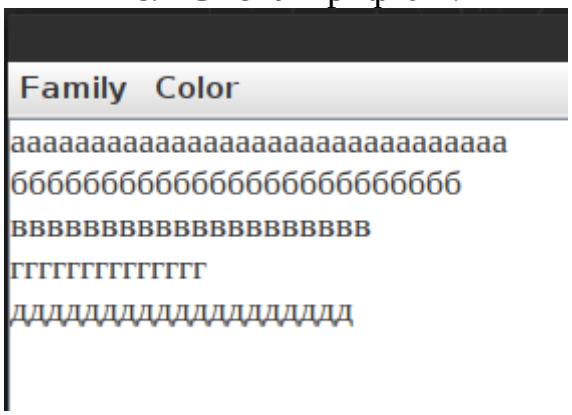


Рис.2 Смена шрифта ч.2

На рис.3, 4 показана смена шрифта.

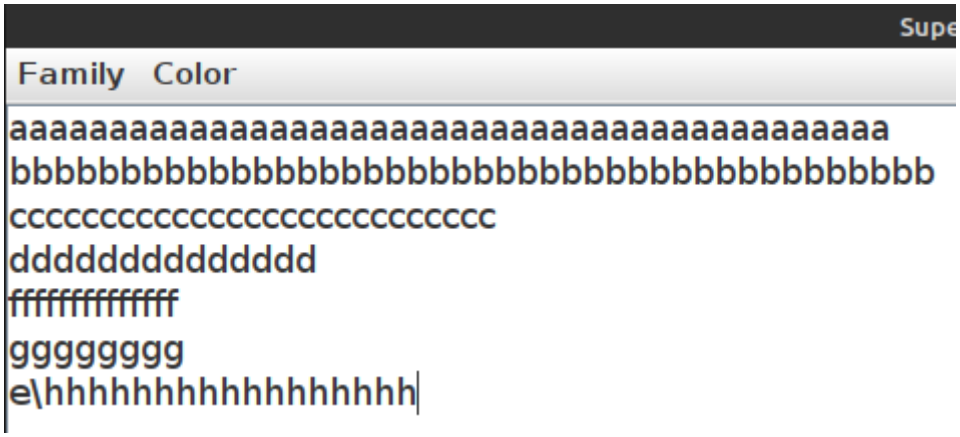


Рис.3 Смена шрифта ч.1

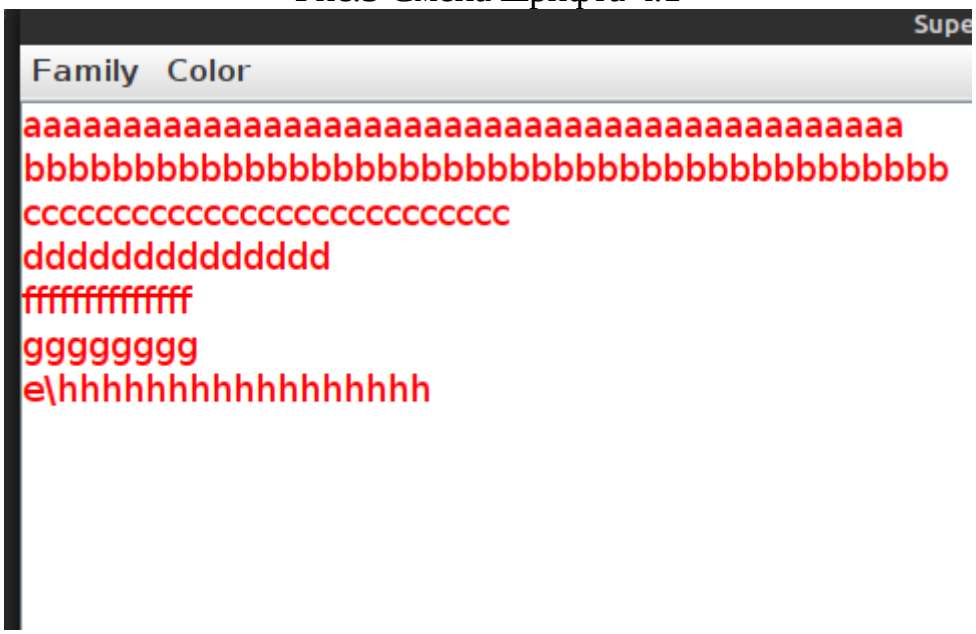


Рис.4 Смена шрифта ч.2

# ВЫВОД

В ходе выполнения научился обрабатывать различные события для разных компонентов (кнопок, меню и т. д.).