

Лабораторная работа №9. Создание меню, графика в языке Java. Требование к отчету.

Отчет должен содержать:

- Титульный лист согласно образца с указанным номером и названием лабораторной работы.
- Оглавление с указанием номера страницы для каждого раздела.
- Цель работы.
- Теоретическое обоснование. Пишется самостоятельно и должно охватывать вопросы, затрагиваемые в лабораторной работе.
- Задание.
- Документированные листинги программ. После каждого листинга программы должен приводиться скриншот с результатом её работы.
- Выводы.

Теоретическое обоснование.

Для создания меню в языке Java есть удобный класс `JMenuBar`, который представляет собой строку меню, куда можно вставлять пункты меню с нужными подпунктами. Для пунктов меню и подпунктов можно использовать класс `JMenuItem`, вставляя подпункты в соответствующие пункты меню. Чтобы установить созданное меню на нужное окно, нужно использовать метод `setJMenuBar` для выбранного объекта типа `JFrame`. Например, создадим окно с меню и текстовой строкой, в которую будут выводиться названия выбранных подпунктов меню (рис. 1):



Рисунок 1

```
import java.awt.BorderLayout; import java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
```

```
import javax.swing.*;
```

Рис. 9. При выборе пункта меню его название выводится в текстовую строку. `public class MyMenu extends JFrame implements ActionListener{ private JTextField myText;`

```
public static void main(String[] args) {
new MyMenu("Пример окна с меню");//создаем окно
с меню
}
```

```
public MyMenu(String name){//конструктор окна с меню, параметр - заголовок окна
super(name); //передаем заголовок окна в конструктор класса-родителя - в JFrame
JMenuBar myMenuBar=new JMenuBar();// создаем строку меню
```

```
JMenu menu1=new JMenu("Пункт1");//создаем первый
пункт меню JMenu first=new JMenu("Пункт1_1"); //создаем
подпункт меню menu1.add(first);//добавляем подпункт в
первый пункт меню
JMenuItem[] first_1=new JMenuItem[3];// создаем массив из трех подпунктов меню
```

```

        for (int i=0;i<3;i++){ //в цикле создается каждый подпункт, добавляется в
нужное место меню        first_1[i]=new JMenuItem("Пункт1_1_"+(i+1));
            first.add(first_1[i]);
            first_1[i].addActionListener(this);//и к нему подключаем слушатель, описанный в
конце
        }
        //класса
        JMenu second=new JMenu("Пункт1_2"); //создаем подпункт меню
        menu1.add(second); //добавляем его в меню
        JMenuItem[] second_1=new JMenuItem[3]; // создаем массив из трех
подпунктов меню        for (int i=0;i<3;i++){ // и т.д.
            second_1[i]=new JMenuItem("Пункт1_2_"+(i+1));
            second.add(second_1[i]);
            second_1[i].addActionListener(this);
        }

        JMenu menu2=new JMenu("Пункт2"); //создаем второй пункт меню и далее аналогично
первому
        JMenu first2=new JMenu("Пункт2_1");
        menu2.add(first2);
        JMenuItem[] first2_1=new JMenuItem[3];
        for (int i=0;i<3;i++){
            first2_1[i]=new JMenuItem("Пункт2_1_"+(i+1));
            first2.add(first2_1[i]);
            first2_1[i].addActionListener(this);
        }
        JMenu second2=new JMenu("Пункт2_2");
        menu2.add(second2);
        JMenuItem[] second2_1=new JMenuItem[3];
        for (int i=0;i<3;i++){
            second2_1[i]=new
JMenuItem("Пункт2_2_"+(i+1));
            second2.add(second2_1[i]);
            second2_1[i].addActionListener(th
is);
        }

        myMenuBar.add(menu1); //в строку меню добавляем
главные пункты меню        myMenuBar.add(menu2);
myText=new JTextField();
        setJMenuBar(myMenuBar); //устанавливаем для окна созданное меню
add(myText, BorderLayout.SOUTH);
        setSize(300, 200);
setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {//описываем метод слушателя, отвечающий
за действия        myText.setText(e.getActionCommand()); //при выборе пункта меню
    } // e.getActionCommand() возвращает название пункта меню, который был
выбран }

```

Для использования графических возможностей языка Java необходимо создать свой компонент, на котором будет происходить рисование, унаследовав его от стандартного компонента пакета Swing, и переопределить в нем метод paintComponent. Например, создадим панель, на которой будет происходить рисование:

```

//файл
JMyPanel.java
import
java.awt.*;
import

```

```

javax.swing.J
Panel;

public class JMyPanel extends JPanel{ // наш класс является наследником класса JPanel
    //создаем перечисление используемых
    параметров    public static enum Figure
    {LINE,OVAL,RECT,ROUNDRECT,CLEAR};
    private Figure vibor=Figure.CLEAR; //объявляем переменную типа созданного
    перечисления

    //и присваиваем ей значение CLEAR

    public JMyPanel() {    } //конструктор нашего класса
    public void ris(String s) {//метод, вызов которого приводит к
    перерисовке панели

        //параметр s принимает значение во время вызова данного метода (см.
        MyGraph.java)    vibor=Figure.valueOf(s); //устанавливаем, что нужно
        нарисовать    repaint(); //перерисовываем нашу панель, т.е. вызываем метод
        paintComponent    }

    public void paintComponent(Graphics gr){ //переопределяемый метод с параметром типа
    Graphics

        super.paintComponent(gr); // вызов такого же метода родительского класса
        // и передача ему параметра типа Graphics
        Graphics2D g = (Graphics2D)gr; //преобразование параметра к типу Graphics2D
        //задание параметров для сглаживания графики (антиалиасинг)
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON);    switch (vibor){ //в зависимости от
        установленного методом ris значения для vibor        case LINE:
        g.drawLine(20, 20, 100, 100); break; //рисует линию        case OVAL:
        g.drawOval(20, 20, 100, 100); break; // круг        case RECT: g.drawRect(20,
        20, 100, 100); break; // прямоугольник        case ROUNDRECT:
        g.drawRoundRect(20, 20, 100, 100,60,60); break; // прямоугольник со
        //скругленными краями, первые четыре параметра - такие же, как у
        прямоугольника,
        //еще 2 - скругление углов по x и по y
        case CLEAR: g.clearRect(0, 0, getSize().width, getSize().height);break;
    //очищаем
    }
    }
}

```

В этом примере использован тип данных `enum` – перечисление. Этот тип данных используется для хранения константных значений в виде их перечисления. При этом всем значениям присваивается порядковый номер в зависимости от порядка, в котором эти константы перечислены. В нашем примере значение `LINE` имеет порядковый номер 0, `OVAL` – 1, `RECT` – 2, `ROUNDRECT` – 3, `CLEAR` – 4. Узнать порядковый номер текущего значения переменной перечисляемого типа можно, вызвав метод `ordinal()`, например, `vibor.ordinal()` вернет нам порядковый номер текущего значения переменной `vibor` (0, 1, 2, 3 или 4). Также можно узнать текстовое значение переменной перечисляемого типа, вызвав метод `toString()`. В нашем случае `vibor.toString()` выдаст одно из значений: `LINE`, `OVAL`, `RECT`, `ROUNDRECT` или `CLEAR`. Все эти особенности позволяют удобно использовать переменные перечисляемого типа (в нашем случае – для организации выбора рисуемой фигуры).

Также в приведенном примере используется метод `paintComponent(Graphics gr)` для перерисовки компонента. Его параметр `gr` типа `Graphics` в дальнейшем используется при создании объекта `g` типа `Graphics2D`, методами которого и рисуются все фигуры. `Graphics2D` является расширением класса `Graphics` и позволяет выполнять различные преобразования графики, например, сглаживание, чего нет в классе `Graphics`, который использовался для рисования раньше.

После описанных манипуляций можно использовать созданный класс для рисования. Создадим окно, в котором, в зависимости от нажатой кнопки, рисуется нужная фигура или все стирается (рис. 2):



Рис. 2. Рисование фигуры в зависимости от нажатой кнопки.

```
// файл
MyGraph.java
import
java.awt.*;
import
java.awt.event.*; import
javax.swing.*;
*;

public class MyGraph extends JFrame implements ActionListener{

    private JMyPanel myPanel=new JMyPanel();//объявляем и создаем нашу панель для
    рисования

    public static void main(String[]
args) {        new MyGraph("Окно с
графикой");//создаем окно
    }

    public MyGraph(String s){ //конструктор с параметром для заголовка окна
super(s);//вызываем конструктор суперкласса и передаем ему параметр
Box myBox=new Box(BoxLayout.X_AXIS);//создаем компоновку в виде
горизонтального ящика
        JButton[] figs=new JButton[5]; //массив кнопок
        for (int i=0;i<5;i++){
            //каждая кнопка создается с параметром надписи на ней, надпись берется из
перечисления,
            //объявленного в классе JMyPanel, values()[i].toString() переводит в текст название
i-го
            //параметра из Figure
            figs[i]=new JButton(JMyPanel.Figure.values()[i].toString());
figs[i].addActionListener(this); //добавляем слушатель, который реализуется в конце
            //описания класса
myBox.add(figs[i]); //добавляем кнопку в компоновку
            if (i!=4){ //для всех кнопок кроме последней вставляем пружину после
кнопки
                myBox.add(Box.createHorizontalGlue());
            }
        }

        myBox.setAlignmentX(CENTER_ALIGNMENT);//устанавливаем для компоновки выравнивание по
центру

            //хотя в нашем случае это не важно, т.к. мы
используем пружины
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
add(myBox, BorderLayout.NORTH);    add(myPanel, BorderLayout.CENTER);
        Dimension size=getSize();//записываем в переменную size текущий размер окна
        size.setSize(size.width, size.height+200);//устанавливаем новый размер окна,
увеличивая
```

```

        //текущий по высоте на 200    setMinimumSize(size);
pack();
    setVisible(true);
}

```

```

public void actionPerformed(ActionEvent e) { //при нажатии на одну из кнопок
    myPanel.ris(e.getActionCommand()); //вызываем метод ris нашей панели (см.
JMyPanel.java)    }    //и передаем в качестве параметра название нажатой
кнопки (e.getActionCommand()) }

```

Добавим немного разнообразия в созданный класс JMyPanel, изменив его метод paintComponent:

```

public void paintComponent(Graphics gr){
super.paintComponent(gr);
    Graphics2D g = (Graphics2D)gr;
    BasicStroke pen;//создаем перо, параметры которого будут определять стиль линий
    g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
    switch
(vibor){
case LINE:
//определяем перо толщиной 20 точек, с закругленными концами линий и закругленными
стыками линий    pen=new
BasicStroke(20,BasicStroke.CAP_ROUND,BasicStroke.JOIN_ROUND);
    g.setStroke(pen);//делаем текущим пером созданное нами
        g.setColor(Color.blue);//задаем цвет пера
        g.drawLine(20, 20, 100,
100); break;    case OVAL:
//задаем массив, определяющий вид линии
// элементы массива с четными индексами задают длину штриха в пикселах, элементы с
нечетными
//индексами — длину промежутка; массив перебирается циклически;
        float[] dash = {10, 30};
//определяем перо толщиной 10 точек, с квадратными концами линий, закругленными
стыками линий,
//расстоянием в 10 точек, с которого начинает действовать закругление, массив,
определяющий вид
//линии, и с какого элемента массива начинать узор
        pen=new BasicStroke(10,BasicStroke.CAP_SQUARE,BasicStroke.JOIN_ROUND,10, dash,0
);
        g.setStroke(pen);
        g.setColor(Color.red);
//устанавливаем стиль заливки, в качестве параметра задаем градиент от красного к
зеленому,
//30, 30 - начальная точка первого цвета, 50, 50 - начальная точка второго цвета,
true -
//циклическость градиента
        g.setPaint(new GradientPaint(30, 30, Color.red, 50, 50, Color.green, true));
//g.fill - создание объекта с заливкой, в качестве параметра задается объект из
пакета Graphics2D,
//в нашем случае - эллипс
        g.fill(new Ellipse2D.Double(20, 20, 100, 100));
break;    case RECT:
        float[] dash2 = {20, 20};
        pen=new BasicStroke(5,BasicStroke.CAP_SQUARE,BasicStroke.JOIN_BEVEL,1, dash2,0
);
        g.setStroke(pen);
        g.setColor(Color.magenta);
        g.drawRect(20, 20, 100,
100); break;    case ROUNDERECT:
        float[] dash3 = {20, 20,2,20,2,20};    pen=new
BasicStroke(10,BasicStroke.CAP_ROUND,BasicStroke.JOIN_BEVEL,1, dash3,0 );

```

```

        g.setStroke(pen);
        g.setColor(Color.yellow);
        g.drawRoundRect(20, 20, 100, 100, 60, 60); break;
    case CLEAR: g.clearRect(0, 0, getSize().width,
        getSize().height);break;
    }
}

```



```

}

```

Задание.

Задачи: I. Обязательная задача для всех:

Переделать пример с рисованием фигур так, чтобы фигуры рисовались выбором соответствующего пункта меню.

II. Индивидуальное задание.

Нарисовать свою фамилию линиями разной толщины, цвета и стиля (и вообще – проявить фантазию).