

ГИП. Диалоги

Контейнеры верхнего уровня

Три типа контейнеров верхнего уровня, один из них должен присутствовать в Swing-приложении

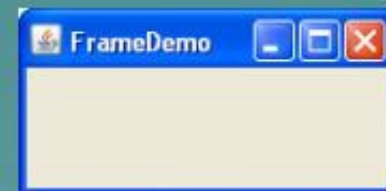
JApplet
(апплет)



JDialog
(диалог)



JFrame
(фрейм)



Другие контейнеры общего назначения: JPanel, JScrollPane, JSplitPane, JTabbedPane, JToolBar.

Имеется ряд контейнеров специального назначения.

Диалоги

Способ взаимодействия программы и пользователя

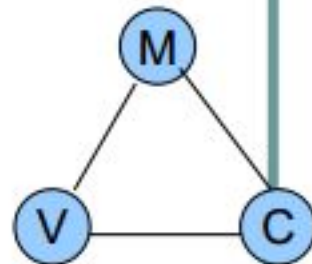
- Программа: сообщение для пользователя
- Программа: прими решение
- Программа: введи данные

- Что хранит
- Как выглядит
- Как управляется

Модель

Вид (View)

Управление
(Controller)



Диалоги

- **Действие пользователя**
 - закрыть окно диалога
 - нажать одну из предусмотренных кнопок
 - ввести необходимые данные (и нажать одну из предусмотренных кнопок)
- **Окна диалога бывают или модальными или немодальными.**
 - При открытом модальном диалоге переход в другие окна блокируется

Модальное окно диалога должно быть закрыто , прежде чем вы сможете продолжить работу с остальной частью приложения

JOptionPane – стандартные диалоги

- JOptionPane – простой способ применения окон стандартных диалогов
- Стандартные диалоги информируют пользователя о некоторой ситуации или предлагают ему принять или изменить некоторое значение

Методы JOptionPane

Класс содержит большое число методов (>50)

Наиболее применяемые методы (в алф. пор.)

- `showConfirmDialog` -- Вопрос с ответами `yes/no/cancel`.
- `showInputDialog` -- Запрос на ввод данных
- `showMessageDialog` -- Сообщение о чем-либо
- `showOptionDialog` – Обобщение предыдущих методов

JOptionPane



Типовое расположение элементов в окне стандартного диалога

- иконка – для С.Д. определяется типом сообщения
- типы сообщений
 - ERROR_MESSAGE
 - INFORMATION_MESSAGE
 - WARNING_MESSAGE (предупреждающее)
 - QUESTION_MESSAGE
 - PLAIN_MESSAGE (простое)

JOptionPane

- Кнопки ответов
 - DEFAULT_OPTION
 - YES_NO_OPTION
 - YES_NO_CANCEL_OPTION
 - OK_CANCEL_OPTION
- Ответы (если диалог возвращает целое)
 - YES_OPTION
 - NO_OPTION
 - CANCEL_OPTION
 - OK_OPTION
 - CLOSED_OPTION

Пример 1 – стандартные диалоги

```
import java.awt.*; //Проект - Диалоги 1-стандартные
import javax.swing.*;

public class DialogDemo extends JComponent {
    private static void createAndShowGUI(){
        JFrame frame = new JFrame("Диалоги");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel content = JPanel();
        frame.setContentPane(content); //можно так или как раньше
        content.setBackground(Color.BLUE);
        content.setBorder (
            BorderFactory.createLineBorder(Color.RED,7));
        frame.setSize(400,400);
        frame.setLocation(10,10);
        frame.setVisible(true);
    }
}
```

```
while (true) {
    JOptionPane.showMessageDialog(frame,
        "Пример для JOptionPane.showMessageDialog");

    String response = JOptionPane.showInputDialog(frame,
        "Пример для JOptionPane.showInputDialog\n"+
        "Введите текст и нажмите кнопку");
    System.out.println("Ответ: " + response);

    int iresponse = JOptionPane.showConfirmDialog(null,
        "Пример для JOptionPane.showConfirmDialog\n"+
        "Продолжить?"); //объясните расположение диалога
    System.out.println("Ответ: " + iresponse);

    if (iresponse == 1) break;
}
} // createAndShowGUI(){
```

```
public static void main (String[] args){  
    javax.swing.SwingUtilities.invokeLater(new Runnable(){  
        public void run(){createAndShowGUI();});  
    }  
} // DialogDemo
```



Диалоги

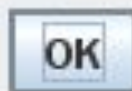


Ю

Message



Пример для `JOptionPane.showMessageDialog`



Диалоги



Input



Пример для JOptionPane.showInputDialog
Введите текст и нажмите кнопку

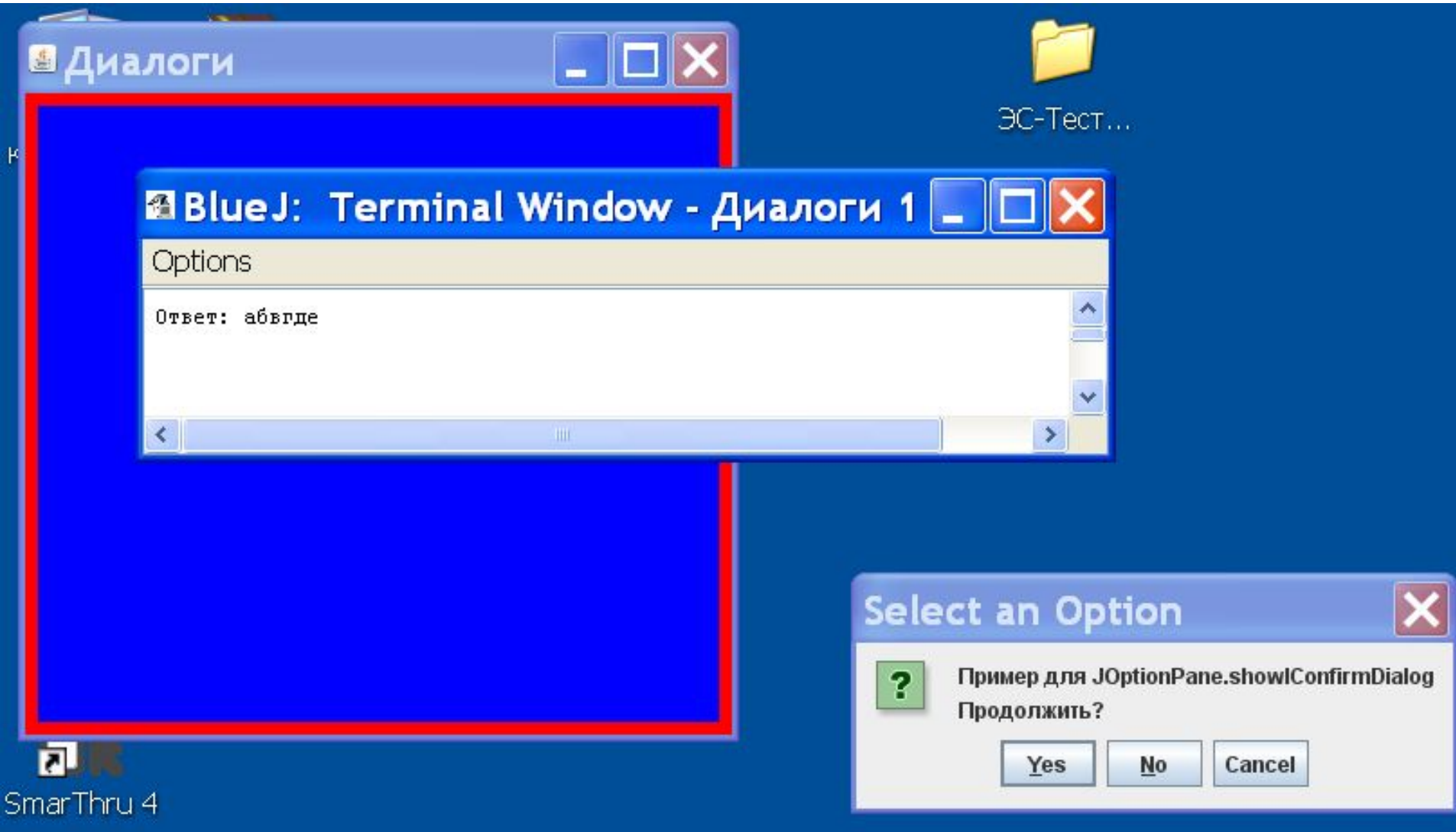
абвгде

OK

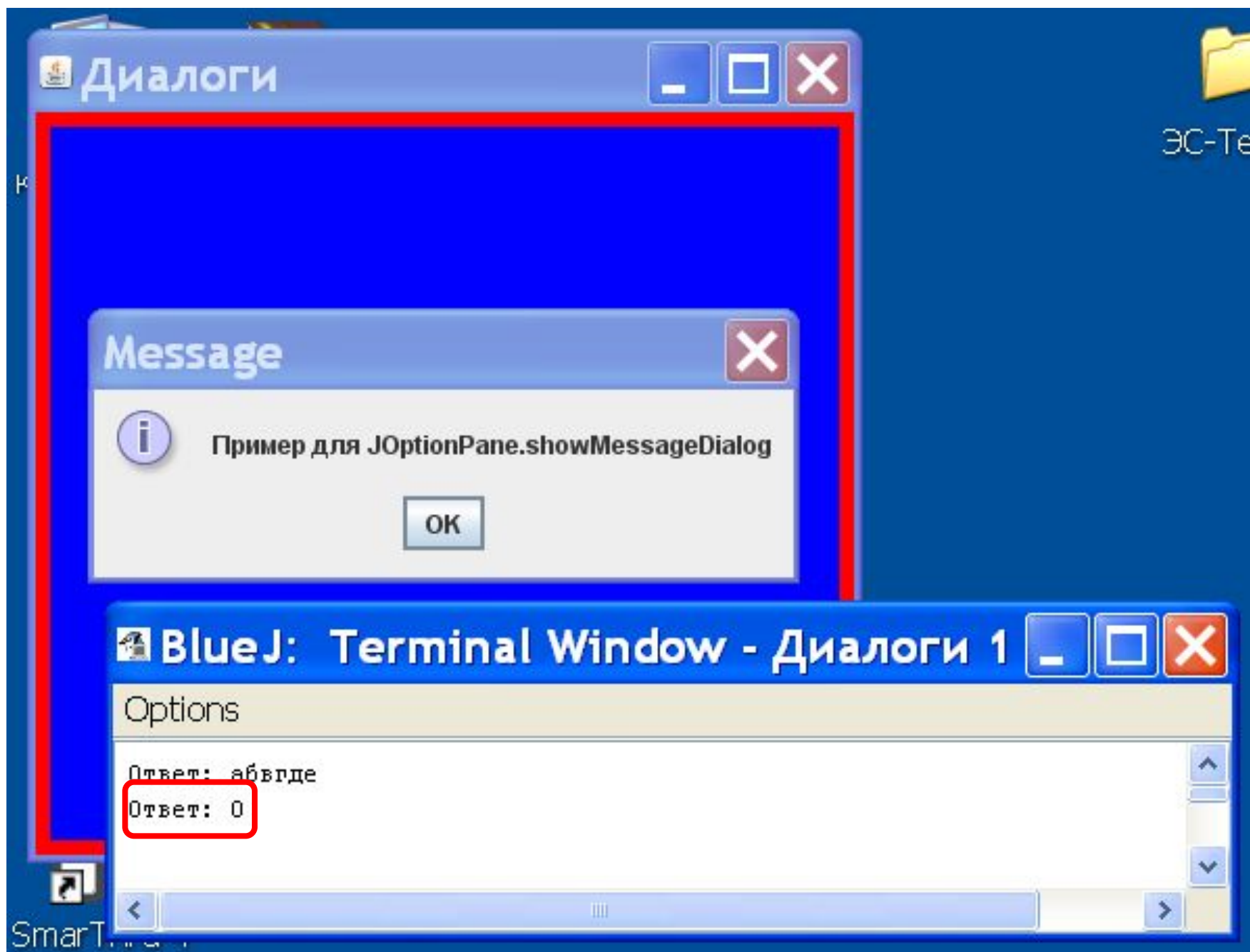
Cancel

После
нажатия
кнопки
<OK> в
предыдущем
диалоге

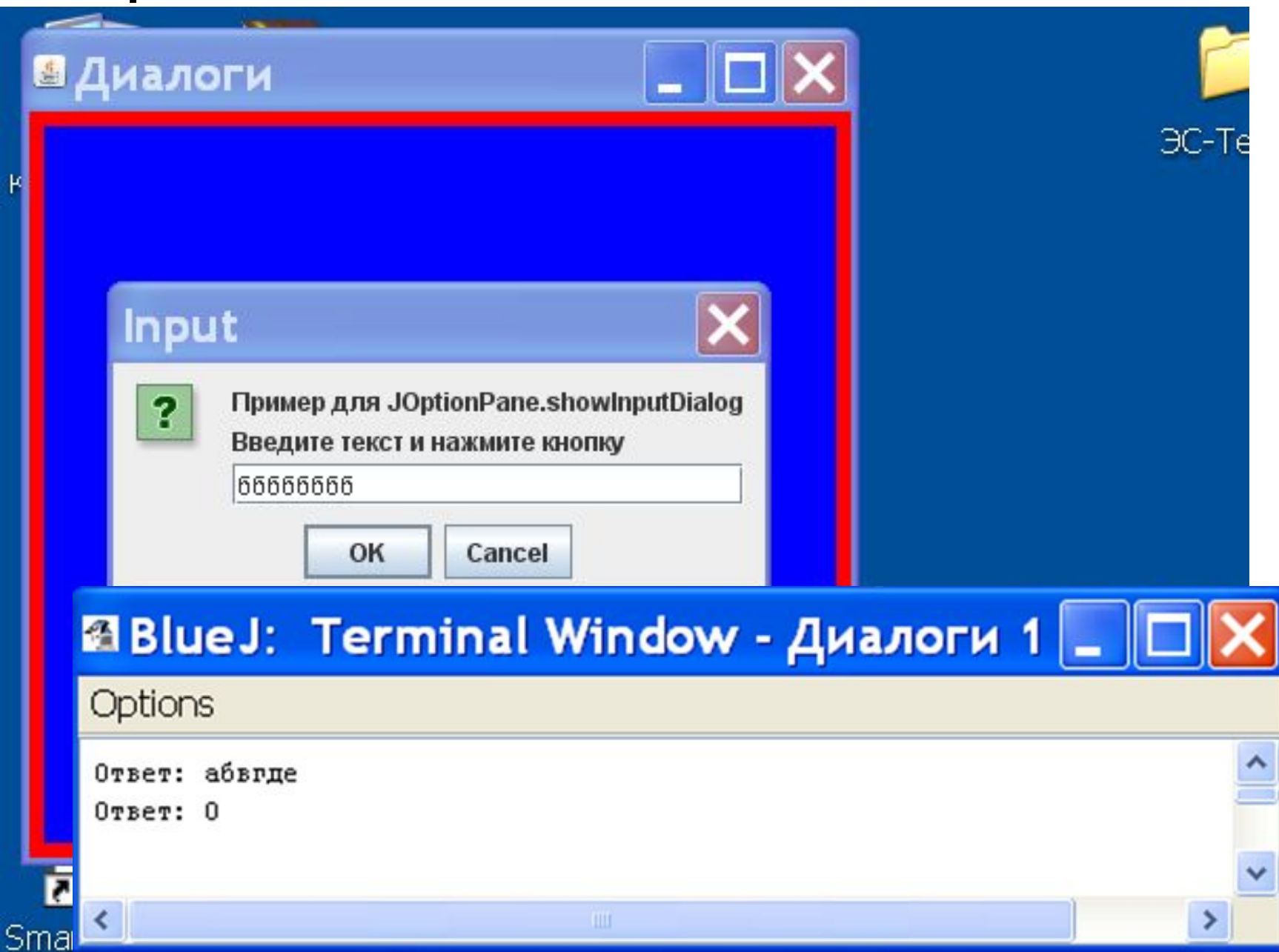
После нажатия кнопки <OK> в предыдущем диалоге



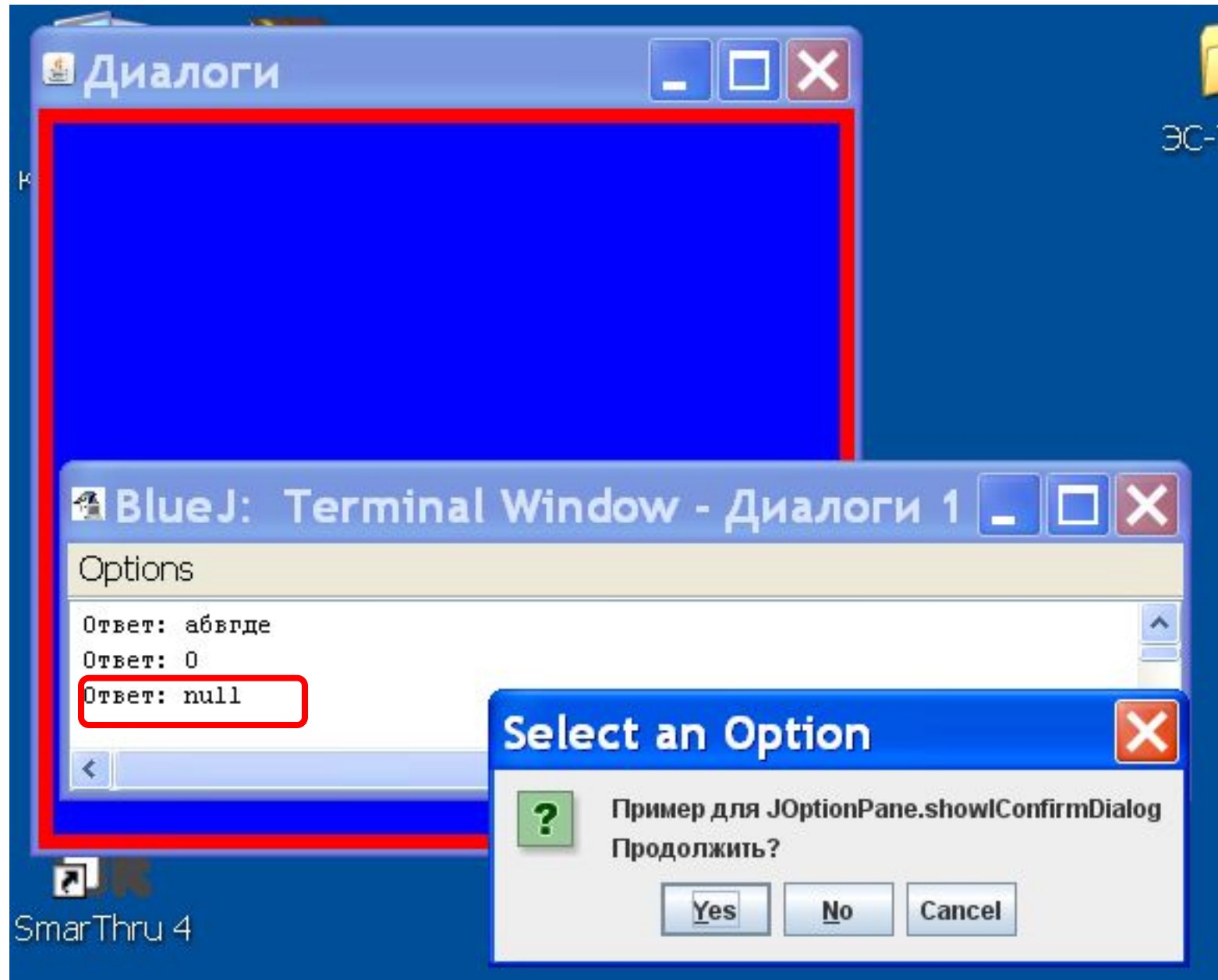
После нажатия кнопки <Yes> в предыдущем диалоге



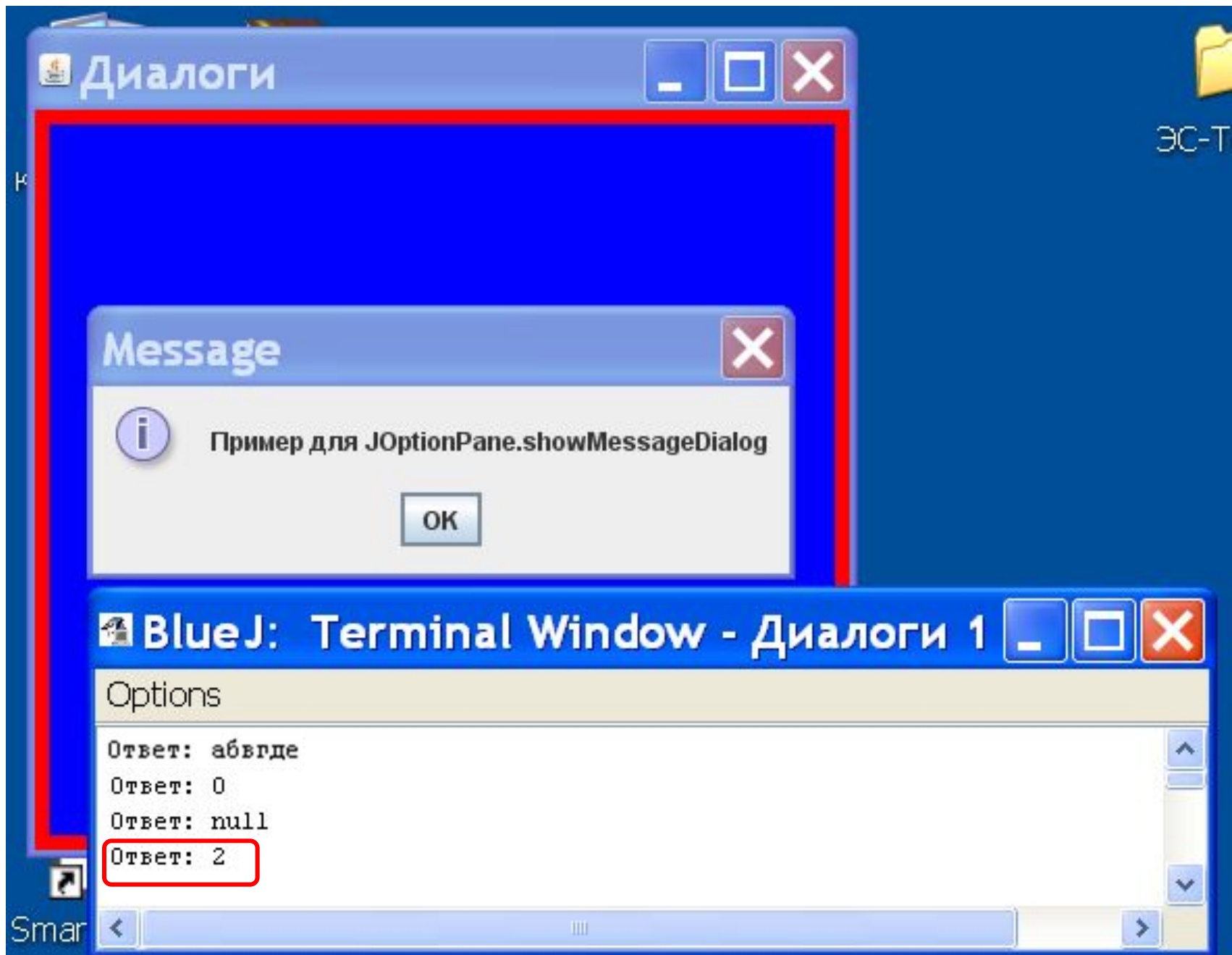
После нажатия кнопки <ОК> в предыдущем диалоге и набора текста сообщения



После нажатия кнопки <Cancel> в предыдущем диалоге

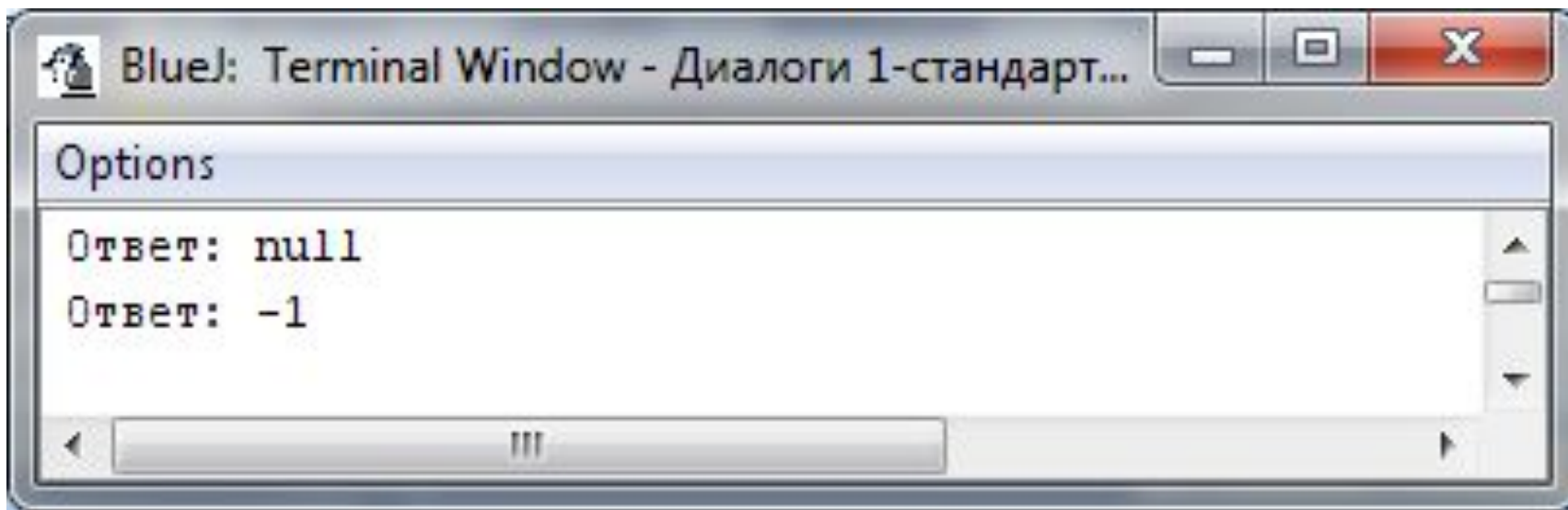


После нажатия кнопки <Cancel> в предыдущем диалоге



Приложение запущено заново и во всех трех диалоговых окнах был нажат значок «заккрыть».

Результат одного прохода цикла:



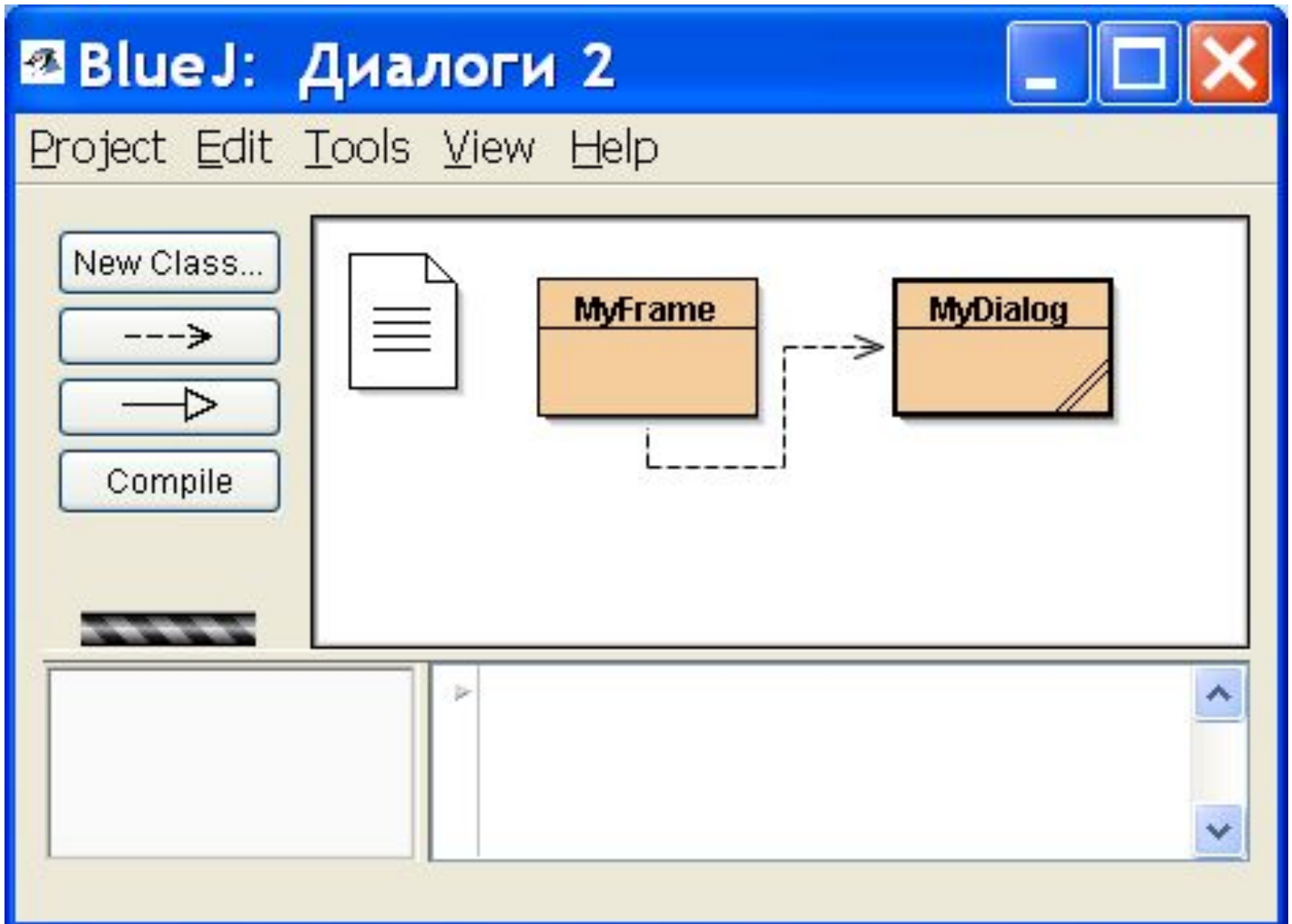
Пояснения к программе 1

- Для создания окна диалога используется "главный" контейнер – окно приложения
- Диалоги – модальные, на экране появляются в заданном порядке
- Для получения информации от окна диалога не нужны обработчики

Диалоги

- JOptionPane – Стандартные диалоги
- JDialog – класс для создания диалогов пользователя
- JColorChooser – диалог для выбора цвета
- JFileChooser – диалог для выбора файла

Пример 2 – создание собственного диалога



```
import java.awt.*; //Проект - Диалоги 2-собственный  
import javax.swing.*;  
import java.awt.event.*;
```

```
class MyDialog extends JDialog  
    implements ActionListener{
```

```
public MyDialog (JFrame parent){ //конструктор  
    // вызов конструктора базового класса  
    super (parent, "Мой диалог", true);  
    //родитель, название, true - модальный диалог  
    Container cp = getContentPane();  
    cp.setLayout(new FlowLayout());  
    cp.add (new JLabel("Мой диалог"));
```

```
JButton ok = new JButton ("OK");  
ok.addActionListener(this); //слушаем
```

```
JButton cancel = new JButton ("Отмена");  
cancel.addActionListener(this); //слушаем
```

```
cp.add(ok); cp.add(cancel);
```

```
setSize(250,125);
```

```
setLocation(50,100);
```

```
setDefaultCloseOperation(  
    WindowConstants.DO_NOTHING_ON_CLOSE);
```

```
//не будет закрываться при нажатии на X
```

```
} //конструктор
```

```
public void actionPerformed (ActionEvent e){
```

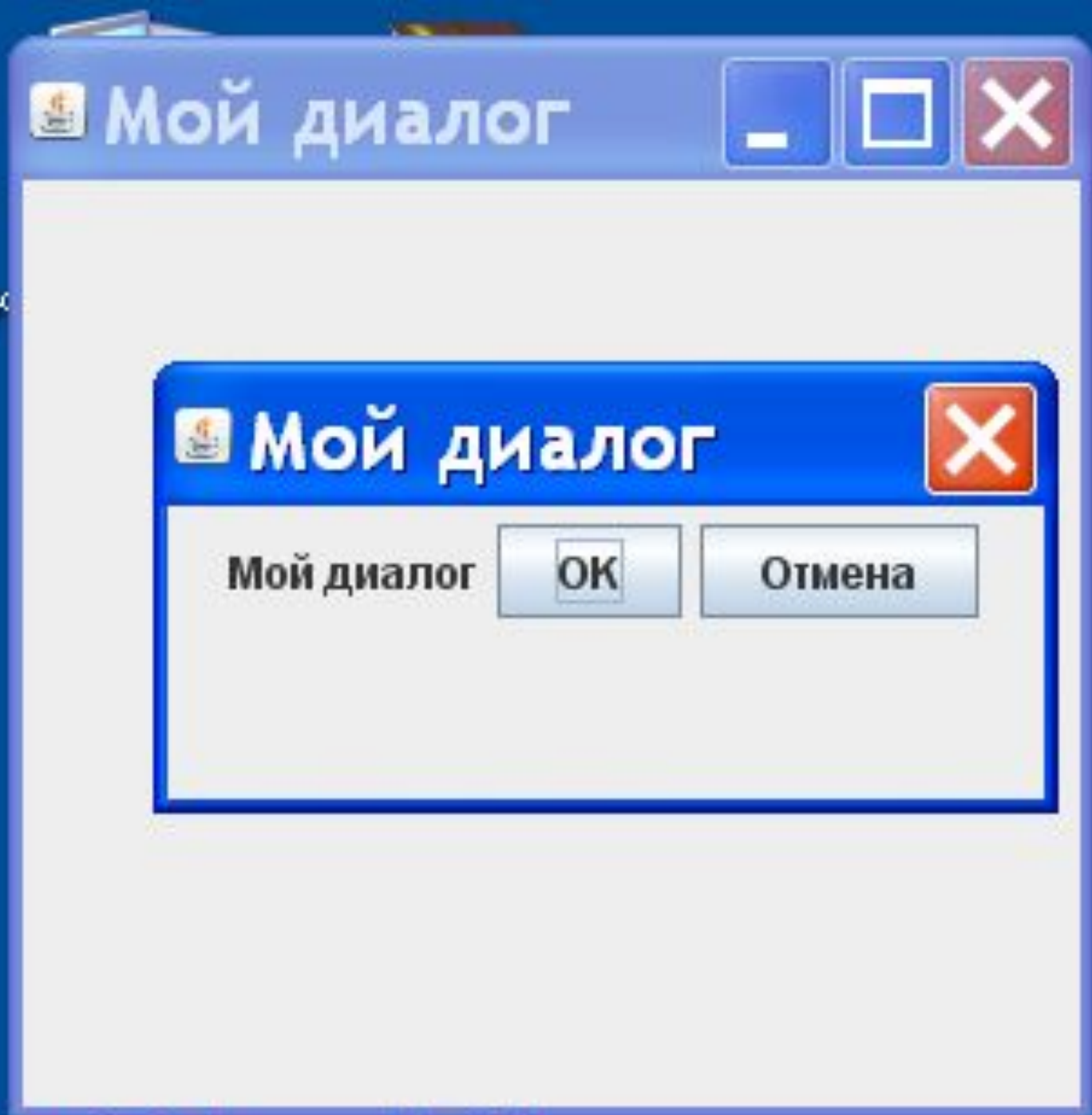
```
//обработчик событий
```

```
System.out.println (e);  dispose();  } }
```

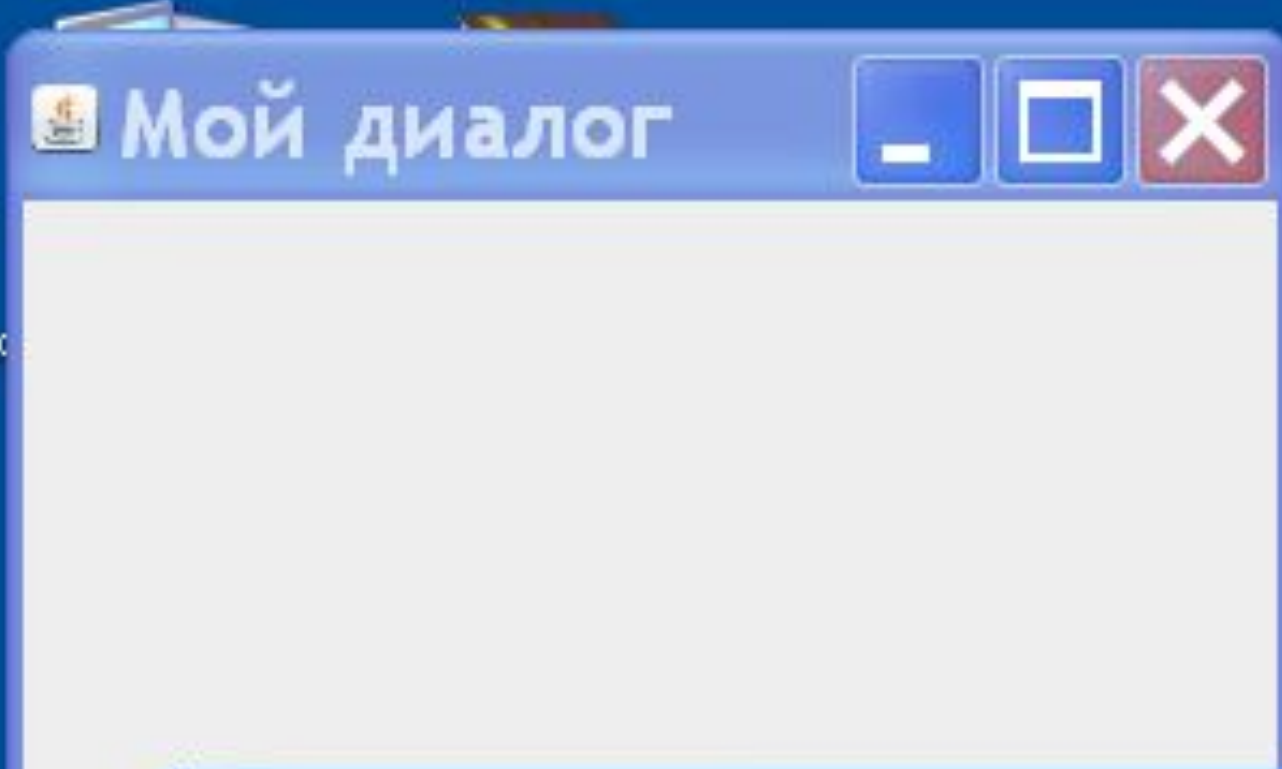


```
import java.awt.*;
import javax.swing.*;
public class MyFrame{
    private static void createAndShowGUI(){
        JFrame frame = new JFrame("Мой диалог");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,400);
        frame.setLocation(10,10);
        frame.setVisible(true);
        MyDialog md = new MyDialog (frame);
        md.setVisible(true);
    }
    public static void main (String[ ] args){
        javax.swing.SwingUtilities.invokeLater(new Runnable(){
            public void run(){createAndShowGUI();});
        }
    }
}
```

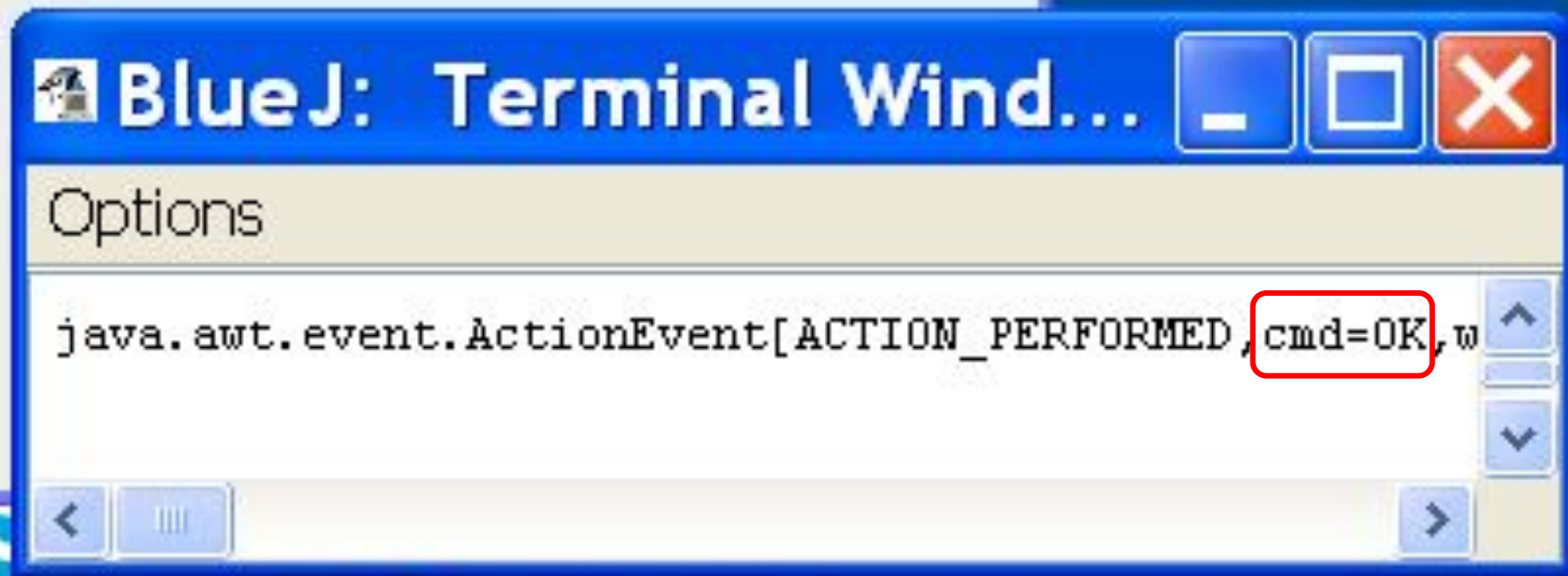
Т



После
запуска.



После нажатия
кнопки <OK>



После нажатия
кнопки
<Отмена>

Мой диалог



BlueJ: Terminal Windo...



Options

```
java.awt.event.ActionEvent[ACTION_PERFORMED, cmd=Отмена,
```



Skype

1ver 0

Пример 2 – с классом MyDialog

- Окно диалога ведет себя как обычное окно
- Для простоты обработчик сделан простым – только регистрация
- В реальных случаях надо анализировать команды, связанные с кнопками, и выполнять их

JDialog – полезные методы (1)

- public void

setDefaultCloseOperation(int operation)

- Задаёт действие при закрытии диалога пользователем. Значения:
 - DO_NOTHING_ON_CLOSE /из WindowConstants/ Ничего не делать; программа должна предусмотреть обработку в методе windowClosing зарегистрированного слушателя WindowListener.
 - HIDE_ON_CLOSE /из WindowConstants/): Автоматически прячет окно после вызова зарегистрированного слушателя
 - DISPOSE_ON_CLOSE /из WindowConstants/): Автоматически прячет и уничтожает диалог после вызова зарегистрированного слушателя

JDialog – полезные методы (2)

- **setJMenuBar(JMenuBar menu)**
устанавливает полосу меню в окне диалога
- **setLayout(LayoutManager manager)** –
задает менеджер расположения
- **getContentPane()** – панель контента для диалога
- **setContentPane(Container contentPane)**
задает панель контента

JDialog – полезные методы (3)

- `public Graphics getGraphics()`
получает графический контекст для окна диалога
- `repaint(long time, int x, int y, int width, int height)` перерисовать указанный прямоугольник в окне диалога

Диалог JColorChooser

- создает панель с элементами управления для выбора цвета.

3 уровня программного интерфейса

- статический метод с модальным диалогом, возвращает выбранный цвет
- статический метод с передачей управления обработчикам actionPerformed
- создание объекта вне контейнера, с обработкой события в слушателе PropertyChanged при изменении свойства 'color'

Статические методы:

**public static Color showDialog(Component component,
String title,
Color initialColor)
throws HeadlessException**

Показывает модальный диалог выбора цвета. Если пользователь нажимает кнопку "ОК", то диалог возвращает выбранный цвет и закрывается. Если пользователь нажимает кнопку "Cancel" или закрывает диалог, не нажимая "ОК", то метод скрывает диалог и возвращает пустой указатель.

Параметры:

component - родительский компонент для диалога

title - строка, содержащая название диалога

initialColor - начальный цвет

Исключение класса **HeadlessException** выдается, когда код, зависящий от клавиатуры, дисплея или мыши, вызывается в среде, которая не поддерживает клавиатуру, дисплей или мышь.

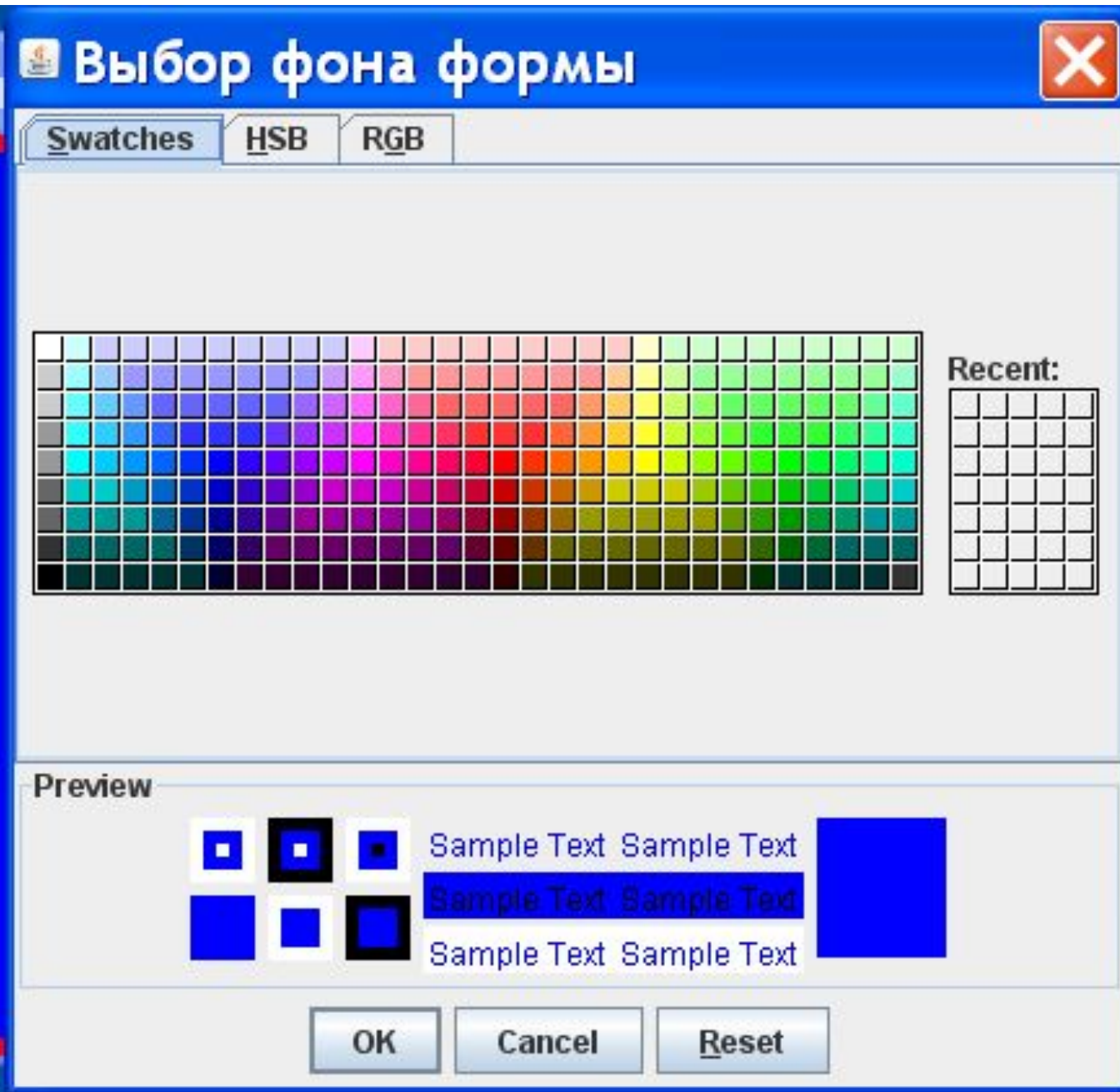
Метод демонстрируется в примере 3

`import java.awt.*; //Проект - 3 JColorChooser static method-showDialog`
`import javax.swing.*;`

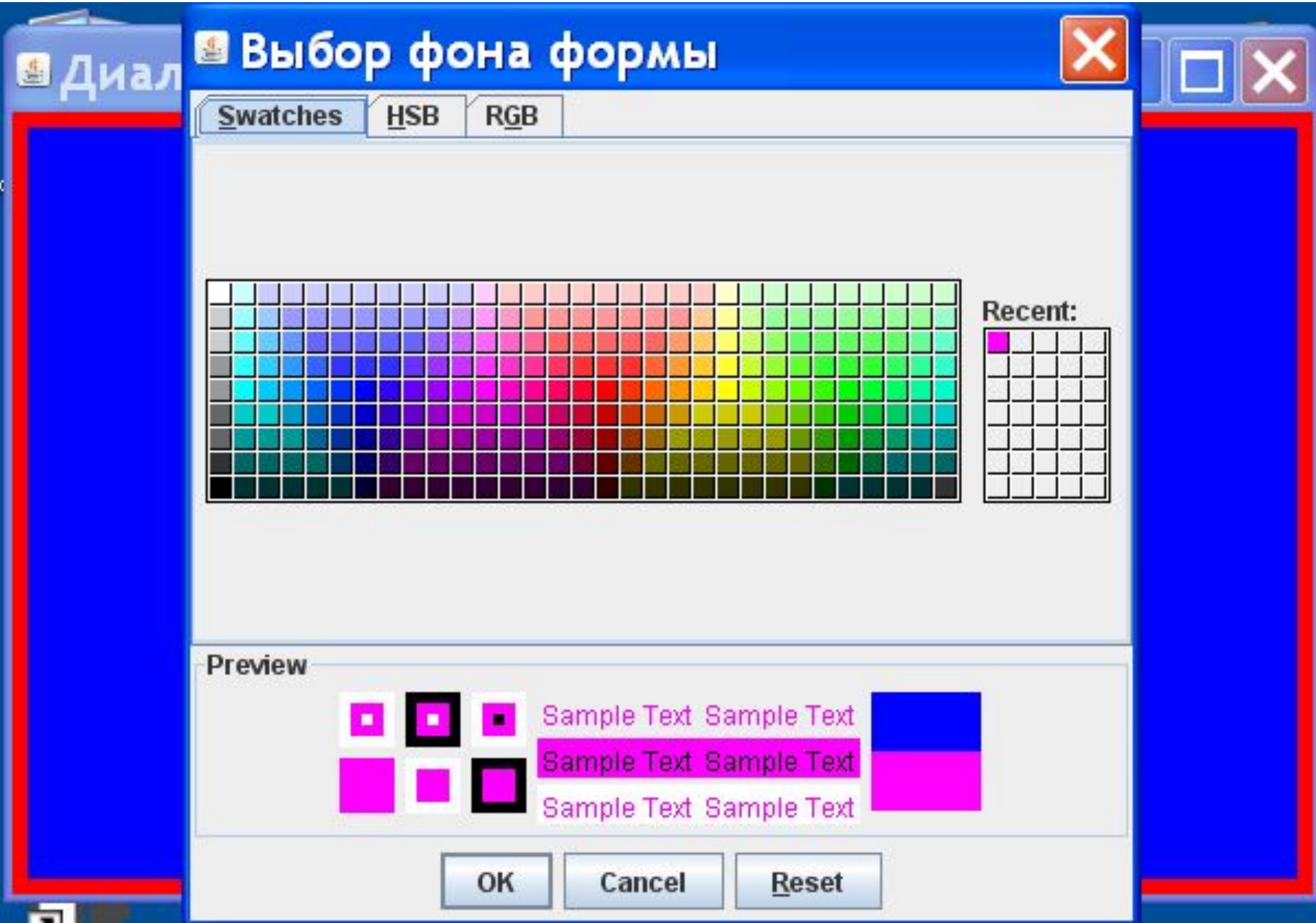
Пример 3

```
public class ColorDialogDemo extends JComponent{
    private static void createAndShowGUI(){
        JFrame frame = new JFrame("Диалоги");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel content = new JPanel();
        frame.setContentPane(content); //можно так или как раньше
        content.setBackground(Color.BLUE);
        content.setBorder (
            BorderFactory.createLineBorder(Color.RED,7));
        frame.setSize(600,400);
        frame.setLocation(10,10);
        frame.setVisible(true);
        Color rez = JColorChooser.showDialog (frame,
            "Выбор фона формы", Color.BLUE);
        content.setBackground(rez);
    }
    public static void main (String[ ] args){
        javax.swing.SwingUtilities.invokeLater(new Runnable(){
            public void run(){createAndShowGUI();});
    }
}
```

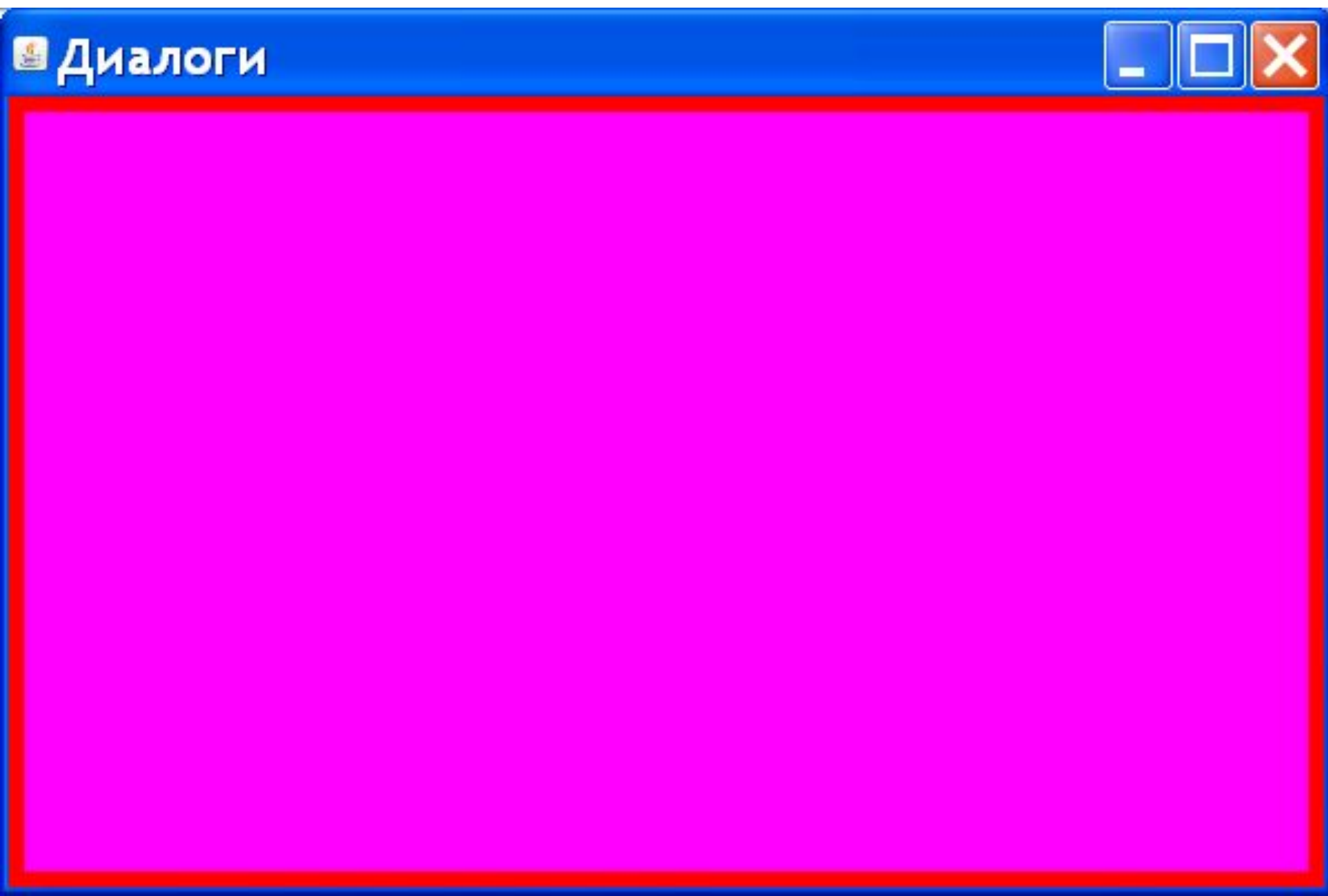
После запуска



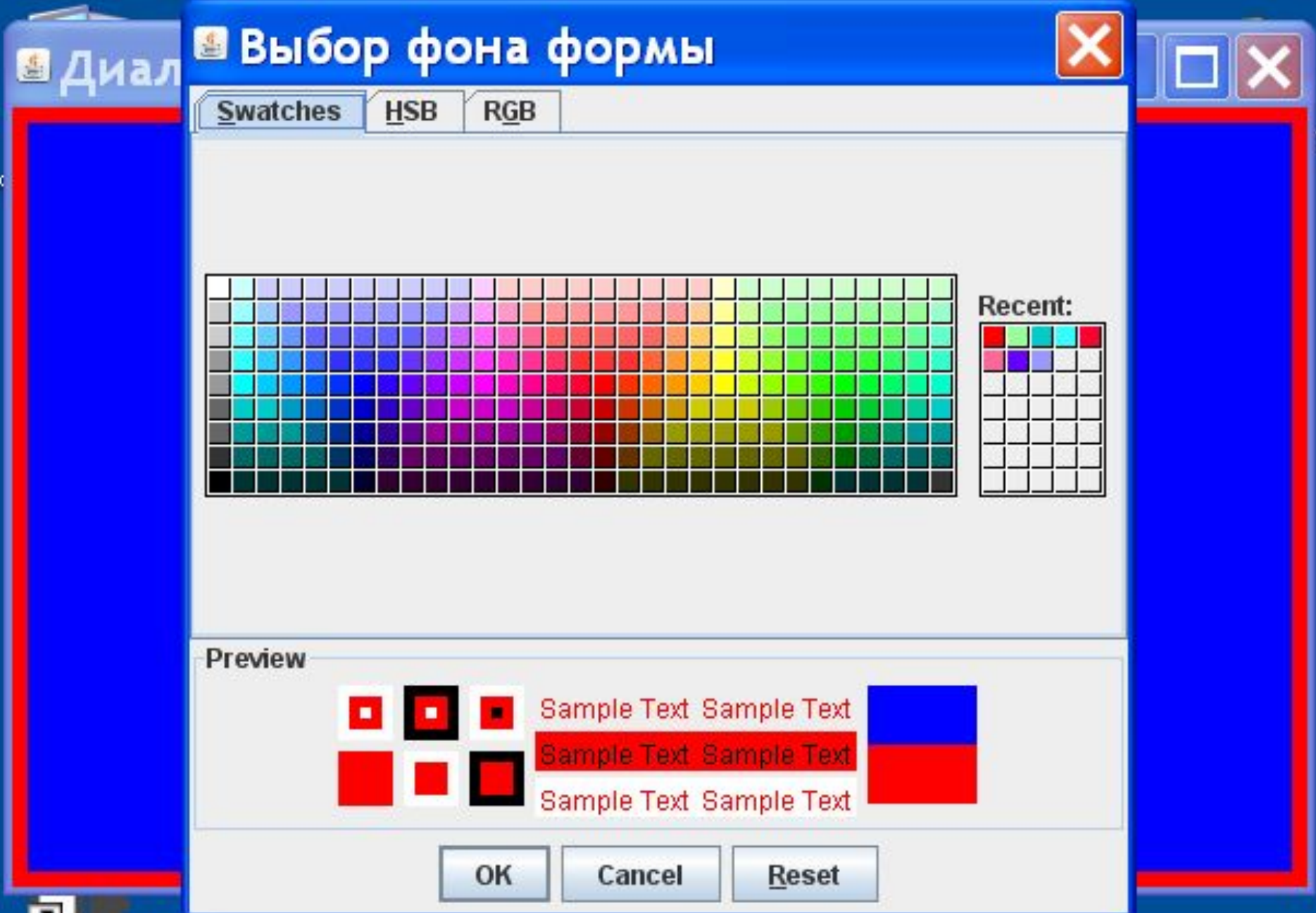
Выбран новый цвет



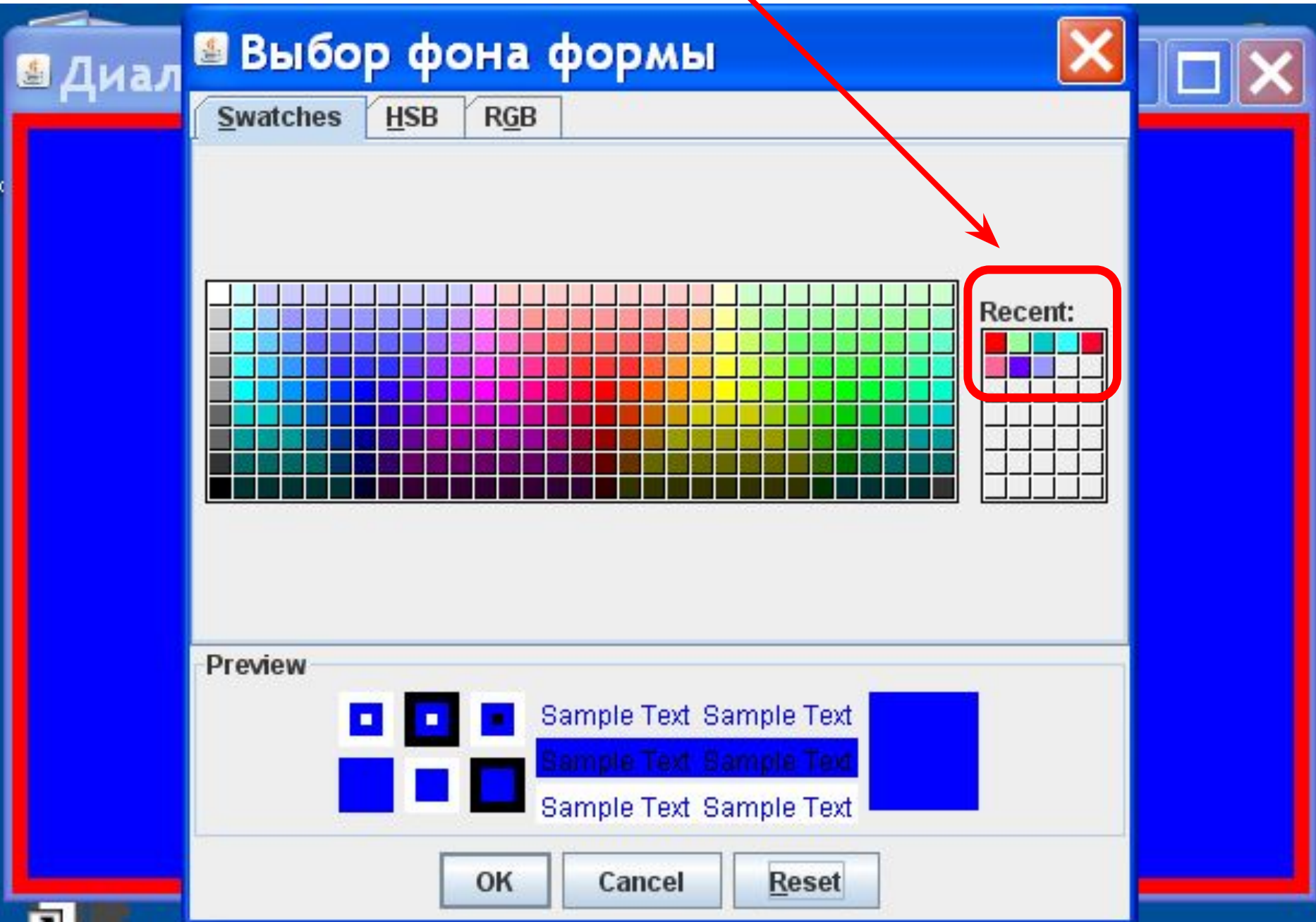
После нажатия кнопки <ОК>



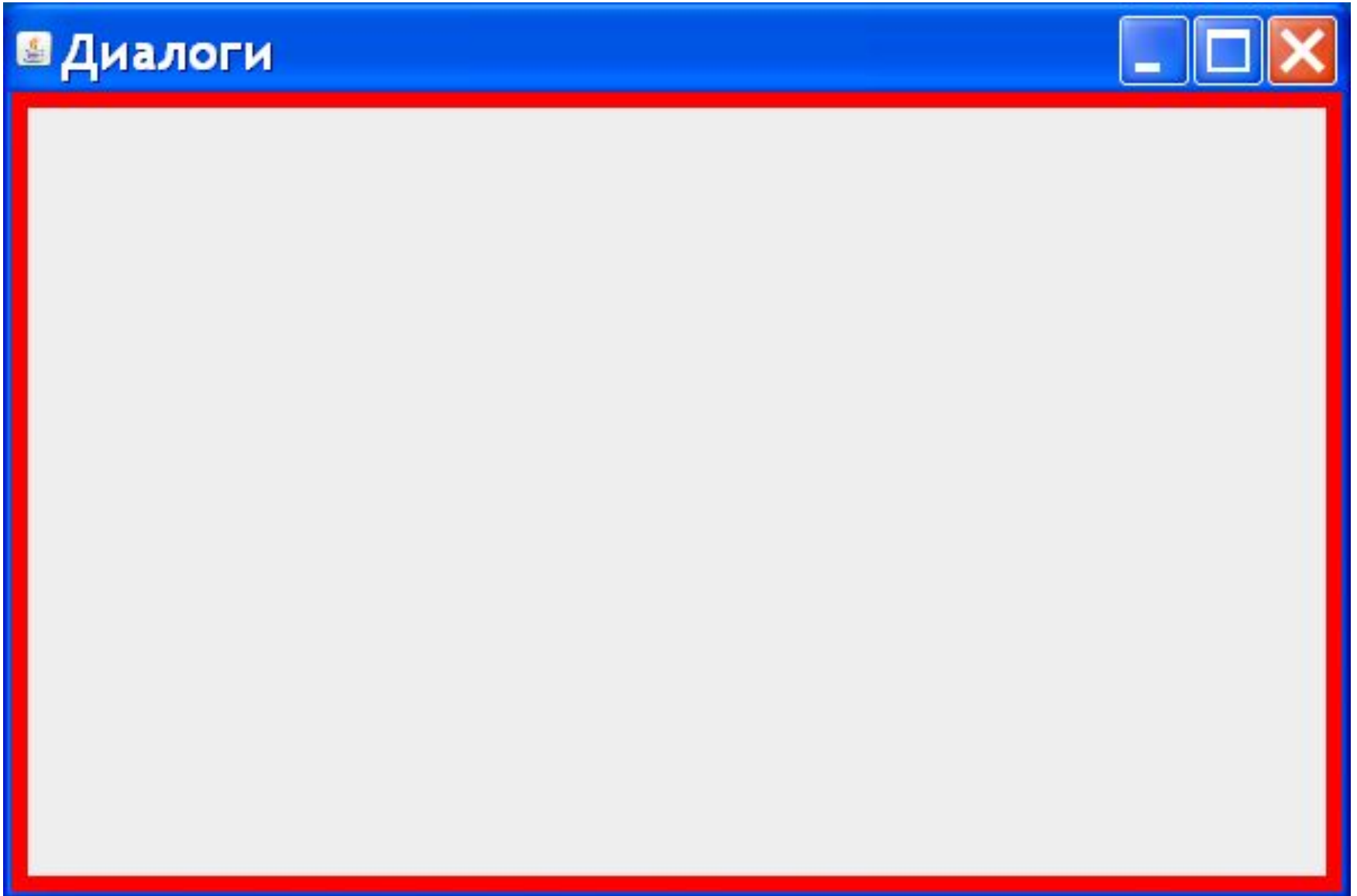
Пользователь тренировался в выборе цветов (видно на панели Recent)



А потом нажал кнопку <Reset> и получил :



После нажатия кнопки <Cancel> цвет фона изменился на цвет по умолчанию, а нам нужно, чтобы он остался первоначальным (синим). Что делать?



Статические методы:

```
public static JDialog createDialog(Component c,  
    String title,  
    boolean modal,  
    JColorChooser chooserPane,  
    ActionListener okListener,  
    ActionListener cancelListener)  
    throws HeadlessException
```

Создает и возвращает новый диалог, содержащий панель с указанным объектом класса `JColorChooser` и кнопками "OK", "Cancel", и "Reset". Если кнопки "OK" или "Cancel" нажаты, диалог автоматически скрывается (но не разрушается). Если кнопка "Reset" будет нажата, то цвет вернется к предыдущему выбранному, и диалог останется видимым.

Параметры:

`c` - родительский компонент для диалога

`title` - название для диалога

`modal` - булево (`true` - диалог модальный)

`chooserPane` - объект класса `JColorChooser`, который будет помещен в диалоге

`okListener` - `ActionListener`-слушатель нажатия кнопки "OK"

`cancelListener` - `ActionListener`-слушатель нажатия кнопки "Cancel»

Метод демонстрируется в примере 4

// Проект - 4 JColorChooser static method-createDialog and show-со слушанием

```
import java.awt.*;  
import javax.swing.*;  
import java.awt.event.*;
```

```
public class ColorDialogDemo extends JFrame  
    implements ActionListener{
```

```
    JPanel content;
```

```
    JColorChooser jcc;
```

```
    public ColorDialogDemo(){ //конструктор
```

```
        super ("Демонстрация JColorChooser.createDialog");
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        content = new JPanel();
```

```
        setContentPane(content);
```

```
        content.setBackground(Color.BLUE);
```

```
        content.setBorder (
```

```
            BorderFactory.createLineBorder(Color.RED,7));
```

```
        setSize(600,400);
```

```
        setLocation(10,10);
```

```
        setVisible(true);
```

```
jcc = new JColorChooser (content.getBackground());
```

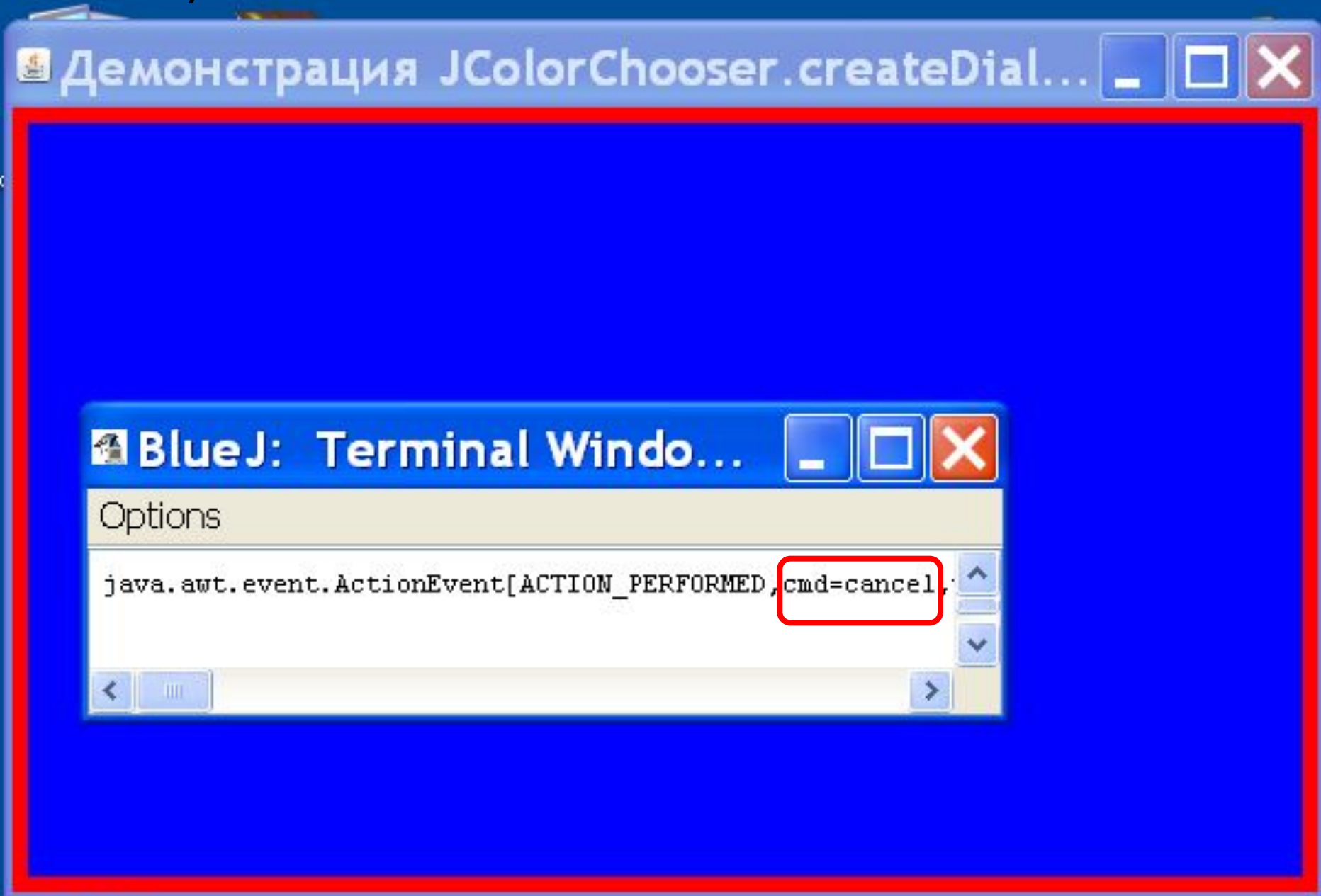
```
JDialog d = JColorChooser.createDialog (this,  
    "Выбор фона формы", true, jcc, this, this);  
d.show();
```

```
}  
public void actionPerformed (ActionEvent e){ // обработчик  
    //чтобы при нажатии "Cancel" цвет не менялся  
    System.out.println(e);  
    if("OK".equals(e.getActionCommand()))  
        content.setBackground(jcc.getColor());  
    //фон меняется только при нажатии <OK>  
}
```

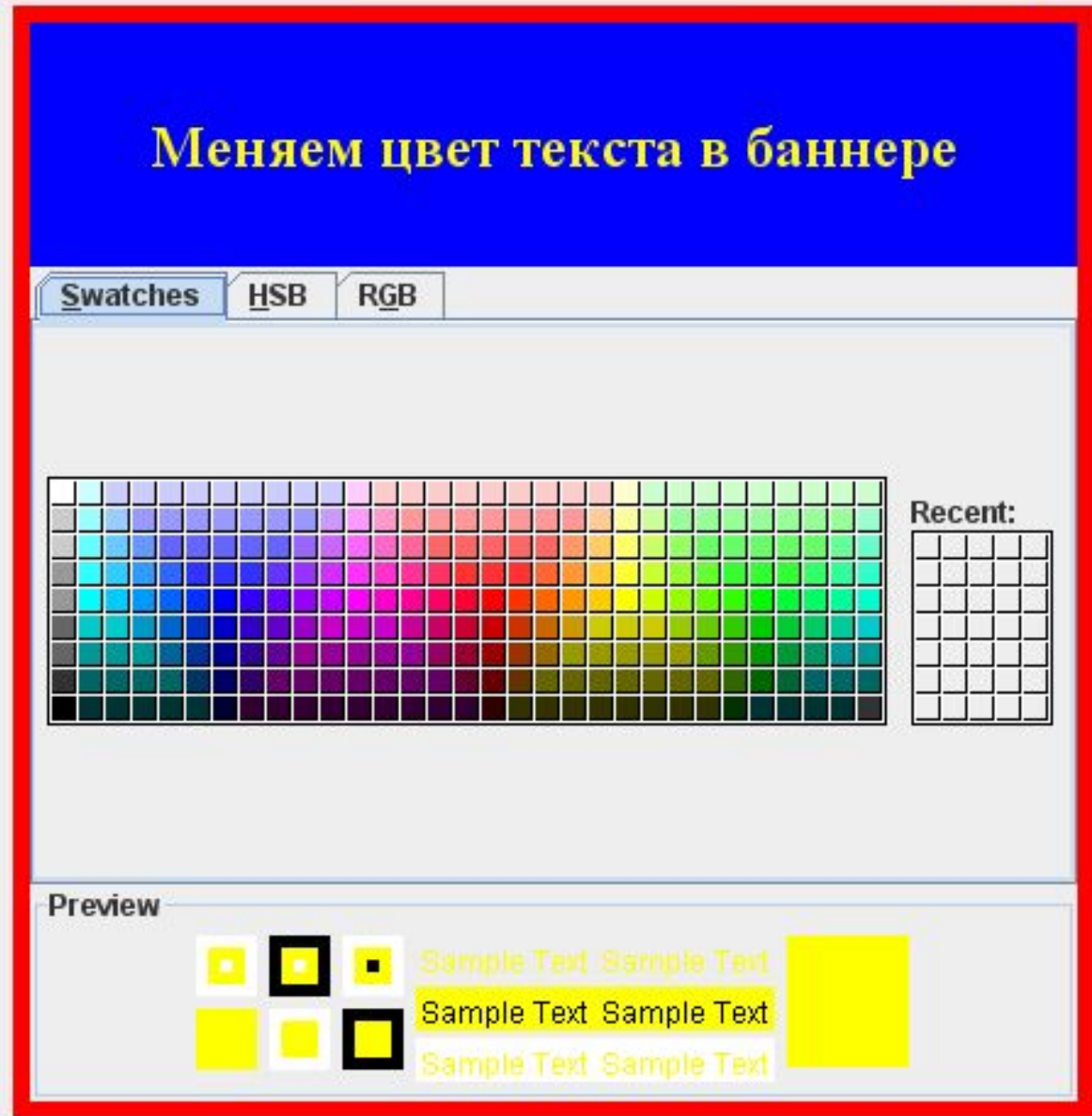
//запускающий класс

```
import javax.swing.*;  
public class Main_class {  
    public static void main (String[] args){  
        javax.swing.SwingUtilities.invokeLater(new Runnable(){  
            public void run(){ ColorDialogDemo cdm =  
                                new ColorDialogDemo();}});  
    }  
}
```

После запуска и нажатия клавиши <Cancel> (цвет теперь не меняется)



Пример 5.
Стационарное
размещение
компонента
JColorChooser
на панели,
которая будет
слушать его
событие
изменения цвета
(без кнопок).



//Проект - 5 JColorChooser со слушателем ChangeListener

import java.awt.*;

import javax.swing.*;

import javax.swing.event.*;

public class ColorDialog extends JPanel

implements ChangeListener {

JColorChooser jcc;

JLabel banner;

public ColorDialog(){ *//конструктор*

setLayout (new BorderLayout());

setBorder (

BorderFactory.createLineBorder(Color.RED,7));


```
banner = new JLabel("Меняем цвет текста в баннере",  
                    JLabel.CENTER);
```

```
banner.setPreferredSize(new Dimension(100,100));
```

```
banner.setForeground(Color.yellow);
```

```
banner.setBackground(Color.blue);
```

```
banner.setOpaque(true);
```

```
banner.setFont(new Font ("Serif", Font.BOLD, 24));
```

```
jcc = new JColorChooser(banner.getForeground());
```

```
jcc.getSelectionModel().addChangeListener(this);
```

```
add(jcc, BorderLayout.CENTER);
```

```
add(banner, BorderLayout.NORTH);
```

```
}
```

```
public void stateChanged(ChangeEvent e) { // обработчик
```

```
    Color newColor = jcc.getColor();
```

```
    banner.setForeground(newColor);
```

```
}
```

```
}
```

```
import java.awt.*;
import javax.swing.*;
```

```
public class MyFrame{
    private static void createAndShowGUI(){
        JFrame frame = new JFrame("Диалог JColorChooser");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel content = new JPanel();
        frame.setContentPane(content);
        ColorDialog p = new ColorDialog();
        content.add(p); // добавляем как обычный визуальный
                       // компонент
        frame.setSize(550,550);
        frame.setLocation(10,10);
        frame.setVisible(true);
    }
    public static void main (String[ ] args){
        javax.swing.SwingUtilities.invokeLater(new Runnable(){
            public void run(){createAndShowGUI();});
        });
    }
}
```

После
запуска

Меняем цвет текста в баннере

Swatches

HSB

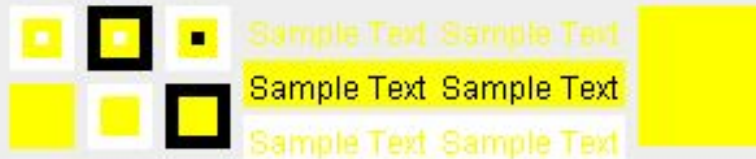
RGB



Recent:



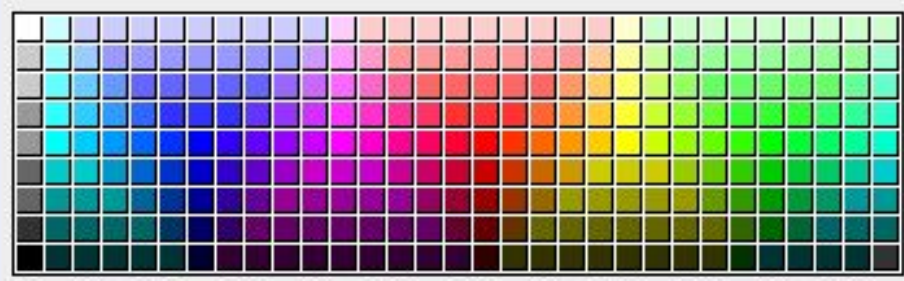
Preview



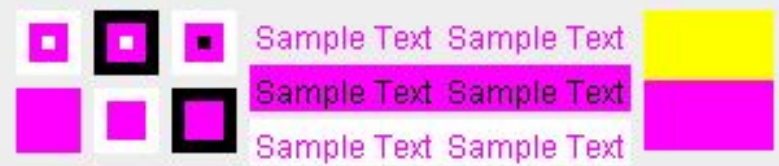
После
выбора
цвета

Меняем цвет текста в баннере

Swatches HSB RGB



Preview



В следующем примере мы будем использовать анонимный внутренний класс для реализации слушателя событий.

Это удобно, когда событие одного типа может прийти от нескольких источников.

Преимущество: каждое событие слушает его собственный слушатель со своим собственным обработчиком (программисту не надо определять в программе от какого источника пришло событие – не надо лишних if).

Пример 6. То же, но **с анонимным слушателем**.

//Проект - 6 JColorChooser с анонимным слушателем

ChangeListener

import java.awt.*;

import javax.swing.*;

import javax.swing.event.*;

**public class DialogDemo extends JComponent { *//не включаем*
*//интерфейс ChangeListener***

static JColorChooser jcc;

static JLabel banner;

private static void createAndShowGUI(){

JFrame frame = new JFrame("Диалог JColorChooser");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

JPanel content = new JPanel();

frame.setContentPane(content);

content.setLayout (new BorderLayout());

content.setBorder (

BorderFactory.createLineBorder(Color.RED,7));

```
banner = new JLabel("Меняем цвет текста в баннере",
                    JLabel.CENTER);
banner.setPreferredSize(new Dimension(100,100));
banner.setForeground(Color.yellow);
banner.setBackground(Color.blue);
banner.setOpaque(true);
banner.setFont(new Font ("Serif",Font.BOLD,24));

jcc = new JColorChooser(banner.getForeground());

jcc.getSelectionModel().addChangeListener(
    new ChangeListener() { //анонимный слушатель
        public void stateChanged(ChangeEvent e) {
            Color newColor = jcc.getColor();
            banner.setForeground(newColor);
        }
    }
);
```

```
content.add(jcc, BorderLayout.CENTER);
content.add(banner, BorderLayout.NORTH);
frame.setSize(500,400);
frame.setLocation(10,10);
frame.setVisible(true);
```

```
}
```

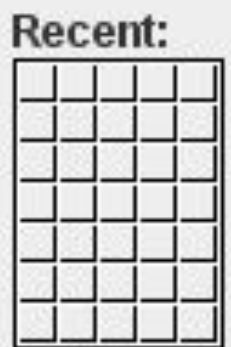
```
public static void main (String[ ] args){
    javax.swing.SwingUtilities.invokeLater(new Runnable(){
        public void run(){createAndShowGUI();});
```

```
}
```

```
}
```


Меняем цвет в баннере

Swatches HSB RGB



Preview

			Sample Text	Sample Text	
			Sample Text	Sample Text	
			Sample Text	Sample Text	

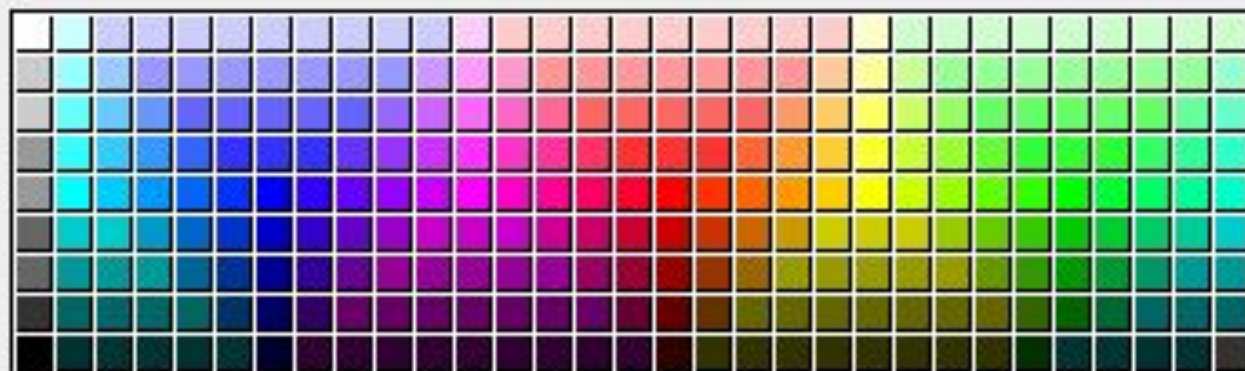


Меняем цвет в баннере

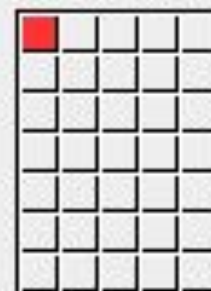
Swatches

HSB

RGB



Recent:



Preview



Sample Text Sample Text

Sample Text Sample Text

Sample Text Sample Text



Пример 7. Стационарное размещение компонента JColorChooser и кнопки JButton на панели, которая будет слушать событие кнопки.

//Проект - 7 JColorChooser с кнопкой и слушателем ActionListener

```
import java.awt.*;  
import javax.swing.*;  
import java.awt.event.*;  
import javax.swing.event.*;  
  
public class ColorDialog extends JPanel  
implements ActionListener {  
  
    JColorChooser jcc;  
    JLabel banner;  
    JButton b1;  
public ColorDialog(){  
    setLayout (new BorderLayout());  
    setBorder (  
        BorderFactory.createLineBorder(Color.RED,7));
```

```
banner = new JLabel("Меняем цвет текста в баннере",  
                    JLabel.CENTER);  
banner.setPreferredSize(new Dimension(100,100));  
banner.setForeground(Color.yellow);  
banner.setBackground(Color.blue);  
banner.setOpaque(true);  
banner.setFont(new Font ("Serif",Font.BOLD,24));  
jcc = new JColorChooser(banner.getForeground());  
b1 = new JButton ("УСТАНОВИТЬ ВЫБРАННЫЙ ЦВЕТ");  
b1.addActionListener(this);  
add (jcc, BorderLayout.CENTER);  
add (banner, BorderLayout.NORTH);  
add (b1, BorderLayout.SOUTH);  
}
```

```
public void actionPerformed (ActionEvent e){  
    Color newColor = jcc.getColor();  
    banner.setForeground(newColor);  
}
```

```
}
```

```
import java.awt.*;
import javax.swing.*;
public class MyFrame{
    private static void createAndShowGUI(){
        JFrame frame = new JFrame("Диалог JColorChooser");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel content = new JPanel();
        frame.setContentPane(content);
        ColorDialog p = new ColorDialog();
        content.add(p);
        frame.setSize(550,550);
        frame.setLocation(10,10);
        frame.setVisible(true);
    }
    public static void main (String[ ] args){
        javax.swing.SwingUtilities.invokeLater(new Runnable(){
            public void run(){createAndShowGUI();});
    }
}
```



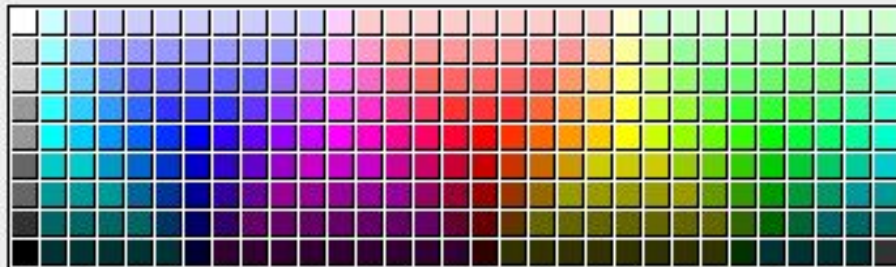
При выборе цвета, цвет текста в баннере не изменяется

Меняем цвет текста в баннере

Swatches

HSB

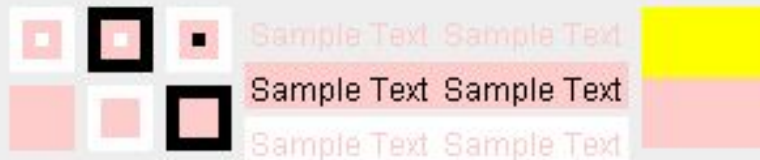
RGB



Recent:



Preview

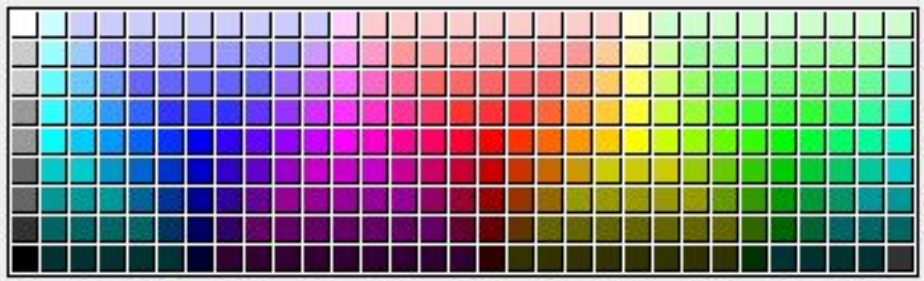


Установить выбранный цвет

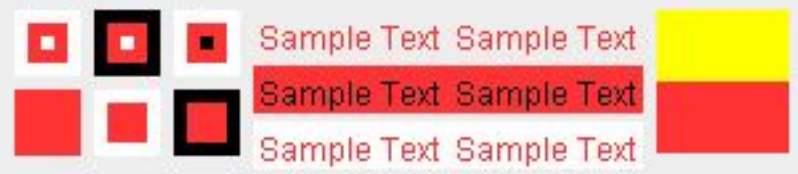
Установка
выбранного
цвета текста в
баннере
осуществляется
при нажатии
кнопки
«Установить
выбранный
цвет»

Меняем цвет текста в баннере

Swatches HSB RGB



Preview



Установить выбранный цвет

Контейнер JFileChooser

Контейнер JFileChooser — это модальное окно с владельцем типа Frame, содержащее стандартное окно выбора файла для открытия (Open) или сохранения (Save). В классе три конструктора:

JFileChooser () — создает окно для открытия файла; устанавливается директория пользователя по умолчанию

JFileChooser (File текущий каталог) — создает окно выбора файла;

Как и JColorChooser, может работать без слушателей событий и со слушателями.

Контейнер JFileChooser (2)

Одна из кнопок окна диалога называется ApproveButton (Open для открытия, Save для сохранения, можно дать свое название)

getApproveButtonText дает текст этой кнопки, т.е. *команды*

setApproveButtonText задает текст этой кнопки

Можно установить начальный каталог для поиска файла методом setCurrentDirectory(File dir)

getSelectedFile() дает объект типа File, соответствующий выбранному файлу

setDialogTitle(String title)

showDialog(Component parent, String ApproveButtonText)

showOpenDialog(Component parent) диалог для "открытия" файла

showSaveDialog(Component parent) диалог для "сохранения" файла



Открытие файла



Look in: Локальный диск (C:)



- | | |
|------------------------|---------------------------------|
| 09-05-2015_16-33-42 | 09-05-2015_16-33-42.zip |
| BlueJ | bar.emf |
| Documents and Settings | WPI_Log_2014.12.25_17.55.05.txt |
| Install | |
| OpenOfficePortable | |
| Program Files | |
| WINDOWS | |

File Name: bar.emf

Files of Type: All Files

Open

Cancel

Пример 8. Применение JFileChooser

// Проект - 8 JFileChooser - использование

import java.io.*; // на будущее

import java.awt.*;

import javax.swing.*;

import java.awt.event.*;

import javax.swing.event.*;

public class MyFrame implements ActionListener{

public JFrame frame; //окно системы

*//*****Имена файла и папки******

String DirectoryName = "C:/";

String FileName = "";

*//******

File cur_File; // текущий файл

*//******

public MyFrame(){ // конструктор

int WinSizeG = 600; //начальный размер окна по горизонтали

int WinSizeV = 500; //начальный размер окна по вертикали

frame = new JFrame("Диалоги");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

```
MenuSim s = new MenuSim();
Container myC = frame.getContentPane();
frame.setJMenuBar(s.mb1); //добавление меню в окно
//****Организация прослушивания пунктов меню****
s.newFile.addActionListener(this);
s.openFile.addActionListener(this);
s.saveFile.addActionListener(this);
s.closeFile.addActionListener(this);
s.startOtl.addActionListener(this);
s.stopOtl.addActionListener(this);
//*****
frame.setSize(WinSizeG,WinSizeV);
frame.setLocation(10,10);
frame.setVisible(true);
}
```

В принципе, можно было сделать прослушивание и обработку событий в классе MenuSim (меню), но можно и здесь.

******методы для пункта меню "Файл"******

```
public void NewFile(){ //заглушка  
    System.out.println ("Очистка визуальных и внутренних компонентов");  
}
```

```
public void OpenDialog(){  
//открывает окно диалога для сохранения файла  
    int rez; int n;  
    JFileChooser FCH = new JFileChooser(DirectoryName);  
    FCH.setDialogTitle("Открытие файла");  
    rez = FCH.showDialog(frame,"Open");  
    if (rez == FCH.APPROVE_OPTION){  
        cur_File = FCH.getSelectedFile();  
        //Сохраняем имя директории и имя файла.  
        //В следующий раз при открытии диалога  
        //будет выбрана директория с  
        //сохраненным именем  
        DirectoryName = cur_File.getAbsolutePath();  
        n = DirectoryName.lastIndexOf("\\");  
        FileName = DirectoryName.substring(n+1);  
        DirectoryName = DirectoryName.substring(0,n+1);  
        System.out.println("Ввод текста из файла " + FileName +  
            " папки "+ DirectoryName); //заглушка  
    }  
}
```

```
private void SaveDialog(){
    //открывает окно диалога для сохранения файла
    int rez; int n;
    JFileChooser FCH = new JFileChooser(DirectoryName);
    FCH.setDialogTitle("Сохранение файла");
    rez = FCH.showDialog(frame,"Save");
    if (rez == FCH.APPROVE_OPTION){
        cur_File=FCH.getSelectedFile();
        DirectoryName = cur_File.getAbsolutePath();
        n = DirectoryName.lastIndexOf('\\');
        FileName = DirectoryName.substring(n+1);
        DirectoryName = DirectoryName.substring(0,n+1);
        System.out.println("Сохранение текста в файле " +
            FileName + " папки " + DirectoryName); //заглушка
    }
}

public void CloseWindow() { frame.dispose(); }
```

*******методы для пункта меню "Отладчик"*******

```
public void StartDebug(){
    System.out.println("Старт отладчика"); //заглушка
}
public void StopDebug(){
    System.out.println("Отладка завершена"); //заглушка
}
```

*******Обработчик событий*******

```
public void actionPerformed (ActionEvent e){
    if("Новый".equals(e.getActionCommand()))NewFile();
    else if("Открыть".equals(e.getActionCommand()))OpenDialog();
    else if ("Сохранить".equals(e.getActionCommand()))SaveDialog();
    else if ("Закрыть".equals(e.getActionCommand()))CloseWindow();
    else if ("Начать отладку".equals(e.getActionCommand()))
        StartDebug();
    else StopDebug();
}
```

*******Главный метод программы*******

```
public static void main (String[ ] args){
    javax.swing.SwingUtilities.invokeLater(new Runnable(){
        public void run(){MyFrame MF = new MyFrame();});
    }}
}
```

```
import java.io.*; import java.awt.*; import javax.swing.*;
import java.awt.event.*; import javax.swing.event.*;
```

```
public class MenuSim { //меню симулятора
```

```
    //подпункты меню
```

```
    JMenuItem newFile;  JMenuItem openFile;
    JMenuItem saveFile; JMenuItem closeFile;
    JMenuItem startOtl;  JMenuItem stopOtl;
```

```
    JMenu m1, m2; //пункты меню
```

```
    JMenuBar mb1;//панель меню
```

```
    public MenuSim(){ //конструктор
```

```
        m1 = new JMenu("Файл");
```

```
        newFile = new JMenuItem("Новый");
        m1.add(newFile);
```

```
        openFile = new JMenuItem("Открыть");
        m1.add(openFile);
```



```
saveFile = new JMenuItem("Сохранить");  
m1.add(saveFile);
```

```
closeFile = new JMenuItem("Закреть");  
m1.add(closeFile);
```

```
m2 = new JMenu("Отладчик");
```

```
startOtl = new JMenuItem("Начать отладку");  
m2.add(startOtl);
```

```
stopOtl = new JMenuItem("Закончить отладку");  
m2.add(stopOtl);
```

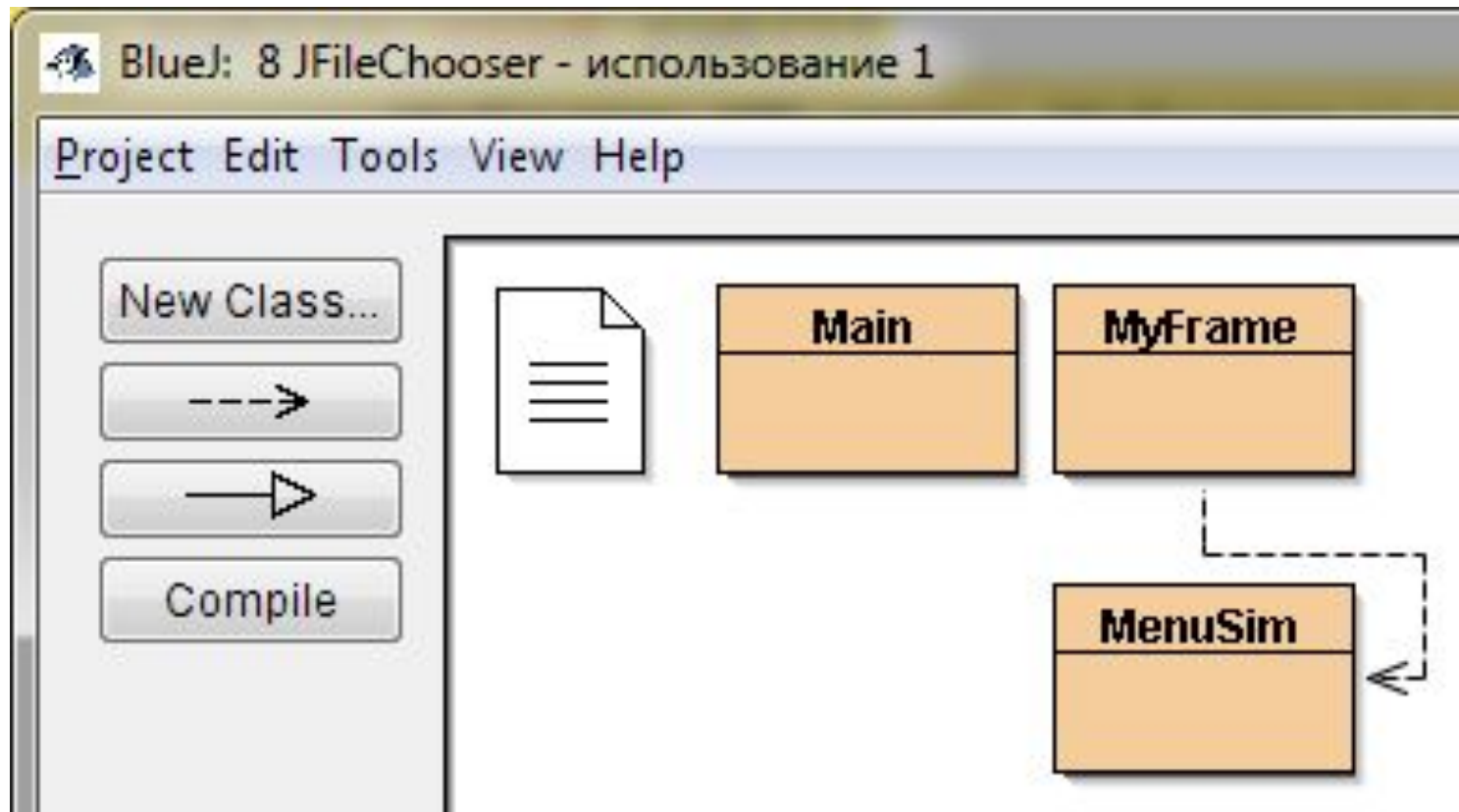
```
mb1 = new JMenuBar();
```

```
mb1.add(m1);  
mb1.add(m2);
```

```
}  
}
```

Статический метод `main`, запускающий приложение, лучше определить в отдельном классе, как это сделано в следующем проекте.

```
public class Main {  
public static void main (String[ ] args){  
    javax.swing.SwingUtilities.invokeLater(new Runnable(){  
        public void run(){  
            MyFrame MF = new MyFrame();});});  
}
```



Открытие файла

Look In: Локальный диск (C:)

- | | |
|------------------------|---------------------------------|
| 09-05-2015_16-33-42 | 09-05-2015_16-33-42.zip |
| BlueJ | bar.emf |
| Documents and Settings | WPI_Log_2014.12.25_17.55.05.txt |
| Install | |
| OpenOfficePortable | |
| Program Files | |
| WINDOWS | |

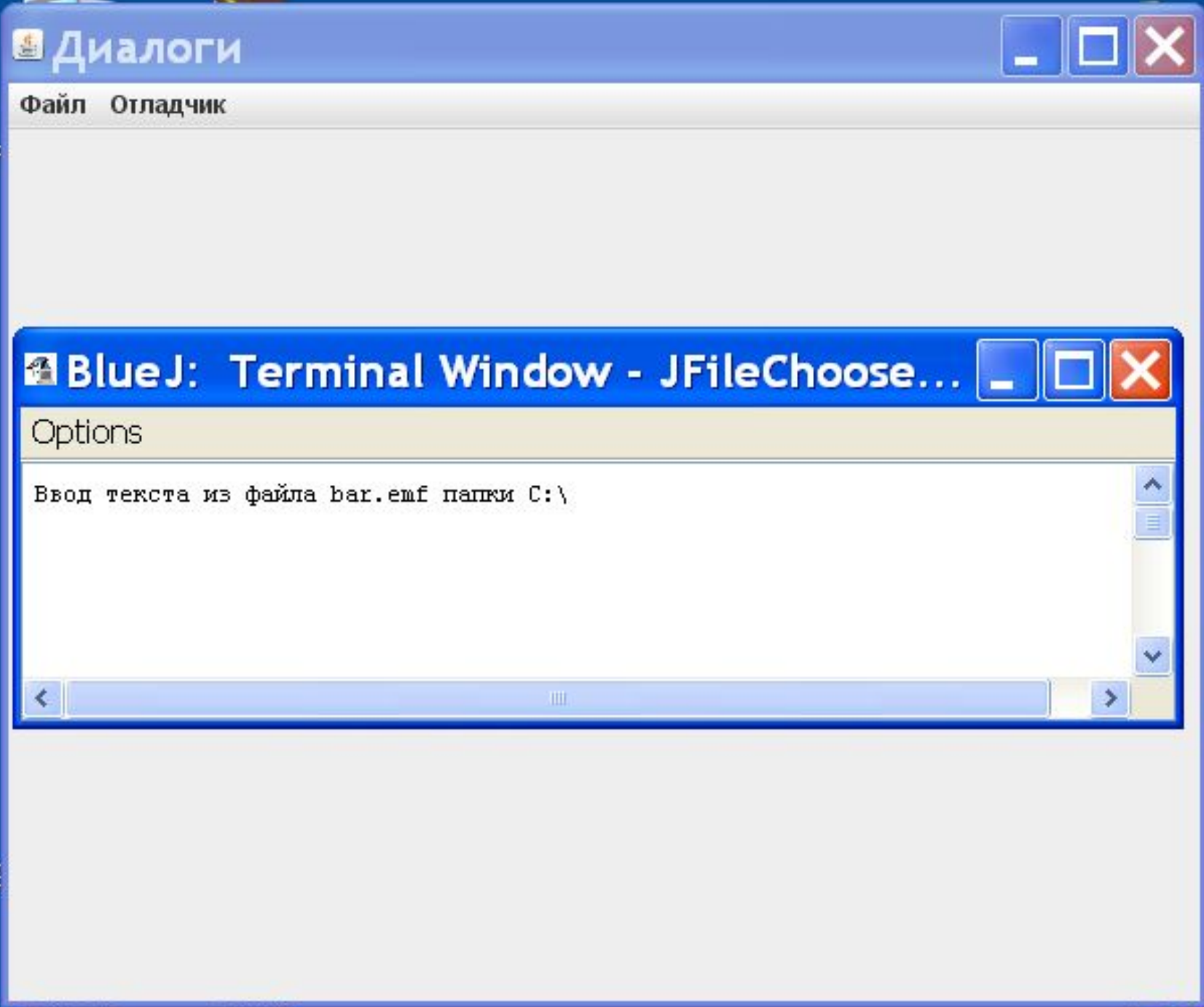
*Выводятся все файлы,
независимо от их
расширения*

File Name: bar.emf

Files of Type: All Files

Open

Cancel



После
<Open>



Сохранение файла



Look In: Локальный диск (C:)



- 09-05-2015_16-33-42
- BlueJ
- Documents and Settings
- Install
- OpenOfficePortable
- Program Files
- WINDOWS
- 09-05-2015_16-33-42.zip
- bar.emf
- WPI_Log_2014.12.25_17.55.05.txt

File Name: ggg.txt

Files of Type: All Files

Save

Cancel

Диалоги

Файл Отладчик

BlueJ: Terminal Window - JFileChoose...

Options

```
Ввод текста из файла bar.emf папки C:\  
Сохранение текста в файле ggg.txt папки C:\
```

После
<Save>

Изменение внешнего вида приложения

- Вид и поведение (Look and Feel) в определенной степени зависят от операционной системы
- В примерах диалогов используется "Look&Feel" по умолчанию, и системный (Windows) "Look&Feel")

O Look & Feel

```
public void run() {  
    try{ UIManager.setLookAndFeel(  
        UIManager.getSystemLookAndFeelClassName( )  
    );  
        } catch (Exception e) { };  
        createAndShowGUI( );}
```

Красным отмечена пара try-catch, которую нужно добавить, чтобы получить "системный" вид

**В последний пример внесены изменения:
(Проект - 9 JFileChooser - использование_вид Windows)**

```
public static void main (String[] args){  
    javax.swing.SwingUtilities.invokeLater(new Runnable(){  
        public void run(){  
            try{UIManager.setLookAndFeel (  
                UIManager.getSystemLookAndFeelClassName());  
            } catch (Exception e){ };  
            MyFrame MF = new MyFrame();}});  
}
```

Диалоги

Файл Отладчик

Открытие файла

Look in: Локальный диск (C:)



Недавние документы



Рабочий стол



Мой компьютер



Сетевое окружение

- 09-05-2015_16-33-42
- BlueJ
- Documents and Settings
- Install
- OpenOfficePortable
- Program Files
- WINDOWS
- 09-05-2015_16-33-42.zip
- bar.emf**
- WPI_Log_2014.12.25_17.55.05.txt

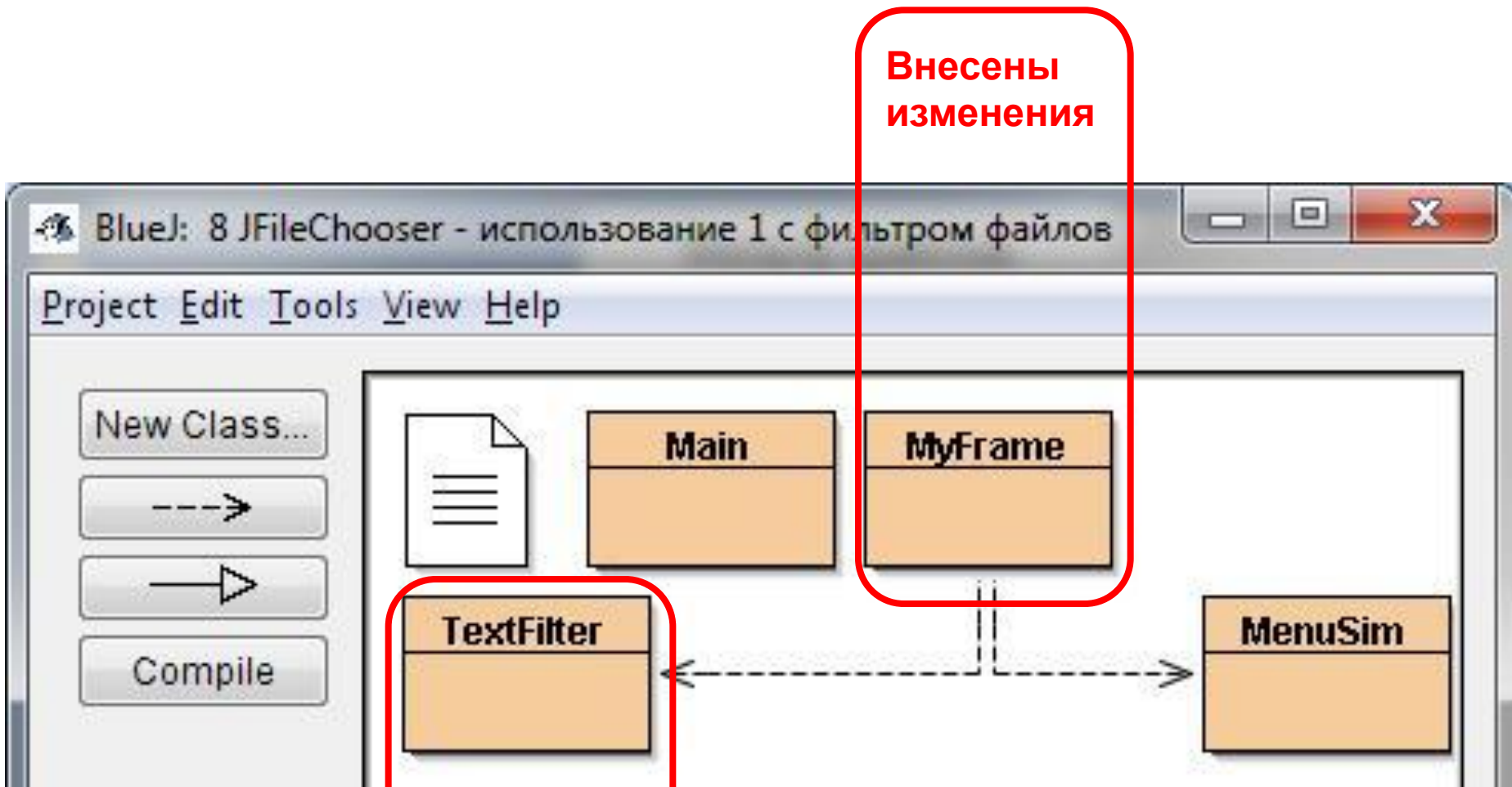
File name: bar.emf

Files of type: All Files

Open

Cancel

Диалог выбора с фильтром файлов.



**extends
FileFilter**

FileFilter - это абстрактный класс, который не имеет реализации по умолчанию. После реализации **FileFilter** можно установить в **JFileChooser**, чтобы нежелательные файлы не появлялись в списке каталогов.

```
import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;
```

```
//фильтр файлов *.txt, *.bd
```

```
//Нужен, чтобы в окне диалога выбора файлов отображались
```

```
// только файлы с указанными расширениями
```

```
public class TextFilter extends FileFilter {
```

```
    public boolean accept(File f) {
```

```
        if (f.isDirectory()) {
```

```
            return true;
```

```
        }
```

```
        //Получаем расширение файла
```

```
        String extension = getExtension (f); //метод определен ниже
```

```
        if (extension != null) {
```

```
            if (extension.equals("txt") ||
```

```
                extension.equals("bd")) {return true;}
```

```
            else {return false;}
```

```
        }
```

```
        return false; }
```

//Описание фильтра

```
public String getDescription() {  
    return "Текстовые файлы";  
}
```

//метод класса, не входящий в родительский класс FileFilter

```
private static String getExtension(File f) {  
    //возвращает расширение файла f  
    String ext = null;  
    String s = f.getName();  
    int i = s.lastIndexOf('.');  
  
    if (i > 0 && i < s.length() - 1) {  
        ext = s.substring(i+1).toLowerCase();  
    }  
    return ext;  
}  
}
```

Изменения в классе MyFrame

```
public void OpenFileDialog(){
    //открывает окно диалога для открытия файла
    boolean f=false; String s=""; int rez; int n;
    JFileChooser fch = new JFileChooser(DirectoryName);
    fch.setDialogTitle("Открытие файла");
    TextFilter text_filter = new TextFilter(); //создание фильтра файлов
    fch.setFileFilter(text_filter); //установка фильтра на чужер
    rez = fch.showDialog(frame,"Open");
    ...

private void SaveDialog(){
    //открывает окно диалога для сохранения файла
    int rez; int n;
    JFileChooser fch = new JFileChooser(DirectoryName);
    fch.setDialogTitle("Сохранение файла");
    TextFilter text_filter = new TextFilter(); //создание фильтра файлов
    fch.setFileFilter(text_filter); //установка фильтра на чужер
    rez = fch.showDialog(frame,"Save");
    ...
```

Выводятся только файлы с заданными расширениями:

