Язык программирования Java

Мультимедийный курс

автор: Васильев А.Н.

www.vasilev.kiev.ua

Киев 2017

Основы создания приложений с графическим интерфейсом

- Принципы создания приложений с GUI
- Библиотека AWT
- Библиотека Swing
- Классы графических компонентов
- Классы событий
- Принципы обработки событий
- Примеры программ

Общие подходы к созданию приложений с GUI

Создание графического интерфейса

Программирование компонентов на основе библиотек **AWT** и **Swing**	Использование класса **JOptionPane** из библиотеки **Swing**	**Графический конструктор** (в составе интегрированной среды разработки)
Преимущества: Высокая гибкость программного кода и возможность создавать многофункциональные приложения.	**Преимущества:** Минимальный объем программного кода для отображения диалоговых окон.	**Преимущества:** Удобный режим создания и настройки компонентов интерфейса, вспомогательный код генерируется автоматически.
Недостатки: Необходимость самостоятельно создавать большой объем вспомогательного кода.	**Недостатки**: Ограниченный набор функциональных элементов.	**Недостатки**: Неоптимальный автоматически сгенерированный код.

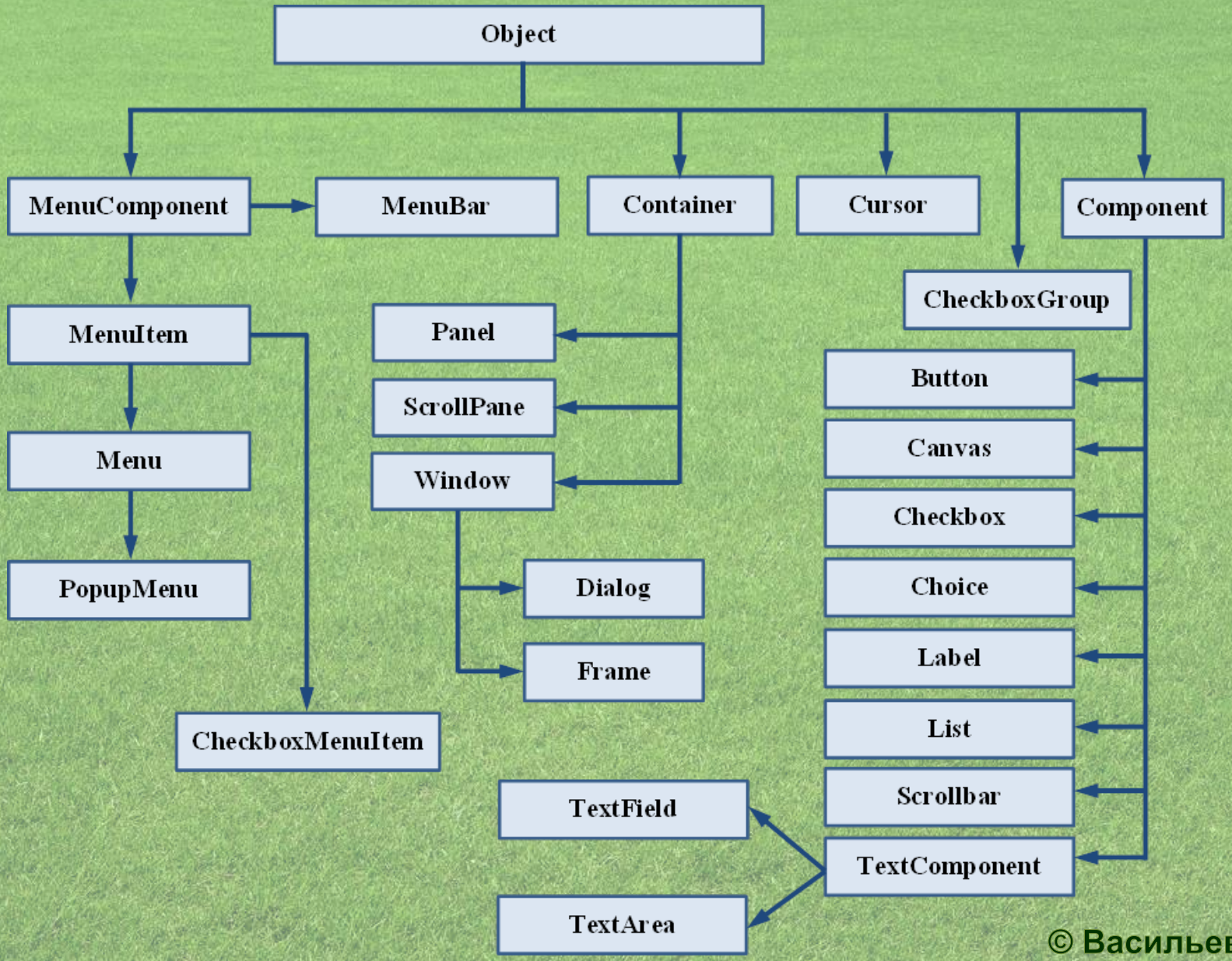
Основные компоненты

При создании приложений с графическим интерфейсом (GUI) используются библиотеки Swing (пакет `javax.swing`) и AWT (пакет `java.awt`)

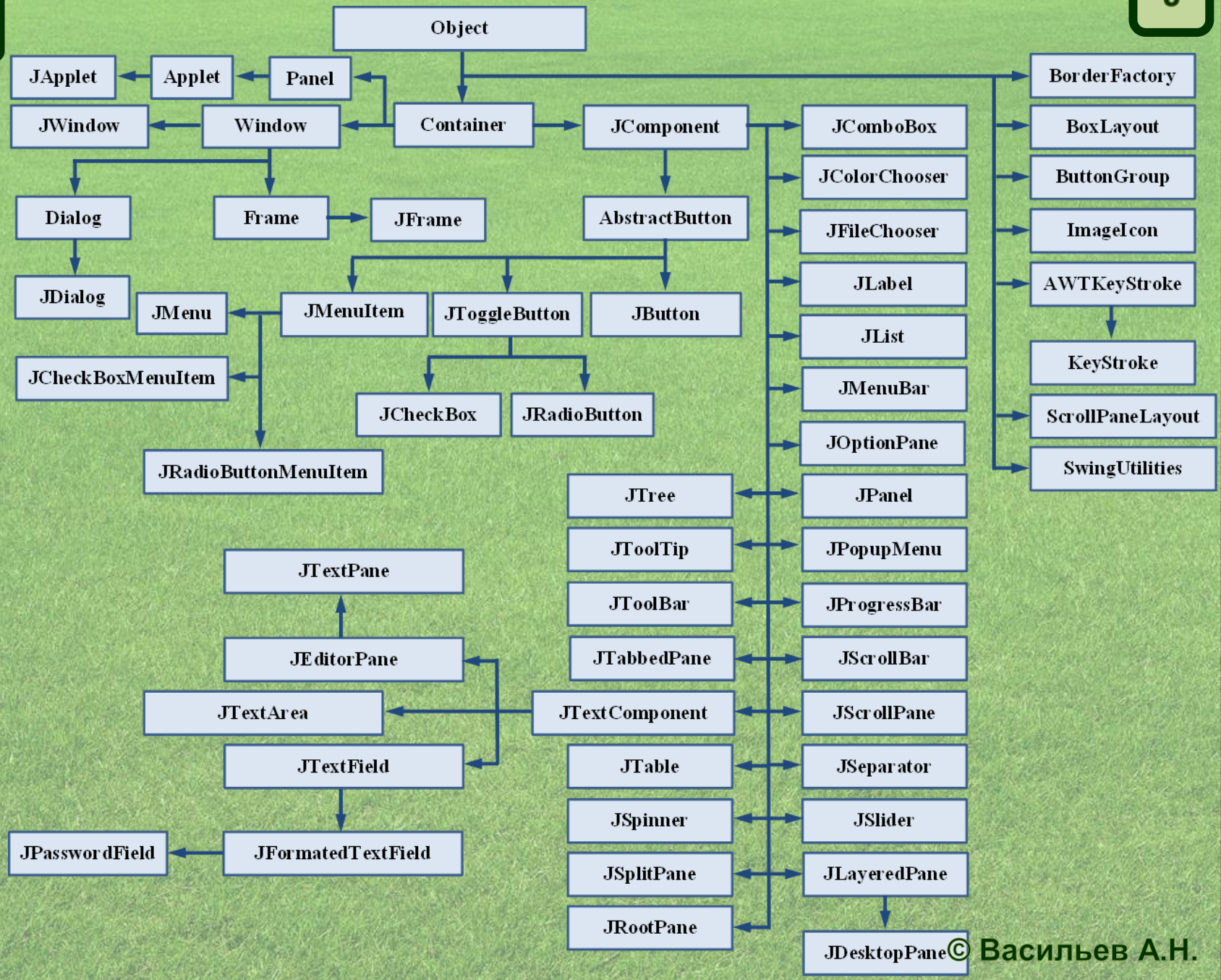
Некоторые классы библиотеки Swing

Графический компонент	Класс
Окно	`JFrame`
Кнопка	`JButton`
Текстовая метка	`JLabel`
Поле ввода	`JTextField`
Раскрывающийся список	`JComboBox`
Список	`JList`
Опция	`JCheckBox`
Переключатель	`JRadioButton`
Панель	`JPanel`

Библиотека AWT

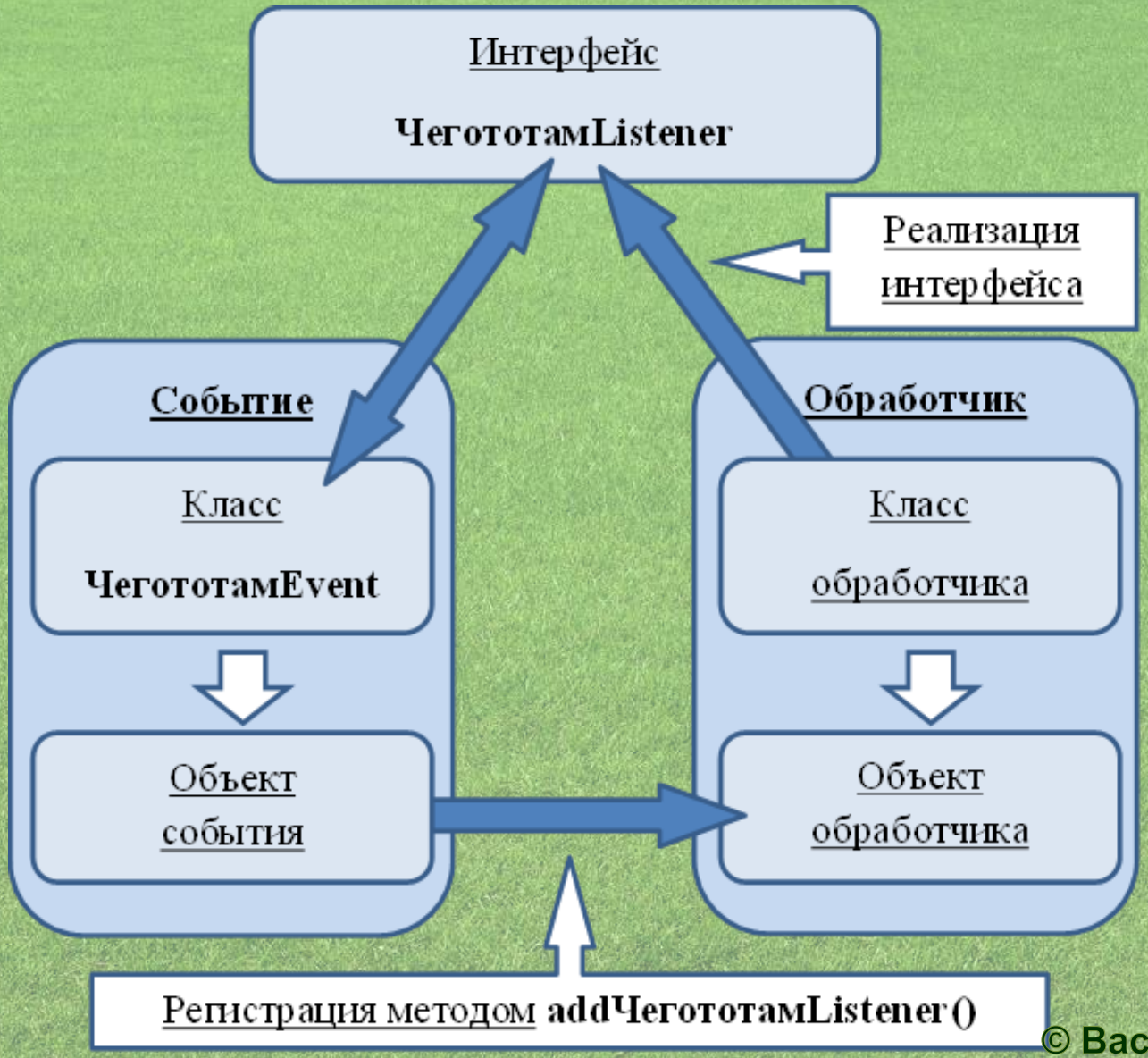


Библиотека Swing



Классы событий

Обработка событий



Пример 1. Окно с кнопкой и меткой

Требования к программе

- При выполнении программы на экране появляется окно, с названием и функциональными системными пиктограммами;

- В центральной части окна содержится текстовое сообщение;

- В нижней части окна - кнопка, щелчок на которой приводит к завершению выполнения программы.

Нам понадобятся следующие элементы графического интерфейса:

- Основное окно, которое называется окном формы или просто формой. Для создания объекта окна формы используем класс `JFrame`.

- Кнопка, чтобы разместить ее в форме. Кнопка реализуется через объект класса `JButton`.

- Текст в центральной области окна размещается не напрямую, а с помощью специального элемента, который называется меткой. Для создания объекта метки используем класс `JLabel`.

Пример 1. Окно с кнопкой и меткой

Основные моменты

- Для добавления элемента в форму (или куда-то еще) вызываем метод `add()`. Метод вызывается из того объекта, который соответствует элементу-контейнеру. Объект добавляемого элемента интерфейса указывается аргументом метода.

- Метод `setSize()`, когда он вызывается из объекта, позволяет задать его линейные размеры - ширину и высоту в пикселях. Этот метод имеет смысл применять в том случае, если расположение компонентов в контейнере определяется пользователем "вручную", без использования менеджера компоновки.

- Метод `setBounds()` позволяет задать не только размер, но и положение элемента в содержащем его контейнере. Положение задается по отношению к левому верхнему углу контейнера в пикселях (для формы положение определяется по отношению к левому верхнему углу монитора).

Пример 1. Окно с кнопкой и меткой

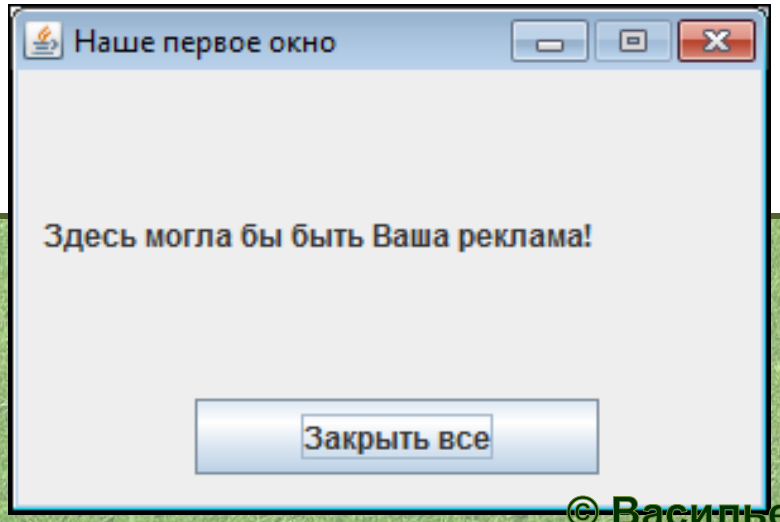
Программа: окно с кнопкой и меткой - 1

```
import javax.swing.*;
import java.awt.event.*;
// Класс для обработчика кнопки.
// Реализуется интерфейс ActionListener:
class MyHandler implements ActionListener{
   public void actionPerformed(ActionEvent ae){
   // Завершается работа программы:
   System.exit(0);
   }
}
class MyFirstFrameDemo{
   public static void main(String[] args){
      // Создаем объект формы:
      JFrame frame=new JFrame("Наше первое окно");
      // Реакция на щелчок системной пиктограммы:
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      // Размеры формы:
      frame.setSize(300,200);
      // Окно постоянных размеров:
      frame.setResizable(false);
      // Отключение менеджера компоновки:
      frame.setLayout(null);
      // Создаем объект метки:
      JLabel label=new JLabel("Здесь могла бы быть Ваша реклама!");
      // Продолжение на следующем слайде!!!
```

Пример 1. Окно с кнопкой и меткой

Программа: окно с кнопкой и меткой - 2

```
        // Положение и размер метки:
        label.setBounds(10,50,280,30);
        // Добавляем метку в форму:
        frame.add(label);
        // Создаем объект кнопки:
        JButton button=new JButton("Закрыть все");
        // Положение и размер кнопки:
        button.setBounds(70,130,160,30);
        // Создаем объект обработчика для кнопки:
        MyHandler handler=new MyHandler();
        // Регистрируем обработчик в кнопке:
        button.addActionListener(handler);
        // Добавляем кнопку в форму:
        frame.add(button);
        // Отображаем форму на экране:
        frame.setVisible(true);
    }
}
```



Пример 1. Окно с кнопкой и меткой

Комментарии - 1

- Имеем дело с двумя классами: `MyHandler` реализует интерфейс `ActionListener` и нужен нам для создания объекта-обработчика щелчка на кнопке. В классе описан единственный метод `actionPerformed()`, объявленный в интерфейсе `ActionListener`. В теле метода выполняется команда `System.exit(0)`, что приводит к завершению работы программы (к завершению работы главного метода программы) с кодом `0` (нормальное завершение работы).

Метод `main()` описан в классе `MyFirstFrameDemo`.

- Командой `JFrame frame=new JFrame("Наше первое окно")` создается объект формы `frame`. Текстовый аргумент, переданный конструктору класса `JFrame`, определяет название окна.
- Командой `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)` определяется реакция окна формы на щелчок на системной пиктограмме.
- Размеры формы задаются командой `frame.setSize(300,200)`.
- Команда `frame.setResizable(false)` - окно неизменных размеров.
- Команда `frame.setLayout(null)` - отключаем менеджер компоновки.

Комментарии - 2 (менеджер компоновки)

- Методу `setLayout()` аргументом передается объект менеджера компоновки - объект класса, реализующего интерфейс `LayoutManager`.

- Существует несколько библиотечных классов, реализующих интерфейс `LayoutManager` и предназначенных для создания менеджеров компоновки: `BorderLayout` (размещает компоненты по границам и в центре контейнера), `BoxLayout` (размещает компоненты в контейнере, группируя их по горизонтали и вертикали), `CardLayout` (размещает компоненты в контейнере по принципу формирования картотеки), `FlowLayout` (размещает компоненты в контейнере в одну с року слева направо), `GridLayout` (располагает компоненты в контейнере по принципу формирования таблицы с ячейками разного размера), `GridBagLayout` (располагает компоненты в контейнере по принципу формирования таблицы с ячейками одинакового размера), `SpringLayout` (позволяет контролировать взаимное расположение компонентов в контейнере).

Пример 1. Окно с кнопкой и меткой

Комментарии - 3

- Объект метки `label` создаем командой `JLabel label=new JLabel("Здесь могла бы быть Ваша реклама!");`
- Положение и размер метки задаем командой `label.setBounds(10,50,280,30);`
- Добавляем метку в форму с помощью команды `frame.add(label);`
- Командой `JButton button=new JButton("Закрыть все")` создаем объект кнопки `button`.
- Чтобы задать положение и размер кнопки, используем команду `button.setBounds(70,130,160,30);`
- Командой `MyHandler handler=new MyHandler()` создаем объект handler класса `MyHandler`. Назначение объекта `handler` - определять реакцию кнопки на щелчок.
- Для регистрации обработчика используем команду `button.addActionListener(handler);`
- Командой `frame.add(button)` добавляем кнопку в форму.
- Командой `frame.setVisible(true)` отображаем окно формы на экране.

Пример 1. Окно с кнопкой и меткой

Комментарий - 4 (обработка событий)

Исходные данные:

- Событию "щелчок на кнопке" соответствует класс `ActionEvent`.
- Регистрировать в кнопке обработчик события будем методом `addActionListener`.
- Обработчик - это объект класса, реализующего интерфейс `ActionListener`.

Реализация:

- Класс `MyHandler` реализует интерфейс `ActionListener`.
- В этом интерфейсе объявлен метод `actionPerformed()`.
- Этот метод будет запускаться для обработки события щелчка на кнопке. У метода один аргумент - объект класса `ActionEvent`.
- В главном методе программы мы создали объект класса `MyHandler`, и с помощью метода `addActionListener()` зарегистрировали его в объекте кнопки.

Пример 1. Окно с кнопкой и меткой

Пояснения

- Метод `setLocation()` задает положение элемента в контейнере. То есть то, что можно сделать с помощью метода `setBounds()` также можно выполнить посредством методов `setLocation()` и `setSize()`.

- Есть еще метод `setPreferredSize()`. В чем разница между методами `setSize()` и `setPreferredSize()`?

- Если мы не используем менеджер компоновки и размещаем компоненты "вручную", разумно использовать метод `setSize()`. Но этот метод малополезен, если за размещение компонентов в контейнере "отвечает" менеджер компоновки (более того, по умолчанию используется менеджер компоновки, и в случае потребности его надо отключать явным образом).

- Метод `setPreferredSize()` используют при работе с менеджером компоновки.

Пример 1. Окно с кнопкой и меткой

Вариации на тему "окно с кнопкой"

- Создается пользовательский класс для кнопки и пользовательский класс для формы
- Для кнопки отдельный класс не создается, а интерфейс реализуется в классе пользовательской формы
- Используется анонимный объект анонимного класса для обработчика события
- Используется лямбда-выражение для обработчика события

Пример 1. Класс для кнопки и формы

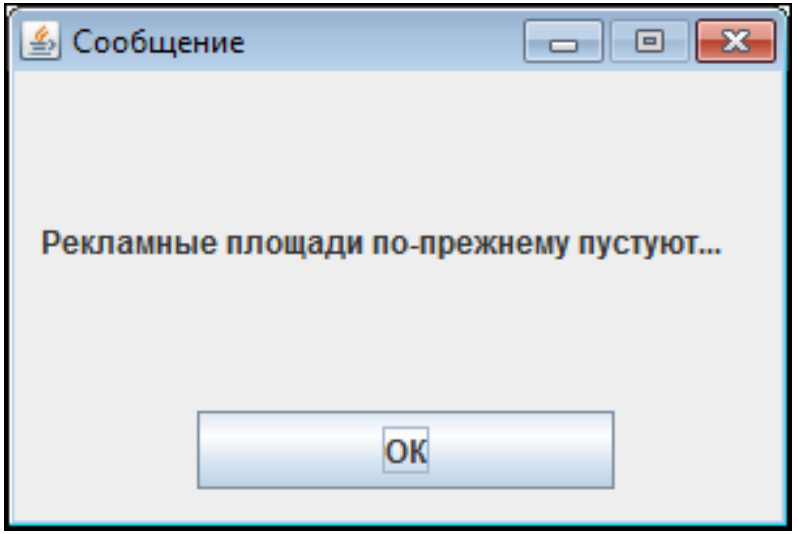
Программа: класс для кнопки и формы - 1

```
import javax.swing.*;
import java.awt.event.*;
// Класс для кнопки пользователя.
// Наследуется класс JButton и реализуется интерфейс ActionListener:
class MyButton extends JButton implements ActionListener{
   public void actionPerformed(ActionEvent ae){
      // Завершается работа программы:
      System.exit(0);
   }
   // Конструктор класса:
   MyButton(String name){
      // Название кнопки:
      super(name);
      // Регистрация обработчика:
      addActionListener(this);
   }
}
// Класс для окна формы пользователя. Наследуется класс JFrame:
class MyFrame extends JFrame{
   // Метка реализуется как поле класса:
   JLabel label;
   // Кнопка реализуется как метод класса:
   MyButton button;
   // Конструктор класса:
   MyFrame(String name,String text,String btnName){
      // Название окна формы:
      super(name);
      // Продолжение на следующем слайде!!!
```

Пример 1. Класс для кнопки и формы

Программа: класс для кнопки и формы - 2

```
        // Положение и размеры окна формы:
        setBounds(500,400,300,200);
        // Реакция на щелчок системной пиктограммы:
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // Отключение менеджера компоновки:
        setLayout(null);
        // Окно постоянных размеров:
        setResizable(false);
        // Создаем объект метки:
        label=new JLabel(text);
        // Положение и размер метки:
        label.setBounds(10,50,280,30);
        // Добавляем метку в форму:
        add(label);
        // Создаем объект кнопки:
        button=new MyButton(btnName);
        // Положение и размер кнопки:
        button.setBounds(70,130,160,30);
        // Добавляем кнопку в форму:
        add(button);
        // Отображаем форму на экране:
        setVisible(true);
}}
class MyFirstFrameDemoTwo{
   public static void main(String[] args){
        // Создаем анонимный объект для окна формы:
        new MyFrame("Сообщение","Рекламные площади по-прежнему пустуют...","OK");
}}
```



Пример 1. Интерфейс реализуется в классе формы

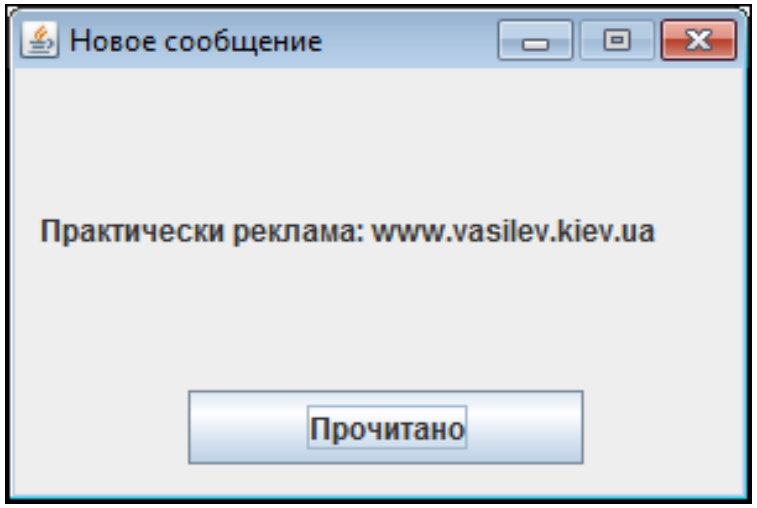
Программа: интерфейс реализуется в классе формы - 1

```
import javax.swing.*;
import java.awt.event.*;
// Класс для окна формы пользователя.
// Наследуется класс JFrame и реализуется интерфейс ActionListener:
class MyFrame extends JFrame implements ActionListener{
   // Метка реализуется как поле класса:
   JLabel label;
   // Кнопка реализуется как метод класса:
   JButton button;
   public void actionPerformed(ActionEvent ae){
      // Завершается работа программы:
      System.exit(0);
   }
   // Конструктор класса:
   MyFrame(String text){
      // Название окна формы:
      super("Новое сообщение");
      // Положение и размеры окна формы:
      setBounds(500,400,300,200);
      // Реакция на щелчок системной пиктограммы:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      // Отключение менеджера компоновки:
      setLayout(null);
      // Окно постоянных размеров:
      setResizable(false);
      // Создаем объект метки:
      label=new JLabel(text);
      // Продолжение на следующем слайде!!!
```

Пример 1. Интерфейс реализуется в классе формы

Программа: интерфейс реализуется в классе формы - 2

```
        // Положение и размер метки:
        label.setBounds(10,50,280,30);
        // Добавляем метку в форму:
        add(label);
        // Создаем объект кнопки:
        button=new JButton("Прочитано");
        // Положение и размер кнопки:
        button.setBounds(70,130,160,30);
        // Регистрируем обработчик:
        button.addActionListener(this);
        // Добавляем кнопку в форму:
        add(button);
        // Отображаем форму на экране:
        setVisible(true);
    }
}
class MyFirstFrameDemoThree{
    public static void main(String[] args){
        // Создаем анонимный объект для окна формы:
        new MyFrame("Практически реклама: www.vasilev.kiev.ua");
    }
}
```



Пример 1. Анонимный объект анонимного класса

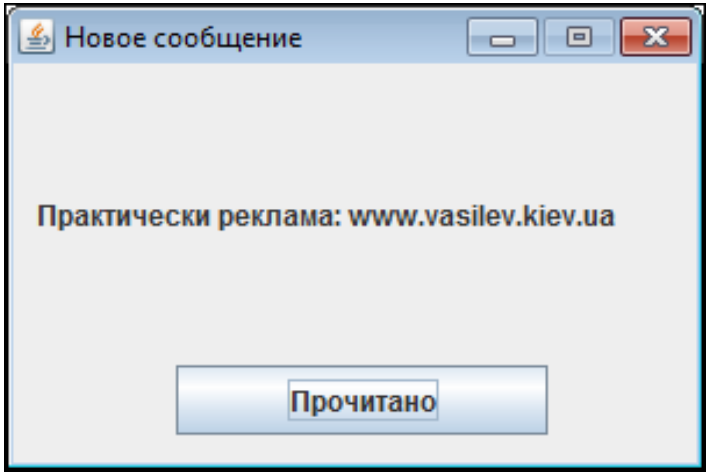
Программа: анонимный объект анонимного класса - 1

```
import javax.swing.*;
import java.awt.event.*;
// Класс для окна формы пользователя.
// Наследуется класс JFrame:
class MyFrame extends JFrame{
   // Метка реализуется как поле класса:
   JLabel label;
   // Кнопка реализуется как метод класса:
   JButton button;
   // Конструктор класса:
   MyFrame(String text){
      // Название окна формы:
      super("Новое сообщение");
      // Положение и размеры окна формы:
      setBounds(500,400,300,200);
      // Реакция на щелчок системной пиктограммы:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      // Отключение менеджера компоновки:
      setLayout(null);
      // Окно постоянных размеров:
      setResizable(false);
      // Создаем объект метки:
      label=new JLabel(text);
      // Положение и размер метки:
      label.setBounds(10,50,280,30);
      // Добавляем метку в форму:
      add(label);
      // Продолжение на следующем слайде!!!
```

Пример 1. Анонимный объект анонимного класса

Программа: анонимный объект анонимного класса - 2

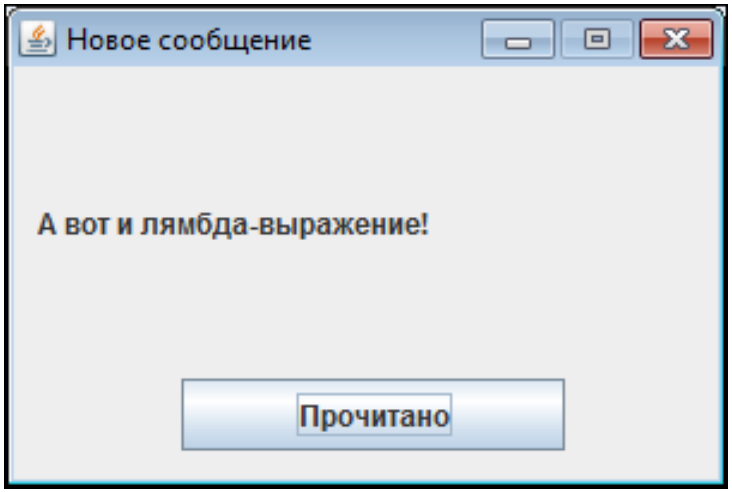
```
        // Создаем объект кнопки:
        button=new JButton("Прочитано");
        // Положение и размер кнопки:
        button.setBounds(70,130,160,30);
        // Регистрируем обработчик.
        // Использован анонимный объект обработчика,
        // созданный на основе внутреннего анонимного класса,
        // реализующего интерфейс ActionListener:
        button.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent ae){
                            // Завершается работа программы:
                            System.exit(0);
                        }
                                    });
        // Добавляем кнопку в форму:
        add(button);
        // Отображаем форму на экране:
        setVisible(true);
    }
}
class MyFirstFrameDemoFour{
    public static void main(String[] args){
        // Создаем анонимный объект для окна формы:
        new MyFrame("Практически реклама: www.vasilev.kiev.ua");
    }
}
```



Пример 1. Использование лямбда-выражения

Программа: использование лямбда-выражения

```
import javax.swing.*;
import java.awt.event.*;
class MyFrame extends JFrame{
   JLabel label;
   JButton button;
   MyFrame(String text){
      super("Новое сообщение");
      setBounds(500,400,300,200);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(null);
      setResizable(false);
      label=new JLabel(text);
      label.setBounds(10,50,280,30);
      add(label);
      button=new JButton("Прочитано");
      button.setBounds(70,130,160,30);
      // Регистрируем обработчик. Использовано лямбда-выражение:
      button.addActionListener(e->System.exit(0));
      add(button);
      setVisible(true);
   }
}
class MyFirstFrameDemoFive{
   public static void main(String[] args){
      new MyFrame("А вот и лямбда-выражение!");
   }
}
```



Пример 2. Окно с кнопками

Что нового

- Определяем для диалогового окна пользовательскую пиктограмму - вместо стандартной Java-пиктограммы в диалоговом окне будет такая, которую выбираем мы.

- В метку, кроме текста, добавляем изображение (пиктограмму метки).

- Узнаем, как в метке выравнивать текст (и не только).

- Узнаем, как задается шрифт для элементов графического интерфейса.

- Узнаем, как компоновать графические элементы в диалоговом окне с использованием панели и задавать рамку границы элементов.

- Научимся определять и рассчитывать параметры элементов интерфейса.

- Узнаем, как использовать один и тот же обработчик для нескольких кнопок.

Пример 2. Окно с кнопками: новые методы

Метод	Описание
setIconImage()	Метод для применения пиктограммы для окна формы. Аргументом передается объектная ссылка класса **Image**, определяющая "картинку" пиктограммы. Соответствующий объект создается методом **getImage()**, аргументом которому передается текст с полным путем к соответствующему графическому файлу. Метод вызывается из объекта, реализующего инструментарий окна. Ссылку на объект-инструментарий можем получить, вызвав из объекта окна метод **getToolkit()**. То есть метод **getImage()** вызывается в формате **getToolkit().getImage()**

Пример 2. Окно с кнопками: новые методы

Метод	Описание
setHorizontalAlignment()	Метод для определения способа выравнивания содержимого метки вдоль горизонтали. Аргументом указывается целочисленная константа - например, **SwingConstants.CENTER** для выравнивания по центру
setHorizontalTextPosition()	Методом задается способ выравнивания (вдоль горизонтали) текста метки по отношению к изображению (пиктограмме метки). Используется аргумент **SwingConstants.CENTER** - выравнивание по центру

Пример 2. Окно с кнопками: новые методы

Метод	Описание
setVerticalTextPosition()	Методом задается способ выравнивания (вдоль вертикали) текста метки по отношению к изображению (пиктограмме метки). Например, **SwingConstants.BOTTOM** - текст снизу, под меткой
setFont()	Метод, с помощью которого применяется шрифт для компонента (из которого вызывается метод). Аргументом указывается объектная ссылка класса **Font** на объект шрифта. Например, инструкция new **Font("Serif",Font.ITALIC,25)** означает, что применяется (логический) шрифт типа Serif, курсивный размера 25

Пример 2. Окно с кнопками: новые методы

Метод	Описание
getFont()	Метод позволяет определить шрифт, установленный для объекта, из которого вызывается метод. В качестве результата возвращается объектная ссылка класса **Font**, которую можно, например, использовать в качестве аргумента метода **setFont()**
setBorder()	Метод для определения типа рамки для компонента. Аргумент - объект класса Border, который можно сформировать статическими методами класса **BorderFactory**. С помощью метода **createEtchedBorder()** формируется объект для рамки с "вдавленной" границей

Пример 2. Окно с кнопками: новые методы

Метод	Описание
getWidth()	Метод в качестве результата возвращает ширину (в пикселях) компонента
getHeight()	Метод возвращает в качестве результата высоту (в пикселях) компонента
getSize()	Метод в качестве результата возвращает объект класса **Size** с размерами компонента. Такой объект можно передать в качестве аргумента, например, методу **setSize()**
getY()	Метод в возвращает вертикальную "координату" компонента (по отношению к своему контейнеру). Горизонтальную "координату" можно определить с помощью метода **getX()**

Пример 2. Окно с кнопками: новые методы

Метод	Описание
setFocusPainted()	Метод позволяет задать режим отображения рамки фокуса кнопки. Чтобы рамка фокуса не отображалась, методу при вызове передается аргумент **false**
getActionCommand()	Метод вызывается из объекта события и в качестве результата возвращает текст компонента (в нашем случае это название кнопки), который вызвал событие
getText()	Метод позволяет определить текст компонента (метки)
setText()	Метод позволяет задать текст компонента (метки)

Пример 2: Окно с кнопками

Программа: окно с кнопками - 1/5

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class FootballFrame extends JFrame implements ActionListener{
   // Массив для записи счета:
   private int[] goals={0,0};
   // Значения для текстовой метки:
   private String[] result={"Ну, так не интересно...",
                            "Динамо - чемпион!",
                            "Спартак - чемпион!"};
   // Массив кнопок:
   private JButton[] btns;
   // Текстовые метки (сообщение и счет):
   private JLabel showResult,score;
   // Конструктор класса:
   FootballFrame(){
      // Конструктор суперкласса:
      super("Футбольное обозрение");
      // Окно постоянных размеров:
      setResizable(false);
      // Задаем положение и размеры диалогового окна:
      setBounds(500,250,400,300);
      // Для завершения работы приложения щелкаем системную пиктограмму:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      // Пиктограмма для диалогового окна:
      setIconImage(getToolkit().getImage("/pictograms/ball.png"));
      // Отключение менеджера компоновки:
      setLayout(null); // Продолжение на следующем слайде!!!
```

Пример 2: Окно с кнопками

Программа: окно с кнопками - 2/5

```
// Создаем метку с сообщением о результате:
showResult=new JLabel(result[0]);
// Положение и размеры метки:
showResult.setBounds(5,5,getWidth()-15,50);
// Текст в метке выравнивается по центру:
showResult.setHorizontalAlignment(SwingConstants.CENTER);
// Задаем шрифт для метки:
showResult.setFont(new Font("Serif",Font.ITALIC,25));
// Добавление метки в окно формы:
add(showResult);
// Создаем объект панели:
JPanel panel=new JPanel();
// Положение и размеры панели:
panel.setBounds(5,10+showResult.getHeight(),
                showResult.getWidth(),getHeight()-150);
// Задаем тип границы панели:
panel.setBorder(BorderFactory.createEtchedBorder());
// Отключаем менеджер компоновки для панели:
panel.setLayout(null);
// Создаем метку с пиктограммой "Динамо":
JLabel dynamo=new JLabel("Динамо (Киев)", // Текст метки
              new ImageIcon("/pictograms/dynamo.png"), // Пиктограмма
              SwingConstants.CENTER); // Способ выравнивания
// Выравнивание по вертикали - текст снизу:
dynamo.setVerticalTextPosition(SwingConstants.BOTTOM);
// Выравнивание по горизонтали - текст по центру:
dynamo.setHorizontalTextPosition(SwingConstants.CENTER);
// Продолжение на следующем слайде!!!
```

Пример 2: Окно с кнопками

Программа: окно с кнопками - 3/5

```
// Положение и размеры метки:
dynamo.setBounds(0,0,panel.getWidth()/2,panel.getHeight()*2/3);
// Шрифт для метки:
dynamo.setFont(new Font("Arial",Font.PLAIN,18));
// Создаем метку с пиктограммой "Спартак":
JLabel spartak=new JLabel("Спартак (Москва)", // Текст метки
              new ImageIcon("/pictograms/spartak.png"), // Пиктограмма
              SwingConstants.CENTER); // Способ выравнивания
// Выравнивание по вертикали - текст снизу:
spartak.setVerticalTextPosition(SwingConstants.BOTTOM);
// Выравнивание по горизонтали - текст по центру:
spartak.setHorizontalTextPosition(SwingConstants.CENTER);
// Размеры метки:
spartak.setSize(dynamo.getSize());
// Положение метки:
spartak.setLocation(dynamo.getWidth()+1,0);
// Шрифт для метки:
spartak.setFont(dynamo.getFont());
// Создаем метку со "счетом":
score=new JLabel(goals[0]+" : "+goals[1]);
// Положение и размеры метки:
score.setBounds(0,dynamo.getHeight()+1,
   panel.getWidth(),panel.getHeight()-dynamo.getHeight());
// Выравнивание текста метки по центру:
score.setHorizontalAlignment(SwingConstants.CENTER);
// Шрифт для метки:
score.setFont(new Font("Monospaced",Font.BOLD,30));
// Продолжение на следующем слайде!!!
```

Пример 2: Окно с кнопками

Программа: окно с кнопками - 4/5

```
// Добавляем метку "Динамо" на панель:
panel.add(dynamo);
// Добавляем метку "Спартак" на панель:
panel.add(spartak);
// Добавляем метку для "счета" на панель:
panel.add(score);
// Добавляем панель в форму:
add(panel);
// Массив кнопок:
btns=new JButton[2];
// Первая кнопка:
btns[0]=new JButton("Динамо, вперед!");
// Вторая кнопка:
btns[1]=new JButton("Спартак, вперед!");
// Положение и размеры первой кнопки:
btns[0].setBounds(20,getHeight()-70,150,30);
// Размеры второй кнопки:
btns[1].setSize(btns[0].getSize());
// Положение второй кнопки:
btns[1].setLocation(getWidth()-btns[1].getWidth()-20,
                    btns[0].getY());
// Задаем свойства кнопок:
for(int i=0;i<2;i++){
   btns[i].addActionListener(this); // Регистрируем обработчик
   btns[i].setFocusPainted(false); // Рамка фокуса не отображается
   add(btns[i]); // Добавляем кнопку в окно формы
} // Отображаем окно формы:
setVisible(true);} // Продолжение на следующем слайде!!!
```

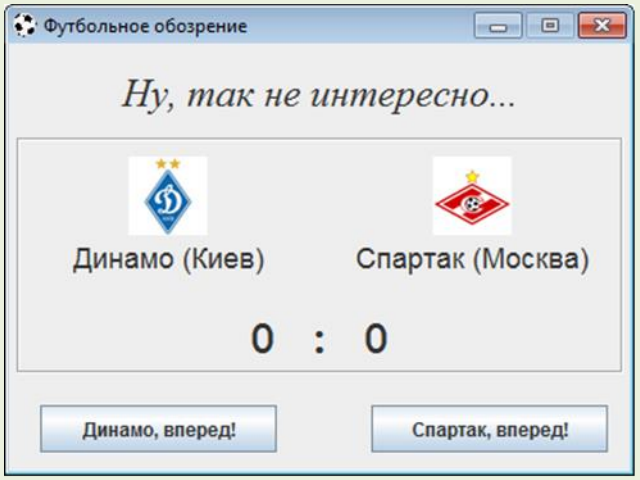
Пример 2: Окно с кнопками

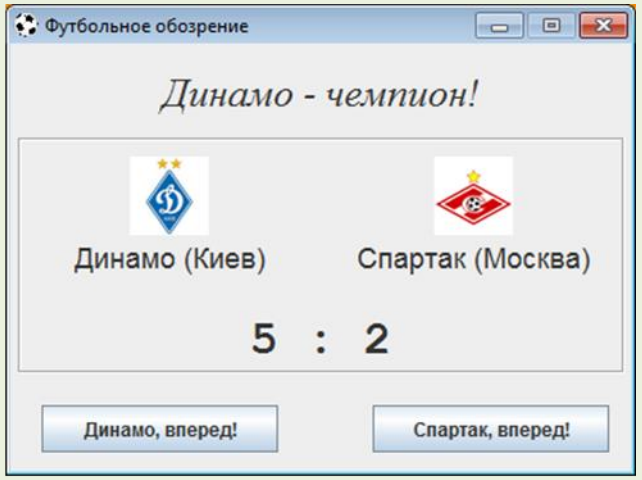
Программа: окно с кнопками - 5/5

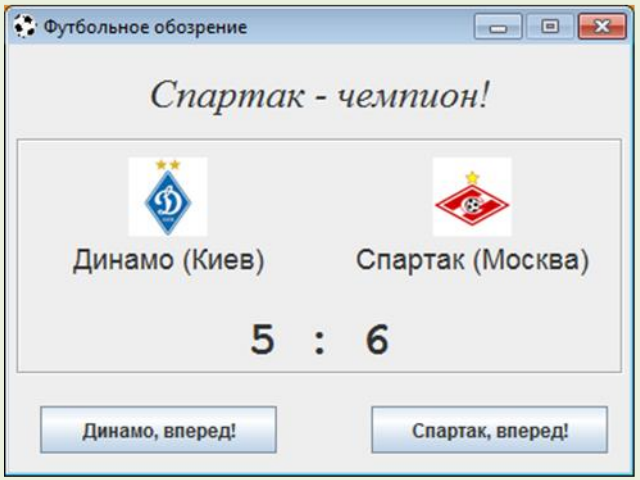
```
   // Метод для обработки щелчков на кнопках, общий для обеих кнопок:
   public void actionPerformed(ActionEvent ae){
      // Определяем текст кнопки, на которой выполнен щелчок:
      String button=ae.getActionCommand();
      // Переменная для записи индекса кнопки:
      int index;
      // Определяем индекс кнопки, на которой щелкнул пользователь:
      if(button.equals(btns[0].getText())){
         index=0;
      }else{
         index=1;
      } // Увеличиваем счет:
      goals[index]++;
      // Отображаем изменение счета в соответствующей текстовой метке:
      score.setText(goals[0]+" : "+goals[1]);
      // Проверяем результат и определяем лидера:
      if(goals[0]>goals[1]){
         showResult.setText(result[1]);
      }else{
         if(goals[0]<goals[1]){
            showResult.setText(result[2]);
         }else{
            showResult.setText(result[0]);
}}}}
class MyFootballDemo{
   public static void main(String[] args){
      // Создаем анонимный объект окна формы:
      new FootballFrame();}}
```

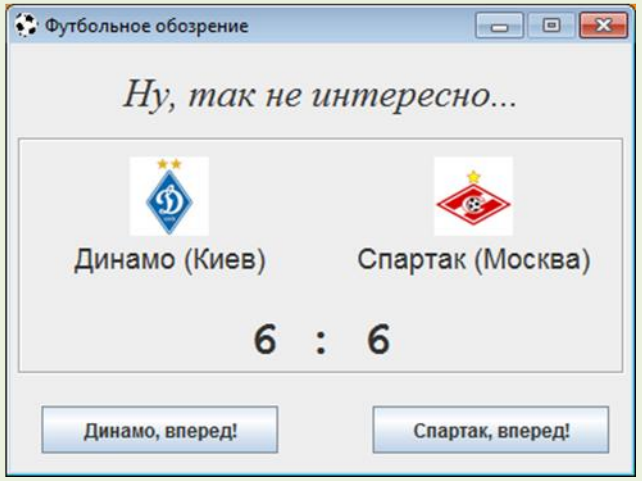
Пример 2: Окно с кнопками

Программа: окно с кнопками - результат выполнения









Домашнее задание

- В программе "Футбольное обозрение" предложите альтернативные способы обработки событий, связанным со щелчком на кнопках

- Напишите программу, в которой отображается окно с текстовой меткой и двумя кнопками. В области метки есть целое число (начальное значение равно нулю). При щелчке на одной кнопке значение целого числа увеличивается на единицу, а при щелчке на другой кнопке значение числа уменьшается на единицу. Предложите разные способы обработки событий

- Напишите программу, в которой отображается окно с меткой и двумя кнопками. В области метки могут отображаться разные картинки (например, изображения животных: лиса, волк, заяц и енот). Щелчок на кнопках приводит к смене картинки. Для одной кнопки картинки циклически меняются в прямом порядке, для другой - в обратном