Язык программирования Java

Мультимедийный курс

автор: Васильев А.Н.

www.vasilev.kiev.ua

Киев 2017

Приложения с графическим интерфейсом

- Использование поля ввода
- Обработка ввода символов
- Использование раскрывающегося списка
- Обработка событий для раскрывающихся списков

Пример 1. Окно с полем ввода

Алгоритм работы приложения

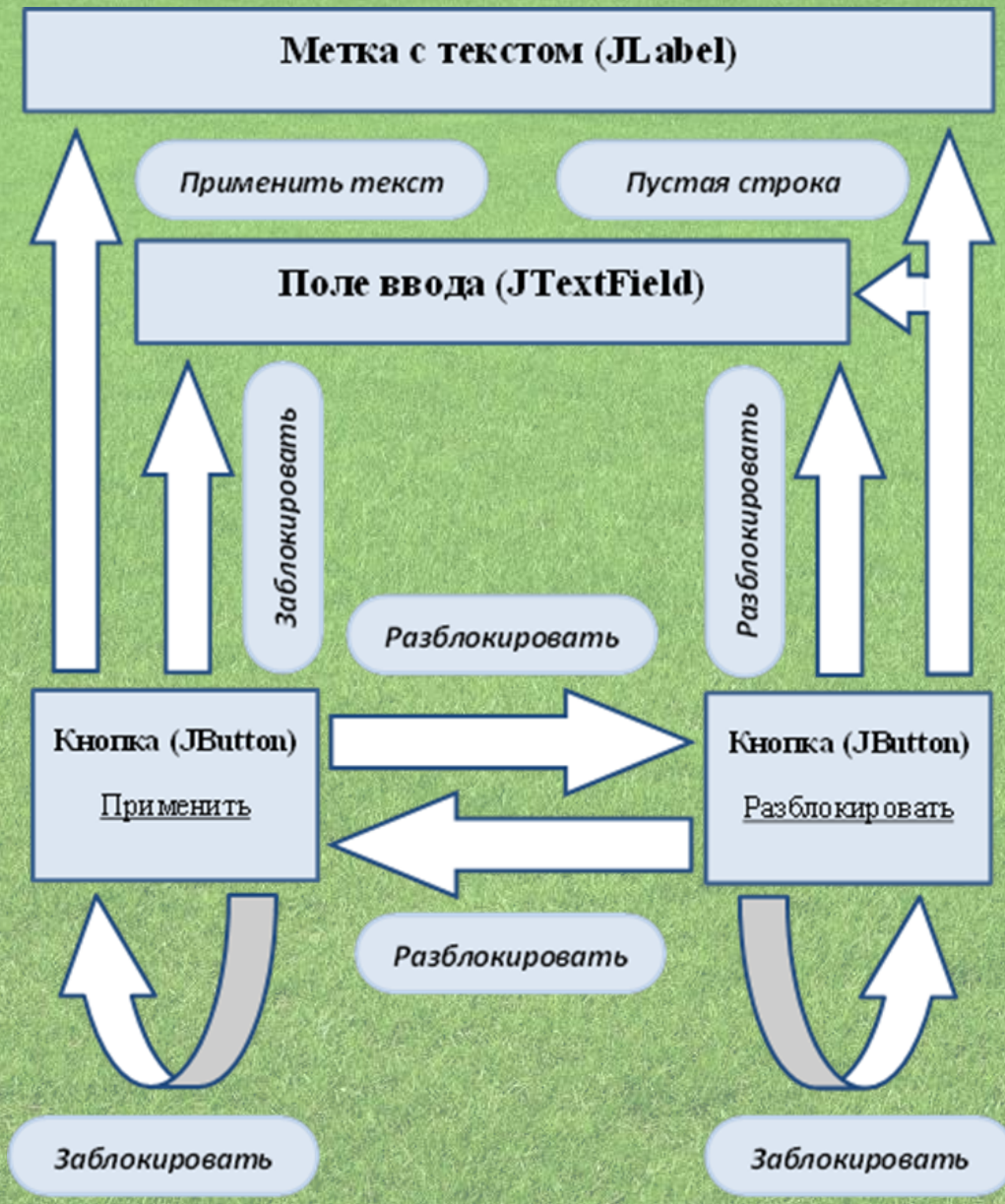
- Открывается диалоговое окно, в котором столбиком расположены текстовая метка, под ней поле ввода и под ним, в одном ряду, две кнопки (называются Применить и Разблокировать).

- Вначале поле ввода пустое, метка не содержит текста, а кнопка справа (кнопка Разблокировать) неактивна.

- В поле ввода можно ввести текст. Если после этого щелкнуть кнопку Применить, содержимое поля ввода будет отображено в текстовой метке, поле ввода и кнопка Применить станут неактивными, зато активной станет кнопка Разблокировать.

- После щелчка на кнопке Разблокировать метка и поле ввода "очищаются" (значение - пустая строка), поле ввода и кнопка Применить становится активными, а кнопка Разблокировать - неактивной. Таким образом, мы возвращаемся к тому, с чего начинали.

Пример 1. Окно с полем ввода



Пример 1. Окно с полем ввода

Программа: окно с полем ввода - 1/4

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class MyFrame extends JFrame{
   private JTextField textField; // Текстовое поле ввода
   private JButton apply,unlock; // Кнопки
   private JLabel label; // Текстовая метка
   MyFrame(){ // Конструктор класса
      // Вызов конструктора суперкласса:
      super("Текстовое поле - не игрушка");
      // Переменная определяет высоту элементов интерфейса:
      int h=30;
      // Реакция окна на щелчок системной пиктограммы:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      // Положение и размеры окна формы:
      setBounds(500,250,400,160);
      // Окно постоянных размеров:
      setResizable(false);
      // Отключаем менеджер компоновки:
      setLayout(null);
      // Создаем объект тестовой метки:
      label=new JLabel();
      // Для метки добавляем рамку:
      label.setBorder(BorderFactory.createEtchedBorder());
      // Положение и размеры метки:
      label.setBounds(10,10,getWidth()-25,h);
      // Добавляем метку в окно формы:
      add(label); // Продолжение на следующем слайде!!!
```

Пример 1. Окно с полем ввода

Программа: окно с полем ввода - 2/4

```
// Объект текстового поля ввода:
textField=new JTextField();
// Положение и размеры поля ввода:
textField.setBounds(label.getX(),label.getY()+h+10,
                    label.getWidth(),h);
add(textField); // Добавляем поле ввода в окно формы
// Создаем объект первой кнопки:
apply=new JButton("Применить");
// Положение и первой размеры кнопки:
apply.setBounds(textField.getX(),
         textField.getY()+textField.getHeight()+10,
         textField.getWidth()/2-20,h);
// При передаче фокуса кнопке рамка фокуса не отображается:
apply.setFocusPainted(false);
// Регистрация обработчика для первой кнопки:
apply.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent ae){
         textField.setEnabled(false); // Поле ввода недоступно
         // Содержимое поля ввода записывается в метку:
         label.setText(textField.getText());
         // Первая кнопка становится недоступной:
         apply.setEnabled(false);
         // Вторая кнопка становится доступной:
         unlock.setEnabled(true);
         }
      });
// Создаем объект второй кнопки:
unlock=new JButton("Разблокировать");// Продолжение на следующем слайде
```

Пример 1. Окно с полем ввода

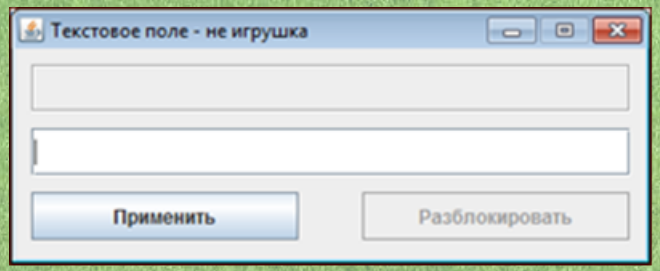
Программа: окно с полем ввода - 3/4

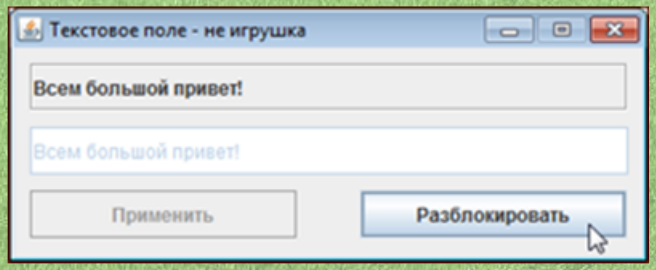
```
// Размеры второй кнопки:
unlock.setSize(apply.getSize());
// Положение второй кнопки:
unlock.setLocation(apply.getX()+apply.getWidth()+40,
                   apply.getY());
// При передаче фокуса кнопке рамка фокуса не отображается:
unlock.setFocusPainted(false);
// В начальный момент кнопка недоступна:
unlock.setEnabled(false);
// Регистрация обработчика для кнопки:
unlock.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent ae){
          textField.setText(""); // Пустая строка в текстовом поле
          // Текстовое поле доступно для ввода:
          textField.setEnabled(true);
          // Метка получает значение текстового поля (пустая строка):
          label.setText(textField.getText());
          // Первая кнопка становится доступной:
          apply.setEnabled(true);
          // Вторая кнопка становится недоступной:
          unlock.setEnabled(false);
       }
});
// Первая кнопка добавляется в окно формы:
add(apply);
// Вторая кнопка добавляется в окно формы:
add(unlock);
// Продолжение на следующем слайде!!!
```

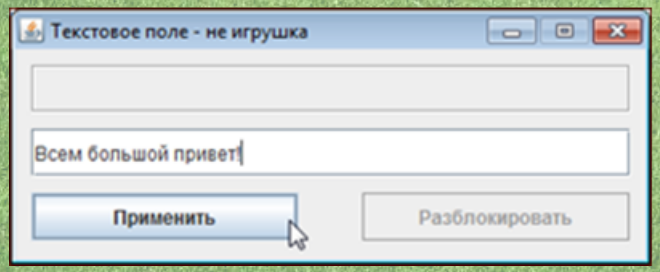
Пример 1. Окно с полем ввода

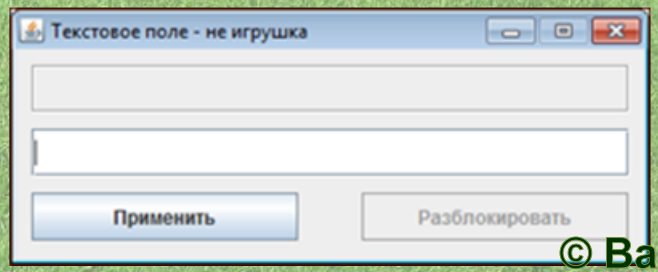
Программа: окно с полем ввода - 4/4

```
        // Окно формы отображается на экране:
        setVisible(true);
    }
}
class JTextFieldDemo{
    public static void main(String[] args){
        // Для отображения окна создается анонимный объект:
        new MyFrame();
    }
}
```









Пример 1. Окно с полем ввода

Комментарии - 1

- На основе класса `JFrame` путем наследования создаем класс `MyFrame` окна формы. В результате для отображения окна на экране в главном методе программы нам достаточно будет создать объект класса `MyFrame`. Все элементы окна формы (а это метка, поле и две кнопки) реализуются в виде закрытых полей класса:

- текстовое поле реализуется через объектную ссылку `textField` класса `JTextField`;

- ссылки `apply` и `unlock` класса `JButton` определяют кнопки;

- для текстовой метки используем объектную ссылку `label` класса `JLabel`.

Пример 1. Окно с полем ввода

Комментарии - 2

- В конструкторе класса объект текстового поля ввода создается с помощью команды `textField=new JTextField()`. Положение и размеры поля ввода определяем на основе положения и размеров ранее созданной метки, для чего используем команду
`textField.setBounds(label.getX(), label.getY()+h+10,label.getWidth(),h)`.

- Методы `getX()` и `getY()` возвращают в качестве значения горизонтальную и вертикальную координаты левого верхнего угла компонента.

- Поле в окно формы добавляется командой `add(textField)`.

- Командой `unlock.setEnabled(false)` кнопка `unlock` делается недоступной.

- Метод `setEnabled()` есть не только у кнопок. У него один логический аргумент: `true` если компонент доступен, и `false` если компонент недоступен.

Пример 1. Окно с полем ввода

Комментарии - 3 (регистрация обработчиков)

- При регистрации обработчиков используем внутренний анонимный класс (для каждой кнопки свой собственный), в котором все описано в методе `actionPerformed()`.

Для кнопки `apply` (кнопка с названием `Применить`) в методе `actionPerformed()` выполняются такие команды:

- Командой `textField.setEnabled(false)` делаем недоступным поле ввода.
- Содержимое поля ввода записывается в метку командой `label.setText(textField.getText())`, в которой результат считывания содержимого поля ввода `textField.getText()` передается аргументом методу метки `setText()`.
- После выполнения команды `apply.setEnabled(false)` первая кнопка становится недоступной.
- Антипод кнопки `apply` - кнопка `unlock` с помощью команды `unlock.setEnabled(true)` становится доступной.

Пример 1. Окно с полем ввода

Комментарии - 4 (регистрация обработчиков)

Схожие команды выполняются в методе `actionPerformed()` обработчика события щелчка на кнопке `unlock` (кнопка `Разблокировать`). А именно:

- С помощью команды `textField.setText("")` текстовое поле получает в качестве значения (содержимого поля) пустую текстовую строку.

- Текстовое поле становится доступным для ввода значений благодаря команде `textField.setEnabled(true)`.

- Метка получает значение текстового поля (пустая строка), для чего используется команда `label.setText(textField.getText())`.

- После выполнения команды `apply.setEnabled(true)` кнопка `Применить` становится доступной.

- Как следствие выполнения команды `unlock.setEnabled(false)` кнопка Разблокировать становится недоступной.

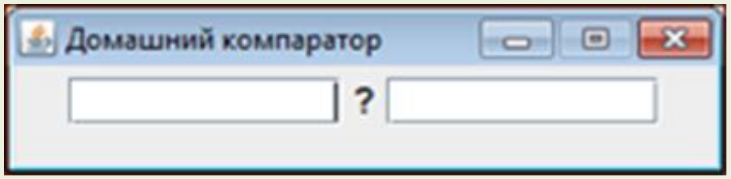
Пример 2. Сравнение чисел

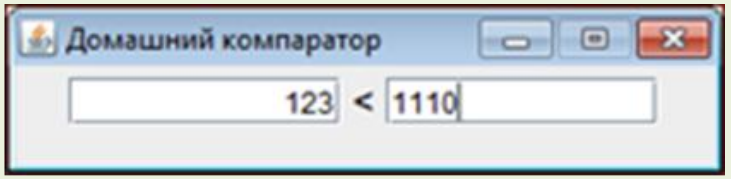
Алгоритм работы приложения

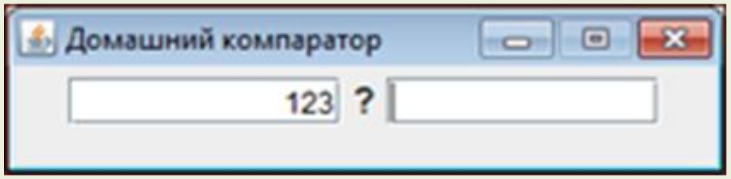
- Отображается диалоговое окно, в котором в одну строку слева направо расположены: поле ввода, текстовая метка и еще одно поле ввода.
- В начальный момент поля пустые, а в текстовой метке отображается знак вопроса.
- В поля ввода можно вводить значения (числа). Если в полях вводя числа, то в текстовой метке отображается оператор сравнения (больше, меньше или равно) в зависимости от числовых значений в полях справа и слева.
- Процесс сравнения чисел происходит автоматически еще при их вводе.
- Если в одном из полей нечисловое значение: во-первых, области текстовой метки отображается вопросительный знак, и, во-вторых, для отображения нечислового содержимого в поле ввода используется шрифт красного цвета.

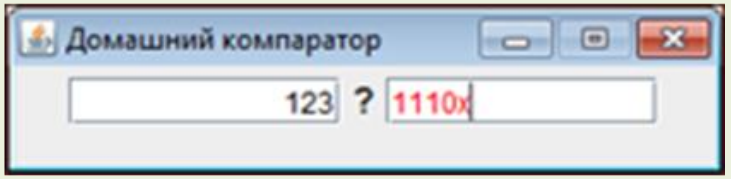
Пример 2. Сравнение чисел

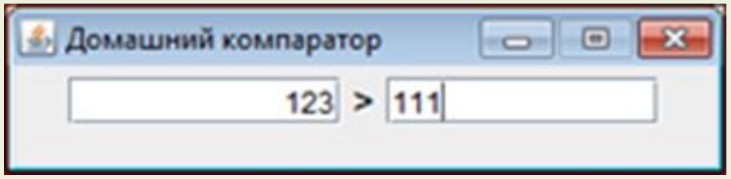
Результат

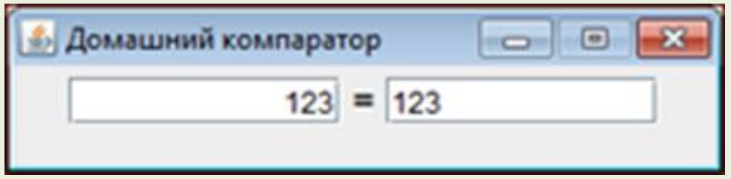












Пример 2. Сравнение чисел

Программа: сравнение чисел - 1/3

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
class MyFrame extends JFrame implements KeyListener{
   private JTextField num1,num2; // Поля для ввода чисел
   // Метка для отображения результата сравнения чисел:
   private JLabel result;
   MyFrame(){ // Конструктор класса
      // Название окна:
      super("Домашний компаратор");
      // Реакция на щелчок системной пиктограммы:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      // Используем менеджер компоновки:
      setLayout(new FlowLayout());
      // Положение и размер окна формы:
      setBounds(500,300,300,70);
      setResizable(false); // Окно постоянных размеров
      // Первое поле ввода:
      num1=new JTextField(10);
      // Выравнивание по правому краю:
      num1.setHorizontalAlignment(SwingConstants.RIGHT);
      // Красный цвет шрифта для вводимого в поле значения:
      num1.setForeground(Color.RED);
      // Регистрация обработчика для первого поля:
      num1.addKeyListener(this);
      // Второе поле ввода:
      num2=new JTextField(10);
      // Продолжение на следующем слайде!!!
```

Пример 2. Сравнение чисел

Программа: сравнение чисел - 2/3

```
        // Выравнивание по левому краю:
        num2.setHorizontalAlignment(SwingConstants.LEFT);
        // Красный цвет шрифта для вводимого в поле значения:
        num2.setForeground(Color.RED);
        // Регистрация обработчика для второго поля:
        num2.addKeyListener(this);
        // Создаем объект метки:
        result=new JLabel("?");
        // Шрифт Arial, жирный размера 15:
        Font fnt=new Font("Arial",Font.BOLD,15);
        // Применяем шрифт к метке:
        result.setFont(fnt);
        // Добавляем первое поле в окно формы:
        add(num1);
        // Добавляем метку в окно формы:
        add(result);
        // Добавляем второе поле в окно формы:
        add(num2);
        // Отображаем окно формы на экране:
        setVisible(true);
    }
    // Метод обрабатывает событие "отпускание клавиши":
    public void keyReleased(KeyEvent ke){
        // Идентифицируем текстовое поле, вызвавшее событие:
        JTextField tf=(JTextField)ke.getSource();
            try{ // Обработка исключительных ситуаций
                // Пытаемся преобразовать текст в число:
                Integer.parseInt(tf.getText());// Продолжение на следующем слайде...
```

Пример 2. Сравнение чисел

Программа: сравнение чисел - 3/3

```
            // Если введено число, то применяем черный цвет в поле ввода:
            tf.setForeground(Color.BLACK);
            // Если в обеих полях введены числа:
            if(num1.getForeground()==num2.getForeground()){
               int n1,n2;
               n1=Integer.parseInt(num1.getText()); // Число в первом поле
               n2=Integer.parseInt(num2.getText()); // Число во втором поле
               // Сравниваем числа и определяем оператор отношения для метки:
               if(n1>n2) result.setText(">");
               else if(n1<n2) result.setText("<");
                    else result.setText("=");
            }
         }catch(Exception e){ // Если в поле ввода не число
            // Применяем красный цвет шрифта для поля ввода:
            tf.setForeground(Color.RED);
            // Знак вопроса отображается в метке:
            result.setText("?");
          }
   } // Методы интерфейса KeyListener с пустой реализацией:
   public void keyPressed(KeyEvent ke){}
   public void keyTyped(KeyEvent ke){}
}
class TwoTextFieldsDemo{
   public static void main(String[] args){
      // Отображаем окно:
      new MyFrame();
   }
}
```

Пример 2. Сравнение чисел

Комментарий - 1

- В классе `MyFrame` не только наследуется класс `JFrame`, но еще и реализуется интерфейс `KeyListener`. Методами интерфейса обрабатываются события, связанные с нажатием клавиш клавиатуры.

- Если в названии `KeyListener` вначале добавить ключевое слово `add`, получим название метода для регистрации обработчиков: `addKeyListener()`.

- Аргументом этому методу передается объект класса, реализующего интерфейс `KeyListener`.

- Название класса событий, которые обрабатываются, можно получить, заменив в названии `KeyListener` слово `Listener` на слово названии `Event`. В результате получим имя класса `KeyEvent`.

- Объекты этого класса будем предавать аргументами методам из интерфейса `KeyListener` - их три: `keyReleased()` (событие отпускания клавиши), `keyPressed()` (событие нажимания клавиши) и `keyTyped()` (событие ввода символа).

Пример 2. Сравнение чисел

Комментарий - 2

- Поля ввода `num1` и `num2` (объектные переменные класса `JTextField`) в классе MyFrame описаны как закрытые поля.
- Метка (объектная ссылка класса `JLabel`) для отображения результата сравнения чисел `result` также является закрытым полем класса `MyFrame`.
- В конструкторе при компоновке элементов интерфейса в окне формы мы менеджер компоновки не отключаем: командой `setLayout(new FlowLayout())` для окна формы используем встроенный менеджер компоновки, который при добавлении элемента в контейнер размещает элементы в окне в один ряд слева направо.
- При создании полей ввода объектов класса `JTextField`, аргументом конструктору передается целое число. Оно определяет количество "столбцов" (или позиций) в поле ввода другими словами длину (ширину) поля ввода (ширина поля ввода определяет горизонтальный размер поля, но не ограничивает длину текста в поле; если текст превышает ширину поля, то в поле отображается "хвост" текста).

Если аргументом конструктору класса **JTextField** передать текстовое значение, то это значение будет отображаться в поле ввода при первом отображении поля.

Пример 2. Сравнение чисел

Комментарий - 3

- Для определения способа выравнивания текста в поле ввода используем метод `setHorizontalAlignment()`.

- Для первого поля метод вызывается с аргументом `SwingConstants.RIGHT` (команда `num1.setHorizontalAlignment(SwingConstants.RIGHT))` - выравнивание по правому краю.

- Для второго поля метод вызывается с аргументом `SwingConstants.LEFT` (команда `num2.setHorizontalAlignment(SwingConstants.LEFT))` - выравнивание по левому краю.

- Цвет отображения содержимого задается методом `setForeground()`. Аргументом методу передается статическая константа класса `Color`, определяющая цвет: `RED` для красного цвета или `BLACK` для черного цвета (команды `num1.setForeground(Color.RED)` и `num2.setForeground(Color.RED))`.

Пример 2. Сравнение чисел

Комментарий - 4

- Командами `num1.addKeyListener(this)` и `num2.addKeyListener(this)` в полях регистрируется один и тот же обработчик - объект класса окна формы `MyFrame` (в классе окна формы должны быть описаны методы интерфейса `KeyListenet`).

- У интерфейса `KeyListener` три метода, которые следует описать: `keyPressed()`, `keyTyped()` и `keyReleased()`.

- Все три метода не возвращают результат и каждый из них имеет по одному аргументу - объекту класса `KeyEvent`. Описывая каждый из этих методов, определяем реакцию элементов на то или иное событие. Нажатие и отпускание клавиши являются событиями низкоуровневыми - зависят от платформы и раскладки клавиатуры. Эти события генерируются при нажатии/отпускании любой клавиши, а не только клавиши с символом. Событие ввода символа платформенно независимое и относится в основном к вводу символов. Только в метод `keyReleased()` добавляем "полноценную начинку" - нас интересует отпускание клавиши.

Пример 2. Сравнение чисел

Комментарий - 5 (метод keyReleased())

- Метод `keyReleased()` описан с аргументом `ke` - объектной ссылкой класса `KeyEvent`.
- В теле метода командой `JTextField tf=(JTextField)ke.getSource()` определяем объектную ссылку класса `Object` на объект, вызвавший событие (инструкция `ke.getSource()`), приводим эту ссылку к объектному типу `JTextField` и записываем результат в объектную переменную `tf` класса `JTextField`.
- Используем `try-catch` блок. При попытке преобразования содержимого поля в число (команда `Integer.parseInt(tf.getText())`) если в поле введено целое число, преобразование пройдет нормально, в противном случае возникнет ошибка, которая перехватывается.
- Если ошибки не возникло, то командой `tf.setForeground(Color.BLACK)` для поля применяется черный шрифт.
- С помощью условного оператора проверяем, число ли содержится в другом поле: проверяется условие `num1.getForeground()==num2.getForeground()`.

Пример 2. Сравнение чисел

Комментарий - 6 (метод keyReleased())

- **Если в полях содержатся числа, то они считываются, запоминаются и сравниваются.**
- **В зависимости от результатов сравнения текстовая метка получает свое значение - один из трех операторов сравнения.**
- **Какая бы ошибка ни возникла при попытке преобразовать текст в поле ввода в число, она будет перехвачена `catch-блоком`. В этом случае командой `tf.setForeground(Color.RED)` для проверяемого поля применяем красный цвет для отображения одержимого поля, а текстовая метка получает в качестве значения знак вопроса.**

Формально здесь сравнивается цвет отображения содержимого для первого и второго поля. Цвет можно узнать с помощью метода **getForeground()**. Чтобы понять, почему такой прием с проверкой цвета срабатывает, необходимо учесть, что до выполнения условного оператора дело дойдет, только если в проверяемом поле введено число и установлен черный цвет для отображения содержимого поля. Поэтому совпадать цвета в двух полях могут, исключительно если и в одном, и в другом поле установлен черный цвет. А это означает, что в обоих полях введены целочисленные значения.

Пример 3. Раскрывающийся список

Алгоритм работы приложения

- Технически раскрывающийся список реализуется через объект класса `JComboBox`.

- Для раскрывающегося списка, размещенного в окне формы, определяется выбор пользователя, и на основе этого выбора рассчитываются параметры для следующего окна.

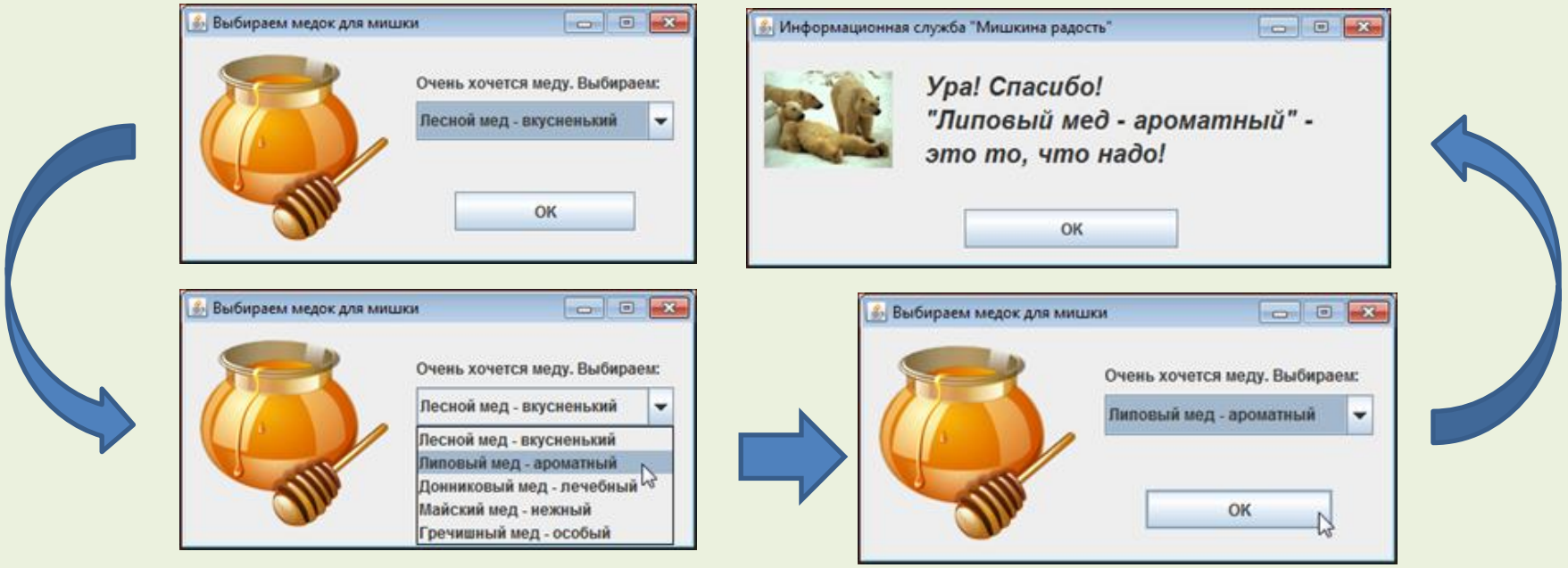
- Обработка событий для раскрывающегося списка в этом случае не предусмотрена.

- В рассматриваемо примере на экране последовательно открывается два окна. В первом окне - раскрывающийся список с названиями сортов меда. В списке нужно выбрать один из сортов и щелкнуть кнопку `OK`.

- Первое окно закрывается, а вместо него отображается другое, в котором содержится информация о том, каков был выбор пользователя на предыдущем этапе.

Пример 3. Раскрывающийся список

Результат



Пример 3: Раскрывающийся список

Программа: раскрывающийся список - 1/4

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
// Класс для отображения окна с сообщением:
class BearMessageFrame extends JFrame{
   BearMessageFrame(String text){                          // Конструктор класса
      super("Информационная служба \"Мишкина радость\""); // Название окна
      // Реакция на щелчок системной пиктограммы:
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setBounds(500,250,500,200);        // Положение и размер окна
      setResizable(false);               // Окно постоянных размеров
      setLayout(null);                   // Отключаем менеджер компоновки
      // Пиктограмма для отображения в окне формы:
      Icon img=new ImageIcon("/pictograms/bears.jpg");
      JLabel imLabel=new JLabel(img);    // Создаем метку с пиктограммой
      imLabel.setBounds(10,10,100,100); // Положение и размеры метки с пиктограммой
      JLabel txLabel=new JLabel(text);   // Метка с текстом
      // Шрифт для текстовой метки:
      Font labelFont=new Font(txLabel.getFont().getFamily(), // Тип тот же
                        Font.BOLD|Font.ITALIC,               // Жирный курсив
                        txLabel.getFont().getSize()+8);      // Размер на 8 больше
      txLabel.setFont(labelFont);                            // Применяем шрифт
      // Положение и размеры метки:
      txLabel.setBounds(imLabel.getX()+imLabel.getWidth()+25,
                        imLabel.getY(),
                        getWidth()-imLabel.getWidth()-50,
                        imLabel.getHeight());
      // Продолжение на следующем слайде!!!
```

Пример 3: Раскрывающийся список

Программа: раскрывающийся список - 2/4

```
        add(imLabel);                    // Добавляем метку с пиктограммой в окно формы
        add(txLabel);                    // Добавляем метку с текстом в окно формы
        JButton btn=new JButton("OK"); // Создаем объект кнопки
        // Положение и размеры кнопки:
        btn.setBounds(getWidth()/3,getHeight()-70,getWidth()/3,30);
        // При передаче фокуса кнопке рамка фокуса не отображается:
        btn.setFocusPainted(false);
        // Регистрация обработчика в кнопке:
        btn.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent ae){
                   System.exit(0);
                }
        });
        add(btn);              // Добавление кнопки в окно формы
        setVisible(true);      // Отображение окна формы на экране
}}
// Класс для отображения окна с раскрывающимся списком:
class HoneyChoiceFrame extends JFrame implements ActionListener{
   // Массив с названиями сортов меда:
   private String[] honey={"Лесной мед - вкусненький",
                           "Липовый мед - ароматный",
                           "Донниковый мед - лечебный",
                           "Майский мед - нежный",
                           "Гречишный мед - особый"};
   // Текстовое поле для запоминания выбора пользователя:
   private String choice;
   // Раскрывающийся список (с портами меда):
   private JComboBox honeyCB; // Продолжение на следующем слайде!!!
```

Пример 3: Раскрывающийся список

Программа: раскрывающийся список - 3/4

```
private JButton button; // Кнопка
// Метка для отображения текста над раскрывающимся списком:
private JLabel txt;
// Метка для отображения пиктограммы в области окна формы:
private JLabel img;
HoneyChoiceFrame(){                          // Конструктор класса
   super("Выбираем медок для мишки");  // Название окна формы
   // Реакция на щелчок системной пиктограммы:
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   setBounds(500,250,400,200);               // Положение и размеры окна
   setResizable(false);                      // Окно постоянных размеров
   setLayout(null);                          // Отключаем менеджер компоновки
   txt=new JLabel("Очень хочется меду. Выбираем:"); // Метка с текстом
   txt.setBounds(180,20,200,30);             // Положение и размеры метки с текстом
   img=new JLabel(new ImageIcon("/pictograms/honey.png")); // Метка с пиктограммой
   img.setBounds(10,10,150,150);             // Положение и размеры метки с пиктограммой
   // Создаем раскрывающийся список.
   // Аргумент конструктора - массив элементов списка:
   honeyCB=new JComboBox(honey);
   // Положение и размеры раскрывающегося списка:
   honeyCB.setBounds(180,50,200,30);
   button=new JButton("OK");                 // Создаем кнопку
   button.setBounds(210,120,140,30);         // Положение и размеры кнопки
   button.addActionListener(this);           // Регистрация обработчика в кнопке
   add(txt);        // Добавляем текстовую метку в область окна формы
   add(img);        // Добавляем метку с пиктограммой в область окна формы
   add(honeyCB);    // Добавляем раскрывающийся список в окно формы
   // Продолжение на следующем слайде!!!
```

Пример 3: Раскрывающийся список

Программа: раскрывающийся список - 4/4

```
        // Добавляем кнопку в окно формы:
        add(button);
        // Отображаем окно формы на экране:
        setVisible(true);
    }
    // Метод для обработки щелчка на кнопке:
    public void actionPerformed(ActionEvent ae){
        // Идентифицируем выбор пользователя в раскрывающемся списке:
        choice=honey[honeyCB.getSelectedIndex()];
        // Отображаем окно с текстовым сообщением:
        new BearMessageFrame("<html>Ура! Спасибо!<br>\""+
                             choice+"\" - это то, что надо!</html>");
        // Закрываем текущее окно с раскрывающимся списком:
        dispose();
    }
}
class HoneyDemo{
    public static void main(String[] args){
        // Отображается окно с раскрывающимся списком:
        new HoneyChoiceFrame();
    }
}
```

Пример 3. Раскрывающийся список

Комментарий - 1

- Для каждого из двух типов окон, используемых в программе, описывается отдельный класс.
- Класс для отображения окна с сообщением называется `BearMessageFrame` и создается наследованием класса `JFrame`.
- У конструктора класса текстовый аргумент - он определяет сообщение в окне (реализуется в текстовой метке `txLabel`).
- Для метки создается шрифт (объект `labelFont` класса `Font`): переопределяется на основе шрифта по умолчанию: инструкцией `txLabel.getFont()` поучаем ссылку на объект текущего шрифта метки, а из этого объекта вызываем метод `getFamily()`, который в качестве результата возвращает текстовую строку с названием типа шрифта (первый аргумент конструктора класса `Font` имеет вид `txLabel.getFont().getFamily()`). Второй аргумент конструктора - инструкция `Font.BOLD|Font.ITALIC` (используется жирный курсив). Третий аргумент - `txLabel.getFont().getSize()+8` (текущий размер будет увеличен на 8). Для применения созданного шрифта к метке используем метод `setFont()`.

Текст, определяющий название окна, содержит двойные кавычки. Чтобы отличить такие двойные кавычки от двойных кавычек, в которые заключаются текстовые литералы, в первом случае используют косую черту. Поэтому в тексте двойные кавычки добавляются комбинацией \".

Пример 3. Раскрывающийся список

Комментарий - 2

- Через класс `HoneyChoiceFrame` реализуется окно с раскрывающимся списком.

- В классе объявлен текстовый массив `honey` с названиями сортов меда. Его используем при создании раскрывающегося списка: элементы массива послужат основой для создания элементов раскрывающегося списка.

- Для запоминания выбора пользователя (текста того пункта, который пользователь выберет в раскрывающемся списке) объявляется текстовое поле `choice`.

- Раскрывающийся список связан с полем `honeyCB` - объектная переменная класса `JComboBox`.

- Полями класса объявлены ссылка на кнопку button и две метки: `txt` для определения надписи над раскрывающимся списком и `img` для отображаемой пиктограммы в окне.

Пример 3. Раскрывающийся список

Комментарий - 3

- В конструкторе командой `honeyCB=new JComboBox(honey)` создается объект раскрывающегося списка. Аргументом конструктору класса `JComboBox` передан массив `honey` - элементы списка создаются на основе массива.
- Положение и размер списка задаем методом `setBounds()`.
- Добавляем список в окно командой `add(honeyCB)`.
- Метод для обработки щелчка на кнопке - `actionPerformed()`: в команде `choice=honey[honeyCB.getSelectedIndex()]` метод `getSelectedIndex()` возвращает индекс выбранного элемента списка. Значение переменной `choice` - соответствующий элемент массива `honey`.
- Для отображения окна с текстовым сообщением создается анонимный объект класса `BearMessageFrame`. Текстовый аргумент конструктора этого класса содержит HTML-теги и значение текстовой переменной `choice`.
- Чтобы закрыть окно без завершения работы всего приложения, вызываем метод `dispose()` (при этом окно не просто убирается с экрана, а удаляется).
- В главном методе программы путем создания анонимного объекта класса `HoneyChoiceFrame` отображается окно с раскрывающимся списком.

Пример 4. Еще раз список

Алгоритм работы приложения

- Открывается диалоговое окно, в котором имеется раскрывающийся список с перечислением разных видов медведей.

- В этом же окне отображается пиктограмма с изображением медведя и приводится краткое описание этого дружественно настроенного зверя.

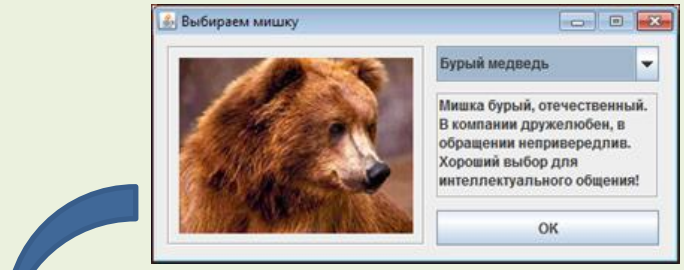
- Как только выбираем другой элемент в раскрывающемся списке, в соответствии с нашим выбором меняется мишкина "фотография" и мишкина "биография".

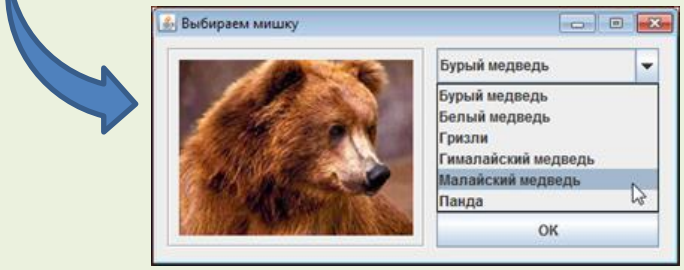
- Щелчок на кнопке OK приводит к завершению работы приложения.

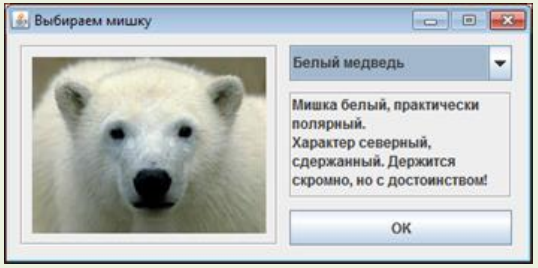
- В этом примере (в отличие от предыдущего) используется обработка событий для раскрывающегося списка.

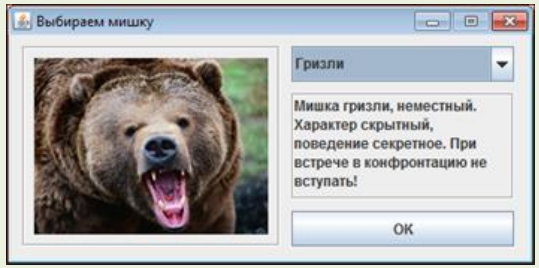
Пример 4. Еще раз список

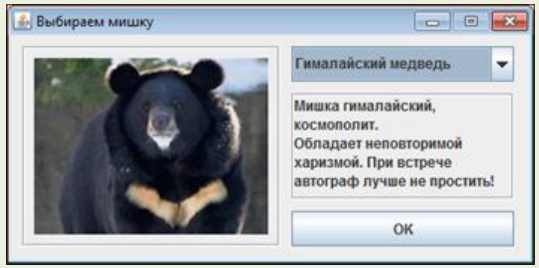
Результат

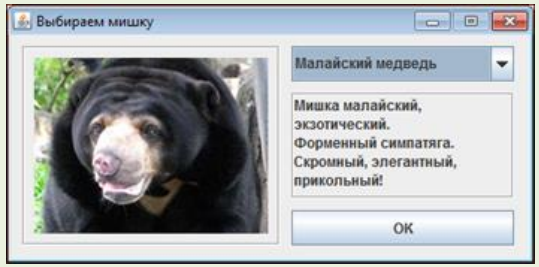


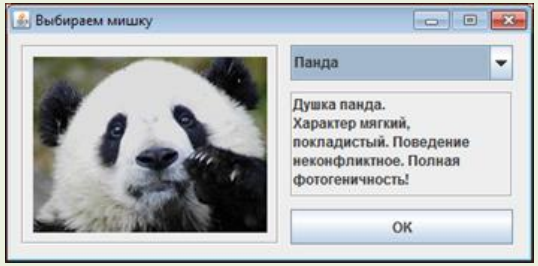












Пример 4: Еще раз список

Программа: еще раз список - 1/3

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
// Класс реализует сразу два интерфейса:
class BearChoiceFrame extends JFrame implements ActionListener,ItemListener{
   private ImageIcon[] imgs; // Массив из пиктограмм
   // Текстовый массив с видами медведей и названиями соответствующих файлов:
   private String[][] bears={{"Бурый медведь","Белый медведь","Гризли","Гималайский
медведь","Малайский медведь","Панда"},
{"bury.jpg","bely.jpg","grizli.jpg","hymalai.jpg","malai.jpg","panda.jpg"}};
   // Текстовый массив с кратким описанием медведей:
   private String[] msgs={"<html>Мишка бурый, отечественный.<br>В компании дружелюбен,
в обращении неприveредлив. Хороший выбор для интеллектуального общения!</html>",
                          "<html>Мишка белый, практически полярный.<br>Характер
северный, сдержанный. Держится скромно, но с достоинством!</html>",
                          "<html>Мишка гризли, неместный.<br>Характер скрытный,
поведение секретное. При встрече в конфронтацию не вступать!</html>",
                          "<html>Мишка гималайский, космополит.<br>Обладает
неповторимой харизмой. При встрече автограф лучше не простить!</html>",
                          "<html>Мишка малайский, экзотический.<br>Форменный
симпатяга. Скромный, элегантный, прикольный!</html>",
                          "<html>Душка панда.<br>Характер мягкий, покладистый.
Поведение неконфликтное. Полная фотогеничность!</html>"};
   private JComboBox cb;      // Раскрывающийся список
   private JLabel lbl,info;   // Метки
   private JButton btn;       // Кнопка
    // Продолжение на следующем слайде!!!
```

Пример 4: Еще раз список

Программа: еще раз список - 2/3

```
BearChoiceFrame(){                          // Конструктор класса
   super("Выбираем мишку");                 // Название окна формы
   // Реакция на щелчок системной пиктограммы:
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   setBounds(500,250,450,220);              // Положение и размеры окна
   setResizable(false);                     // Окно постоянных размеров
   setLayout(null);                         // Отключаем менеджер компоновки
   imgs=new ImageIcon[bears[0].length];     // Создаем массив пиктограмм
   // Заполняем массив пиктограмм:
   for(int i=0;i<imgs.length;i++){
      imgs[i]=new ImageIcon("/pictograms/"+bears[1][i]);
   }
   // Создаем метку с пиктограммой (первая пиктограмма в массиве пиктограмм):
   lbl=new JLabel(imgs[0]);
   lbl.setBorder(BorderFactory.createEtchedBorder()); // Рамка вокруг метки
   lbl.setBounds(10,10,220,170);            // Положение и размеры метки
   btn=new JButton("OK");                   // Создаем кнопку
   btn.addActionListener(this);             // Регистрируем обработчик в кнопке
   btn.setBounds(240,150,190,30);           // Положение и размеры кнопки
   cb=new JComboBox(bears[0]);              // Создаем раскрывающийся список
   cb.setBounds(240,10,190,30);    // Положение и размеры раскрывающегося списка
   cb.addItemListener(this);       // Регистрация обработчика в списке
   info=new JLabel(msgs[0]);       // Создаем метку с текстом
   // Способ выравнивания текста в метке по вертикали (вверху):
   info.setVerticalAlignment(SwingConstants.TOP);
   info.setBounds(240,50,190,90); // Положение и размеры метки
   info.setBorder(BorderFactory.createEtchedBorder()); // Рамка вокруг метки
   // Продолжение на следующем слайде!!!
```

Пример 4: Еще раз список

Программа: еще раз список - 3/3

```
        add(info);          // Добавляем метку с текстом в окно формы
        add(lbl);           // Добавляем метку с пиктограммой в окно формы
        add(btn);           // Добавляем кнопку в окно формы
        add(cb);            // Добавляем раскрывающийся список в окно формы
        setVisible(true); // Отображаем окно формы на экране
    }
    // Метод для обработки события, состоящего в изменении
    // состояния (выбранного пункта) раскрывающегося списка:
    public void itemStateChanged(ItemEvent ie){
        // Определяем индекс выбранного элемента:
        int index=cb.getSelectedIndex();
        // Меняем пиктограмму в метке с пиктограммой:
        lbl.setIcon(imgs[index]);
        // Меняем текст-описание в текстовой метке:
        info.setText(msgs[index]);
    }
    // Метод для обработки щелчка на кнопке:
    public void actionPerformed(ActionEvent ae){
        System.exit(0);
    }
}
class BearsDemo{
    public static void main(String[] args){
        // Отображаем окно формы:
        new BearChoiceFrame();
    }
}
```

Пример 4. Еще раз список

Комментарий - 1

- Класс `BearChoiceFrame` наследует `JFrame` и реализует два интерфейса: `ActionListener` (обработка щелчка на кнопке) и `ItemListener` (обработка события выбора элемента в списке).

Поля класса:

- Массив `imgs` пиктограмм (ссылки класса `ImageIcon`), предназначенный для "хранения" мишкиных фотографий.
- Текстовый двумерный массив `bears`: первый ряд - виды медведей, второй ряд - файлы с изображениями мишек.
- Текстовый массив `msgs` с описанием медвежьих "характеров" (описания достаточно громоздки и содержат HTML-теги).
- Раскрывающийся список - ссылку `cb` класса `JComboBox`.
- Метка `lbl` для отображения мишкиных фотографий.
- Метка `info` для отображения краткой справки по медведям.
- Кнопка реализуется через поле `btn`. Это стандартная кнопка, щелчок на которой приводит к закрытию окна и завершению работы приложения.

Пример 4. Еще раз список

Комментарий - 2

- Командой `imgs=new ImageIcon[bears[0].length]` создаем массив пиктограмм. Размер массива определяется количеством элементов в первом "ряду" двойного текстового массива `bears`.

- Заполняем массив пиктограмм в теле оператора цикла командой `imgs[i]=new ImageIcon("/pictograms/"+bears[1][i])`.

- Раскрывающийся список создаем командой `cb=new JComboBox(bears[0])` (`bears[0]` - первый "ряд" двумерного массива `bear`, то есть одномерный массив с названиями медведей).

- Регистрация обработчика события выбора нового элемента в списке выполняется командой `cb.addItemListener(this)`.

- При создании метки с изображением и текстовой метки с информацией о медведе в качестве "начальных" значений соответственно используются элементы массивов `imgs[0]` и `msgs[0]`.

Пример 4. Еще раз список

Комментарий - 3

- С помощью метода `setVerticalAlignment()` и переданного ему аргумента `SwingConstants.TOP` для текстовой метки выполняется выравнивание вдоль вертикали по верхнему краю.

- Вокруг обоих меток отображается рамка.

- В классе описывается метод `itemStateChanged()` для обработки события, состоящего в изменении состояния раскрывающегося списка.

- У метода один аргумент - объектная ссылка класса `ItemEvent`. Поскольку обработчиком для раскрывающегося списка регистрируется объект окна формы, этот метод вызывается каждый раз, когда пользователь выбирает тот или иной пункт раскрывающегося списка.

- В теле метода командой `index=cb.getSelectedIndex()` определяем индекс выбранного элемента, и затем меняем пиктограмму в метке с пиктограммой (команда `lbl.setIcon(imgs[index])`) и текст-описание в текстовой метке (команда `info.setText(msgs[index])`).

Домашнее задание

- Напишите программу, в которой отображается окно с полем ввода. текстовой меткой и кнопкой. Щелчок на кнопке приводит к закрытию окна. В текстовой метке дублируется текст, который пользователь вводит в поле ввода
- То же, что в предыдущем случае, но предполагается, что в поле ввода пользователь вводит число. Если в поле ввода введено число, в текстовой метке оно отображается синим цветом. Если в поле ввода введено не число, то в текстовой метке красным цветом отображается слово "Ошибка"
- Напишите программу, в которой отображается окно с раскрывающимся списком. Список содержит названия цветов (например, красный, желтый, зеленый). Также окно содержит область (например, панель или текстовую метку) закрашенную цветом, выбранным в раскрывающемся списке. Выбор цвета в раскрывающемся списке приводит к автоматическому изменению цвета заливки области
- Напишите программу, в которой отображается окно с полем ввода, двумя раскрывающимися списками и панелью с текстом (панель, на которой размещена метка). В раскрывающихся списка выбирается цвет для шрифта текста на панели цвет для фона заливки панели. Текст, отображаемый в области панели, дублирует текст, введенный в поле ввода