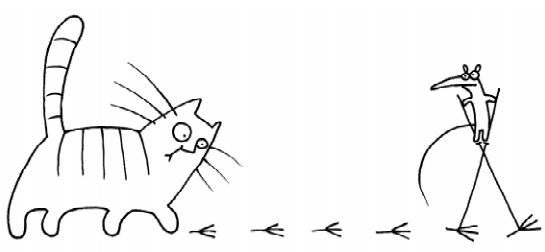
*Дмитриева Марина Валерьевна*

# ОСНОВЫ ПРОГРАММИРОВАНИЯ ДЛЯ ИНТЕРНЕТ. АППЛЕТЫ. ЗАНЯТИЕ 6. КОМПОНЕНТЫ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА. СТРОКА ВВОДА

Статья посвящена методам обработки данных, заданных компонентой «строка ввода». Приводятся программы обработки текстовых данных, проверки пароля, построения фигур заданных размеров, манипулирования изображениями.

Пользовательский интерфейс строится из компонент, представляющих собой элементы интерфейса. Одним из распространенных элементов интерфейса является кнопка, созданию которой и обработке событий, с ней связанных, была посвящена предыдущая статья. Другим часто используемым компонентом AWT является строка ввода. Пользователь может водить текст в строку ввода и редактировать этот текст с помощью стрелок, клавиш «Удалить», «Вставить» и т. п.

Для ввода текста AWT содержит два компонента: строка ввода, представленная классом **TextField**, и поле ввода, представленное классом **TextArea**. Класс **TextField** позволяет ввести только одну строку текста, класс **TextArea** – любое количество строк. Оба класса являются потомками класса **TextComponent**, поэтому имеют общие методы.



*Компонент TextField – это поле для ввода одной строки текста.*

## КОМПОНЕНТ TEXTFIELD

Компонент **TextField** — это поле для ввода одной строки текста. Ширина поля измеряется в колонках (**column**). Ширина колонки – это средняя ширина символа в шрифте, которым вводится текст. В классе предусмотрены следующие конструкторы:

- **TextField()** создает пустое поле шириной в одну колонку;
- **TextField(int columns)** создает пустое поле с числом колонок **columns**;
- **TextField(string text)** создает поле с текстом **text**;
- **TextField(String text, int columns)** создает поле с текстом **text** и числом колонок **columns**.

Для получения результата пользовательского ввода или редактирования можно применить метод **getText()**. Для ввода текста используется метод **setText()**.

Ширина поля или размер строки – это размер видимой части текста. Текст может превышать эти размеры, его можно сдвинуть в видимой области с помощью стрелок влево или вправо.

Текстовыми компонентами **`TextField`** и **`TextArea`** генерируются события типа **`TextEvent`** при изменении содержащегося в них текста. Обработка событий **`TextEvent`** производится с помощью объектов классов, реализующих интерфейс **`TextListener`**, у которого имеется только один метод **`textValueChanged`**.

## РИСОВАНИЕ КРУГА ЗАДАННОГО РАДИУСА

Создадим апплет, в котором с помощью поля для ввода строки пользователь может ввести размер радиуса круга. После ввода значения в строку в области апплета в центре будет нарисован круг заданного радиуса. В программе определяется **`rad`** – экземпляр класса **`TextField`**, затем создается строка ввода текста с помощью конструктора **`rad = new TextField(3)`**. Далее в программе созданный компонент размещается в области апплета.

Слушателем события является строка для ввода текста. После нажатия на клавишу «ENTER» в центре апплета будет нарисован круг заданного радиуса. Описанный класс реализует интерфейс **`TextListener`**, переопределяя его единственный метод **`textValueChanged`**. Проверяется, введено ли значение в строку для ввода текста, и если значение введено, то оно преобразовывается в число, и рисуется круг заданного радиуса.

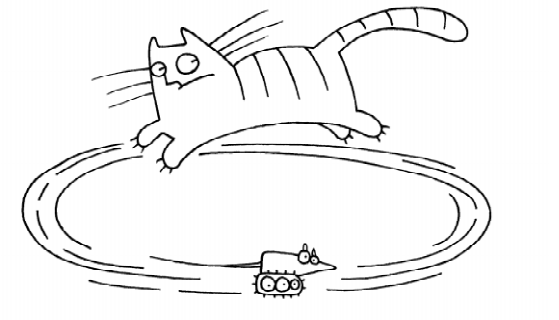
При рисовании круга цвет фона и цвет переднего края зависят от радиуса. Если ввести в строку для ввода текста другое значение радиуса, то и цвет фона и цвет круга будет изменен. В листинге 1 представлен исходный текст программы. Апплет показан на рис. 1.

## ВЫЧИСЛЕНИЕ НАЛОГА

Напишем апплет, содержащий три строки для ввода текста. В первую строку вводится гонорар, после нажатия на клавишу «ENTER» во вторую сроку помещается значение, равное налогу (13% от суммы гонорара), в третью строку – сумма, выдаваемая на руки после уплаты налога. Можно считать, что первая строка служит для ввода данных, остальные две – для вывода результата работы программы.

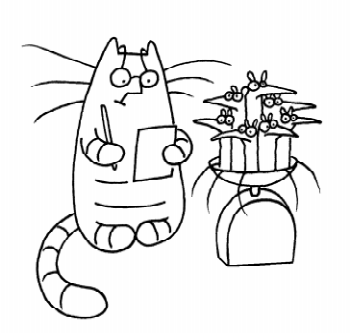
Напомним, что многие компоненты и поле для ввода строки, в частности, вызывают события, которые можно обрабатывать при помощи метода апплета **`action()`**. Первый параметр метода класса **`Event`** хранит информацию о событии, второй параметр позволяет получить дополнительную информацию о событии.

При обработке события проверяется, что источником события является строка ввода, более того, проверяется, что ввод



*Рисование круга заданного радиуса*


**Рис. 1.** Строка ввода для задания радиуса круга



Вычисление налога

осуществлен в первую строку. Далее извлекается введенный текст, преобразовывается к целому значению, и выполняются требуемые вычисления.

Метод **setEditable()** разрешает или запрещает пользователю вводить и редактировать текст. Проверить, разрешен или запрещен ввод текста, можно с помощью метода **isEditable**: если ввод разрешен, то выдается значение **true**, в противном случае – **false**. Ввод во вторую и третью строки запрещен.

**Листинг 1.** Рисование круга заданного радиуса

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//рисование окружности заданного радиуса
public class Rad extends Applet
                    implements TextListener{
     TextField rad;//строка ввода для радиуса
//координаты центра и радиус круга
     int xC, yC, r;
//начальные установки
     public void init(){
          xC = this.getWidth()/2;
          yC = this.getHeight()/2;
          Label radp = new Label("Введите радиус ", Label.RIGHT);
          rad = new TextField(3);
//добавление элементов в апплет
          add(radp); add(rad);
          rad.addTextListener(this);
   } // init
//обработка событий
      public void textValueChanged (TextEvent e){
           TextField t = (TextField)e.getSource();
           if (t.getText().length() != 0){
                r = Integer.parseInt(t.getText());
               repaint();
           }
      }
//рисование
    public void paint(Graphics g){
       Color col= new Color(r%255,2*r%255,3*r%255);
       setBackground(col);
       g.fillOval(xC-r,yC-r,2*r,2*r);
       col= new Color(5*r%255,4*r%255,3*r%255);
       setForeground(col);
       g.drawOval(xC-r,yC-r,2*r,2*r);
       showStatus("радиус: " + r);
     }
}
```

В программе для размещения данных используются три панели. На первую панель помещается первая строка для ввода текста и метка, на вторую панель – вторая компонента и, наконец, на третью – строка для ввода суммы, выдаваемой на руки, и соответствующая метка.

В листинге 2 приведен текст программы. Апплет показан на рис. 2.

### СТРОКА ДЛЯ ВВОДА ПАРОЛЯ

Строку для ввода текста можно использовать следующим образом: пользователь вводит текст, а в строке ввода отображается какой-либо символ-заменитель, чаще всего – звездочка. Так происходит, когда пользователю предложено, например, ввести пароль. Установить символ-заменитель

**Листинг 2.** Вычисление налога

```
import java.awt.*;
import java.applet.*;
//вычисление налога
public class Nalog extends Applet {
// строки ввода для гонорара, налога и суммы на руки
  TextField ti1, ti2, ti3;
  float a,  b, c;
  Panel p1=new Panel();
  Panel p2=new Panel();
  Panel p3=new Panel();
// начальные установки
  public void init(){
     Label i1 = new Label("гонорар", Label.LEFT);
     Label i2 = new Label("налог  ", Label.LEFT);
      Label i3 = new Label("на руки", Label.LEFT);
     ti1 = new TextField(6);
     ti2 = new TextField(6);
      ti3 = new TextField(6);
// размещение компонентов в окне
      p1.add(ti1); p1.add(i1);   add(p1);
      p2.add(ti2); p2.add(i2);   add(p2);
      p3.add(ti3); p3.add(i3);   add(p3);
      ti2.setEditable(false); // в строке не разрешен ввод
      ti3.setEditable(false); // в строке не разрешен ввод
 }// init
// обработка событий
      public boolean action (Event evt, Object whichAcion) {
        if (evt.target instanceof TextField){
          TextField cur = (TextField)evt.target;
          if (cur == ti1) {// событие от первой строки
              a = Integer.parseInt(cur.getText());
              b = a*13/100;
              c = a - b;
              ti2.setText(""+b);
              ti3.setText(""+c);
              showStatus("на руки: " + a +" - "+ b + " = " + c );
            return true;
         } else return false; // событие не обработано
        } else return false;
      }
}
```


**Рис. 2.** Три строки ввода при вычислении налога

**(echoCharacter)** можно с помощью метода **setEcnoCnar(char echo)**. Параметр **echo** – это символ, который будет появляться в строке ввода. Проверить, установлен ли эхо-символ, можно логическим методом **echoCharisSet()**, получить эхо-символ – методом **getEchoChar()**.

## ВВОД ПАРОЛЯ

Напишем программу, в которой будем использовать две строки ввода. В первую строку может быть введен пароль. Вводимые символы заменяются символом звездочка. Ввод и редактирование во второй строке запрещено. После вода пароля и нажатия на клавишу «Enter» во второй строке отображается введенный пароль. Информация о том, что пароль введен, отображается и в строке статуса. Для размещения компонент используется табличный способ. В листинге 3 приведен текст программы. Апплет показан на рис. 3. Обратите внимание на способ размещения компонент в апплете.


**Листинг 3.** Ввод пароля

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//ввод пароля
public class pass_1 extends Applet {
// строки для ввода пароля и результата
  TextField ti1, ti2 ;
  Label i1 = new Label("Ведите пароль и нажмите <Enter>: ", Label.LEFT);
  Label i2 = new Label("Вы ввели");
// начальные установки
     public void init(){
         setLayout(new GridLayout(4,1));
         ti1 = new TextField(15);
         ti1.setEchoChar("*");
         ti2 = new TextField(15);
         ti2.setEditable(false);
 // размещение компонентов в окне
      add(i1); add(ti1);      add(i2); add(ti2);
 }// init
// обработка событий
      public boolean action (Event e, Object whichAcion) {
        if (e.target instanceof TextField){
            TextField cur = (TextField)e.target;
            ti2.setText("Пароль: "+ cur.getText());
            showStatus("пароль введен "  );
       return true;
          } else return false; // событие не обработано
        }
 }
```

### ОБМЕН ИЗОБРАЖЕНИЙ ПО НОМЕРАМ

Напишем программу, в которой с помощью строк ввода пользователь может задавать номера рисунков. Считаем, что рисунки перенумерованы от 1 до 4. После нажатия на кнопку «Поменять» осуществляется перестановка рисунков с указанными номерами. Изображения объединяются в массив. Создаваемый класс реализует интерфейс **`ActionListener`**, содержащий метод **`actionPerformed`**. После анализа введенных значений в массиве переставляются элементы с указанными индексами. В листинге 4 приведена программа, решающая задачу.

Апплет показан на рис. 4. Если какое-либо из двух значений не введено, то выдается сообщение о том, что не заданы значения. Если значения введены, но не попадают в заданный диапазон, то выдается сообщение о том, что ошибка в номере рисунка.

### ЗАДАЧА О РАЗРЕЗАНИИ ПРЯМОУГОЛЬНИКА НА КВАДРАТЫ

Рассмотрим задачу разрезания прямоугольника на квадраты максимальной площади. Первоначально пользователь задает ширину и высоту прямоугольника.

Метод демонстрации разрезания прямоугольника – рекурсивный. При проектировании рекурсивного алгоритма следует выделить параметры, от которых зависит решение задачи. В рассматриваемом случае это координаты левого угла

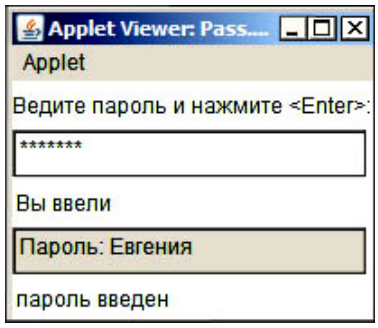
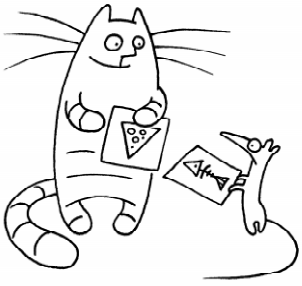



**Рис. 3.** Использование строки ввода для ввода пароля

прямоугольника, ширина и высота. Далее следует решить задачу в тривиальном случае, то есть в случае, когда не требуется рекурсивный вызов. В данной задаче тривиальный случай: ширина и высота прямоугольника совпадают, то есть прямоугольник является квадратом. И, наконец, сведение решения задачи к ней самой, но с другими, «более простыми» параметрами. Если же значения ширины и высоты различны, то вычисляется длина меньшей стороны, которая и будет стороной квадрата, отрезаемого вдоль большей стороны прямоугольника. Затем аналогичные действия применяются к прямоугольнику, полученному после отрезания квадрата.

Кроме двух строк для ввода ширины и высоты прямоугольника, в апплете предусмотрены две кнопки. После ввода ширины и высоты прямоугольника при нажатии на кнопку с названием «Построить» в центре графической области апплета будет построен прямоугольник и квад-



*Обмен изображений по номерам*

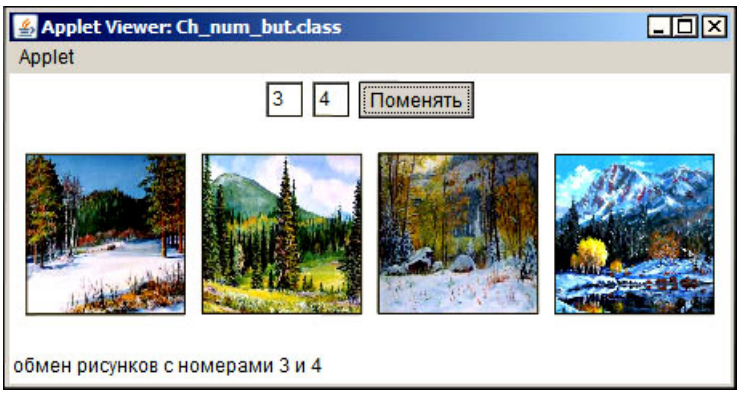
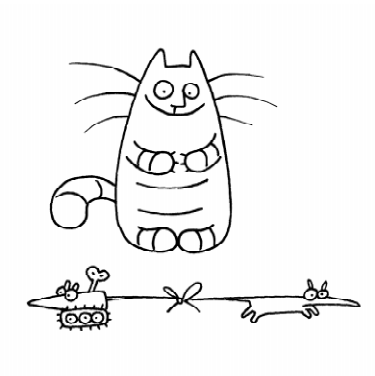



**Рис. 4.** Перестановка рисунков с заданными номерами

**Листинг 4.** Обмен изображений по номерам

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
 // обмен изображений по номерам
public class Ch_num_but extends Applet implements ActionListener{
  int pictLength = 4;
  Image[] pict = new Image[pictLength];
  Button b;
  TextField i1, i2;//две строки для ввода номеров рисунков
  int w=100, h=100;
// начальные установки
  public void init(){
    resize(450,170);//размер окна
    pict[0]=getImage(getCodeBase(),"images/g1.jpg");
    pict[1]=getImage(getCodeBase(),"images/g2.jpg");
    pict[2]=getImage(getCodeBase(),"images/g3.jpg");
    pict[3]=getImage(getCodeBase(),"images/g4.jpg");
// создание и размещение полей для ввода строки
    i1 = new TextField();
    i2 = new TextField();
    add(i1); add(i2);
// создание и размещение кнопки
    b = new Button(«Поменять»); add(b);
    b.addActionListener(this);
 }//init
// рисование
  public void paint (Graphics g){
    g.drawImage(pict[0], 10,  50, w, h, this);
    g.drawImage(pict[1], 120, 50, w, h, this);
    g.drawImage(pict[2], 230, 50, w, h, this);
    g.drawImage(pict[3], 340, 50, w, h, this);
  }//paint
// обработка события щелчок по кнопке
  public void actionPerformed(ActionEvent ae) {
   if (!i1.getText().equals("")&!i2.getText().equals(«»)){
    String str = ae.getActionCommand();
    if(str.equals("Поменять")) {
    int a = Integer.parseInt(i1.getText());
    int b = Integer.parseInt(i2.getText());
    if(a >= 1 & a<= 4 & b>=1 & b <= 4){
      Image  cur = pict[a-1];
      pict[a-1] = pict[b-1];
      pict[b-1] = cur;
      repaint();
      showStatus("обмен рисунков с номерами "+ a +" и "+ b);
    } else showStatus("ошибка в номере рисунка");
   }
  }else showStatus("Не заданы значения");
 }//actionPerformed
}//Ch_num_but
```



*...читается одинаково как слева направо, так и справа налево.*

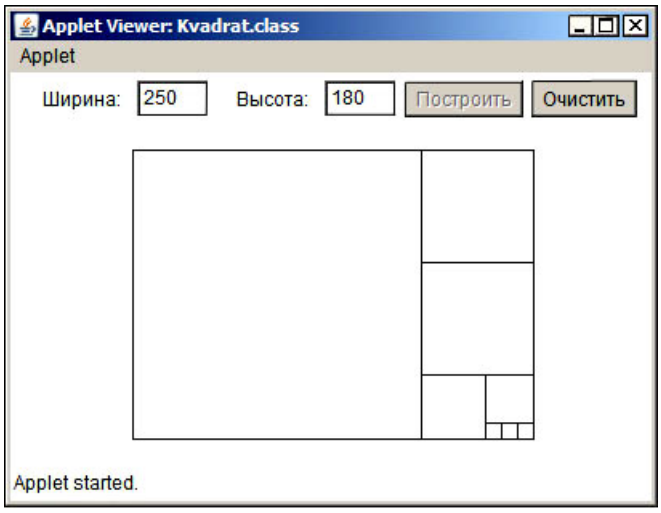



**Рис. 5.** Разрезание прямоугольников на квадраты наибольшей площади

раты, на которые он разрезается. Кнопка «Построить» становится недоступной. Пользователю доступна кнопка с названием «Очистить». При щелчке по ней очищаются текстовые строки и графическая область. Доступной становится кнопка «Построить», и после ввода значений ширины и высоты можно получить решение задачи.

Созданный класс реализует интерфейс **ActionListener**, содержащий метод **actionPerformed**, который реализован в описанном классе. Проверяется, введены ли значения в строки для ввода текста, и если строки не пусты, то проверяется, какая кнопка была нажата. Далее либо строится прямоугольник, либо стирается предыдущий вариант решения задачи. В листинге 5 приведена программа, решающая задачу.

Апплет показан на рис. 5. Обратите внимание на то, что прямоугольник располагается в центре области апплета.

### ФРАЗА – ПАЛИНДРОМ

Напишем программу, определяющую, является ли заданная фраза палиндромом. Палиндромом считается строка, которая читается одинаково как слева направо, так и справа налево. Палиндромами являются слова: казак, кок, шалаш и др. Палиндромами могут быть фразы, обычно считается, что пробелы между словами, знаки препинания и различия между маленькими и большими буквами игнорируются. Например, палиндромами являются фразы: «А роза упала на лапу Азора», «Кит на море не романтик» и др.

**Листинг 5.** Разрезание прямоугольника на квадраты

```
import java.applet.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class Kvadrat extends Applet implements ActionListener{
     int xc;  // координаты центра
     int yc;  // графической области
     TextField i1, i2; // строки ввода для операндов
     Button cr; // кнопка для рисования
```

```
    Button res; // кнопка для очистки области рисования
// действия при открытии страницы
    public void init (){
       setSize (500, 500); // размер области
       xc = getWidth()/2;
       yc = getHeight()/2;
       Label i1p = new Label("Ширина: ", Label.RIGHT);
       Label i2p = new Label("Высота: ", Label.RIGHT);
       i1 = new TextField(3);
       i2 = new TextField(3);
       cr = new Button("Построить");
       res = new Button("Очистить");
       add(i1p); add(i1);    add(i2p); add(i2);
       add(cr); add(res);
       cr.addActionListener(this);
       res.addActionListener(this);
    }
// рекурсивный метод для демонстрации построения квадратов
  public void div (int x, int y, int w, int h){
     Graphics g = this.getGraphics();
     g.drawRect(x, y, w, h);
     if (w!=h){
         if (w > h) {
               x = x + h;
               w = w - h;
         } else {
               y = y + w;
               h = h - w;
         }
     div (x, y, w, h);
     }
  }
//обработка событий     от действий пользователя
  public void actionPerformed(ActionEvent ae){
     if (!i1.getText().equals("")&!i2.getText().equals("")){
//значения введены
     int  w = Integer.parseInt(i1.getText());
     int  h = Integer.parseInt(i2.getText());
//определение источника события
     String str = ae.getActionCommand();
     if(str.equals("Построить")){
          div (xc-w/2, yc-h/2, w, h);
          cr.setEnabled(false);
          res.setEnabled(true);
    } else {
           if(str.equals(«Очистить»)){
              i1.setText(«»); // очистить первую строку ввода
              i2.setText(«»); // очистить вторую строку ввода
              cr.setEnabled(true);
              res.setEnabled(false);
              repaint();
           }
         }
      }
  }
}
```

**Листинг 6.** Фраза-палиндром

```
import java.awt.*;
import java.applet.*;
public class Pal_str extends Applet{
// программа проверяет, является ли фраза палиндромом.
// слова разделяются произвольным числом пробелов
      TextField ti1, ti2;
      Label i1, i2;
      public void init(){
          i1 = new Label("Введите текст: ", Label.LEFT);
          ti1 = new TextField("Кит   на море   не романтик",20);
          i2 = new Label("результат  ", Label.LEFT);
          ti2 = new TextField(12);
          ti2.setEditable(false);
          add(i1);  add(ti1); add(i2);  add(ti2);
      }
      boolean fpol (String s){
          int n = s.length(); //длина фразы
          int i = 0;    //индекс очередного символа строки слева
          int j = n-1; //индекс анализируемого символа справа
            char c1 = s.charAt(i); //очередной символ слева
           char c2 = s.charAt(j);; //очередной символ справа
          boolean p = true; //изменится на false, если символы не равны
           while ((i < j) & p) {
             if (c1 == " "){ // пробел слева пропускаем, сдвигаясь вправо
                       i = i+1; c1 = s.charAt(i);
              } else {
                     if (c2 == " "){ //пробел справа пропускаем, сдвигаясь влево
                         j = j-1; c2 = s.charAt(j);
                       } else  //оба символа не пробелы
                         if (c1 != c2) {
                           p = false;  //фраза не палиндром
                           break;
                         }
                         else { i = i+1; c1 = s.charAt(i);
                                j = j-1; c2 = s.charAt(j);
                         }
                    }
             };
           return p;
      }
// обработка событий
      public boolean action (Event evt, Object whichAcion) {
        if (evt.target instanceof TextField){
          TextField cur = (TextField)evt.target;
          if (cur == ti1) { // событие от первой строки
      String s = cur.getText().toLowerCase();
      if (fpol (s))
          ti2.setText("палиндром");
      else
          ti2.setText("не палиндром");
               return true;
         } else return false; // событие не обработано
        } else return false;
      }
}
```

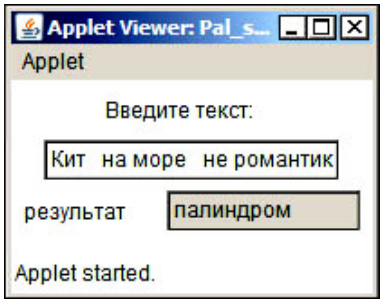



**Рис. 6.** Фраза-палиндром

Будем анализировать введенный текст с обоих концов, пропуская пробелы. Переменная `i` равна индексу левого анализируемого символа строки, переменная `j` равна индексу правого анализируемого символа строки. В начальный момент значение `i` равно `0`, значение `j` равно индексу последнего символа в строке.

Если оба анализируемых символа не являются пробелами, то они сравниваются. Если символы различны, то анализ текста можно завершить, – фраза не является палиндромом. Если же очередные символы одинаковы, то анализ фразы следует продолжать.

Заметим, что значение `i` в программе может только увеличиваться, значение `j` только уменьшаться. Процесс проверки следует прекратить в случае, когда значение `i` станет больше или равно значению `j`. Это означает, что просмотрена вся строка, и символы на соответствующих местах одинаковы. В листинге 6 приведена программа, содержащая метод, определяющий, является ли предложение палиндромом. Апплет показан на рис. 6.

## УПРАЖНЕНИЯ

1. Написать программу, которая проверяет, является ли идентификатором последовательность символов, введенных в строку ввода. Будем считать идентификатором непустую последовательность букв латинского алфавита и цифр, начинающуюся с буквы. Буквы можно использовать как прописные, так и строчные.

2. В области апплета расположено четыре изображения. В строке ввода указывается номер удаляемого изображения. После ввода номера указанное изображение должно быть удалено.

3. Написать программу, рисующую салфетку Серпинского. Салфетка Серпинского строится следующим образом. Рисуется треугольник и в нем средние линии. В получившихся треугольниках, на которые разбивается исходный, опять рисуются средние линии и т. д. до заданного уровня вложенности. С помощью строки ввода задается уровень вложенности.



***Дмитриева Марина Валерьевна,***
***доцент кафедры информатики***
***математико-механического***
***факультета Санкт-Петербургского***
***государственного университета.***