Программирование GUI

...

Принципы и примеры реализации

GUI - разновидность пользовательского интерфейса, элементы которого выполнены в виде графических изображений располагаются на дисплее устройства.

Отличие от CUI

Достоинства

- Легкость для неопытных пользователей
- Функция Drag’n’Drop
- Более “дружелюбный интерфейс”

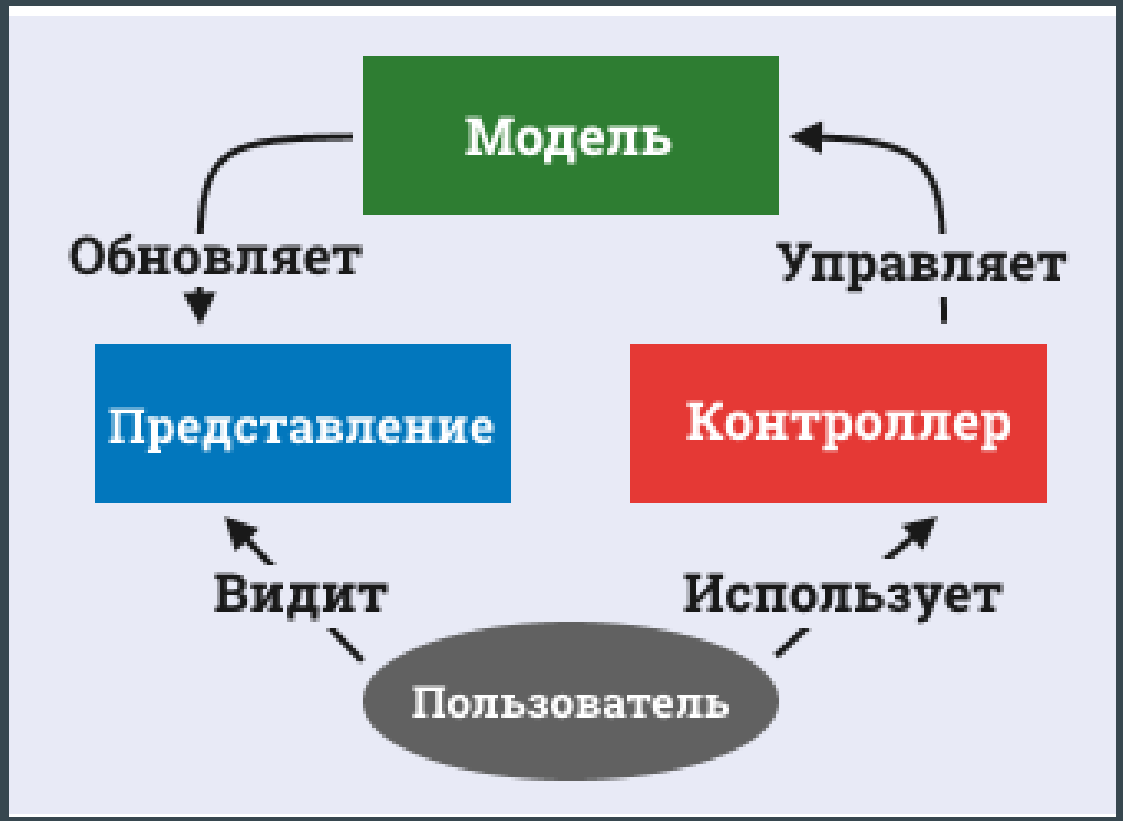
Недостатки

- Большее потребление памяти в сравнении с текстовым интерфейсом
- Невозможность автоматизации
- Требует повышенного внимания
- Длительное время разработки

Принципы организации и реализации

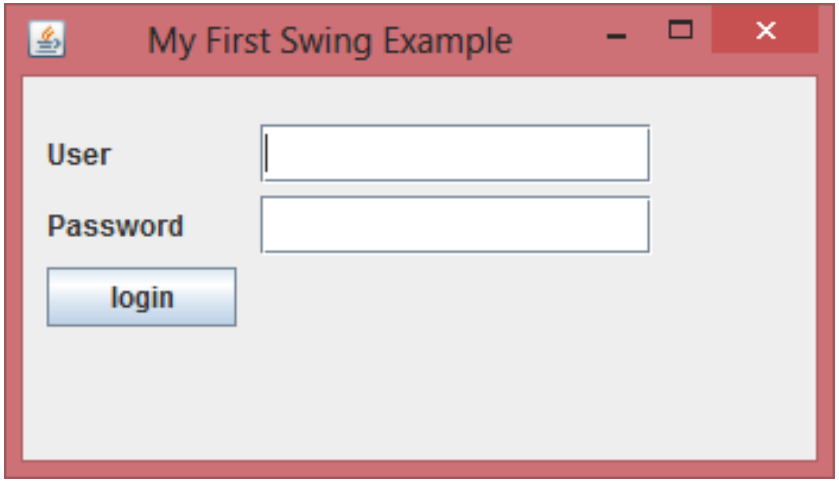
- Видимость статуса системы
- Соответствие между системой и реальным миром
- Управляемость и свобода для пользователя
- Согласованность и стандарты
- Предотвращение ошибок
- Распознавать лучше, чем вспоминать
- Гибкость и эффективность использования
- Эстетический и минималистический дизайн
- Помочь пользователю понять и исправить ошибку
- Справка и документация

MVC

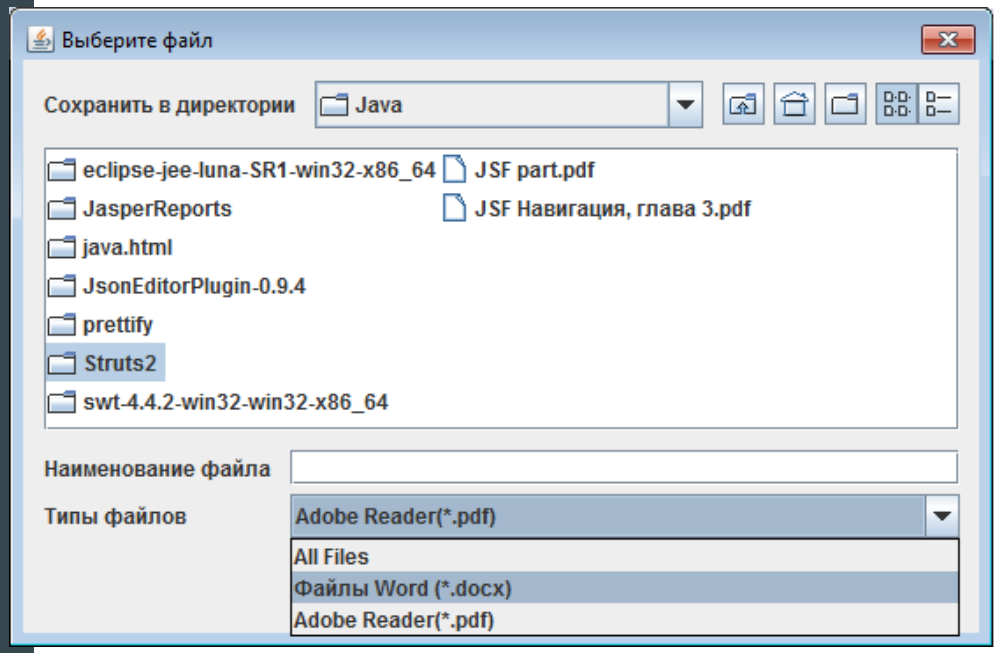

Модель
Обновляет
Управляет
Представление
Контроллер
Видит
Использует
Пользователь

Библиотеки в языке Java

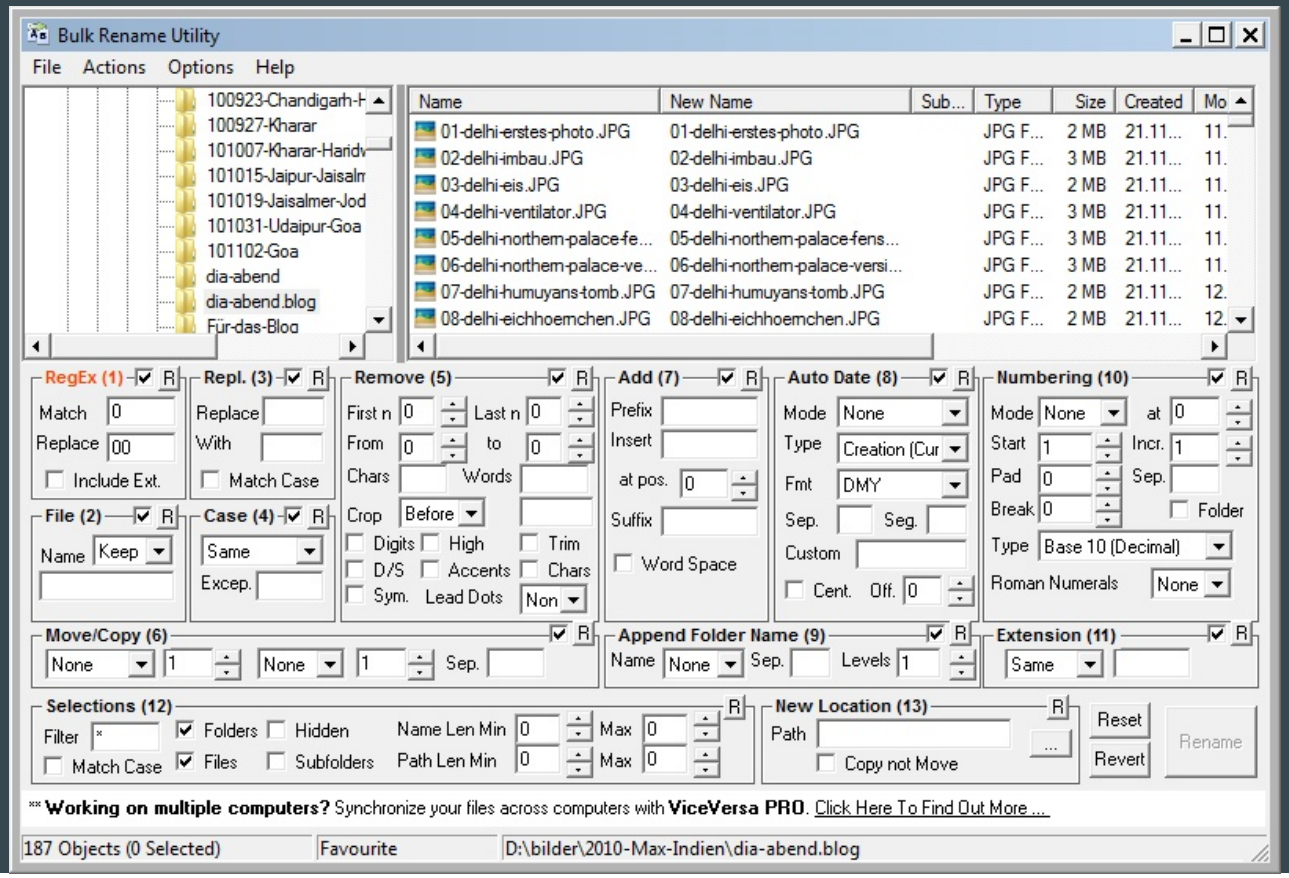
Пример простого GUI



Пример сложного GUI

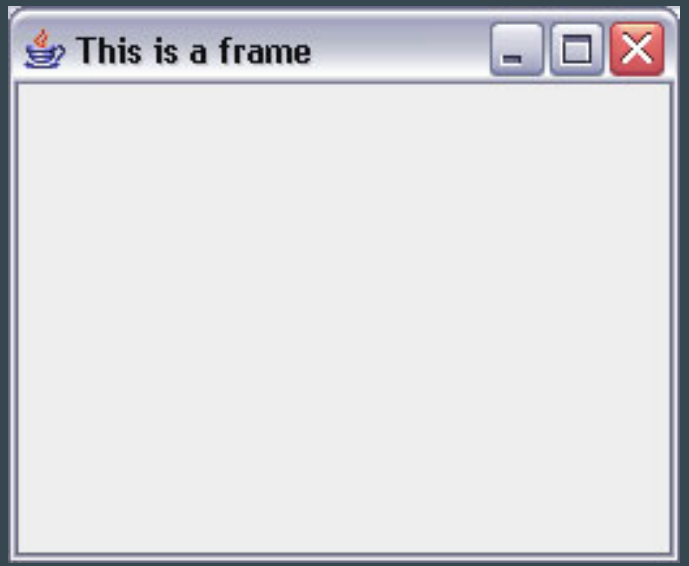


Пример плохого GUI



Основные Swing-компоненты

JFrame




```
public class JMainExample {
 public static void main(String[] arguments) {
    JFrame frame = new JFrame("This is a frame");
    frame.setSize(300,200);
  }
}
```

JComponent

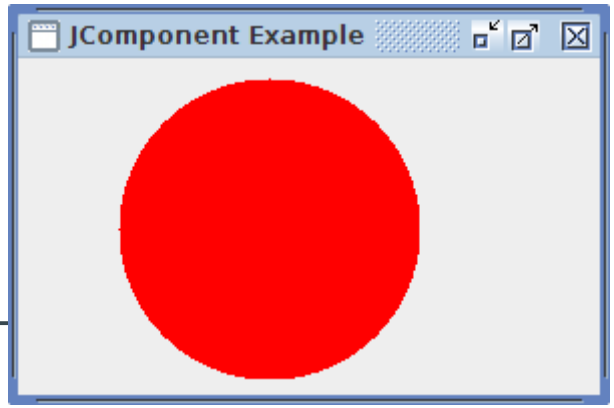
Основной класс всей библиотеки визуальных компонентов Swing

```
class MyOval extends JComponent {
  public void paint(Graphics g) {
    g.setColor(Color.red);
    g.fillOval(50, 10, 150, 150);
  }
}
```

Добавление компонента в JFrame

```
// ... we are in the main function

MyOval oval = new MyOval();
      frame.add(oval);
```




```
JLabel label = new JLabel();
jLabel.setText("This is a label");
```

JButton


```
JButton button = new JButton();
button.setBounds(103, 110, 71, 27);
button.setText("OK");
```

JTextField


```
JTextField textField = new JTextField();
textField.setBounds(96, 49, 160, 20);
```

Дополнительные Swing-компоненты

JComboBox, JPasswordField, JCheckBox, JRadioButton, JMenu, JMenuItem, JMenuBar, JSlider, JSpinner, JToolBar, JToolTip, JOptionPane, JTextArea, JScrollPane, JList, JTable, JTree ...

Схемы (Layout)

setBounds(x, y, w, h)

```
button.setBounds(103, 110, 71, 27);

textField.setBounds(96, 49, 160, 20);
```

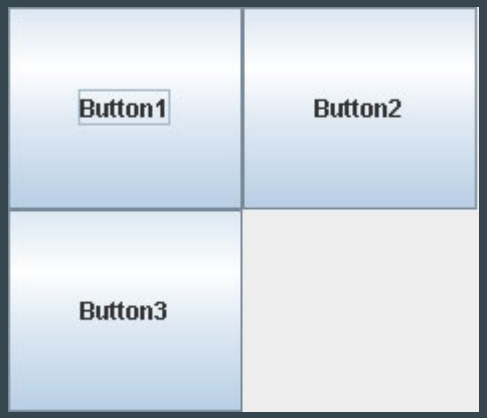
---

x - координата по оси OX, y - координата по оси OY,
w - ширина, h - высота

FlowLayout

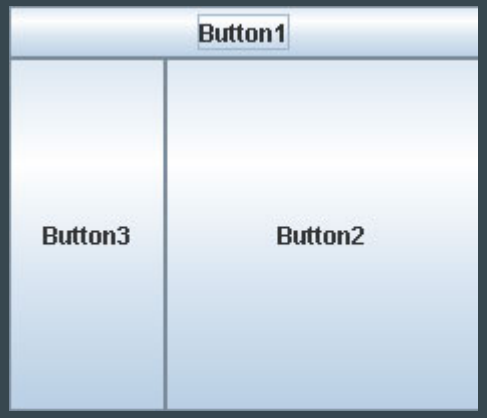

```
setLayout(new FlowLayout());
add(new JButton("Button1"));
add(new JButton("Button2"));
add(new JButton("Button3"));
```

GridLayout




```
setLayout(new GridLayout(1, 2));
add(new JButton("Button1"));
add(new JButton("Button2"));
add(new JButton("Button3"));
```

BorderLayout




```
setLayout(new BorderLayout());
add(new JButton("Button1"), "North");
add(new JButton("Button2"), "Center");
add(new JButton("Button3"), "West");
```

События и слушатели событий

ActionListener и ActionEvent

```
JButton b = new JButton("Button");
b.addActionListener(new HelloListener());

class HelloListener implements ActionListener
{
  // Метод интерфейса для получения нажатий кнопки
  public void actionPerformed(ActionEvent e)
  {
    System.out.println("Hello");
  }
}
```

ListSelectionListener и ListSelectionEvent

```
//  myList - это JList, заполненный данными

myList.addListSelectionListener(new
ListSelectionListener() {
  public void valueChanged(ListSelectionEvent e)
  {
    Object o = myList.getSelectedItem();
    System.out.println(o.toString());
  }
});
```

Модель

ComboBoxModel

Пример реализации пользовательской модели на основе базовой

```
MyComboModel model = new MyComboModel(alphaList);
myComboBox.setModel(model); // JComboBox component


public class MyComboModel implements ComboBoxModel
{
  private List data = new ArrayList();
  private int selected = 0;

  public MyComboModel(List list) {
    data = list;
  }

  public void setSelectedItem(Object o) {
    selected = data.indexOf(o);
  }

  public Object getSelectedItem() {
    return data.get(selected);
  }
}
```

Пример реализации приложения

Шаг 1

Выбрать расположение компонентов в окне

С помощью Layout'ов настроить расположение компонентов.

(современные редакторы кода позволяют делать это визуально при помощи Drag'n'Drop)

Шаг 2

Привязка данных

Приложению необходимо работать с данными.

“Представление” отрисовывает полученные данные.

Данные создаются в классе DataHandler.

Шаг 3

Управление событиями

Необходимо рассмотреть сценарий работы приложения и описать все действия[1] с помощью программного кода.

---

[1] - Необходимо использовать метод addActionListener()

Шаг 4

Использование моделей

Модели обеспечивают двустороннюю привязку между “контроллером” и “представлением” приложения.

```
comboModel1 = new
ItemsComboModel(DataHandler.getItems());
comboModel2 = new
ItemsComboModel(DataHandler.getItems());
```

DataHander необходим для извлечения и хранения данных

Шаг 5

Обработка исключений и ошибок

```
try
{
  // выполнить действие, которое может создать ошибку в
работе приложения
  DataHandler.updateRecords(o, tixx);
}
catch (Exception ex) {
  // отобразить здесь сообщение об ошибке
  JOptionPane.showConfirmDialog(
        this, // указатель на наш класс
        ex.getMessage(), // берем текст ошибки из ex
        "Error", // название окна
        JOptionPane.OK_CANCEL_OPTION, // кнопки
        JOptionPane.ERROR_MESSAGE // вид окна
  );

}
```

Спасибо за внимание!