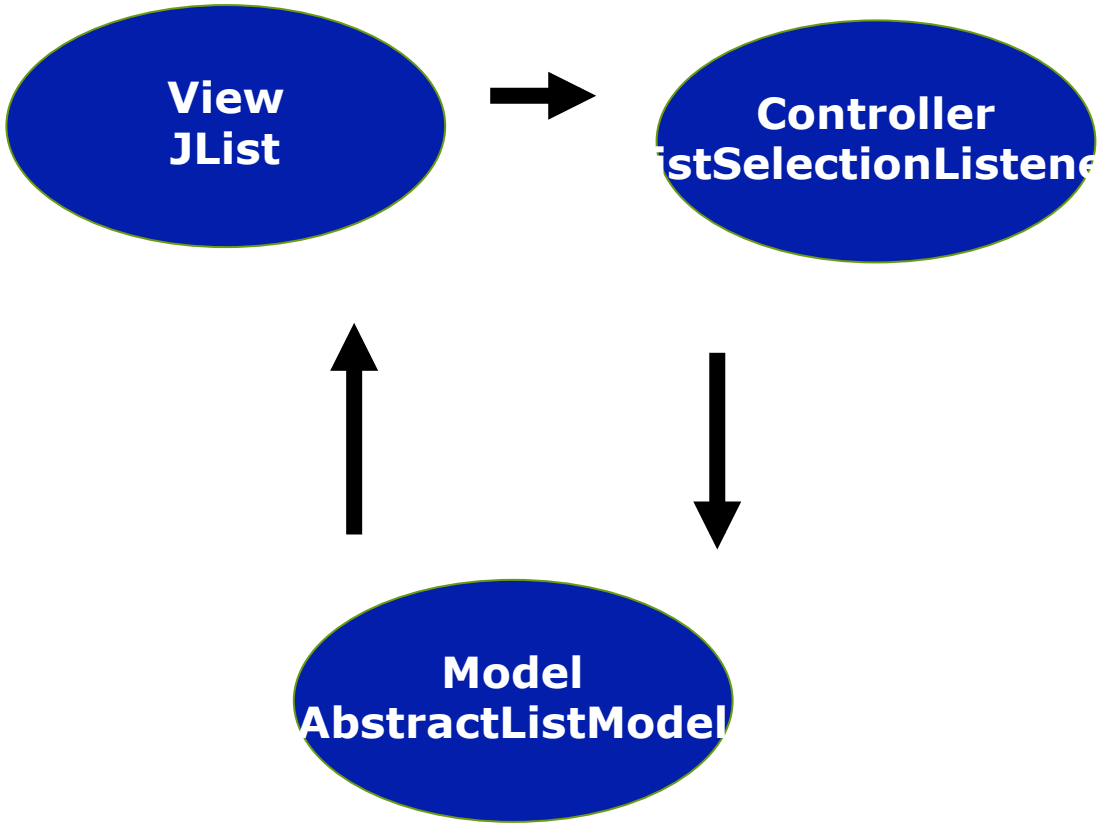
JList und MVC-Architektur

MVC bei der JList



AbstractListModel - Rahmen

Diese Methoden müssen überschrieben werden!

```
public class JListData extends AbstractListModel {

    public JListData() {
        // TODO Auto-generated constructor stub
    }

    public Object getElementAt( int arg0) {
        // TODO Auto-generated method stub
        return null;
    }

    public int getSize() {
        // TODO Auto-generated method stub
        return 0;
    }
}// end of class
```

Beispiel: AbstractListModel

```
public class JListData extends AbstractListModel {

    /**
    * Deklarieren eines Vector-Datenspeichers.
    */
    private Vector <String> dlist;

    /**
    * Referenz auf das Hauspfenster.
    */
    private JFrame fenster;

    /**
    * Konstruktor
    * @param Referenz auf das Hauspfenster.
    */
    public JListData( JFrame f) {
          fenster = f; // Zuweisung von f zu fenster
          dlist = new Vector <String> (); // Initialisierung eines neuen
    Vector-Objekts
          makeData();//Registerkartenbennenung wird im Vector
    gespeichert
    }
```

Fortsetzung nächste Seite

Beispiel: AbstractListModel

```
/**
* Die Methode makeData liest Daten ein und speichert sie im Datenspeicher
* @return Datenspeicher
*/
private Vector <String> makeData()
{
	String[] data = {"Otto", "Gustav", "Hugo"};  // Daten

	for(int i = 0; i < data.length; i++){
		dlist.addElement( (String) data[i]); // Speicherung der Daten
	}
	return dlist; // Rückgabe des Datenspeichers
}
```

Fortsetzung nächste Seite

Beispiel: AbstractListModel

```
    /**
    * Die Methode getElementAt holt das selektierte Element an der Stelle des
    * Indexes.
    * @param index ist der Index, der anzeigt, welches Element geholt werden
    * soll.
    * @return String am Index i
    */
    public Object getElementAt( int index )
    {
          return dlist.elementAt( index ); // Rückgabe des Elements an der Pos.
    }


    /**
    * Die Methode getSize gibt die Größe des Datenspeichers zurück.
    * @return Größe des Datenspeichers
    */
    public int getSize()
    {
          return dlist.size(); //Rückgabe der Größe des Datenspeichers
    }
}// end of class
```

JList konstruieren

im Konstruktor des Hauptfensters

```
JListData dataModel = new JListData( this );  /* Instanzierung eines JListData-Objektes*/

JList liste = new JList ( dataModel );
/* Instanzierung einer JList, die als Übergabeparameter das JListData-Objekt erhält.*/

liste.setLayoutOrientation(JList.VERTICAL); // Layout der liste festlegen
liste.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
/* Selektionmodus wählen,  hier:  Mehrfachselektion*/

liste.setVisibleRowCount(-1); // keine Zeile vorselektiert

JScrollPane scrollPane = new JScrollPane( liste );
/* Instanzierung einer JScrollPane, die als Übergabeparameter das JList-Objekt hat.*/

scrollPane.setPreferredSize(new Dimension(450, 260));
//Festlegung der Grösse des scrollPanels

liste.addListSelectionListener( new ListSelectionAdapter( liste ) );
/* Anbinden des ListSelectionListener an die liste */
```

ListSelectionListener implementieren

```
class ListSelectionAdapter implements ListSelectionListener
{
    JList meineListe; // Deklaration eines JList-Objektes

    /** Der Konstruktor */
    public ListSelectionAdapter( JList liste ) {
          meineListe= liste; // JList liste wird in meineListe abgespeichert
    }

    /**
    * Die Methode valueChanged wird aufgerufen, bei Auswahl der Werte
    * @param ev ist das eingetretene Ereignis.
    */
    public void valueChanged( ListSelectionEvent ev ) {
          if( ev.getValueIsAdjusting() == false){
                String  selectedName =
    (String)meineListe.getSelectedValue();
                int selectedIndex = meineListe.getSelectedIndex();
          }
    }
}// end of inner class ListSelectionAdapter
```