Workshop GUI (2)

Customer Renderers

- The standard way of rendering data in lists and talbes
  - Example: JList

Example Custom Renderer
Line is a model class

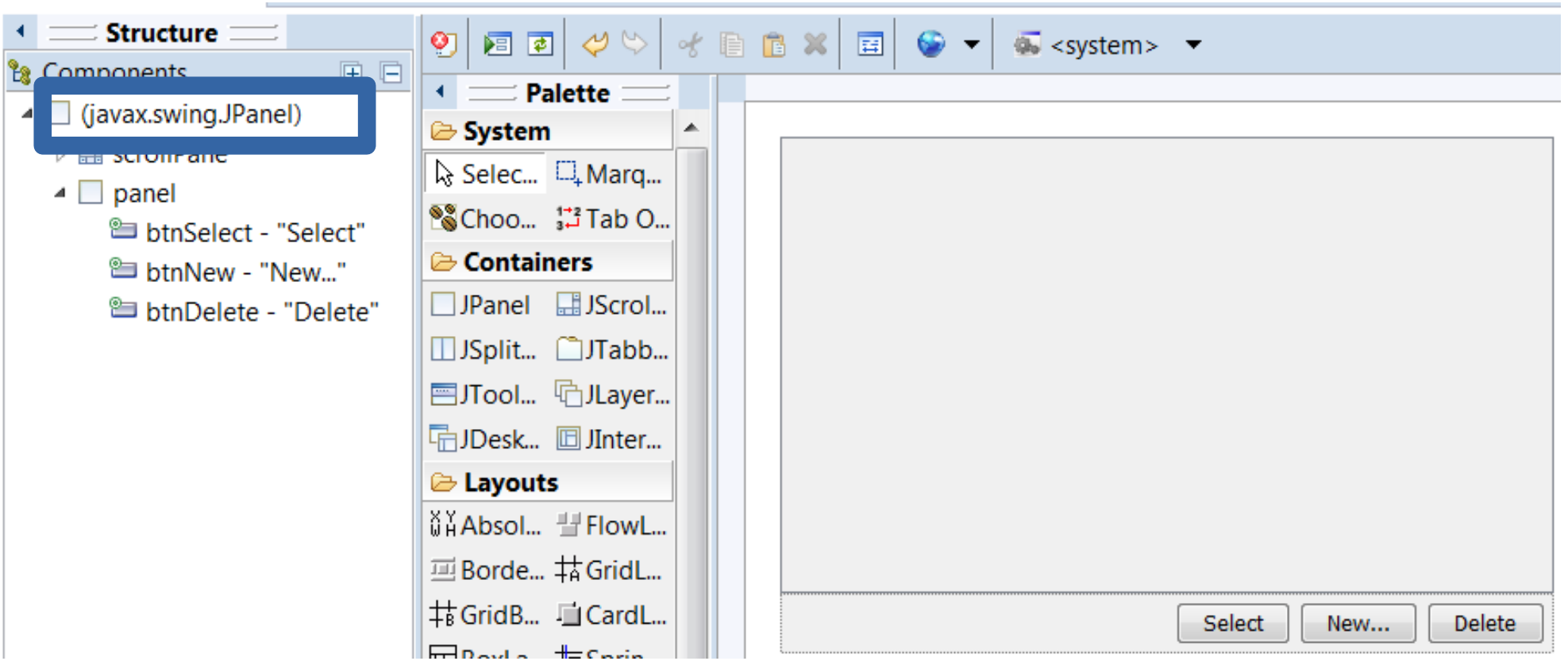
```
package gui;

import ...

public class CustomCellRenderer implements ListCellRenderer<Line> {

	@Override
	public Component getListCellRendererComponent(JList<? extends Line> arg0,
			Line arg1, int arg2, boolean arg3, boolean arg4) {
		JLabel label = new JLabel("");
		label.setText(arg1.getName() + " ~ " + arg1.getQuantity());
		label.setOpaque(true);
		if(arg3) {
			label.setForeground(arg0.getSelectionForeground());
			label.setBackground(arg0.getSelectionBackground());
		} else {
			label.setForeground(arg0.getForeground());
			label.setBackground(arg0.getBackground());
		}
		return label;
	}
}
```

Usage of Customer Renderer

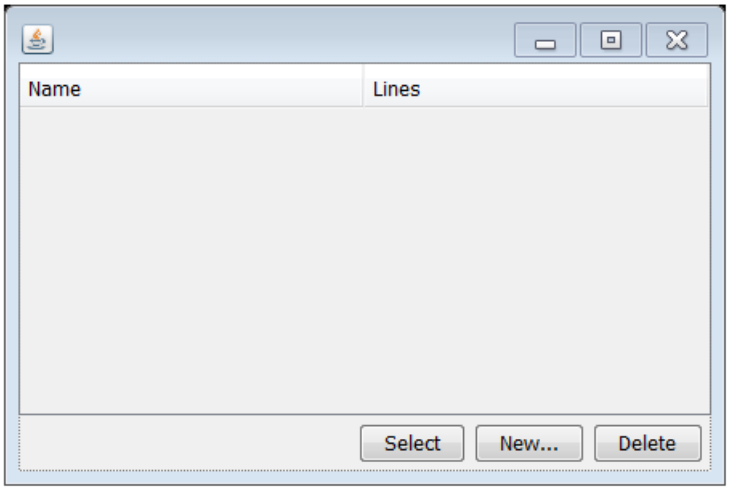
- In the constructor of a GUI class:

```
lstGrocery = new JList<>();
lstGrocery.setCellRenderer(new CustomCellRenderer());
```

Reusable Components




```
public GroceryListsWindow() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new GroceryLists();
        setContentPane(contentPane);
}
```



JTable + DefaultTableModel

- Like JList has ListModel
- JTable has a TableModel
  - A default implementation is called DefaultTableModel
  - We can e.g. subclass this and modify its behavior
  - This is what you find in GroceryList4

MyTableModel (Inner Class) Overriding the interface methods

```
private class MyTableModel extends DefaultTableModel {

    @Override
    public int getColumnCount() {...

    @Override
    public int getRowCount() {...

    @Override
    public String getColumnName(int ix) {...

    @Override
    public Object getValueAt(int row, int col) {...

}
```

https://docs.oracle.com/javase/7/docs/api/javax/swing/table/TableModel.html

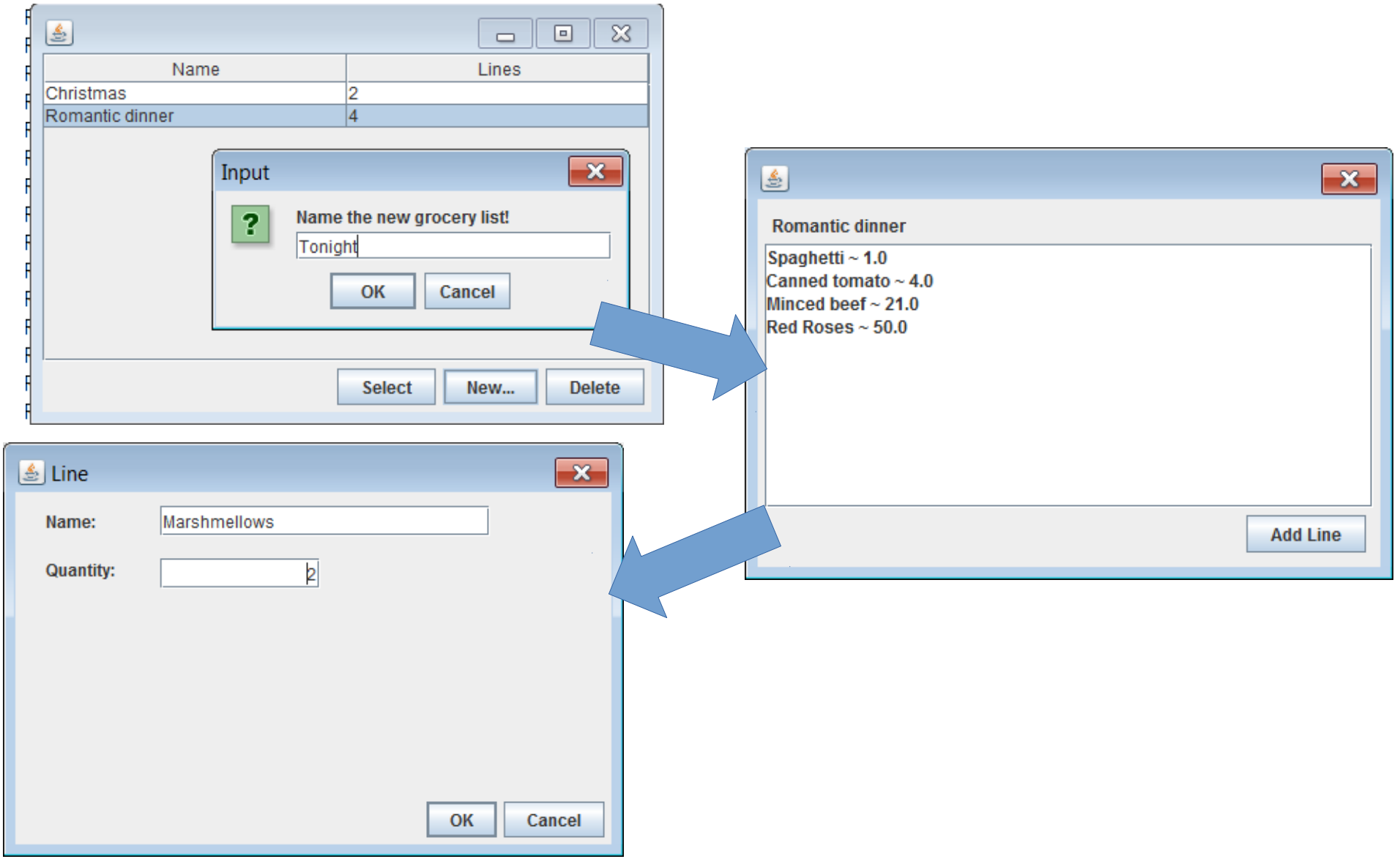
Custom Part

```
private class MyTableModel extends DefaultTableModel {
	private static final long serialVersionUID = 1L;
	private List<GroceryList> gLists;

	public MyTableModel (){
		gLists = new ArrayList<>();
	}

	// Not part of the framework, we add this to be able to change the data model in the table!
	public void setData(List<GroceryList> data) {
		this.gLists = data;
		super.fireTableDataChanged();
	}

	public GroceryList getData(int selectedRow) {
		if(selectedRow >= 0 && selectedRow < gLists.size()) {
			return this.gLists.get(selectedRow);
		}
		return null;
	}

}
```

Exercise

- Assuming that you have a working grocery list editor
- … add a window that displays all your grocery lists
  - Implement Create, Edit, Delete on this window
  - Use a JTable to show your lists
    - Column 1: Name of list
    - Column 2: Line count in each list
  - Also add missing controller and container methods as necessary

Navigation


Name
Lines
Christmas
2
Romantic dinner
4
Input
Name the new grocery list!
Tonight
OK
Cancel
Select
New...
Delete
Romantic dinner
Spaghetti ~ 1.0
Canned tomato ~ 4.0
Minced beef ~ 21.0
Red Roses ~ 50.0
Add Line
Line
Name:
Marshmellows
Quantity:
2
OK
Cancel