Software 1

Recitation No. 10:
SWT GUI Package
The GUI Development Process

- **GUI**: Graphical User Interface

  - User Interface Engineer
  - Graphic Designer
  - GUI Programmer
When implementing a GUI application one should specify:

- the GUI elements
- the 2D arrangement of the GUI elements
- the behavior of the GUI elements

Java GUI libraries:

- AWT (Abstract Windowing Toolkit)
- Swing
- SWT (Standard Widget Toolkit)
Model-View Separation

- Separate between the application logic (model) part and the GUI (view) part.
- Ensures that view changes have no effect on the basic model.
- Enables us to maintain one model for several different views.
Example: Address Book

Address Book Application

- AddressBook class
- GUIAddressBookViewer class

The Model

The View
The Model

**AddressBook**
- void add(Contact c);
- Contact get(String name);
- void delete(String name);
- void modify(Contact c);
- Contact search(String prefix);
- Iterator<Contact> getContacts();
- int getCount();
- Iterator<Contact> search(String prefix);
- void save(String filename);
- void load(String filename);

**Contact**
- String name;
- String email;
- String telephone;
- Address address

**Address**
- String street;
- String zipCode;
- String country

**Notation:**
- `<class>`
- `Contact`
- `Address`
- `has-a`
The View

New Address Book Ctrl+N
Open Ctrl+O
Save Ctrl+S

New Contact Ctrl+M
Edit
Delete
The View

- The class diagram:

- The implementation:
  - based on the SWT GUI library
SWT

- Online Documentation:
    - JavaDoc
    - Snippets
Widgets

Widget

Control

ScrollBar
Menu

Button
Label

Scrollable

Composite

Shell
Table
A Layout controls the position and size of Control widgets in a Composite.
GridLayout

- Lays out the Control widgets in a grid.

Each column is as wide as Wide Button 2.
**GridLayout**

- Lays out the `Control` widgets in a grid.
- **GridLayout Configuration fields:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontalSpacing</td>
<td>5</td>
<td>Horizontal/vertical space between the grid cells</td>
</tr>
<tr>
<td>verticalSpacing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>marginHeight</td>
<td>5</td>
<td>The size of the horizontal/vertical margins of the layout</td>
</tr>
<tr>
<td>marginWidth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>numColumns</td>
<td>1</td>
<td>Number of columns</td>
</tr>
<tr>
<td>makeColumnsEqualWidth</td>
<td>false</td>
<td>If true, all columns will have the same size</td>
</tr>
</tbody>
</table>
GridLayout (cont.)

GridData:

- Use `GridData` objects to configure the Control widgets in a GridLayout.

- Use the `setLayoutData()` method to set a `GridData` object into a Control, e.g.

  ```java
  label.setLayoutData(new GridData(...));
  ```

- Do not reuse `GridData` objects
### GridLayout (cont.)

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grabExcessHorizontalSpace</td>
<td>false</td>
<td>If true, the width/length of the widget will be as large as possible to fit the remaining space.</td>
</tr>
<tr>
<td>grabExcessVerticalSpace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>heightHint</td>
<td>SWT.DEFAULT</td>
<td>A minimum width/height for the widget.</td>
</tr>
<tr>
<td>widthHint</td>
<td>(no minimum)</td>
<td></td>
</tr>
<tr>
<td>horizontalSpan</td>
<td>1</td>
<td>the number of column/row cells that the widget will take up.</td>
</tr>
<tr>
<td>verticalSpan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>horizontalIndent</td>
<td>0</td>
<td>the number of indentation pixels along the left side of the cell.</td>
</tr>
<tr>
<td>horizontalAlignment</td>
<td>GridData.BEGINNING</td>
<td>how controls will be positioned horizontally/vertically within a cell.</td>
</tr>
<tr>
<td>verticalAlignment</td>
<td>GridData.CENTER</td>
<td></td>
</tr>
</tbody>
</table>
FormLayout

- A very flexible layout
FormLayouts

- A very flexible layout
- FormLayout Configuration Properties:

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>marginHeight</td>
<td>0</td>
<td>the margin width/height</td>
</tr>
<tr>
<td>marginWidth</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>spacing</td>
<td>0</td>
<td>the number of pixels between the edge of one control and the edge of its neighbouring control.</td>
</tr>
</tbody>
</table>
FormLayouts (cont.)

- Use FormData objects to configure the Control widgets in a FormLayout.
- Use the setLayoutData() to set a FormData object into a Control widget.
- A FormData object has a FormAttachment object for each edge of the Control.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width/height</td>
<td>the desired width/height in pixels.</td>
</tr>
<tr>
<td>top/bottom/left/right</td>
<td>Specifies the position of the control attachment.</td>
</tr>
</tbody>
</table>
A **FormAttachment** defines where to attach the side of a `Control` by using the equation: $y = ax + b$.

A fraction defined by: 
- **numerator**
- **denominator**

an **offset**, in pixels

the width/height of a `Control` to which the control side is attached (**control**).
FormLayouts (cont.)

Main FormAttachment Constructors:

- public FormAttachment(Control control)
- public FormAttachment(Control control, int offset)
- public FormAttachment(int numerator)
- public FormAttachment(int numerator, int offset)

<table>
<thead>
<tr>
<th>Field</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>Parent Composite</td>
</tr>
<tr>
<td>numerator</td>
<td>100</td>
</tr>
<tr>
<td>denominator</td>
<td>100</td>
</tr>
<tr>
<td>offset</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
y = \frac{numerator}{denominator} \bullet x + offset
\]

\(x\) – control's width/height