Grafische Benutzeroberflächen mit Java

Um eine grafische Benutzeroberfläche (***GUI ... graphical user interface*** – nicht zu verwechseln mit einem Java `interface`) zu erstellen gibt es in Java standardmäßig mehrere Bibliotheken.

1. **Abstract Window Toolkit (AWT)**: Das AWT stellt die älteste GUI-Bibliothek in Java dar. Sie hat folgende grundlegende Eigenschaften und Funktionen:
   - GUI-Komponenten (Buttons, Textfelder, Labels, ...), deren Darstellung und Erstellung an das Betriebssystem weitergeleitet wird. Dadurch ist AWT relativ schnell und robust, allerdings muss sich die Funktionalität auf den kleinsten gemeinsamen Nenner aller Betriebssysteme beschränken
   - Ereignissteuerung für die GUI-Komponenten
   - Basis-Klassen für Grafiken, wie z.B. Zeichenklassen und Schrift-Klassen
   - Layout-Manager, damit die GUI-Komponenten flexibel und größenangepasst im Fenster angeordnet werden können
   - Datenfluss-Klassen, mit denen z.B. Copy/Cut-and-Paste-Funktionen über das Betriebssystem abgewickelt werden können.
2. **swing bzw. Java Foundation Classes (JFC)**: swing ist ein Teil der JFC, mit denen sehr reichhaltige GUI-Oberflächen gestaltet werden können. Die JFC bestehen aus folgenden Bereichen
   - swing-GUI-Komponenten, die komplett von Java selbst gezeichnet werden und daher mehr Möglichkeiten bieten. Dadurch benötigen sie aber mehr Ressourcen (Arbeitsspeicher, Prozessorzeit). Swing-GUIs gelten daher als langsamer.
   - Look-And-Feel, das angepasst werden kann. Unter LAF oder auch PLAF (für pluggable look and feel) versteht man das grundsätzliche Aussehen einer GUI. Eine Windows basierte GUI unterscheidet sich auf den ersten Blick von einer Unix oder Mac basierten GUI in der grundsätzlichen Form der Komponenten (mehr abgerundete Ecken, Pseudo-3D-Effekte, Farbgebungen, ...). Dieses kann bei swing-GUIs während der Laufzeit angepasst werden, so dass sie ihren Stil an das geladene Look-And-Feel anpassen.
   - Bibliotheken zur Unterstützung für Geräte zur Erhöhung der Zugänglichkeit (z.B. für Braille-Anzeigen, das sind Anzeigen in der Blindernschrift)
   - Java 2D-API, die verbesserte Grafik-Funktionalitäten bietet
   - Unterstützung für Internationalisierung (unterschiedliche Sprachpakete)

   Die swing-Bibliotheken bauen auf die AWT-Bibliotheken auf. Daher sind die AWT-Bibliotheken nachwievor für die Entwicklung einer GUI mit swing wichtig.
3. **Java FX**, die neueste Grafik-Bibliothek, die mit dem Ziel entwickelt wurde, die GUI-Entwicklung mit Java zu vereinfachen. Allerdings ist die Basis dieser Vereinfachung XML, d.h. XML ist eine Voraussetzung. Java FX kann daher sowohl als normale GUI-Applikation als auch für den Browser verwendet werden.

Da Java FX durch seine Neuheit noch mit einigen Schwierigkeiten zu kämpfen hat, wird sich dieses Skriptum auf swing stützen.

Top-Level-Container

Damit man eine GUI aufbauen kann braucht man eine Fläche – ein Fenster – in dem alle GUI-Inhalte – das sind die GUI-Komponenten – platziert werden können. D.h. wir haben nun folgende wichtige Begriffe im Zusammenhang mit GUIs:

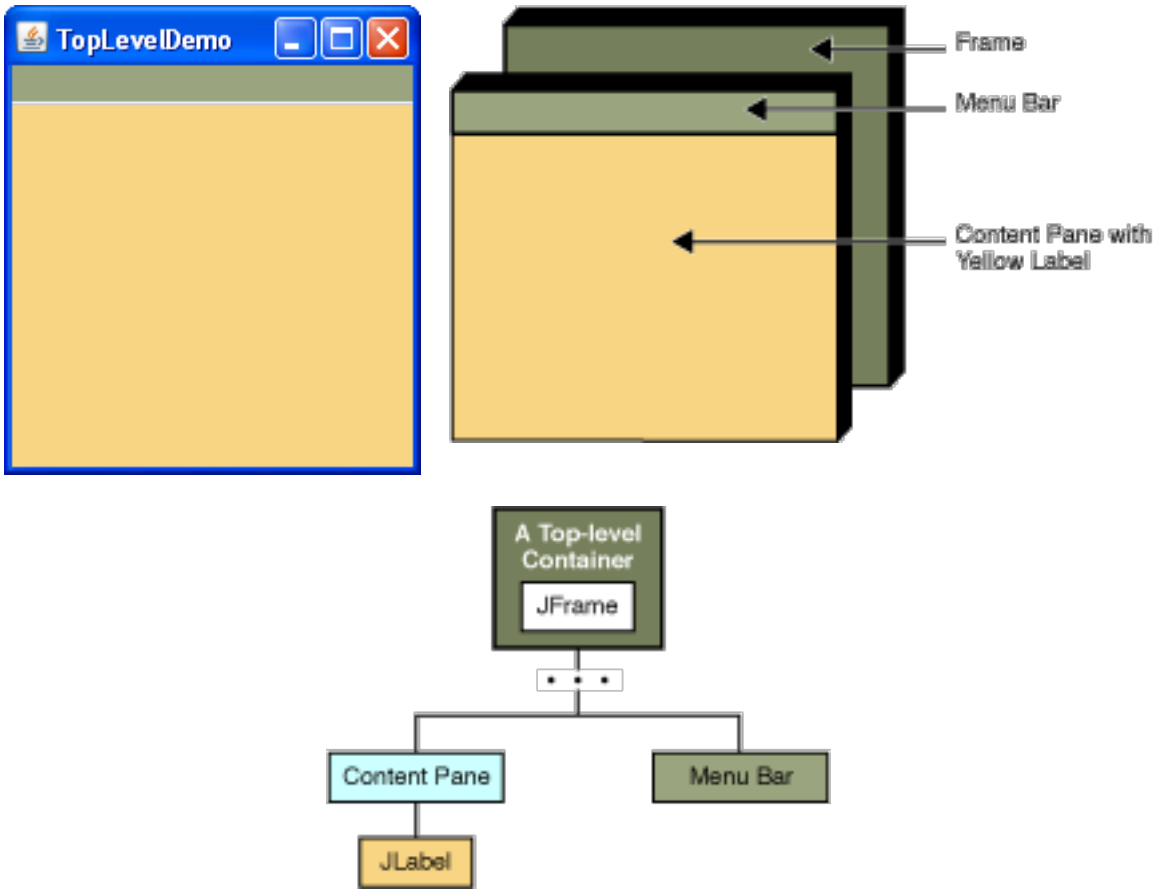
- GUI-Komponente: ein grafisches Objekt, das in einer GUI platziert werden kann (z.B. ein Button, ein Textfeld, eine Beschriftung, ein Bild, ...)
- Container: eine spezielle GUI-Komponente, die andere GUI-Komponenten enthalten kann

- Top-Level-Containter: ein Container, der direkt vom Betriebssystem (oder im Fall eines Applets vom Browser) angezeigt werden kann und der alle anderen GUI-Komponenten enthält

Um mit GUI-Komponenten und Containern zu arbeiten muss man folgende Fakten beachten:

- Damit eine GUI-Komponente angezeigt werden kann, muss sie Teil einer Struktur sein, wo sichergestellt ist, dass sie innerhalb eines Containers ist, der letztendlich in einem Top-Level-Container enthalten ist. Dabei muss dieser Container nicht direkt im Top-Level-Container angezeigt werden, sondern es kann auch sein, dass der Container in einem anderen Container enthalten ist, der wieder um in einem anderen Container vorkommt. Es muss nur ein durchgängiger Pfad vorhanden sein. Man nennt dies ***containment hierarchy***.
- Jede GUI-Komponente kann nur in einem Container platziert sein. Wenn eine GUI-Komponente noch zu einem zweiten Container hinzugefügt wird, wird sie aus dem ersten Container entfernt.
- Jeder Top-Level-Container hat eine Content-Pane, das ist jene Fläche die den eigentlichen Inhalt (engl. content), das sind die sichtbaren GUI-Komponenten, enthalten soll.
- Ein Top-Level-Container kann auch noch eine Menu-Bar enthalten. Diese ist gewöhnlicher Weise außerhalb der Conten-Pane aber noch innerhalb des Top-Level-Containers platziert. Die Platzierung der Menu-Bar kann aber durch das Look-And-Feel geändert werden.

Das folgende Bild zeigt eine Frame-Struktur mit der zugehörigen containment hierarchy[1]:



Wie die Punkte im letzten Diagramm darstellen wurden der Einfachheit halber ein paar Dinge ausgelassen.

[1] Aus https://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html

Damit man dieses Programm schreiben kann braucht man ein Objekt der Klasse `JFrame`. Zu diesem werden dann die entsprechenden Inhalte hinzugefügt.

```
import java.awt.*;
import javax.swing.*;

public class TopLevelDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame("TopLevelDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JMenuBar greenMenuBar = new JMenuBar();
        greenMenuBar.setOpaque(true);
        greenMenuBar.setBackground(new Color(154, 165, 127));
        greenMenuBar.setPreferredSize(new Dimension(200, 20));

        JLabel yellowLabel = new JLabel();
        yellowLabel.setOpaque(true);
        yellowLabel.setBackground(new Color(248, 213, 131));
        yellowLabel.setPreferredSize(new Dimension(200, 180));

        frame.setJMenuBar(greenMenuBar);
        frame.add(yellowLabel);

        frame.pack();
        frame.setVisible(true);
    }
}
```

Importieren der Packages für die swing- und awt-Klassen (swing basiert auf AWT!)

Erzeugen eines neuen Top-Level-Container-Objektes

Damit das Fenster beim Klicken auf X geschlossen wird

Erzeugen eines MenuBar-Objekts und setzen einiger Anzeige-Eigenschaften

Erzeugen eines Label-Objekts (an sich für Beschriftungen) und setzen einiger Anzeige-Eigenschaften

Hinzufügen der GUI-Komponenten zum Top-Level-Container

Größe des Frames auf die Größe der Komponenten beschränken

***Sichtbarmachen** des Top-Level-Containers! (sonst wird nichts angezeigt und das Programm endet gleich wieder)*

Im Folgenden werden einige wichtige Punkte zur Erstellung einer GUI näher beleuchtet:

import-Anweisungen

Auch wenn mit swing gearbeitet wird, werden meistens Klassen aus den Packages `java.awt.*` und `javax.swing.*` benötigt. Es ist für swing-Anwendungen normal, dass immer wieder Klassen aus dem awt-Package gebraucht werden (in diesem Fall die Klassen `Color` und `Dimension`). Immerhin basiert swing auf AWT.

Erzeugen eines `JFrame`-Objekts

Ein `JFrame` ist der übliche Top-Level-Container für swing-Desktop-Anwendungen ("normale GUI-Fenster-Anwendungen" mit swing). Ein Frame hat standardmäßig die Betriebssystem Titelleiste mit einem Titel und den Buttons zum Minimieren, Maximieren und Schließen. Der Titel wird über den Konstruktor gesetzt.

`JFrame``(`**`String`** `title)`
Creates a new, initially invisible Frame with the specified title.

Damit der Button zum Schließen das Programm auch wirklich schließt, muss dies dem Objekt über die Methode `setDefaultCloseOperation` mitgeteilt werden (zweite Anweisung im Beispiel).

void	**setDefaultCloseOperation**(int operation) Sets the operation that will happen by default when the user initiates a "close" on this frame.

Als Parameter für diese Methode (ein `int`) können folgende Klassenkonstanten (erkennbar am `static` und den Blockbuchstaben in der API) verwendet werden:

static int	**DISPOSE_ON_CLOSE** The dispose-window default window close operation.
static int	**DO_NOTHING_ON_CLOSE** The do-nothing default window close operation.
static int	**EXIT_ON_CLOSE** The exit application default window close operation.
static int	**HIDE_ON_CLOSE** The hide-window default window close operation

Da es sich hierbei um Klassenkonstanten handelt, muss der Klassenname immer davor geschrieben werden (genauso wie bei Klassenmethoden), also z.B. `JFrame.DO_NOTHING_ON_CLOSE`.

Erzeugen der GUI-Komponenten

Damit überhaupt etwas im Top-Level-Container angezeigt wird müssen GUI-Komponenten erzeugt werden. Das sind im Allgemeinen Komponenten aus dem package `javax.swing`. Auch wenn immer wieder Klassen aus dem AWT-Package verwendet werden, ist es nicht gut Anzeige-Komponenten aus AWT und swing zu mischen (das kann leider zu ungewollten Ergebnissen führen). Die Klassennamen der swing-Komponenten beginnen alle mit einem `J`.

Im Beispiel wird zunächst eine Komponente für eine Menüleiste – Klasse `JMenuBar` – ohne Inhalt aber mit grünlichem Hintergrund erzeugt. Damit sie auch sichtbar ist wird noch die Eigenschaft `opaque` (engl. für undurchsichtig) auf true gesetzt. Außerdem wird die bevorzugte Größe festgelegt.

Als zweite GUI-Komponente wird eine Beschriftung – Klasse `JLabel` – verwendet, die allerdings auch keinen Text enthält sondern auch nur durch die Hintergrundfarbe erkennbar ist. Wie schon bei der Menüleiste wird neben der Farbe auch noch die `opaque`-Eigenschaft und die bevorzugte Größe gesetzt.

Nach dem Erzeugen müssen die GUI-Komponenten aber auch noch zum Frame hinzugefügt werden, sonst sind sie lediglich Objekte im Arbeitsspeicher und werden nicht angezeigt.

Hinzufügen der GUI-Komponenten und Anzeigen des JFrame-Objekts

Die zuvor erzeugten GUI-Komponenten müssen noch für die Anzeige zum Top-Level-Container hinzugefügt werden. Dies geschieht im Fall einer Menüleiste mit der Methode `setMenuBar` und für alle Komponenten, die zur Content-Pane hinzugefügt werden sollen über die Methode `add`:

void	**setMenuBar**(**MenuBar** mb) Sets the menu bar for this frame to the specified menu bar.
Component	**add**(**Component** comp) Appends the specified component to the end of this container.

Danach muss noch die Größe des Fensters festgelegt werden. Am einfachsten ist dies mit der Methode `pack()`. Diese versucht das Fenster so klein wie möglich zu machen. Dabei wird aber die bevorzugte Größe der Komponenten (*preferred size*) nicht unterschritten.

void	**pack**() Causes this Window to be sized to fit the preferred size and layouts of its subcomponents.

Damit das erzeugte JFrame auch wirklich angezeigt wird, muss es am Schluss noch sichtbar gemacht werden. Dazu wird die Eigenschaft visible auf true gesetzt.

```
void        setVisible(boolean b)
```
Shows or hides this Window depending on the value of parameter b.

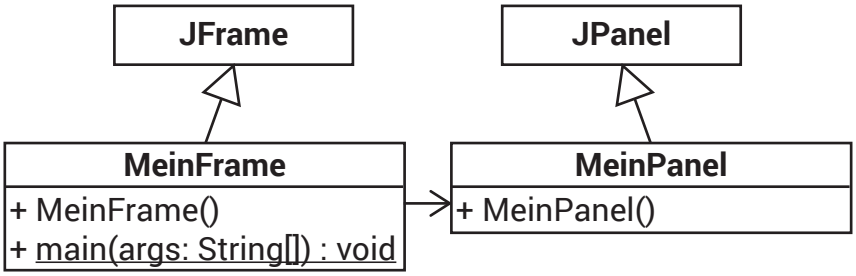
Ohne diesen Schritt würde das Programm gleich nach dem Start wieder beendet werden, da eine GUI ohne sichtbaren Fenstern von Java gleich wieder geschlossen wird.

Strukturierung

Alles in die main-Methode zu schreiben ist zwar schnell und einfach, entspricht aber nicht einem guten Programmierstil. Um eine gute Aufteilung des Programms zu erreichen ist es sinnvoll, die Klassen nach ihren Verantwortlichkeiten zu strukturieren. Für die ersten einfachen GUI-Beispiele haben wir folgende Klassen/Verantwortlichkeiten:

- Anzeige des Fensters: Dazu wird eine eigene `JFrame`-Klasse geschrieben, die von der Klasse `JFrame` erbt. Hier werden die Basis-Einstellung für das Fenster (Größe, was passiert beim Schließen, wo wird es platziert, ...) und eine eventuelle Menüleiste definiert.
- Verwalten des Inhalts: Dazu wird eine eigene Komponenten-Klasse geschrieben, die sämtliche GUI-Komponenten enthält. D.h. diese selbst geschriebene Klasse muss ein Container sein. Diese Container-Funktionalität kann über die Klasse `JPanel` geerbt werden. `JPanel` ist kein Top-Level-Container, sondern ein ganz normaler Container, der einfach nur Komponenten enthalten kann.

Damit ergibt sich eine bisschen komplexere Struktur für das Beispiel:



Dementsprechend wird auch der Quellcode aufgeteilt. Da das Panel für die Verwendung im Frame benötigt wird, muss dieses zuerst geschrieben werden:

```
import javax.swing.*;
import java.awt.*;

public class MeinPanel extends JPanel {
    public MeinPanel() {
        BorderLayout basis = new BorderLayout();
        this.setLayout(basis);

        JLabel yellowLabel = new JLabel();
        yellowLabel.setOpaque(true);
        yellowLabel.setBackground(new Color(248, 213, 131));
        yellowLabel.setPreferredSize(new Dimension(200, 180));

        this.add(yellowLabel);
    }
}
```

Von JPanel erben und damit eine eigene Layout-Container-Klasse machen

Aufruf der geerbten Methoden für das eigene Panel-Objekt

Ein spezielles Layout festlegen, das die Anordnung der GUI-Komponenten bestimmt,

Hinzufügen des Labels zum Panel

Hier wird nur der Inhalt (Content) des Fensters festgelegt, Menüleisten gehören nicht dazu. Üblicherweise enthält eine GUI jedoch viel mehr Komponenten.

Die Klasse `MeinFrame` ist dann folgendermaßen aufgebaut:

```
import javax.swing.*;
import java.awt.*;

public class MeinFrame extends JFrame {
    public MeinFrame() {
        super("TopLevelDemo");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JMenuBar greenMenuBar = new JMenuBar();
        greenMenuBar.setOpaque(true);
        greenMenuBar.setBackground(new Color(154, 165, 127));
        greenMenuBar.setPreferredSize(new Dimension(200, 20));

        MeinPanel content = new MeinPanel();

        this.setJMenuBar(greenMenuBar);
        this.add(content);

        this.pack();
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new MeinFrame();
    }
}
```

Von JFrame erben und damit eine eigene Top-level-Container-Klasse machen

Aufruf des Konstruktors der Superklasse (JFrame) mit String-Parameter, um den Titel festzulegen

Eine Menüleiste ist Teil der Rahmenstruktur und damit im Frame zu setzen

Aufruf der geerbten Methoden für das eigene Frame-Objekt

Erzeugen eines Objekts der zuvor erstellten Klasse MeinPanel, die alle Inhalts-GUI-Komponenten enthält.

Zum Starten muss dann nur noch ein Objekt der selbst erstellten Frame-Klasse erzeugt werden (damit wird dann der Konstruktor aufgerufen)

Größe und Position des Fensters

Mit Hilfe der Methode `pack()` wird das Fenster so klein wie möglich gemacht, so dass die Komponenten trotzdem noch angezeigt werden können. Es gibt aber auch die Möglichkeit die Größe des Frames selbst zu bestimmen. Die bevorzugten Größen der Komponenten werden u.U. ignoriert.

`void`	**`setSize`**`(int width, int height)` Resizes this component so that it has width `width` and height `height`.
`void`	**`setSize`**`(Dimension d)` Resizes this component so that it has width `d.width` and height `d.height`.

Auch die Position am Bildschirm kann festgelegt werden (Achtung: 0/0 ist links oben)

`void`	**`setLocation`**`(int x, int y)` Moves this component to a new location.
`void`	**`setLocation`**`(Point p)` Moves this component to a new location.
`void`	**`setLocationRelativeTo`**`(Component c)` Sets the location of the window relative to the specified component according to the following scenarios.

Wenn bei der letzten Methode `null` anstelle eines `Component`-Objekts übergeben wird, dann wird das Frame am Bildschirm zentriert. Sonst wird das Frame so positioniert, dass der Mittelpunkt des Frames mit dem Mittelpunkt des `Component`-Objekts übereinstimmt. Als Parameter kann jede beliebige GUI-Komponente verwendet werden.

Man kann auch Position und Größe gemeinsam festlegen

```
void   setBounds(int x, int y, int width, int height)
       Moves and resizes this component.
void   setBounds(Rectangle r)
       Moves and resizes this component to conform to the new bounding rectangle r.
```

```
import javax.swing.*;
import java.awt.*;

public class MeinFrame extends JFrame {
    public MeinFrame() {
        super("TopLevelDemo");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        MeinPanel content = new MeinPanel();
        this.add(content);
                                    Anstelle von pack() werden Größe und
                                    Position mit setBounds(...) festgelegt
        this.setBounds(100,100, 300, 200);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        new MeinFrame();
    }
}
```

Natürlich hat die Klasse JFrame noch viel mehr Methoden. Das Anführen aller Methoden im Rahmen dieser Unterlage ist jedoch nicht möglich. Diese Methoden können in der Java-API-Dokumentation[2] nachgeschlagen werden.

Habe ich es verstanden:

- Ich kenne die Abkürzung GUI und kann sie erklären.
- Ich kann alle drei Standard-GUI-Bibliotheken von Java benennen und grundlegend beschreiben.
- Ich kann die grundsätzlichen Funktionen und Eigenschaften von AWT und swing
- Ich kenne die Begriffe GUI-Komponente, Container und Top-Level-Container
- Ich kenne den Begriff containment hierarchy und kann beschreiben, warum er für ein GUI-Komponenten wesentlich ist.
- Ich kenne die grundsätzlichen Regeln um mit Top-Level-Containern und GUI-Komponenten arbeiten zu können.
- Ich kann ein JFrame-Objekt erstellen und anzeigen.
- Ich kann für das JFrame-Objekt unterschiedliche Schließverhalten festlegen.
- Ich kann GUI-Komponenten erstellen und in einem Top-Level-Container anzeigen.
- Ich kann die GUI in Klassen entsprechend ihrer Verantwortlichkeiten aufteilen.
- Ich kann Größe und Position des JFrame-Objektes festlegen.

[2] https://docs.oracle.com/javase/9/docs/api/index.html?javax/swing/JFrame.html