Handout

OP2 Java Programmierung

50-GUI Einführung

GUI entwickeln mit Java Swing

- Swing hat einen ausführlichen Vorrat an GUI-Komponenten. Eine GUI-Komponente ist ein Ding (Widget) auf einer Seite. (Button, Checkbox, Label etc.)
- Vor Swing gab es AWT
- Nachfolger Swing: JavaFX
- Swing verwendet das MVC Architektur Muster. MVC -> Model View Controller

Aufbau Swing-GUI

Eine Swing-GUI besteht aus drei Teilen:

1. UI-Elementen (Komponenten, die der Nutzer sehen und mit denen dieser interagieren kann)
2. Layouts (Look and Feel) kontrollieren die Platzierung und Ausrichtung der Komponenten
3. Behavior / Verhalten (Ereignisse (Events) die auftreten können: z.B. der Nutzer klickt einen Button oder gibt Text ein.)
4. Swing verwendet ein System aus Containern.
5. Der Top-Level Container kann ein JFrame, JDialog oder JApplet sein. Haben wir unseren Top Level Container erstellt, können wir darauf eine Hierarchie von Komponenten entwickeln. Der Top-Level Container ist wie ein leerer Bilderrahmen. Um Zeichnen zu können brauchen wir eine Leinwand. In diesem Fall können wir als Leinwand ein Content Pane verwenden. JPanels sind die Leinwände von Swing.
6. Wir verwenden den Frame, um die Größe des Fensters zu bestimmen, aber JFrame beinhaltet nicht die anderen Komponenten. Diese müssen durch ein Panel angeordnet werden.
7. Um die Komponenten hinzuzufügen können wir entweder einen WYSIWYG[1] Editor benutzen oder per Code codieren. Beides hat Vor- und Nachteile, aber das Anordnen von Komponenten fällt viel leichter mit Hilfe eines Editors. Nachteil es wird viel automatischer Quellcode in verschiedenen Dateien erstellt, die nur schwer wartbar sind.
8. Swing ist sehr flexibel und läuft auf allen Plattformen.
9. Eine GUI erlaubt Nutzerinteraktion.

[1] WYSIWYG ist eine Abkürzung für “What You See Is What You Get”, was auf Deutsch so viel bedeutet wie “Was du siehst, ist das, was du bekommst”. Es handelt sich dabei um eine Art von Editor, bei dem du Elemente wie Texte und Bilder auf einer grafischen Oberfläche nach deinen Wünschen anordnen kannst, ähnlich wie bei einem Puzzle. Du musst keinen speziellen Code eingeben. Der Editor kümmert sich darum, alles automatisch in die entsprechende Programmiersprache zu übersetzen. Das macht das WYSIWYG-Prinzip sehr benutzerfreundlich.

Erstes Frame erstellen

```java
import javax.swing.*;

public class Main{
    public static void main(String[] args){
        Main m = new Main();
    }

    public Main(){
        initialize();
    }

    private void initialize(){
        JFrame meinFrame = new JFrame();
        meinFrame.setTitle("Erster Frame");
        meinFrame.setSize(400,300);
        meinFrame.setResizable(false);
        meinFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        meinFrame.setLocationRelativeTo(null);
        meinFrame.setVisible(true);
    }
}
```

Abb. 1 Ausschnitt des Java-Quellcode für das erste Frame

1. *import javax.swing.*;*
   Dieser Befehl importiert alle Klassen aus dem Swing-Paket. Swing ist eine Bibliothek in Java, die eine Sammlung von Klassen für die Erstellung grafischer Benutzeroberflächen (GUIs) bereitstellt.
2. *public class Main*
   Hier wird eine neue öffentliche Klasse namens Main definiert. Eine Klasse ist ein Bauplan für Objekte mit ähnlichen Eigenschaften und Verhaltensweisen.
3. *public static void main(String[] args)*
   Dies ist die Hauptmethode, die beim Start des Programms aufgerufen wird. Sie ist der Einstiegspunkt für jede Java-Anwendung.
4. *Main m = new Main();*
   In der Hauptmethode wird ein neues Objekt der Klasse Main erstellt. Dies führt zur Ausführung des Konstruktors der Main-Klasse.
5. *public Main() { initialize(); }*
   Dies ist der Konstruktor der Main-Klasse. Ein Konstruktor ist eine spezielle Methode, die aufgerufen wird, wenn ein neues Objekt einer Klasse erstellt wird. In diesem Fall ruft der Konstruktor die Methode initialize auf.
6. *private void initialize() { ... }*
   Die initialize-Methode ist dafür verantwortlich, das Swing-Fenster zu erstellen und seine Eigenschaften zu konfigurieren.
7. *JFrame meinFrame = new JFrame();*
   Hier wird ein neues JFrame-Objekt erstellt und als meinFrame bezeichnet. Ein JFrame ist ein Fenster mit einem Titel und einem Rahmen
8. *meinFrame.setTitle("Erster Frame");*
   Diese Methode setzt den Titel des Fensters auf “Erster Frame”.

9. *meinFrame.setSize(400,300);*
   Mit dieser Methode wird die Größe des Fensters auf 400 Pixel Breite und 300 Pixel Höhe festgelegt.
   Eine weitere Möglichkeit die Größe festzulegen ist mit setBounds():
   Der Code meinFrame.setBounds(0, 0, 800, 600); ist eine Methode in Java, die zur Festlegung der Position und Größe eines Fensters verwendet wird. Die Methode setBounds() nimmt vier Parameter: setBounds(int x, int y, int width, int height).
   x und y bestimmen die linke obere Ecke des Fensters auf dem Bildschirm.
   width und height bestimmen die Breite und Höhe des Fensters in Pixeln.
   In diesem Beispiel-Code meinFrame.setBounds(0, 0, 800, 600); wird das Fenster meinFrame so positioniert, dass seine linke obere Ecke mit dem Ursprung des Bildschirms (0,0) übereinstimmt. Das Fenster hat eine Breite von 800 Pixeln und eine Höhe von 600 Pixeln.
10. *meinFrame.setResizable(false);*
    Diese Methode verhindert, dass das Fenster in der Größe geändert werden kann.
11. *meinFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);*
    Diese Methode stellt sicher, dass das Programm beendet wird, wenn das Fenster geschlossen wird. JFrame.EXIT_ON_CLOSE ist eine Konstante, die angibt, dass das Programm beim Schließen des Fensters beendet werden soll.
12. *setLocationRelativeTo(null)*
    Diese Methode wird verwendet, um ein Fenster mittig auf dem Bildschirm zu positionieren
13. *meinFrame.setVisible(true);*
    Schließlich macht diese Methode das Fenster sichtbar. Standardmäßig sind neue Fenster unsichtbar, daher muss setVisible(true) aufgerufen werden, um das Fenster anzuzeigen.

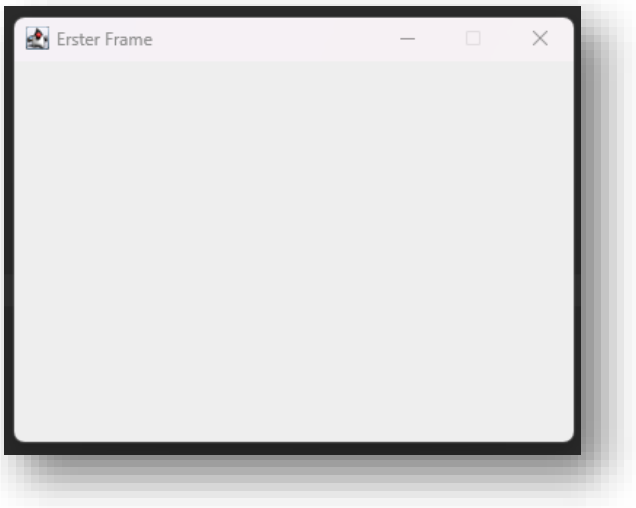

Abb. 2 Darstellung des ersten Frame

Zusammengefasst erstellt dieser Code ein einfaches Swing-Fenster mit dem Titel “Erster Frame”, das 400 Pixel breit und 300 Pixel hoch ist. Das Fenster ist nicht skalierbar und das Programm wird beendet, wenn das Fenster geschlossen wird. Das Fenster wird sichtbar gemacht, indem setVisible(true) aufgerufen wird. Dies ist ein grundlegendes Beispiel für die Erstellung eines GUI-Fensters in Java mit Swing.

Erweiterung 1. des Frame um ein Label

```
private void initialize(){
        JFrame meinFrame = new JFrame();
        meinFrame.setTitle("Erster Frame");
        meinFrame.setSize(400,300);
        meinFrame.setResizable(false);
        meinFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        meinFrame.setLocationRelativeTo(null);

         // Erweiterung 1 durch ein Label->
        JLabel label = new JLabel("Erstes Label");
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setVerticalAlignment(JLabel.CENTER);
        Border border = BorderFactory.createLineBorder(Color.BLACK, 2);
        label.setBorder(border);
        label.setBackground(new Color(171, 205, 247));
        label.setOpaque(true);
        meinFrame.getContentPane().setLayout(null);
        label.setSize(125,50);
        label.setLocation(25, 25);
        meinFrame.add(label);
        // <-

        meinFrame.setVisible(true);
 }
```

Abb. 3 Erweiterung des Frame durch ein Label

1. *JLabel label = new JLabel("Erstes Label");*
   Diese Zeile erstellt ein neues JLabel-Objekt mit dem Text “Erstes Label”.
2. *label.setHorizontalAlignment(JLabel.CENTER);*
   label.setVerticalAlignment(JLabel.CENTER);
   Diese beiden Zeilen setzen die horizontale und vertikale Ausrichtung des Textes innerhalb des Labels auf die Mitte.
3. *Border border = BorderFactory.createLineBorder(Color.BLACK, 2);*
   label.setBorder(border);
   Hier wird ein neues Border-Objekt erstellt, das eine schwarze Linie mit einer Dicke von 2 Pixeln darstellt. Dieser Rand wird dann auf das Label angewendet.
4. *label.setBackground(new Color(171, 205, 247));*
   label.setOpaque(true);
   Diese Zeilen setzen die Hintergrundfarbe des Labels auf einen bestimmten Farbton von Blau (RGB-Werte 171, 205, 247). Das Label wird auch auf “opaque” gesetzt, was bedeutet, dass der Hintergrund sichtbar ist.

5. *meinFrame.getContentPane().setLayout(null);*
   Diese Zeile setzt das Layout des Inhaltsbereichs des Frames auf null[2], was bedeutet, dass Sie die Position und Größe der Komponenten manuell festlegen müssen.
6. *label.setSize(125,50);*
   label.setLocation(25, 25);
   Diese Zeilen setzen die Größe des Labels auf 125x50 Pixel und positionieren es an der Stelle (25, 25) im Frame.
7. *meinFrame.add(label);*
   Schließlich wird das Label zum Frame hinzugefügt, sodass es angezeigt wird, wenn der Frame sichtbar gemacht wird.

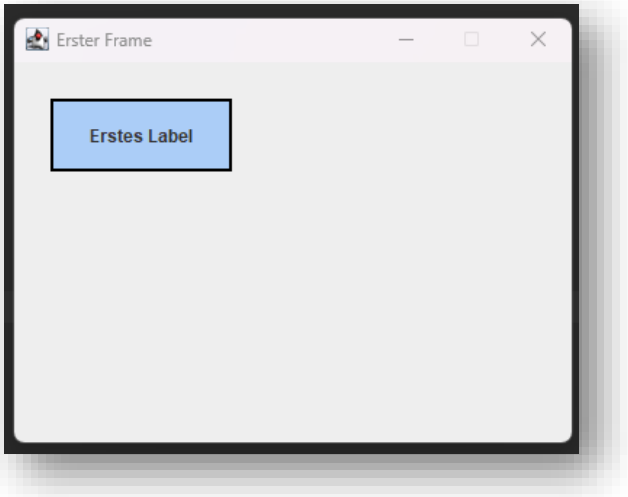

Abb. 4 Darstellung Frame-Erweiterung durch ein Label

[2] In Java Swing wird MEINFRAME.GETCONTENTPANE().SETLAYOUT(NULL); verwendet, um den Standard-Layout-Manager zu deaktivieren und eine vollständige Kontrolle über die Position und Größe der Komponenten zu ermöglichen. Dies erfordert die manuelle Festlegung von Position und Größe jeder Komponente mit SETSIZE und SETLOCATION. Allerdings wird diese Praxis oft als schlecht angesehen, da sie nicht gut mit verschiedenen Fenstergrößen und Bildschirmauflösungen skaliert. Es wird empfohlen, einen geeigneten Layout-Manager zu verwenden.

Erweiterung 2. des Frame durch ein GridBagLayout

```
// Erweiterung 1 durch ein Label->
JLabel label = new JLabel("Erstes Label");
label.setHorizontalAlignment(JLabel.CENTER);
label.setVerticalAlignment(JLabel.CENTER);
Border border = BorderFactory.createLineBorder(Color.BLACK, 2);
label.setBorder(border);
label.setBackground(new Color(171, 205, 247));
label.setOpaque(true);
label.setPreferredSize(new Dimension(125,50)); // Hinzufügen für Erweiterung 2

// Diesen Bereich auskommentieren für Erweiterung 2->
/*
meinFrame.getContentPane().setLayout(null);
label.setSize(125,50);
label.setLocation(25, 25);
meinFrame.add(label);
*/
// <-

// Erweiterung 2 durch das GridBagLayout->
GridBagLayout gridBag = new GridBagLayout();
meinFrame.getContentPane().setLayout(gridBag);
GridBagConstraints constraints = new GridBagConstraints();
constraints.gridx = 0;
constraints.gridy = 0;
meinFrame.getContentPane().add(label, constraints);
//<-
```

Abb. 5 Erweiterung des Frame durch ein GridBagLayout

Der auskommentierte Code wurde durch das GridBagLayout ersetzt, und die setPreferredSize Methode wird verwendet, um die Größe des JLabels aus dem vorherigen Beispiel beizubehalten. Hier ist eine Erläuterung des aktualisierten Codes:

1. *label.setPreferredSize(new Dimension(125, 50));*
   Diese Zeile setzt die bevorzugte Größe des JLabels auf eine Breite von 125 und eine Höhe von 50. Diese Größe wird vom Layout-Manager berücksichtigt, wenn er die Komponente anordnet.
2. *GridBagLayout gridBag = new GridBagLayout();*
   Diese Zeile erstellt ein neues GridBagLayout.
3. *meinFrame.getContentPane().setLayout(gridBag);*
   Diese Zeile setzt das Layout des Inhaltsbereichs des Frames auf das neu erstellte GridBagLayout.
4. *GridBagConstraints constraints = new GridBagConstraints();*
   Diese Zeile erstellt ein neues GridBagConstraints-Objekt, das die Platzierungsregeln für Komponenten im GridBagLayout definiert.
5. *constraints.gridx = 0;*
   constraints.gridy = 0;
   Diese Zeilen setzen die x- und y-Koordinaten des JLabels im GridBagLayout.
6. *meinFrame.getContentPane().add(label, constraints);*

Diese Zeile fügt das JLabel zum Inhaltsbereich des Frames hinzu und verwendet dabei die definierten GridBagConstraints.
In diesem aktualisierten Code wird das JLabel mit der bevorzugten Größe in das GridBagLayout eingefügt. Die tatsächliche Größe und Position des JLabels wird vom GridBagLayout bestimmt, basierend auf den GridBagConstraints und der bevorzugten Größe des JLabels.

Abb. 6 Darstellung des Labels angeordnet durch das GridBagLayout

Erweiterung 3. des Frame durch einen Button und Dialog-Fenster

```
// Erweiterung 3 durch ein Button->
JButton button = new JButton("Erster Button");
button.addActionListener((event) ->
        JOptionPane.showMessageDialog(meinFrame, "Der Button wurde geklickt!")
);

constraints.gridx = 0;
constraints.gridy = 1;
constraints.insets = new Insets(10,10,10,10);
meinFrame.getContentPane().add(button, constraints);
//<-
```

Abb. 7 Erweiterung des Frame durch einen Button

1. *JButton button = new JButton("Erster Button");*
   Dieser Teil des Codes erstellt einen neuen Button mit dem Text “Erster Button”.

2. *button.addActionListener((event) ->*
   JOptionPane.showMessageDialog(meinFrame, "Der Button wurde geklickt!"));
   Hier wird ein ActionListener[3] zum Button hinzugefügt. Wenn der Button geklickt wird, zeigt eine Dialogbox eine Nachricht an, die besagt: “Der Button wurde geklickt!”.
3. *constraints.gridx = 0;*
   constraints.gridy = 1;
   constraints.insets = new Insets(10,10,10,10);
   Diese Zeilen definieren die Position und den Abstand des Buttons im Fenster. Der Button wird in der Zelle positioniert, die durch gridx = 0 und gridy = 1 definiert ist. Der Abstand um den Button herum beträgt 10 Pixel in alle Richtungen.
4. *meinFrame.getContentPane().add(button, constraints);*
   Schließlich wird der Button zum Inhalt des Fensters hinzugefügt, wobei die zuvor definierten Einschränkungen berücksichtigt werden.

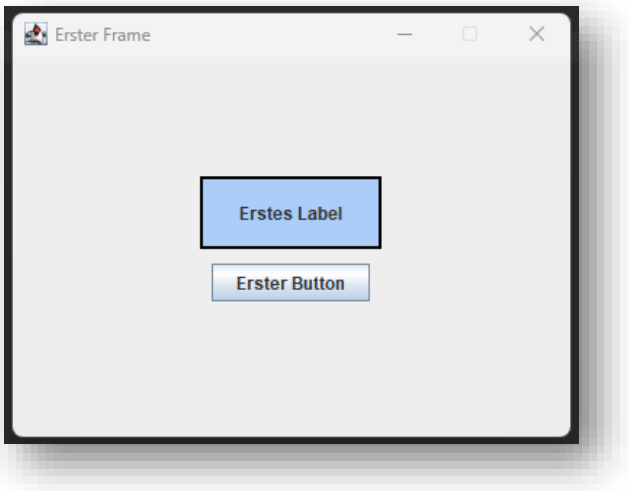

Abb. 8 Darstellung des Frame erweitert durch einen Button

[3] Das Hinzufügen des ActionListener stellt eine Implementierung des Beobachter-Musters dar. Hierbei fungiert der JButton als Herausgeber und der ActionListener als Abonnent. Bei einem Klick auf den Button, also einem Ereignis, informiert der Herausgeber alle seine Abonnenten. In diesem speziellen Szenario wird lediglich eine Aktion ausgeführt: Es erscheint ein Dialogfenster mit der Meldung “Der Button wurde geklickt!”.

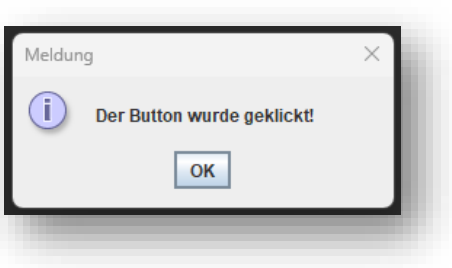

Abb. 9 Darstellung des Dialog Fenster, wenn Button geklickt

Erweiterung 4. des Frame durch ein Menu und Icons

Als erstes wird durch rechten Mausklick auf das Projekt oder Modul ein neuer Ordner mit dem Namen *images* erstellt.

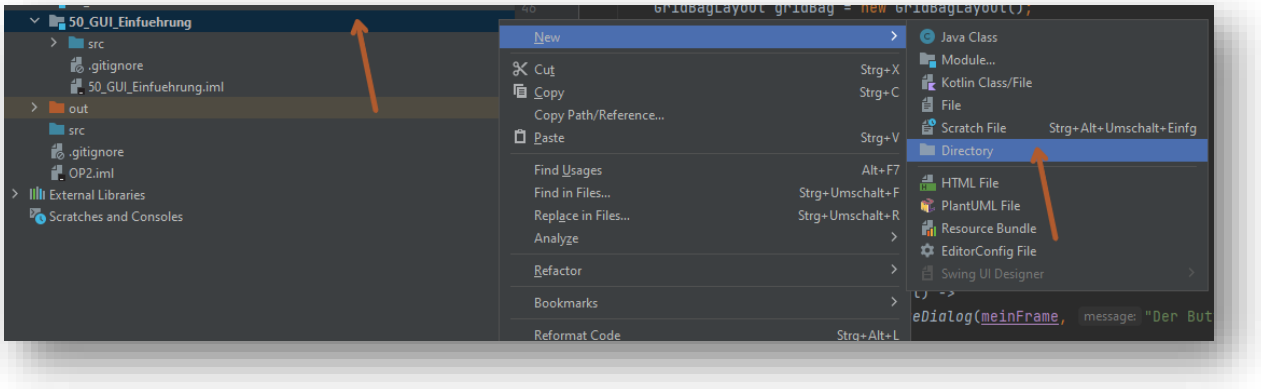

Abb. 10 Ordner images anlegen

Um den Ordner als Root-Ordner für Ressourcen zu markieren, folgen Sie bitte diesen Schritten:

1. Navigieren Sie zu *File -> Project Structure*.
2. Wählen Sie den Reiter *Modules*.
3. Wählen Sie in der Mitte das Projekt oder das Modul aus, in dem sich der neue Ordner befindet.
4. Wählen Sie unter dem Reiter *Sources* den Ordner aus.
5. Markieren Sie ihn als *Resources Root*.
6. Bestätigen Sie unten mit *Ok*.

Jetzt ist der ausgewählte Ordner als Root-Ordner für Ressourcen markiert.

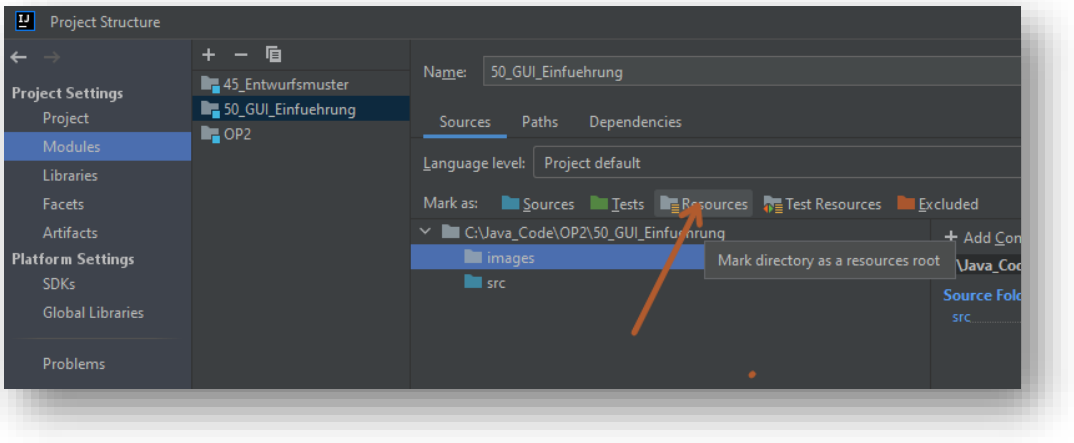

Abb. 11 Ordner images als Resources Root markieren

Nun werden die mitgelieferten Icons in den Ordner kopiert / verschoben, dann wird ein Enum erstellt über das dann die Icons abgefragt(ausgewählt) werden können.

```
private enum Icons {
        ADDBUTTON_16("addButton_16.png"),
        // Definiert eine Konstante mit dem Namen ADDBUTTON_16 und dem Pfad "addButton_16.png"
         EXIT_16("exit_16.png"),
        // Definiert eine Konstante mit dem Namen EXIT_16 und dem Pfad "exit_16.png"
         MEN_16("men_16.png");
        // Definiert eine Konstante mit dem Namen MEN_16 und dem Pfad "men_16.png"

         private String pfad; // Definiert eine private Variable zum Speichern des Pfads

         Icons(String pfad){ // Der Konstruktor der enum, der den Pfad setzt
                this.pfad = pfad;
         }

         public Image getImage(){ // Eine öffentliche Methode zum Abrufen des Bildes

                URL url = getClass().getResource("/" + pfad);
                // Versucht, die Ressource aus dem Pfad zu holen

                if (url != null){ // Wenn die Ressource gefunden wurde

                        try{
                                return ImageIO.read(url);
                                // Versucht, das Bild aus der URL zu lesen und zurückzugeben
                        }
                        catch(IOException ex){ // Wenn ein Fehler beim Lesen des Bildes auftritt
                                return null; // Gibt null zurück
                        }
                }
                return null; // Wenn die Ressource nicht gefunden wurde, gibt null zurück
        }
}
```

Abb.12 Erstellung des Enum für die Icon-Auswahl

Dieser Code definiert ein privates *enum* namens *Icons*. Ein *enum* in Java ist ein spezieller Datentyp, der eine Gruppe von Konstanten repräsentiert. In diesem Fall repräsentiert jede Konstante ein bestimmtes Bild.

Jede Konstante in *Icons* hat einen zugehörigen *String*-Wert, der den Pfad zur Bilddatei darstellt. Dieser Pfad wird im Konstruktor des *enum* gesetzt.

Die Methode *getImage()* ist eine öffentliche Methode, die ein *Image*-Objekt zurückgibt. Sie versucht, die Bilddatei aus dem Ressourcenpfad zu lesen und gibt das Bild zurück. Wenn die Datei nicht gefunden wird oder ein Fehler beim Lesen der Datei auftritt, gibt die Methode NULL zurück.

Der folgende Code stellt die Erstellung des Menü mit den Icons dar.

```
// Erweiterung 4 durch ein Menu und Icons->
JMenuBar menuBar = new JMenuBar();
ImageIcon exitIcon = new ImageIcon(Objects.requireNonNull(Icons.EXIT_16.getImage()));
ImageIcon addButtonIcon =
new ImageIcon(Objects.requireNonNull(Icons.ADDBUTTON_16.getImage()));

 JMenu menu = new JMenu("Menü");
 menu.setMnemonic(KeyEvent.VK_M);

 JMenuItem exitMenuItem = new JMenuItem("Exit", exitIcon);
 exitMenuItem.setMnemonic(KeyEvent.VK_E);
 exitMenuItem.setToolTipText("Beendet das Programm.");

 exitMenuItem.addActionListener((ActionEvent event) -> System.exit(0));
 menu.add(exitMenuItem);

 JMenuItem addMenuItem = new JMenuItem("Add", addButtonIcon);
 addMenuItem.setMnemonic(KeyEvent.VK A);
 addMenuItem.setToolTipText("Fügt ein weiteres Label hinzu.");

 addMenuItem.addActionListener((ActionEvent event) -> {
     JLabel neuesLabel = new JLabel("Ein Label");
     GridBagConstraints labelConstraints = new GridBagConstraints();
     labelConstraints.gridx = 0;
     labelConstraints.gridy = meinFrame.getContentPane().getComponentCount();

     meinFrame.getContentPane().add(neuesLabel, labelConstraints);
     meinFrame.revalidate();
 });

 menu.add(addMenuItem);
 menuBar.add(menu);
 meinFrame.setJMenuBar(menuBar);
 //<-

// Fenster-Icon festlegen:
 meinFrame.setIconImage(Icons.MEN_16.getImage());
```

Abb.13 Erstellung der Icons und des Menu

1. *JMenuBar menuBar = new JMenuBar();*
   Hier wird eine neue Menüleiste erstellt.

2. *ImageIcon exitIcon = new ImageIcon(Objects.requireNonNull(Icons.EXIT_16.getImage()));*
   ImageIcon addButtonIcon =
   new ImageIcon(Objects.requireNonNull(Icons.ADDBUTTON_16.getImage()));
   Diese Zeilen erstellen zwei ImageIcon-Objekte, die Bilder aus den Ressourcen Icons.EXIT_16 und Icons.ADDBUTTON_16 laden.

3. *JMenu menu = new JMenu("Menü");*
   menu.setMnemonic(KeyEvent.VK_M);
   Ein neues Menü mit dem Namen “Menü” wird erstellt und das Mnemonik (Tastenkürzel) wird auf ‘M’ gesetzt.

4. *JMenuItem exitMenuItem = new JMenuItem("Exit", exitIcon);*
   exitMenuItem.setMnemonic(KeyEvent.VK_E);
   exitMenuItem.setToolTipText("Beendet das Programm.");
   exitMenuItem.addActionListener((ActionEvent event) -> System.exit(0));
   menu.add(exitMenuItem);
   Ein Menüpunkt mit dem Namen “Exit” und dem zuvor erstellten exitIcon wird erstellt. Das Mnemonik wird auf ‘E’ gesetzt und ein Tooltip wird hinzugefügt. Ein ActionListener wird hinzugefügt, der das Programm beendet, wenn auf diesen Menüpunkt geklickt wird. Der Menüpunkt wird dann zum Menü hinzugefügt.

5. *JMenuItem addMenuItem = new JMenuItem("Add", addButtonIcon);*
   addMenuItem.setMnemonic(KeyEvent.VK_A);
   addMenuItem.setToolTipText("Fügt ein weiteres Label hinzu.");
   addMenuItem.addActionListener((ActionEvent event) -> {
   JLabel neuesLabel = new JLabel("Ein Label");
   GridBagConstraints labelConstraints = new GridBagConstraints();
   labelConstraints.gridx = 0;
   labelConstraints.gridy = meinFrame.getContentPane().getComponentCount();
   meinFrame.getContentPane().add(neuesLabel, labelConstraints);
   meinFrame.revalidate();
   });
   menu.add(addMenuItem);
   Ähnlich wie oben wird ein weiterer Menüpunkt erstellt, diesmal mit dem Namen “Add” und dem addButtonIcon. Wenn auf diesen Menüpunkt geklickt wird, wird ein neues Label zur Inhaltsfläche des Fensters hinzugefügt.

6. *menuBar.add(menu);*
   meinFrame.setJMenuBar(menuBar);
   Das Menü wird zur Menüleiste hinzugefügt und die Menüleiste wird dann zum Fenster hinzugefügt.

   Hier wird noch dem Fenster (Frame) ein Icon zu geordnet.
   meinFrame.setIconImage(Icons.MEN_16.getImage());
   Setzen des Men_16 Icon als Frame-Icon.

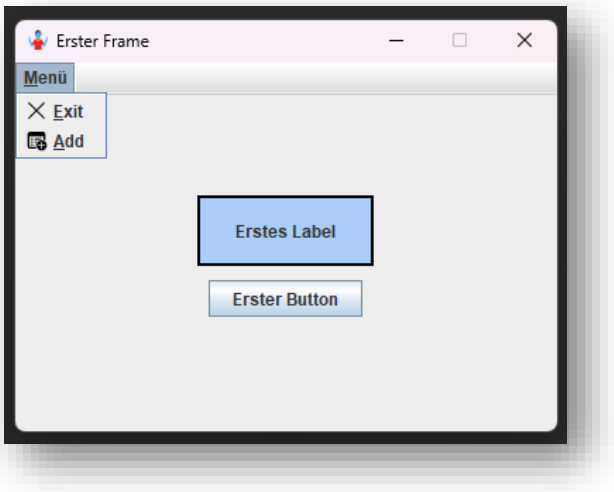

Abb. 14 Darstellung des Frame erweitert durch Menu und Icons

Weitere Java Swing Komponenten

JToggleButton - Schaltfläche, welche zwei Zustände (gedrückt und nicht gedrückt) kennt.

JCheckBox - Auswahlkästchen, das, wenn es ausgewählt wurde, mit einem Häkchen oder Kreuz versehen wird.

JRadioButton - Schaltfläche zur Auswahl zwischen mehreren Optionen, in der Regel sind sie in einer ButtonGroup angeordnet. Im Gegensatz zur JCheckBox kann nur maximal eine Option selektiert werden.

ButtonGroup - Dient der Gruppierung von JRadioButtons. So kann sichergestellt werden, dass tatsächlich nur ein Element aus der Gruppe selektiert wurde.

JComboBox - Dropdown-Liste (auch als Auswahlliste oder Listbox bezeichnet), die zur Auswahl eines Elementes aufgeklappt wird. Wenn die JComboBox editierbar ist, kann über ein Textfeld der ausgewählte Wert auch vom Anwender gesetzt werden.

JList - Einfache Liste, die mehrere Elemente enthalten kann. Einfach- und Mehrfachauswahl möglich.

JTextField - einfache einzeilige Texteingabe

JTextArea - einfache mehrzeilige Texteingabe

JScrollBar - Schieberegler zum Scrollen.

JSlider - Schieberegler, der mit einer Skala versehen werden kann.

JProgressBar - Fortschrittsbalken

JFormattedTextField - Textfeld, für welches eine Formatierung (z.B. für Datumseingaben) festgelegt werden kann.

JPasswordField - Textfeld zur Passworteingabe. Für jeden eingegebenen Buchstaben erscheint ein Sternchen oder ein anderer vorgegebener Character.

JSpinner - Ähnlich der *JComboBox*, allerdings klappt die Liste nicht auf, sondern die Navigation durch die Liste erfolgt über Pfeiltasten.

JSeparator - einfache Trennlinie

JEditorPane - mehrzeiliges Textfeld zur Bearbeitung von formatiertem Text, mit HTML- und RTF-Unterstützung.

JTextPane - Spezialisierung von *JEditorPane*, dient der Bearbeitung und Anzeige grafisch aufbereiteter Texte.

JTree - Baumstruktur, Wird häufig verwendet, um Verzeichnisstrukturen abzubilden, wie man es z.B. vom Windows Explorer kennt.

JTable - einfache Tabelle, kann auch Texteingaben entgegennehmen.