DHBW
Duale Hochschule
Baden-Württemberg

AIRBUS
DEFENCE & SPACE

Java GUI Grundlagen

Lukas Abelt
lukas.abelt@airbus.com

DHBW Ravensburg
Wirtschaftsinformatik

Ravensburg
6. Mai 2019

Inhalt

1 Historie

2 Grundlegender Aufbau

3 Grundlegende Komponenten

Inhalt

1 Historie

2 Grundlegender Aufbau

3 Grundlegende Komponenten

Java GUI Anwendungen

Die Historie (Vgl. [5])

- Anfangs bestanden Computerprogramme in der Regel aus Kommandozeilenanweisungen
- Ab den 1970er Jahren wurden grafische Anwendungen immer wichtiger
- Gründe hierfür unter anderem:
  - 1973 - Release des Alto personal PC's von Xerox
  - 1984 - Erster Macintosh
  - 1985 - Release des Amiga mit GUI Oberfläche
  - 1985 - Release von Windows 1.0
  - Kurz: Der „Personal Computer“ fand immer mehr Verbreitung
- Dadurch Nutzerverschiebung
  - Von „Experten“ zu „unwissenden“ Nutzern

Historie

Nachteile von Konsolenanwendungen

- Kommandozeilenanwendungen sind für „Normalnutzer“ nicht sinnvoll
- Begrenzte Interaktionsmöglichkeiten $\rightarrow$ Nur Texteingabe
- Bedienung nicht intuitiv $\rightarrow$ Kann nicht ohne spezielles Wissen bedient werden
- Sieht nicht ansprechend aus
- GUIs lösen diese Probleme (theoretisch)
  - Der Nutzer hat eine Vielzahl an Interaktionsmöglichkeiten
  - Gut designete GUI benötigt keine Anleitung, der Nutzen lässt sich ableiten
  - GUIs können ansprechend designed werden (zB. über Nutzung von Bildern und Icons)

GUI Anwendungen

Die technischen Implikationen

- Jedes Betriebssystem implementiert ggf. verschiedene GUI-Komponenten und -Logik
- GUI musste also für jedes OS neu implementiert werden
- Dadurch Nachteile:
  - Massiver Mehraufwand für Cross-Platform-Development
  - Kein einheitliches Aussehen auf verschiedenen Plattformen
  - Nicht jedes OS unterstützt jedes GUI Feature $\rightarrow$ Dadurch wieder eingeschränkte Funktionalität

Java to the rescue!

Eine GUI sie zu knechten(?)

- Java wollte eine plattformunabhängige GUI schaffen
- Ergebnis war 1995 das **Abstract Window Toolkit (AWT)**
- Durch die Kopplung zum JRE waren diese Plattformunabhängig
- Intern nutzt AWT jedoch direkt Betriebssystemkomponenten
  - Dadurch sind AWT Komponenten sehr effizient (Geringe Abstraktion)
  - Ist aber auch Grund für die Probleme von AWT

AWT

Der Fall

- Probleme an AWT:
  - Durch enge Betriebssystembindung musste der gemeinsame Nenner aller Betriebssysteme gefunden werden
  - Dadurch auf wenige Komponenten beschränkt
  - Begrenzter Umfang (zB Darstellen von Icons auf Komponenten nicht möglich)
  - AWT war im Grunde „quick and dirty“ zusammengepfuscht um „erstmal zu funktionieren“ (Siehe [4] S. 778)
  - Alternative Namen:
    - *Awkward Window Toolkit*
    - *Annoying Window Toolkit*

AWT

Meinungen (Siehe [3] S. 778)

Quelle: [1]

„The AWT was something we put together in six weeks to run on as many platforms as we could, and its goal was really just to work. So we came out with this very simple, lowest-common-denominator thing that actually worked quite well. But we knew at the time we were doing it that it was really limited. After that was out, we started doing the Swing thing, and that involved working with Netscape and IBM and folks from all over the place.“

—James Gosling, „Vater“ von Java

Komponenten in AWT

Auflistung (Vgl. [3] S. 1070)

- Das AWT besteht aus nur acht Komponenten:
  - Button
  - Checkbox
  - Choice
  - Label
  - List
  - Scrollbar
  - TextArea
  - TextField
- Man nennt diese auch die *Peer-Klassen*

Die Welt nach AWT

Die Geburt von Swing

- AWT wurde schnell erneuert
- Initial begann NetScape mit einer eigenen AWT Erweiterung: Die *Internet Foundation Classes (IFC)*
- Später arbeiten Sun (Heute Oracle), NetScape und IBM zusammen
- 1998 entstanden daraus die *Java Foundation Classes (JFC)*
- Kernelement waren hierbei die neuen *Swing* Komponenten

Grundsätzliche Bestandteile I

Der JFC

- **GUI-Komponenten**
  - Die *Swing* Komponenten bringen ein neues, versatiles, Set an neuen Funktionen. Diese sind, anders als die alten AWT Komponenten, komplett durch Java implementiert und verwaltet.
- **Pluggable Look&Feel**
  - Die Komponenten können zur Laufzeit im Aussehen verändert werden ohne, dass ein Neustart der Anwendung nötig ist. Alle *Swing*-Elemente bringen diese Fähigkeit mit
- **Java 2D API**

Grundsätzliche Bestandteile II

Der JFC

- Die neue 2D Grafik-API bildet automatisch über Objektbeschreibungen Objekte, die auf dem Bildschirm dargestellt werden. Komplexe Objekte können über PFade gebildet werden und darauf Bewegungs- und Verschiebeoperationen ausgeführt werden

- **Drag&Drop**
  - Daten können mittels Drag and Drop zwischen verschiedenen Applikationen übertragen werden. So können Java-Programme auch Daten nutzen, die aus Nicht-Java Anwendungen stammen

- **Accessibility**
  - Die neue API erlaubt es, mehr Interaktionstechniken für körperlich eingeschränkte Nutzer anzubieten. Dies sind zum Beispiel die Vorlesefunktion, eine Spracherkennung oder eine Bildschirmlupe

Swing

Noch etwas trivia

- Swing bietet schlussendlich viel von dem was AWT sein sollte
- Einheitliches Look&Feel auf allen Plattformen
- Hohe konfigurierbarkeit
- Ausgereifter und erweiterbarer Funktionsumfang
- Einziges Problem: Swing wurde mit der Zeit nicht weiterentwickelt
- Dadurch fehlen immer mehr Features von modernen GUIs

JavaFX

Der neue heiße Scheiß!

- Moderne Features vermisst man in Swing, wie:
  - Animationen
  - Medienunterstützung
- Daher begann die Entwicklung von JavaFX
- Dies sollte, anders als bei dem Wechsel von AWT zu Swing, komplett losgelöst vom bisherigen GUI Stack sein
- Bildet einen kompletten modernen Media-Stack ab
- Kann für 3D Anwendungen direkt auf Grafikkartenfunktionen zugreifen

JavaFX

Historie im Überblick

- **2007** - Release von JavaFX 1.0
- **2008** - JavaFX 2.0, entfernen von Java Script und Entscheidung zur puren Java-API
- **2012** - JavaFX 2.2 wird Teil der Standard JRE/JDK (Version 7 Update 6)
- **2018** - Java 11 wird released, jetzt wieder ohne integriertes JavaFX (JavaFX 11)
- **Heute** - JavaFX wird unabhängig von Java weiterentwickelt und ist Open-Source als *OpenJFX*

Java GUI

Die Zukunft

- Ob JFX sich durchsetzen wird bleibt fraglich
- Anwendungsfälle entwickeln und verändern sich rasant
- Neue Peripherie hat ganz andere Anforderungen:
  - Smartphones und -Watches
  - Virtual Reality
  - Mixed Reality (Bspw.: Hololens)
- Meine (persönliche) Meinung: Auch OpenJFX wird langfristig untergehen

Inhalt

1 Historie

2 Grundlegender Aufbau

3 Grundlegende Komponenten

Swing

Grundlegendes

- AWT Komponenten erzeugen Komponenten über Betriebssystemaufrufe
- Dadurch Speicher nicht über Java verwaltet
- Swing erzeugt Komponenten über direkte Low-Level Calls
- Jede Komponente wird auf den Bildschirm gezeichnet
- Dadurch komplette Speicherverwaltung in Java
- Overhead führt ggf. zu Performanceverlust
- Aber mehr Kontrolle über die Features

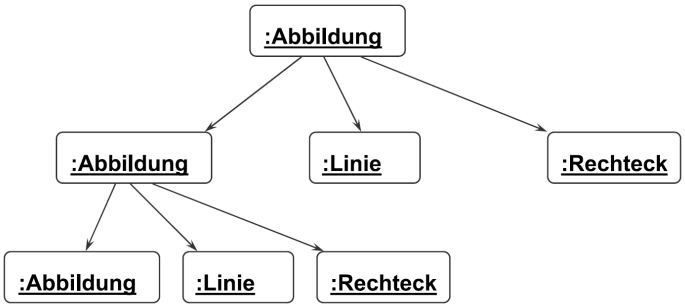
Swing

Konzepte

- Swing nutzt das Entwurfstmuster des *Kompositums*
- Kompositum beschreibt im Grunde eine Art Baumstruktur
- Objekte werden zu Gruppen zusammengefasst und Gruppen wiederum zu größeren Gruppen
- Dadurch kann eine einheitliche Behandlung von Objekten und Aggregaten erreicht werden
- Gemeinsame Eigenschaften von Objekt und Aggregat werden in einer Oberklasse isoliert

Beispiel für Kompositum

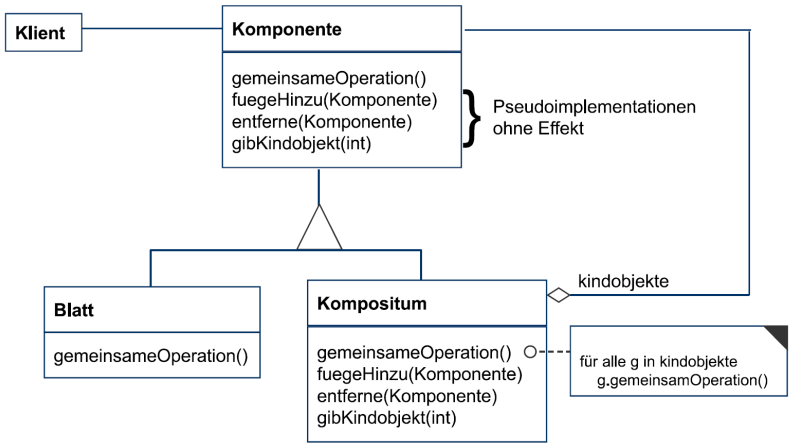
Zusammengesetzte Grapfik-Objekte (Vgl. [2] S. 26)



Gemeinsame Operationen: `zeichne()`, `verschiebe()`, `lösche()`, `skaliere()`

Kompositum

Beispielhaftes Klassendiagramm



Quelle: [2]

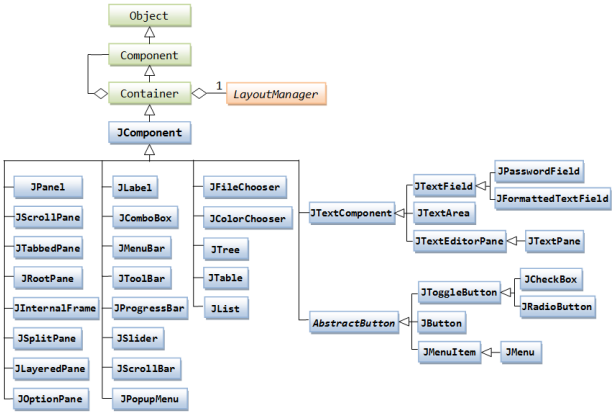
Kompositum

Umsetzung in Swing

- Die Klassenhierarchie stellt ein Kompositum dar
- Swing Objekte unterteilen sich hierbei in zwei Klassen:
  - `Component`
  - `Container`
- `Container` fassen `Components` zusammen
- Jeder Container ist für die Anordnung seiner Komponenten zuständig
- Jeder `Container` ist auch immer eine `Component`

Klassendiagramm

Der Swing Bibliothek



Swing Fenster

JFrame Klasse

- Basis eines Swing Fensters ist die `JFrame` Klasse
- Basiert auf der AWT Klasse `Frame`
- Lässt sich manipulieren z.B. in bezug auf
  - Sichtbarkeit
  - Größe und Position
  - Operation beim schließen
- Leeres Fenster wird erzeugt über den `JFrame()` Konstruktor

Swing Fenster

Ganz simpel

```
1 public static void main(String[] args){
2   JFrame window = new JFrame();
3   window.setDefaultCloseOperation(WindowConstants.
        ↪ EXIT_ON_CLOSE);
4   window.setVisible(true);
5 }
```

JFrame

Ändern der Größe des Fensters

- Größe wird über `setSize()` verändert
- Hier gibt es zwei Überladungen:
  - Als Parameter ein `Dimension` Objekt
  - Zwei `int` Werte als Parameter

Swing Fenster

Weitere Eigenschaften

- Bildet einen *Top-Level-Container*
- `JFrame` wird immer mit Titel und Menüleiste erzeugt
- Erscheinung des Frames kann geändert werden
  - Man spricht in der Regel von „Decorations“
  - Ändern der Form des Fensters (In begrenztem Rahmen)
  - Wählen verschiedener Farbschema
  - (Teil-)transparente Fenster
  - usw

Weitere Fenster in Swing

- `JWindow` – Fenster ohne Menüleiste
- `JDialog` – Zum erstellen von (modalen) Dialogfeldern

Inhalt

1 Historie

2 Grundlegender Aufbau

3 Grundlegende Komponenten

Komponenten

Grundlegendes

- Komponenten in Swing sind frei erweiterbar
- Swing Komponenten i.D.R. über Namensgebung von AWT Komponenten unterscheidbar
- Für viele Anwendungsfälle sind jedoch die Standardkomponenten schon ausreichen
- Neue Komponenten werden über
- add() Methode zu einem beliebigen Container hinzugefügt
- Komponenten können eine bevorzugte oder minimale Größe definieren
- Position wird i.d.R. automatisch bestimmt (Über den Layoutmanager)

Grundlegende Komponenten I

Übersicht

- `JPanel` – Einfacher Bereich ohne spezielle Besonderheiten. Ist ein `Container` und wird in der Regel genutzt um Elemente zu gruppieren und ggf. anzuordnen
- `JButton` – Einfacher Button für den eine Funktion programmiert werden kann(Zum Beispiel das Starten einer Berechung)
- `JCheckBox` – Auswahlbox für eine binäre Eingabe
- `JLabel` – Simples Feld zum Anzeigen von Texten. Kann nicht durch den Nutzer bearbeitet werden
- `JTextField` – Textfeld zum Anzeigen oder Bearbeiten von Texten (Einzelzeilen). Kann durch den Nutzer verändert werden
- `JSlider` – Schieberegler

Grundlegende Komponenten II

Übersicht

- `JRadioButton` – Gibt die Möglichkeit zur Auswahl von einer Option aus mehreren gegebenen Möglichkeiten. Es kann pro Gruppe an Optionen immer nur genau eine gleichzeitig aktiviert sein.
- `JComboBox` – Element, das die Auswahl eines Eintrags aus einer Liste erlaubt oder auch einen eigenen Eintrag ermöglicht
- `JEditorPane` – Ermöglicht die Eingabe von mehrzeiligen Texten

Quellen I

[1] Wikimedia Commons. *File:James Gosling 2008.jpg — Wikimedia Commons, the free media repository*. 2015. URL: `https://commons.wikimedia.org/w/index.php?title=File:James_Gosling_2008.jpg&oldid=149207971` (besucht am 05. 02. 2019).

[2] Prof. Dr. Andreas Judt. *Software Engineering 2. Entwurfsmuster*. 2016.

[3] C. Ullenboom. *Java ist auch eine Insel: Das umfassende Handbuch*. Rheinwerk Computing, 2014. ISBN: 978-3-8362-5869-2.

Quellen II

[4] C. Ullenboom. *Java SE 8 Standard-Bibliothek: das Handbuch für Java-Entwickler ; [Nebenläufigkeit, String-Verarbeitung, Datenstrukturen und Algorithmen, XML, RMI, JDBC, Reflection, JavaFX, Swing, Grafik- und Netzwerkprogrammierung ; JNI, Sicherheit]*. Galileo Computing. Galileo Press, 2014. ISBN: 9783836228749. URL: https://books.google.de/books?id=D3jSnQEACAAJ.

[5] Wikipedia contributors. *History of the graphical user interface — Wikipedia, The Free Encyclopedia*. 2019. URL: https://en.wikipedia.org/w/index.php?title=History_of_the_graphical_user_interface&oldid=882638996 (besucht am 05. 02. 2019).

Kontakt

- E-Mail: `lukas.abelt@airbus.com`
- GitHub: `https://www.github.com/LuAbelt`
- GitLab: `https://www.gitlab.com/LuAbelt`
- Telefon(Firma): 07545 - 8 8895
- Telegram: LuAbelt