

Programmiersprachen  
Vorlesung 8 vom 06.12.21  
Programmierparadigmen

Christoph Lüth

Universität Bremen

Wintersemester 2021/22

## Vortragsthemen

06.01.2022 Do	Jan Walther: Rust	Felix Glinka: LLVM IR
10.01.2022 Mo	Mamia Lehib: JavaScript	Ole Fischer: TypeScript
13.01.2022 Do	Ove von Stackelberg: Golang	Jan Engel: Swift
17.01.2022 Mo	Fida Fahmadi: Kotlin	Hannes Wehrmann: Scala
20.01.2022 Do	Tarek Soliman: LISP	Jona Dirks: Prolog
24.01.2022 Mo	Jakob Gahde: Ocaml	Nele Matschull: Idris
27.01.2022 Do	Philip Hönnecke: Ada	Leon Ehrardt: Cobol
31.01.2022 Mo	Tobias Stage: ABAP	Larissa Fastenau: VHDL
03.02.2022 Do	Lennard Scheurer: Lua	Johanna Tholen: Elm

## Wo sind wir?

- ▶ Einführung
- ▶ Werte und Typen
- ▶ Anweisungen, Variablen und Zustand
- ▶ Kontrollabstraktion
- ▶ Datenabstraktion
- ▶ Fortgeschrittene Typsysteme
- ▶ Nebenläufigkeit
- ▶ **Programmierparadigmen**
- ▶ Skriptsprachen
- ▶ Ab Woche 11: Studentische Vorträge.

# Programmierparadigmen

## Was ist ein Paradigma?

Definition nach dem Duden:

**Bedeutungen (2)**

1. Beispiel, Master; Erzählung mit beispielhaftem Charakter  
Gebrauch **bildungssprachlich**
2. Gesamtheit der Formen der Flexion eines Wortes, besonders als Master für Wörter, die in gleicher Weise flektiert werden  
Gebrauch **Sprachwissenschaft**

Definition nach Merriam-Webster:

**Full Definition of paradigm**

- 1 : EXAMPLE, PATTERN  
*especially*: an outstandingly clear or typical example or archetype  
// ... regard science as the paradigm of true knowledge.  
— G. C. Meigs
- 2 : an example of a conjugation or declension showing a word in all its inflectional forms
- 3 : a philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated  
// the Freudian paradigm of psychoanalysis  
*broadly*: a philosophical or theoretical framework of any kind

## Was ist ein Programmierparadigma?

**Programmierparadigma**

Ein Programmierparadigma ist eine **grundlegende Herangehensweise** an die Programmierung, und umfasst:

- ▶ ein **Berechnungsmodell** — wie rechne ich?
  - ▶ ein **Weltmodell** — was sind die Objekte meiner Berechnungen?
  - ▶ sowie darauf aufbauende Konzepte.
- ▶ Programmierparadigmen sind mehr als nur eine bestimmte Programmiersprache.
  - ▶ Ein Programmierparadigma bedingt auch eine Modellierung und Systemarchitektur.

## Bekannte Programmierparadigmen

- ▶ Prozedural-Imperativ
- ▶ Objektorientiert
- ▶ Funktional
- ▶ Logisch

## Fallstudie

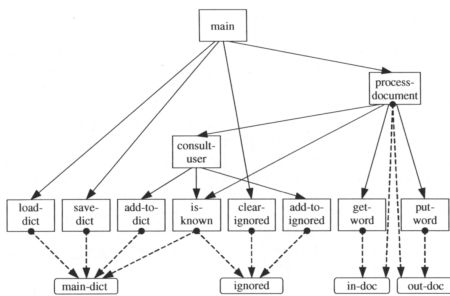
- ▶ Ein einfacher Spellchecker:
- ▶ Ablauf:
  - ▶ Eingabe: Text aus einer Datei
  - ▶ Zerlegt Text in Wörter
  - ▶ Prüft Wort gegen Wörterbuch
  - ▶ Wenn Wort nicht in Wörterbuch: ignorieren oder hinzufügen
  - ▶ Ausgabe: korrigierter Text in neuer Datei
- ▶ Quelle: Watt, Programming Language Design Concepts.

# Imperative Programmierung

## Prozedural-Imperativ

- ▶ Berechnungsmodell: Zustandsübergänge
- ▶ Weltmodell: Abstraktion des Speichers
- ▶ Schlüsselkonzepte:
  - ▶ Variablen und Befehle
  - ▶ Prozeduren und Funktionen
  - ▶ Einfache Datentypen
- ▶ Prozedural: Abstraktion durch Funktionen und Prozeduren
- ▶ Mutter aller Programmierparadigmen
- ▶ Programmiersprachen: C (uvm)

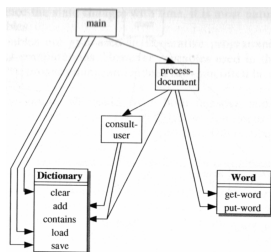
## Imperative Systemarchitektur



## Fallstudie imperativ

- ▶ Ablauf ist imperativ
- ▶ Keine Datenabstraktion:
  - ▶ Wörterbuch und Ignore-Liste globale Variablen
  - ▶ Ein- und Ausgabedatei global
  - ▶ Wörter sind `char *`, müssen anders deklariert werden

## Prozedurale Systemarchitekture



## Fallstudie prozedural

- ▶ Datenabstraktionen:
  - ▶ Funktionen zu `Dictionary` zusammengefasst
  - ▶ Dadurch Vereinfachung: `main_dict` und `ignored` sind beides `Dictionary`s.
  - ▶ Datenabstraktion zu `Word` immer noch unvollständig
- ▶ Erste Anflug von Objekt-Orientierung

# Objekt-Orientierte Programmierung

## Objekt-Orientiertes Paradigma

- ▶ Objekte der Berechnung sind **Objekte**
- ▶ Berechnung sind (abstrakte) Nachrichten (**Methoden**), welche die Objekte austauschen.
- ▶ Im einzelnen:
  - ▶ Verkapselung der Objekte als **abstrakte Datentypen**
  - ▶ Typisierung der Objekte durch Klassen
  - ▶ Strukturierung der Klassen durch **Vererbung**
  - ▶ Methodenaufruf durch **dynamische Bindung**
  - ▶ **Polymorphie** durch Subtyping

## Geschichtliches

- ▶ Erste Sprache: Simula (1960s), Ole-Johann Dahl und Kristen Nygaard
- ▶ Simula kannte Objekte, Klassen, Vererbung, Koroutinen und hatte Speicherverwaltung.
- ▶ Spätere Sprachen: Smalltalk-80, C++, Eiffel, Java
- ▶ Noch spätere Sprachen: Ruby, Python, C#, Scala, OCaml

## Vererbung

- ▶ Vererbung löst ein Problem mit abstrakten Datentypen:
  - ▶ ADTs sind **verkapselt** und könnten gut erweitert und wiederverwendet werden.
  - ▶ Aber: Verkapselung verhindert Wiederverwendung
  - ▶ Ferner: keine **hierarchischen** Definitionen
- ▶ Arten der Vererbung:
  - ▶ Einfache Vererbung (jede Klasse hat **eine** Oberklasse)
  - ▶ Mehrfachvererbung (jede Klasse hat **mehrere** Oberklassen)

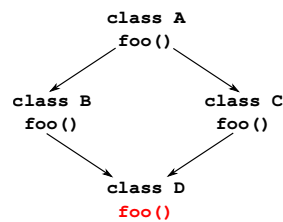
## Probleme mit Mehrfachvererbung

1 Das **Diamanten-Problem** (diamond problem):

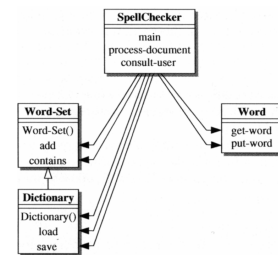
- ▶ B erbt von A und überschreibt `foo`
- ▶ C erbt von A und überschreibt `foo`
- ▶ D erbt von B und C — was ist `foo` in D?

2 Implementation

- ▶ Sprachen mit Mehrfachvererbung: C++, Eiffel
- ▶ Sprachen mit Einfachvererbung: Java, Python, ...
  - ▶ Dafür Interfaces und/oder Traits



## Die Fallstudie objekt-orientiert



# Funktionale Programmierung

## Funktionales Programmierparadigma

- ▶ Berechnungsmodell: rekursive Funktionen
- ▶ Weltmodell: algebraische Datentypen
  - ▶ frei definierbar — SML, Haskell
  - ▶ fest — LISP
- ▶ Schlüsselkonzepte:
  - ▶ Sprachen können pur (Haskell) oder gemischt sein (SML, LISP)
  - ▶ Getypt (Haskell, SML) oder ungetypt (LISP)
  - ▶ Strikt (SML, LISP) oder nicht-strikt (Haskell)

## Die Fallstudie funktional

- ▶ Erste Annäherung:
  - ▶ Modularchitektur ähnlich vorher
  - ▶ Implementation ähnlich Java
- ▶ Zweite Annäherung: funktionale Modellierung
  - ▶ spellcheck als stream processor

```
consultUser :: String -> Dictionaries -> IO (String, Dictionaries)
dropWord :: String -> (String, String)
getWord :: String -> Maybe (String, String)
```

# Zusammenfassung

## Zusammenfassung

- ▶ Programmierparadigmen bestehen aus einem Weltmodell und einem Berechnungsmodell
  - ▶ Was sind die Objekte meiner Berechnung?
  - ▶ Wie berechne ich?
- ▶ Beispiele:
  - ▶ imperativ und prozedural
  - ▶ objekt-orientiert
  - ▶ funktional
  - ▶ logisch
- ▶ Die meisten Programmiersprachen sind **gemischt** und unterstützen mehrere Paradigmen.
  - ▶ Aber meistens ein bestimmtes besonders gut.