

Grundlagen der Programmierung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesungsskript zum Sommersemester 2020

2. Vorlesung (27. April 2020)



Dr. Jin Gerlach
Christian Olt

Fachgebiet Wirtschaftsinformatik | Software & Digital Business
Fachbereich Rechts- und Wirtschaftswissenschaften
Technische Universität Darmstadt

Wir empfehlen euch die spielerische Einführung in diese Konzepte im Rahmen der „Hour of Code“

<http://studio.code.org/>
<http://studio.code.org/s/20-hour>



Minecraft

Baue dein eigene Minecraft-Version. Erschaffe deine eigene Version von Hühnern, Schafen, Creepern, Zombies und mehr.



Star Wars

Lerne, Droiden zu programmieren und erschaffe dein eigenes Star Wars Spiel in einer weit, weit entfernten Galaxie.

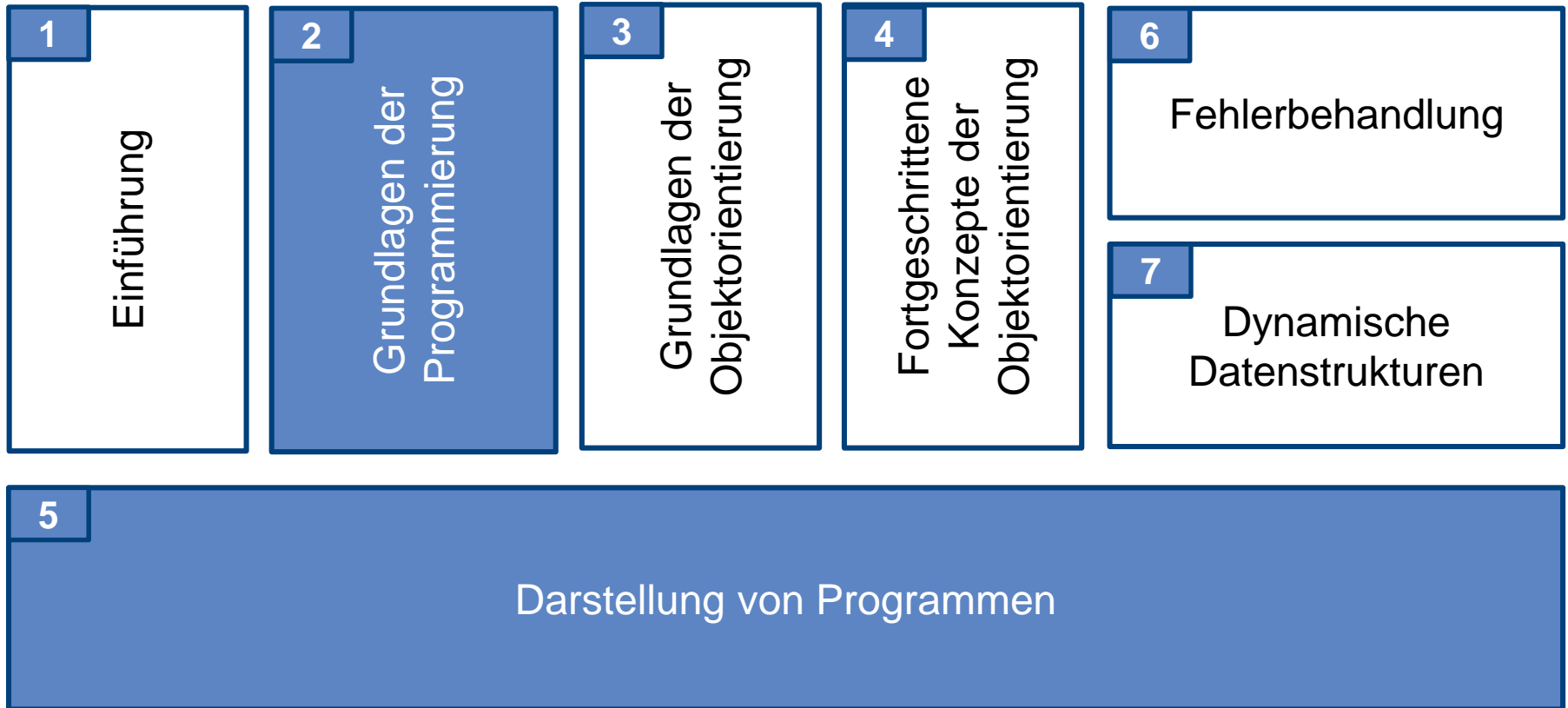


Die Eiskönigin

Verwende Code, um Anna und Elsa bei ihrer Entdeckung der Magie und Schönheit von Eis zu begleiten.



Thematische Übersicht



Kapitel 2: Grundlagen der Programmierung



TECHNISCHE
UNIVERSITÄT
DARMSTADT

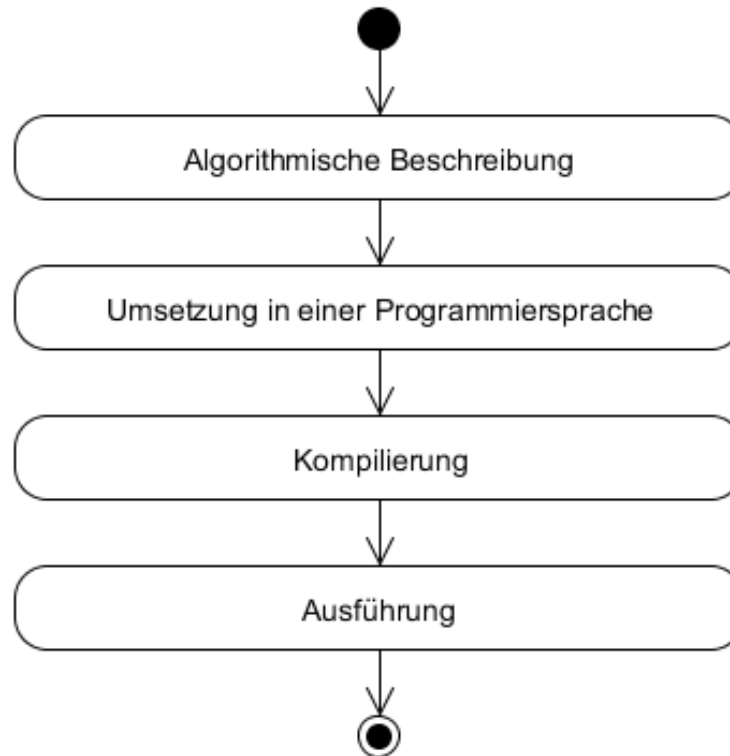
- Aktivitätsdiagramme der Unified Modeling Language (UML)
- Deklaration, Initialisierung und Nutzung von Variablen in Java
- Wichtige Elemente der Programmiersprache Java

Lernziele:

- Den Aufbau von Aktivitätsdiagrammen kennen und gegebene Algorithmen durch Aktivitätsdiagramme modellieren können
- Variablen deklarieren, initialisieren und verwenden können
- Wesentliche Elemente eines Java Programms kennen und verstehen

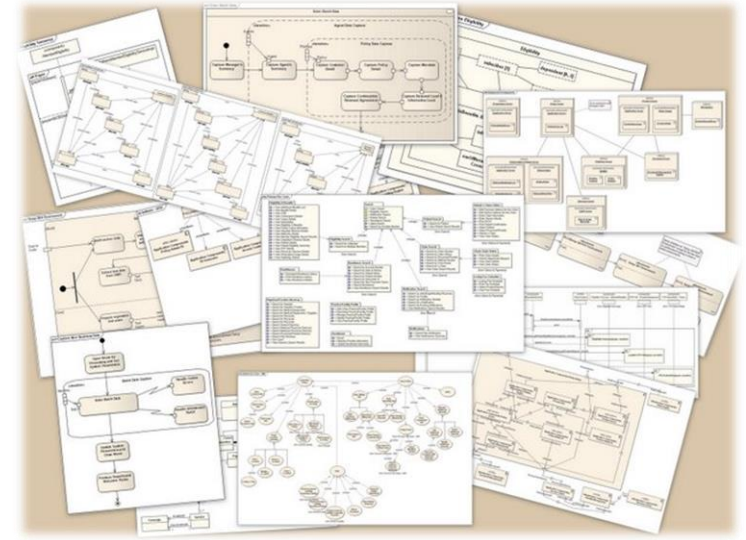


Zusammenfassung: Programmieren



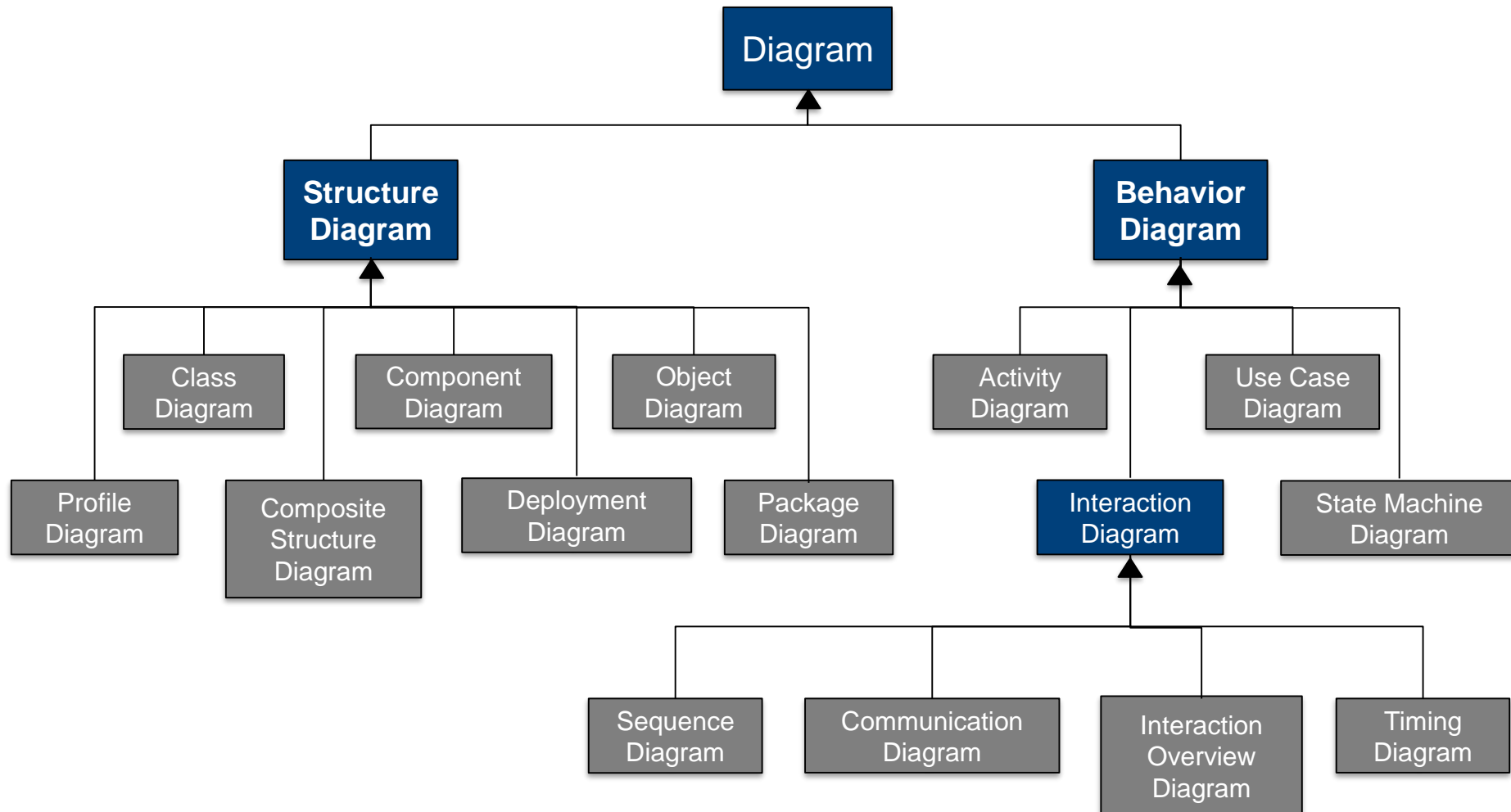
Modellierung mit der Unified Modeling Language (UML)

- Programme können grafisch dargestellt (modelliert) werden, das Ergebnis dieser Visualisierung sind **Diagramme** (Modelle)
- Ein Modell abstrahiert von technischen Feinheiten (visualisiert das Wesentliche) und ermöglicht somit eine effiziente Kommunikation (z. B. zwischen IT und Fachabteilung).
- Die Modellierungssprache **UML (Unified Modeling Language)** ist der Industriestandard bei der Modellierung von Softwaresystemen
- Entstehungsgeschichte:
 - 1990er: zahlreiche Modellierungsansätze
 - 1997: erste Version von UML
 - 2013 UML 2.5





Diagrammtypen der UML

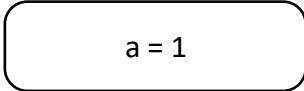








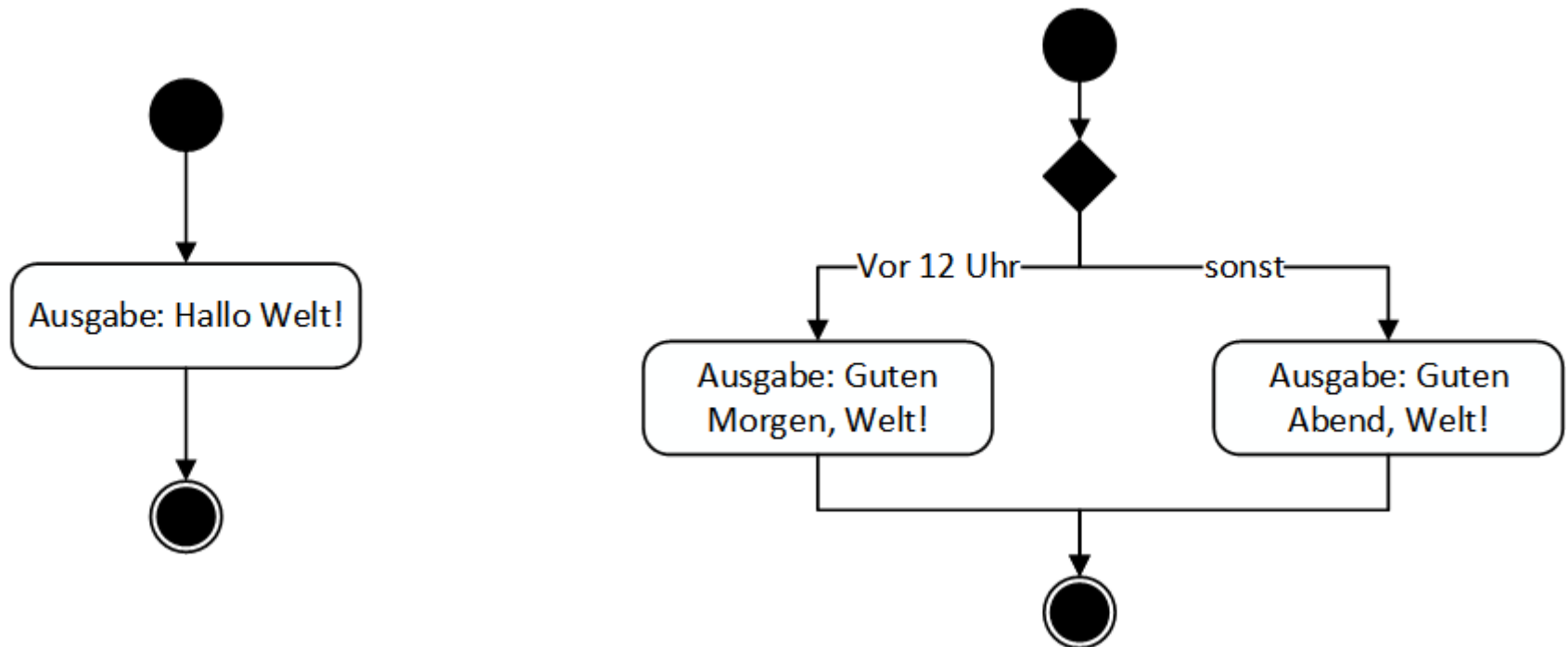
Algorithmen

- **Algorithmen** sind präzise Vorschriften zur Lösung eines Problems, welche die Handlungen und ihre Abfolge beschreiben
- Z. B. Kochrezepte, Bauanleitungen, Spielregeln...
- Ein Algorithmus kann von einem Menschen oder einer Maschine durchgeführt werden
- Wesentliche Bestandteile eines Algorithmus:
 - Eine zu bearbeitende Menge von Objekten in einem definierten Anfangszustand
 - Eine an den Objekten auszuführende Menge von Operationen
 - Gewünschter Endzustand der Objekte

Aktivitätsdiagramme (Notation)

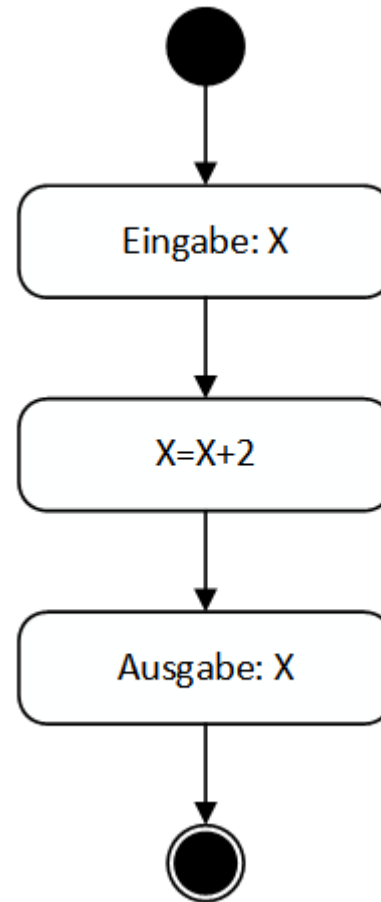
Symbol	Bedeutung
	Eine Aktivität modelliert die elementaren Handlungen (Eingaben, Ausgaben, Operationen) des Programms.
	Kontrollfluss
	Ein Kontrollfluss entspringt aus einem Startknoten.
	Ein Endknoten beendet das Aktivitätsdiagramm.
	Verzweigungsknoten spalten einen Kontrollfluss in mehrere Kontrollflüsse auf. Welcher der ausgehenden Kontrollflüsse verfolgt wird, ist von definierten Bedingungen abhängig.

Zwei erste Beispiele



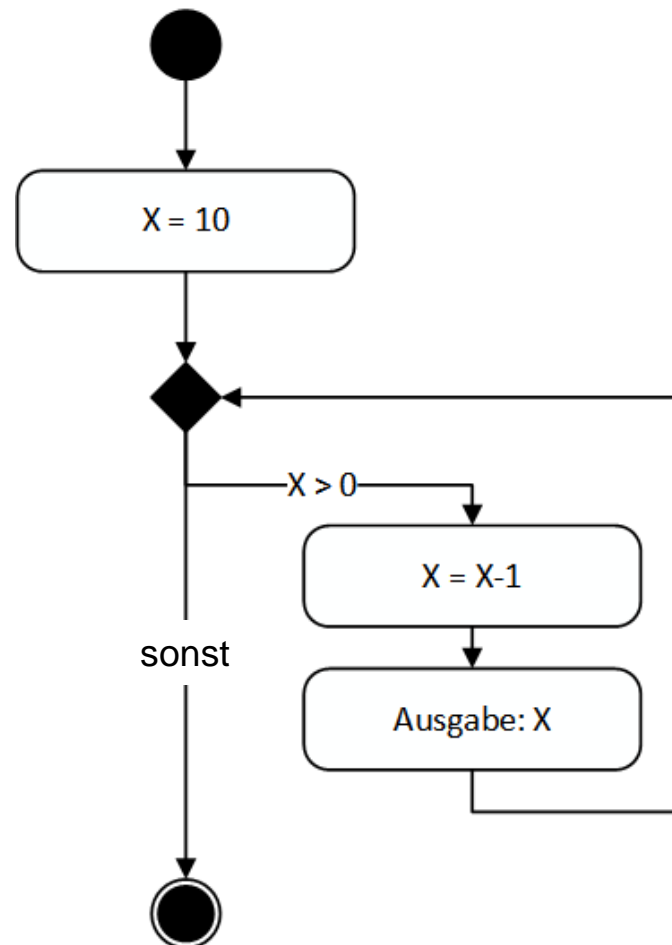
Typen von Aktivitäten

Aktivitäten können **Eingaben**,
Operationen und **Ausgaben**
sein!



Zyklen in Kontrollflüssen

Kontrollflüsse können auch **Zyklen** bilden (sogenannte Schleifen)



Modellierungsregeln für Aktivitätsdiagramme

- Es werden keine Entscheidungen bzw. Verzweigungen des Kontrollflusses innerhalb einer Aktivität modelliert
- Innerhalb einer Aktivität werden nur Handlungen des gleichen Typs zusammengefasst (Hinweis: Wir unterscheiden die drei Typen von Aktivitäten: Eingaben, Ausgaben, Operationen)
- Jedes Diagramm „startet“ mit genau einem Startknoten und „endet“ mit genau einem Endknoten
- Es „entspringen“ niemals mehr als ein Kontrollfluss aus dem Startknoten oder einer Aktivität
- Für jede Verzweigung sind auf den ausgehenden Kanten alle möglichen Fälle anzugeben (Hinweis: „sonst“ verwenden)

Das erste Programm



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
package de.tudarmstadt.helloworld;

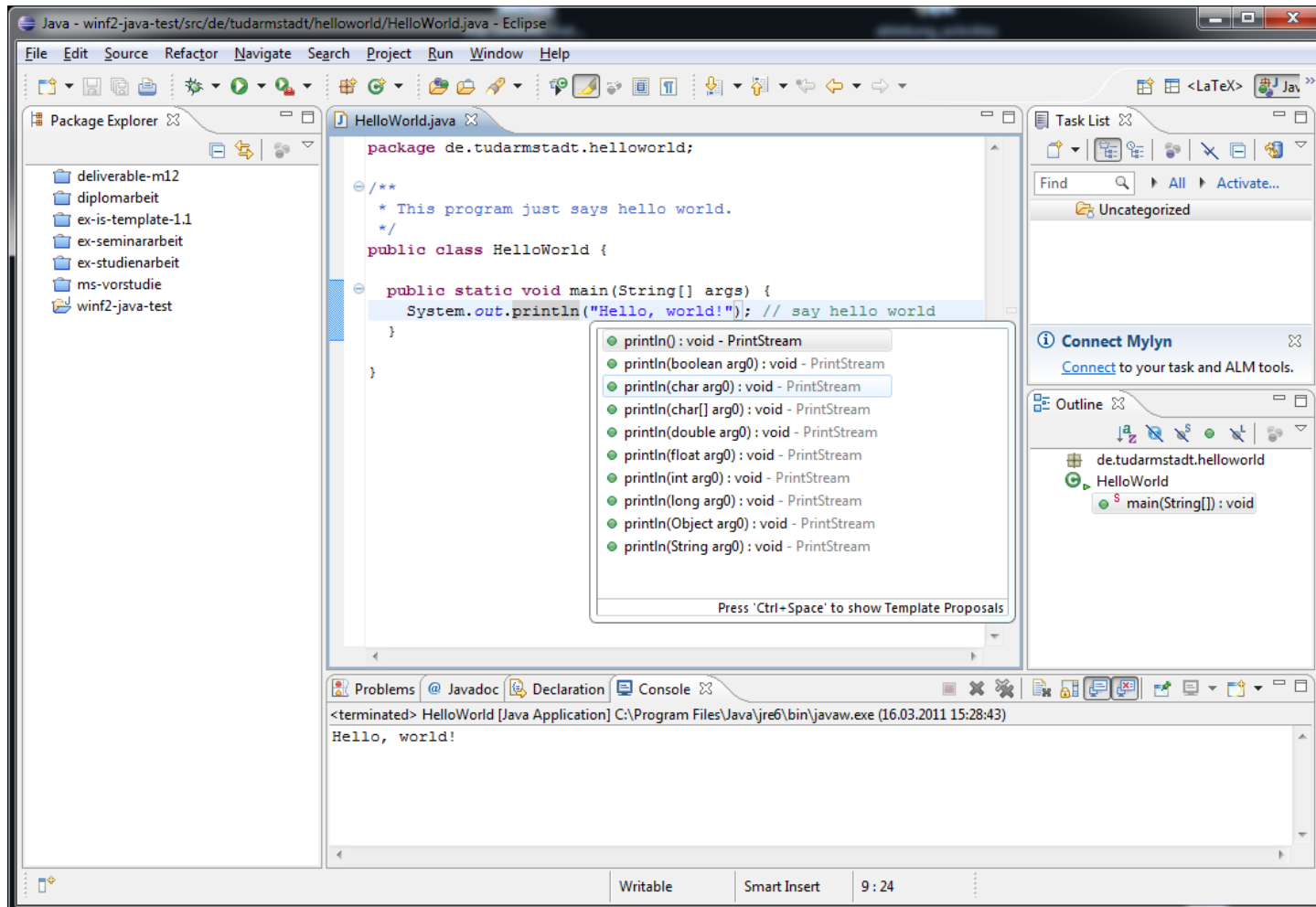
/**
 * This program just says hello world.
 */
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world!"); // say hello world
    }

}
```

- Programmierungsumgebungen (Integrated Development Environment: IDE) vereinfachen die Programmentwicklung
- Die wichtigsten Bestandteile sind:
 - ein Texteditor zum Erstellen und Ändern eines Programmtextes, der zusätzlich grafisch aufbereitet wird
 - ein Compiler bzw. Interpreter zum Ausführen von Programmen
 - ein Debugger für die Fehlersuche

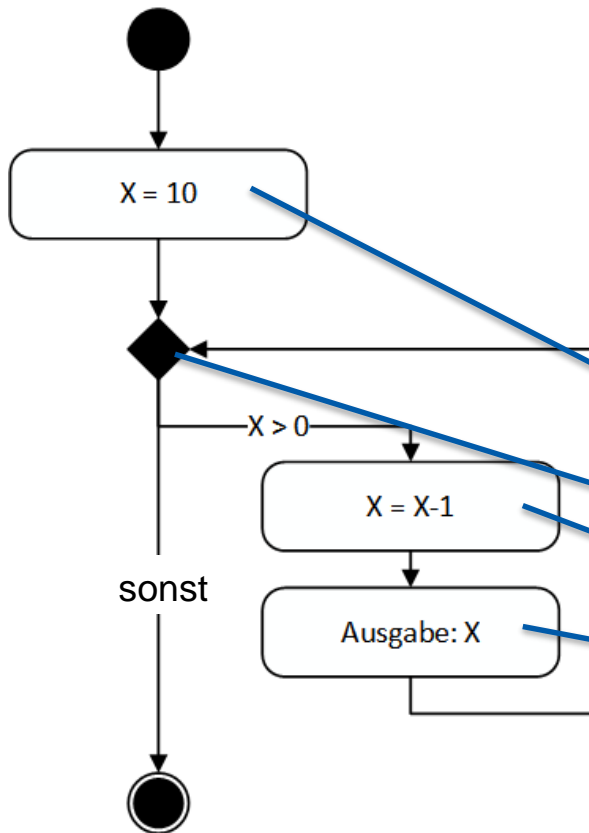
Beispiel: Eclipse IDE



Von UML zu Java



TECHNISCHE
UNIVERSITÄT
DARMSTADT



```
package de.tudarmstadt.helloworld;

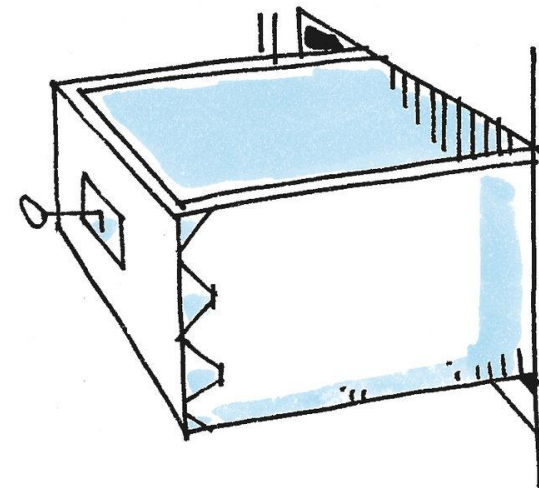
/**
 * This program does a little more.
 */
public class EinProgramm{

    public static void main(String[] args) {

        int x = 10;
        while (x > 0){
            x = x - 1;
            System.out.println(x);
        }
    }
}
```

Variablen und Arten von Datentypen

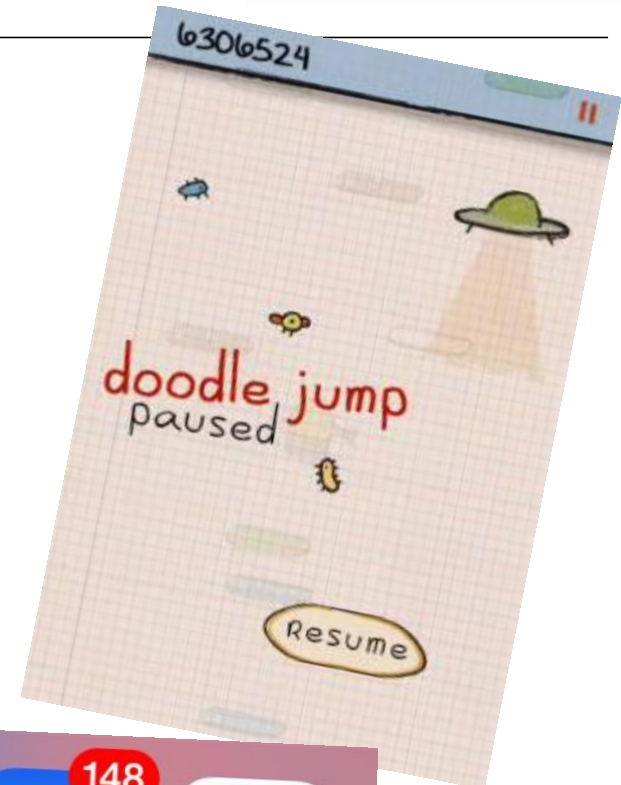
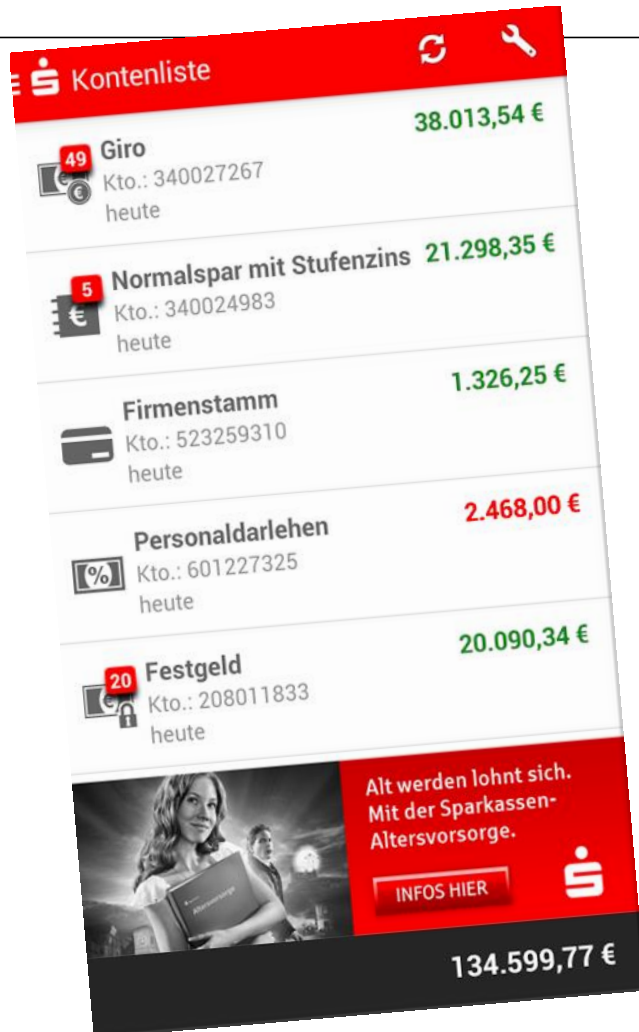
- **Variablen** sind Behälter für genau einen Wert und es gilt:
 - Variablen haben einen **Datentyp** (z. B. `int`)
 - Variablen haben einen **Bezeichner** (z. B. `x` oder `besteVariableDerWelt`)
 - Variablen haben einen **Wert** (z. B. `42`)
- Wir unterscheiden **zwei Arten von Datentypen**:
 - **Einfache** Datentypen:
 - Wahrheitswert (`boolean`) → `true`, `false`
 - Einzelzeichen (`char`) → z. B.: `'a'`, `'b'`, `'1'`, `'2'`, `'%'`, `'_'`
 - Numerische Datentypen
 - Ganzzahlige Datentypen (`byte`, `short`, `int`, `long`) → z. B. `123`
 - Gleitkommatypen (`float`, `double`) → z. B. `1.25`
 - **Strukturierte** Datentypen (auch: Referenz-Datentypen)



Variablen (Beispiele)



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Deklaration und Initialisierung von Variablen

- Variablen müssen vor der ersten Verwendung im Programm **dekliert** werden, dazu gibt man den **Datentyp** an und vergibt einen **Bezeichner**, optional kann die Variable auch **initialisiert** werden, d. h. sie bekommt schon einen ersten Wert zugewiesen. Beispiele für die **Deklaration** einer Variable:

```
int x; // Variablendeklaration ohne Initialisierung
int y = 42; // Variablendeklaration mit Initialisierung
int z = y + x + 1; // Variablendeklaration mit Initialisierung
```

- Variablen können im Programm **gelesen** und **geschrieben** werden. Ein Lesen der Variablen ist notwendig, wenn sie auf der rechten Seite einer Zuweisung verwendet werden

Aufgabe zur Variablendeklaration

Welchen Wert hat die Variable x nach der Ausführung des nachfolgenden Quellcodes?

```
public class VarExample {  
    public static void main(String[] args) {  
        int x = 19;  
        x = x + 23;  
    }  
}
```

Hier wird die Variable x deklariert, ihr wird der Wert 19 zugewiesen (x wird mit dem Wert 19 initialisiert) und später nach der Addition der Wert 42.

Aufgabe zu möglichen Fehlern bei der Variablendeklaration

Der folgende Programmcode enthält Fehler – in welchen Zeilen sind die Fehler?

```
1 package de.tudarmstadt.is.winf2;
2 public class FlaschenCode {
3     public static void main(String[] args) {
4
5         int c = a;
6         int b = 5;
7         int a;
8
9         d = b + 2;
10        c = 2 + b
11        b = b / b;
12        // Rechtschreibfehler;
13
14    }
15 }
```

Hinweis: Der Code enthält genau drei Fehler!



Dürfen wir vorstellen: PINGO

- Webbasiertes Live-Feedback-System
 - keine Installation
 - sofort nutzbar
 - intuitiv
 - anonym
- Wie funktioniert's?
 - **<http://pingo.upb.de>** öffnen und die Zugangsnummer für unsere Vorlesung angeben
 - Oder: den QR-Code scannen
 - mitmachen!



#9456



Aufgabe zu möglichen Fehlern bei der Variablendeklaration



ISCHE
TÄT
T
D
<http://pingo.upb.de>
#9456

Der folgende Programmcode enthält Fehler – in welchen Zeilen sind die Fehler?

```
1 package de.tudarmstadt.is.winf2;  
2 public class FlaschenCode {  
3     public static void main(String[] args) {  
4  
5         int c = a;  
6         int b = 5;  
7         int a;  
8  
9         d = b + 2;  
10        c = 2 + b  
11        b = b / b;  
12        // Rechtschreibfehler;  
13  
14    }  
15 }
```

Hinweis: Der Code enthält genau drei Fehler!

Details zu Datentypen

- Der **Datentyp** einer Variablen definiert:
 - den Wertebereich
 - einen Standardwert (auch: Default-Wert)
 - eine Menge zugehöriger Operationen
- Der Compiler kann prüfen, dass Variablen nur erlaubte Werte enthalten und nur erlaubte Operationen angewendet werden, dadurch können Fehler vermieden werden
- Beispiel:
`int x;`
 - Variable `x` ist vom Datentyp `int`, der Standardwert ist `0`, der Wertebereich ist `[-2147483648; 2147483647]` und ganzzahlig, erlaubt sind alle arithmetischen Operationen

Wertebereiche von einfachen Datentypen

Typ	Wertebereich
boolean	true oder false
char	16-Bit-Unicode-Zeichen (0x0000 ... 0xFFFF)
byte	-2^7 bis $2^7 - 1$ (-128 ... 127)
short	-2^{15} bis $2^{15} - 1$ (-32.768 ... 32.767)
int	-2^{31} bis $2^{31} - 1$ (-2.147.483.648 ... 2.147.483.647)
long	-2^{63} bis $2^{63} - 1$ (-9.223.372.036.854.775.808 ... 9.223.372.036.854.775.807)
float	$1,4 \cdot 10^{-45}$ bis $3,4 \cdot 10^{+38}$ (positiv und negativ)
double	$4,9 \cdot 10^{-324}$ bis $1,7 \cdot 10^{308}$ (positiv und negativ)

Beispiel: Deklaration und Initialisierung von einfachen Datentypen



```
package de.tudarmstadt.is.winf2;

public class BeispieleDatentypen {
    public static void main(String[] args) {

        boolean einBoolean = true;
        char einChar = 'a';

        byte einByte = 42;
        short einShort = 200;
        int einInt = 30000;
        long einLong = 1000000;

        float einFloat = 3.14f;
        double einDouble = 3.14159265;
    }
}
```

- Kommentare dienen der Erläuterung von Quellcode, werden beim Kompilieren verworfen und haben somit keine Auswirkungen auf die Programmlogik
- Zwei wesentliche Arten von Kommentaren:
 - **Einzeilige Kommentare** erstrecken sich bis zum Zeilenende, z. B.:
`System.out.println("text"); // Kommentar`
 - **Mehrzeilige Kommentare** erstrecken sich über mehrere Zeilen, z. B.:
`System.out.println("text"); /* erste Zeile
 zweite Zeile */`

- Schlüsselwörter haben eine vordefinierte symbolische Bedeutung und können nicht als Bezeichner verwendet werden:

<code>abstract</code>	<code>assert</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>
<code>case</code>	<code>catch</code>	<code>char</code>	<code>class</code>	<code>const</code>
<code>continue</code>	<code>default</code>	<code>do</code>	<code>double</code>	<code>else</code>
<code>enum</code>	<code>extends</code>	<code>final</code>	<code>finally</code>	<code>float</code>
<code>for</code>	<code>goto</code>	<code>if</code>	<code>implements</code>	<code>import</code>
<code>instanceof</code>	<code>int</code>	<code>interface</code>	<code>long</code>	<code>native</code>
<code>new</code>	<code>package</code>	<code>private</code>	<code>protected</code>	<code>public</code>
<code>return</code>	<code>short</code>	<code>static</code>	<code>strictfp</code>	<code>super</code>
<code>switch</code>	<code>synchronized</code>	<code>this</code>	<code>throw</code>	<code>throws</code>
<code>transient</code>	<code>try</code>	<code>void</code>	<code>volatile</code>	<code>while</code>

- **Bezeichner** (Namen) benennen Variablen, Methoden, Klassen und Schnittstellen, etc. und ermöglichen es so, die entsprechenden Bausteine anschließend im Programm identifizieren und zu referenzieren
- Hinweis: Bei Bezeichnern und anderen Elementen unterscheidet Java strikt zwischen **Groß- und Kleinschreibung!**
- Folgende Regeln sollen bei **Vergeben von Bezeichnern** beachtet werden:
 - Erstes Zeichen: A-Z, a-z, _, \$
 - Weitere Zeichen: A-Z, a-z, _, \$, 0-9
 - Es dürfen keine Schlüsselwörter verwendet werden
 - „true“, „false“, „null“ sind verboten
 - Drei weitere Bezeichner sind Namen vordefinierter Klassen und dürfen somit nicht verwendet werden: „Object“, „String“, „System“

Beispiel: Ungültige Bezeichner

Ungültige Bezeichner	Grund
2und2macht4	Das erste Symbol muss ein Java-Buchstabe sein und keine Ziffer.
hose gewaschen	Leerzeichen sind in Bezeichnern nicht erlaubt.
Faster!	Das Ausrufezeichen ist, wie viele Sonderzeichen, ungültig.
null, class	Der Name ist schon von Java belegt. Null – Groß-/Kleinschreibung ist relevant – oder cláss wären möglich.