**Softwaretechnik**



1. Einführung und Grundlagen

• Was Software ist und welche Eigenschaften sie hat

• Der Einfluss von Lehmann und Beladys Software-Gesetzen

• Softwaretechnik als Ingenieursdisziplin

2. Vorgehens- und Prozessmodelle

• Arten von Vorgehens- und Prozessmodellen

• Agile Methoden

• SCRUM, DevOps, Continuous Delivery

3. Projektmanagement

• Stellungwert von Planungen

• Bestandteile von Softwareprojekten

• Arten von Projektplänen

• Identifikation und Gliederung von Aufgaben

• Risikomanagement

4. Objektorientierte Modellierung mit der UML

• Bedeutung von der Modellierung

• Diagrammtypen und Einsatzvarianten

• Notationselemente

5. Anforderungsanalyse

• Schritte der Anforderungsanalyse

• Schritte der Ist-Analyse

• Erhebungstechniken

6. Anforderungsspezifikation

• Bedeutung und Eigenschaften

• Gliederung und Kriterien

• Prozesse der Anforderungsspezifikation

• Prüfung der Anforderungsspezifikation

7. Testverfahren

• Testarten und -varianten

• Testabdeckungsmaße

• Testvorbereitung,-durchführung und -protokollierung

8. Änderungs- und Konfigurationsmanagement

• Wartung, Evolution und Reengineering

• Bedeutung der Softwarewartung

• Änderungsprozesse

• Werkzeuge für das Konfigurationsmanagement

9. Implementierung

• Programmiersprachen und -richtlinien

• Code-Qualität und Metriken

• Vermeidung von Code-Redundanz

• Entwicklungsumgebungen

Einführung und Grundlagen

Was ist Software

Aufgabe / Problem

Bedienungsanleitung

Programmdokumentation

Hilfe, Daten

Code: Programmcode

Umgebung/ Interpreter/Compiler/IDE/Sprache/Speicher/Hardware/Betriebssysstem

Anforderungen, Benutzer, Anwender, Sicherheit

Service / Updates

Software als Produkt

Beliebig kopierbar

Key / Lizenz

Wasserfall-Modell

Analyse Entwurf Implementierung Test Wartung & Evolution

Lehman und Beladys „Gesetze“

Gesetz vom fortwährenden Wandel

Software löst ein Problem der realen Welt, die reale Welt ändert sich, Software muss sich anpassen, bis sie abelöst wird

Gesetz der zunehmenden KOmplexxit#t

Die Komplexität der Software erhöht sich, wenn keine Konsequnezen ergriffen werden

Konsequenzen

Standardsoftware wie Word, Excel

Individualsoftware wie kundenspezifisch

Heute Softwareentwicklung

In Phase

Projekt-, Qualitäts und Konfigurationsmanagement

**Manifest** für agile Softwareentwicklung

Wir erschließen bessere Wege, Software zu entwicklen, indem wier es selbst tun und anderen dabei helfen.

Duch diese Tätigkeit haben wir diese Werte zu schätzen glernt:

Individuen und Interaktionen mehr als Prozesse und Werkzeuge

Funktionierende Software mehr als umfasende Dokumenation

Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung

Reagieren auf Veränderung mehr als das Befolgen eines Plans

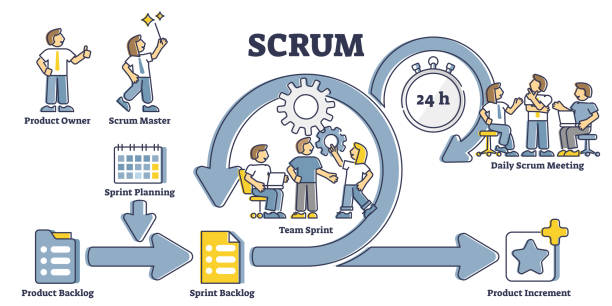
Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden.

Schätzen wir die Werte auf der linken Seite höher ein.

* Menschen und Kuden sind wihtiger Prozesse

Scrum

Prozessbeschreibung um einen Prozess durchzulaufen mit Feeback, Sprint



DevOps Development Operation

Die Programmierer sind auch für den Release etc. zurück

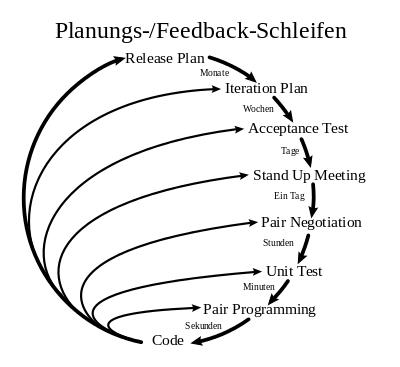
Continuous Delivery

Öfter updaten, Features erstmals mit wenigen Kunden test

Extremeprogramming

Das spezielle ist Pair programming

Es gibt zweier Teams, die sich abwechseln



Es gibt noch Trello:

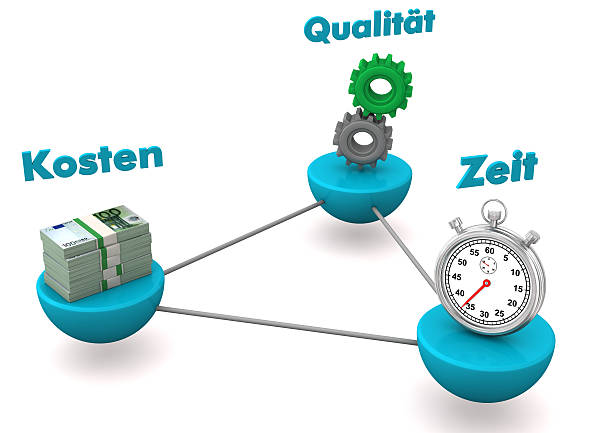
Ist online und kostenlos für kleine Projekte

Zur Doku, man sieht was man geschafft hat, welche Software, welches Pflichtenheft, welche Tests

Säulen des Managements

Kosten, Qualität und Zeit

Besonders im deutschsprachigen Raum ist das “Magische Dreieck” als Symbol für drei zentrale Faktoren des **Projektmanagements** bekannt. Bei den Faktoren handelte es sich um Zeit, Kosten und Leistung (auch Zeit, Kosten, Ergebnis/Qualität). Das Ziel bzw. der Erfolgsfaktor positioniert sich dabei in der Mitte des Dreiecks



**Qualität**

Design / Image

Standard

Fehler / Stabilität

Lifetime / Lebesdauer

Erwatungshaltung

Material

Preis LeistungsVerhältnis

Sicherheit

Preis

Bohrmaschine

Welche Klasse?

Es gibt Preis-, Leistungssieger und Gesamtsieger

Was wird getestet

Sicherheit, Funktionalität, Garantie, Material /Schadstoffe, Abnutzung, Verbrauch, Ergonomie, Lautstärke, Energieverbrauch, Bedienungsanleitung, Sicherheitsanleitung

Bei SW werden Fehler pro Programmzeilge (ohne Kommentare) gerechnet

Speicherzugriffsfehler (Variablen falsch), nebenläufige Fehler,

Es gibt professionelle Fehlersucher

Standardsoftwre

Individualsoftware 100 % Kundenwunsch

Planung am Anfang:

Gliederung in Phasen, Aktivitäten, Arbeitspakete

Zeitlicher Ablauf

Auswahl eines geigneten

Plan macht prinzipiell Aussagen zu den folgenden W-Punkten

Warum

Was getran wird

Für wieviele geld

Von wem

Ist/Soll →←Implementierung

**Anforderungskatalog**

Kann bestehen aus:

Use-Cases

Anwendungsabläufe

Funktionsbeschreibungen

Stakeholders: Kundenfirma wie Mitarbeiter, Anwender, Firmenleitung

Alle gewünschten Features sammeln durch Interview, Beobachtung

Pro Feature:

Titel: may 3 Zeilen

Analyse

User-Stories

Geschäftsanwendungsfälle

Systemanwendungsfälle

Buch S. Analyse & Design S. 39 – S. 114

MVC-Modell als Application

Grafische Benutzerschnittstellen folgen dem Model-View-Controller (**MVC**) Prinzip des Software Engineering. Dieses besagt, dass die grafische Sicht (View) von den Daten (Model) getrennt sein sollte. Das "Model" könnte ein Buchladen oder eine Personalverwaltung sein. Diese Modellschicht sollte unabhängig vom grafischen Zugriff existieren, damit auch bei einer komplett anderen GUI die Modellschicht weiter verwendet werden kann. Der "Controller" verbindet Model und View. Wir sehen uns zunächst die View-Schicht an und kommen dann zur Controller-Schicht (über Lambda-Ausdrücke).

**M**odel wie application wie stage, die Bühne mit verschiedenen Szenen

Unsere API (Schnittstelle) nach außen zur Steuerung

Public Felder, Methoden

**View** wie fxml ist eine scene

**Controller** wie Controller

Singelton: Es gibt nur eine Instanz einer Klasse

Dokumentationn in JavaDoc

Codedokumentation (java.doc)

Klassendiagramme (UML)

Testdokumentation

* Benutzerhandbuch (FAQ)
* Online Hilfe

Vertrag / LizenzvereinbJaarung

Garantie, Service, Support

Schulung(sunterlagen)

ISO 9001

Zertifikat für Dokumentationsprozesse in der Firma

Vorgehen:

1. Features sammeln

2. Features mit voraussichtlichen Arbeitstagen bewerten

3. vom Auftraggeber die Prioritäten geben lassen

4. Nach Prioritäten und Manntagen die Features ordnen

Berechnung der Manntage

Beispiel

5 Tage \* 5 Mann \* 0,7 (Faktor für die tatsächliche Produktivität)

= 17,5 Manntage

Javadoc für Java

Doxygan ist javadoc für nicht java

Kommentar in java

//

/\*…\*/

/\*\*…

\*

\*  
\*/

So waren die Dokumentationen frpher

man 3 printf

man man

<http://www.scalingbits.com/java/javakurs1/javadoc>

link ist ein Verweis

Tripel /\*\* einfügen mit gewünschtem Text

@ einfügen für Author

über cmd, wo ich bin: ls

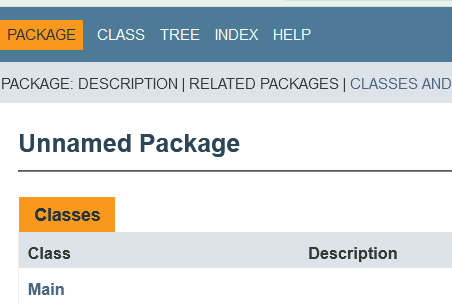
in die src gehen: evtl. cd src

evtl. javadoc -h

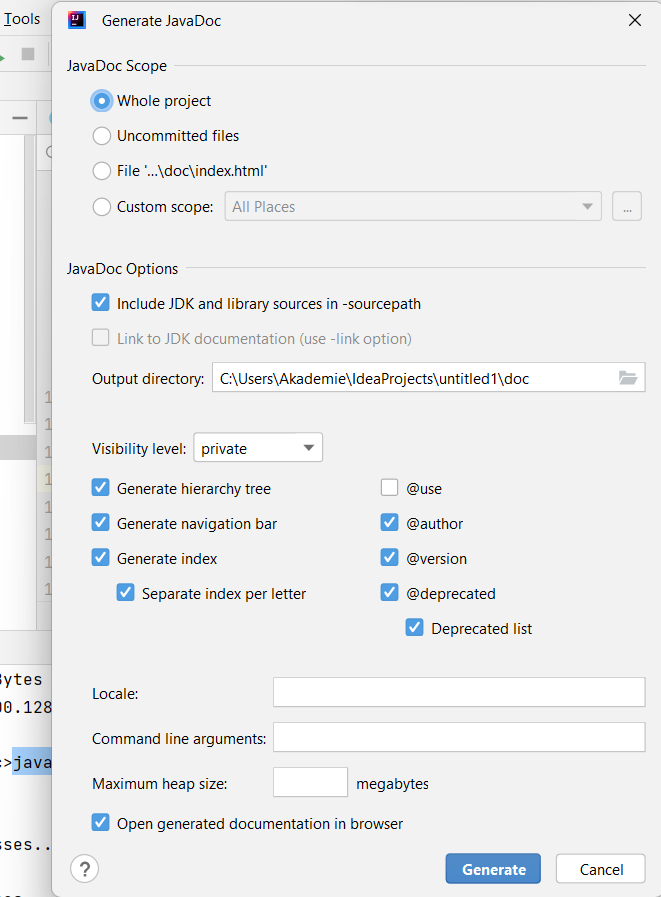
javadoc -author -private -d ../doc Main.java Einkaufsliste.java

In der index.html rechts kliceken und bei auf Firefox

Dann klick auf Main



oder als



UML

Unified Modeling Language

U wie Vereinheitlich

M nicht für Programme

nicht für Programmiersprachen

für Prozesse, Abläufe, Zusammenhänge

l für virtuelle Sprache

nicht eine Diagrammart

auch Zustandsdiagramm

ein Beispiel als Exkurs

FSM wie FiniteStateMaschine

EndlicherAutomat

ReguläreAusdrücke wie TelefonNr wie +49421-830-0

zunächst endlichen Automat

Start wie Anfang

dann reguläre Ausdrücke

+ mindestens 1

+?[0-9-]+

enum = {Start, E09, E09M, Ende}

while input not empty

z = next input

switch zustand

case Start:

if z in (0,9):

zustand = E09M

else if z = +

zustand = E09

else

raise Error oder throw

case E09:

if z in (0,9]

zustand = E09M

else raise Error

case E09M

if z in (0,9 )

//nix

else if z == „-„ zustand = E09

else raise Error

default raise Error

while end

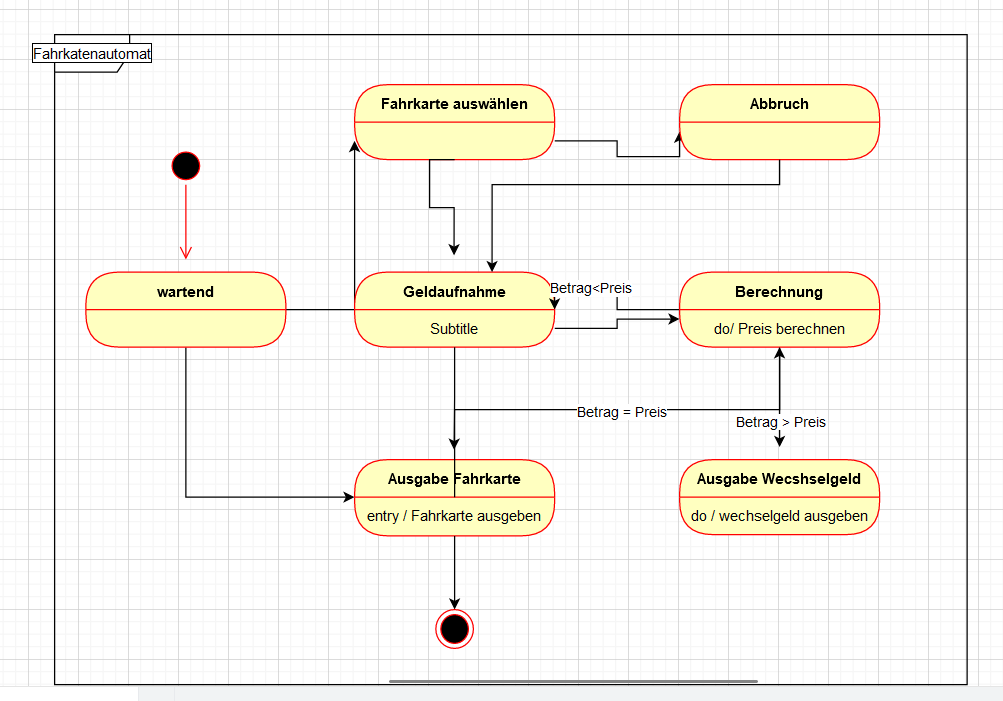
if zustand ==E09M return ok

else raise Error

**Fahrkartenautomat**

als Zustandsdiagramm

normalerweise vom Diagramm zum Programm



pfeil anklicken und Text eingeben

UML-Klassendiagramm

Variable: Typ

Attribute

- name ist private und der Typ

+

-

\* bedeutet, 1 oder mehrere

wer die enthält, bekommt den Diamant, ein Teil vom Ganzen

Beziehungen zwischen den Klassen

Vererbung, die Superklasse bekommt den Pfeil

Buch S. 7 Objektorientierung

Buch S. 107 Klassendiagramme

Klassendiagramme

Konzept objektorientierter Modellierung sind die **Objekte** bzw. deren Abstraktion zu Klassen. Letztere werden mit **Klassendiagrammen** modelliert.

„Innenleben“ der Objekte, d. h. ihre Attribute und Operationen sowie ihre Beziehungen nach außen, Diagrammtyp stellt das „Innenleben“ der Objekte, d. h. ihre Attribute und Operationen sowie ihre Beziehungen nach außen, d. h. Generalisierungs­, Assoziations­ und Abhängigkeitsbeziehungen, dar

für ein Attribut keine Multiplizität angeben, so wird 1..1 als Vorgabewert angenommen. Häufig verwendete Multiplizitäten und ihre Bedeutung:

ƒ0..1: optionales Attribut, d. h. im Objekt ist für das Attribut höchstens ein Wert angegeben.

ƒ1..1 bzw. alternativ 1: zwingendes Attribut, d. h. im Objekt wird genau ein Wert angegeben.

ƒ0..\*: optional beliebig, d. h. beliebig viele, Elementanzahl kann auch null sein. Als abkürzende Schreibweise ist der Stern („\*“) üblich.

ƒ1..\*: beliebig viele Werte, aber mindestens einer.

ƒn..m: fixiert, d. h. mindestens n, aber höchstens m Elemente. Die Multiplizitätsgrenzen werden hierbei durch beliebige natürliche Zahlen, die Ziffer 0 oder das Sternsymbol ausgedrückt. Für den Sonderfall, dass für n und m dieselbe Zahl gewählt wird, darf auch abkürzend diese allein angegeben werden. So kann beispielsweise „5..5“ auch als „5“ dargestellt werden

Klasse

Multiplizität (kommt an die Enden ran

Assoziation 1 oder Pfeile, die Klassen kennen sich

Aggregation leerer Diamant bzw. Raute

Komposition gefüllter Diamant bzw. Raute

gestrichelte Linie: die Klassen kennen sich

Realisation spezifischer, mehr als normale Verbung Dreiecke, Pfeil geht gegen Superklasse

Vererbung

Ich benutze die Datetime-Klasse, also kommt der Pfeil zu mir

Ich gehe zum Supermarkt, dann kommt der Pfeil zu mir, der Supermarkt kennt uns nicht, aber wir kennen den Supermarkt

Aggregation: meine Einkaufsliste kann sowohl im Supermarkt als auch auf dem Wochenmarkt genutzt werden

Kompostion: meine fünf Finger gehören zu mir

die Komposition ist stärker als Aggretation, die Komposition ist ein Bestandteil der anderen Klasse

eine Bestellung wird bei genau einem Kellner aufgegeben, ein Kellner kümmert sich um mehrere Bestellungen

Bestellung \*------1 Kellner

Eine Messe wird von mehreren Besuchern

Eine Reise kann mehrere Ausflüge beeinhatlen, ein Auflug

Im laufe einer Fußballsaison nehmen mehrere Spieler an mehren Spielen teil. Jeder Spieler schießt bei jdem Spiel eine bestimmte unterschiedliche Anzahl an Toren

Spiel \* \* Spieler

Tor

Jedes Restaurant beinhaltet mindestens eine Küche, eine Küche gehört zu genau einem Restaurant

Restaurant Küche

Jeder Vortrag wird entweder von einem Professor oder von einem Assistenten gehalten

Vortrag

\*

1 Professor

Abstract Mitarbeiter Assistentin

Sequenzdiagramm

Beim Sequenzdiagramm steht der zeitliche Verlauf der Nachrichten im Vordergrund. Die Reihenfolge der Nachrichten entspricht ihrer horizontalen Position im Diagramm. Es wird festgelegt, wann ein Objekt erstellt wird und wann Nachrichten zu welchem Objekt gesendet werden.

Kommunikation zwischen verschiedenen Komponenten(Programme, Menschen)

Teilnehmer

Zitat Client Zitat Server

Zeit →

← als Antwort nur eine gestrichtelte Linie

Zitat ausgabe

TCP/IP UDP/IP

→ (Pfeil gefüllt) synchrone Kommunikation (ich schicke ein Paket hin und warte auf eine Antwort)

→ (Pfeil ungefüllt) asynchrone Kommunikation

Zunächst einen Saandarddiagramm, was brauche ich

Je nach Situation, was man vermitteln möchte

Als beispiel ein Bankautomat

Um eine Verbindung aufzubauen, wird eine Linie gezogen

Karte eingeben

Pin-abfrage

PinEingabe

AktionAbfrage

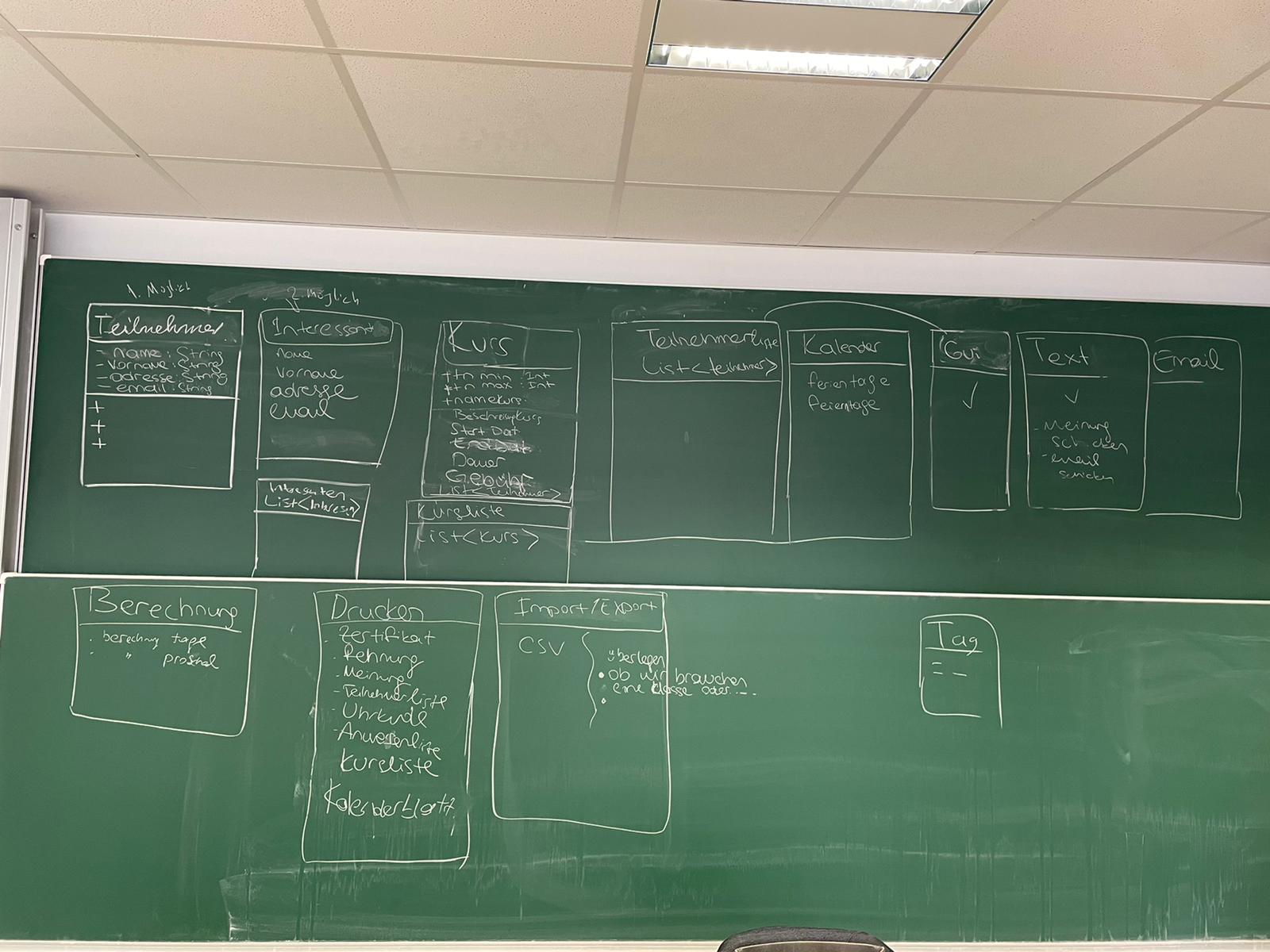
Alt für Alternative

Par für parallel

Ref für Referenz auf ein anderes Diagramm

+→+ als Punkt, es kommt von irgendwoher

uml sequenztcp



GUI besteht aus View, Controller und Application

Was haben die Pfeile zu bedeuten?

Kurs hat eine Teilnehmerliste Kurs → Teilnehmerliste

Eine Kursliste ist genau einem Kurs zugeordnet Kurs ● Kursliste

Kurs benutzt FTage Kurs → FTage

Invarianz

DDD: Domain driven design

Benutzerschnittstelle 0X userexperience

UI userInterface Produkt

Computer zur Eingabe wie Tastatur, Maus, Sprache, Kopfhörer, Touch, Kamera mit Gesten, Login, Datenerfassung, Joyst

ick, Braile

Zur Ausgabe wie Monitor, LCD, Buttons, Serialprint bzw. Serialread

Graphisch, textuell, audiovisuell, auditorisch wie mit GUI

**responsive Design** wie Mobil, Uhr, PC

textuell wie CLI wie (Commando Line Interface ) VBA für Microsoft, LUA als freie Sprache, Skriptsprachen

Menüsteuerung

Makenrelevant

Graphical User Interface Design, auch als Studium Digitale Medien

Mockup

Scene Builder oder VisualStudio

Marvelapp.com für Apps

MockFlow

UserCenticDesignProcess

Interview – Fragebogen

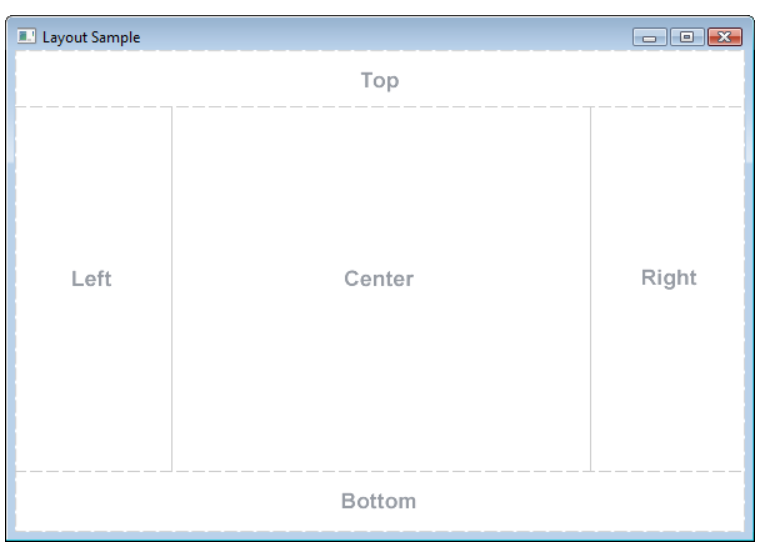
User stories, Mockups, WireFrames

usabilityTest: Wo ist die Aufmerksamkeit des Users => Dark Patterns, die Kunden sollen machen, was wir wollen

SoftwareDesign

SoftwareImplementierung

Container (Sachen wo was reinkommt)

* Pane,
* VBox, übereinander
* Hbox, nebeneinander
* Grid, wie ein 3 \* 5 großes Fenster
* BorderPane, wie oben, unter, links, rechts, Mitte
* 
* FlowPane (kommen hintereinander rein)
* tab wie eine Karteikarte (Hauptfenster oder andere Fenster auswählen)

titel

MenüBar

toolbar angeordnet wie eine HBar

…

StatusBar

Daneben noch DialogFenster

modal / nicht modal (Hilfsfenster, kann geschlossen werden)

Controller

Button

RadioButton

CheckBox

Date Picker

ChoiceBox / DropDown

Label

ListView

TableView (Array mit Slider), Slider mit Option, ob eingeblendet oder nicht

TextField oder TextArray (evtl. RichTextArray)

C:\Users\Akademie\AppData\Local\SceneBuilder

@FXML  
protected void onMouseClick(MouseEvent mouseEvent) {  
 System.*out*.println("Clicked!");  
}

# How to apply MVC in JavaFX

Zusammenfassung

es gibt 2 Fenster

von jedem View gibt es eine

es soll im View automatisch aktualisiert werden

Controller aufsetzen, Guiteile im Controller, mit dem Model verbinden

Statt dem StandardDatentyp wie String gibt es ejetzt ein StringProperty

es gibt Methoden dposit und withdraq

Controller mit Model und View und Model verbinden mit der Methode intiialize()

SimpleGUIDemo

Public void initialize()

cölumnStatus.setCellValueFactory(ne ProprtyValueFactor<User, String> (s: status))

cölumnStauts.setCell

ChoiceBox zur Auswahl, aber mit Häckchen

Combox zur Auswhal, aber ohne Häckchen

onTextFieldAction(), auslesen und hinzufügen

listView

javaFx

level0 Stage stage.show() jede stage hat aktuell eine scene

level1 scene

level2 wie layout, containern oder pane

stage

scene

pane als view bekannt in Fxml

VBox

HBox HBox

Label Textfield Label Textfield

Fxml include Kunde

Fxml include Kurs

Fxml main include Fxml Kunde

xml

<fxml > start

…

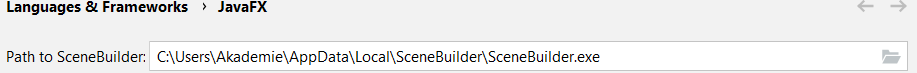
<fxml > ende

oder als Kurzform: <H/>

Kommentare:

<!-- 🡪

scenebuilder



Listen laden und speichern:

xmlns = „url“ 18

xmlns:fx = „url“ 19

xmlns = „url“ 17

wml.namespace

<include

<v

Controller: iniitalize()

onAction fxml

on Mouse

on Drag

.add

Lambda-Funktionen

addListener,Klasse erzeuge, neue Klasse

addListener(Obs, oldval, newval)

Items in die Liste hinzufügen

item\_list.getItems().addAll(meineListe) ->

item\_list.getItems().addAll(lit.of(„Anne“, „Berta“)

alle ausgewählten Items löschen

itemList.getItems().removeAll(itemList.getSelectionModel().getselectedItems());

ItemList.getSeletionModel():setSelectionMode(Selction

Te

Laden und Speichern

In der Main das Model laden und nach dem launch save

EinkaufslistenModel.model.load(filepath: „liste.txt“);

launch();

EinkaufslistenModel.model.load(filepath: „liste.txt“);

Sachen vom Model mit bind verbinden, wenn das eine sich ändert, dann ändert sich das andere

alle Klassen ins Modell, auch speichern

sout(„Init“)

itemList.itemsProperty().bind(EinkaufsListenModel.model.einkaufsliste);

actionbutton.setText(„Action“);

actionButton.setDisable(true),

Liste in eine Datei rausschreiben:

textdateien in Java schreiben:

PrintWriter out = new PrinterWrite(filepath));

for (var item: einkaufsliste.toArray()){

out.println((String) item)

}

out.close();

für Load:

public void load (String filepath){

try{

Scanner in = new Scanner(new FileReader(filepath));

while (in.hasNext()){

String item = in.nextline();

einkaufliste.add(item)

}

in.close();

public void start(Stage stage

**Versionskontrolle**, Repositories GIT oder GitHUB oder BitBucket

(main oder master)

branch main branch als Hauptzweig mit Abzweigungen zum Testen (zum Entfernen oder zum Übernehmen)

add 1.staging, Sachen hinufügne, bereitet vor dem commit

push 3.alle ÄNderunge mit dem Server synchronisieren

update aktuelle Sachen vom Server werden mit meinen lokalen Sachen synchronisiert,

commit 2. aktuelle Version speichern lokal

merge Konflilte zusammenführen

.class-Dateien und Binärdateien werden nicht auf den Server geschickt

Ignore, wenn Dateien nicht gepusht werden sollen

diif bei zwei Versionen, zum Vergleich