Prüfungsrelevant

- Stichpunktliste
- Grundlagen
- Projektmanagement
- Klassendiagramm und OOP
- Pseudocode, PAP, Struktogramm
- Datenbanken

Stichpunktliste

Prüfungsthemen

Grundlagen

- Hexadezimal, Dezimal etc. Umrechnen
- Daten und Information Unterschied
- Unicode, ASCII
- Grundlegenden Datentypen

Projektmanagement

- Projektmanagement (Funktional, Nicht-Funktional, Pflichtenheft/Lastenheft)
- Softwarequalitätskriterien
- Wasserfallmodell, Scrum etc.
- Aufgaben Projektleiter
- Welche Risiken können auftreten in PM
- Informationssicherheit
- Testen (Blackbox, Whitebox)
- Qualitätskriterien der Daten

Diagramme und Schemas

- Pseudocode
- Klassendiagramm
- Aktivitätsdiagramm
- Zustandsdiagramm
- Relationales Datenmodell, ER Modell

Programmierung

- Datenmengen berechnen + Einheiten kennen
- Algorithmen
- CSV, JSON
- Soap, Rest inklusive Fehlermeldung
- Datenbanken nennen, Vor und Nachteile
- SQL
- Objektprogrammierung Grundlagen
- Singleton, Factory Pattern etc.
- Hashing, Entschlüsseln unterschied

- Rekursion
- Clean Code
- UI/UX
- Wireframe, Mockup, Skizze unterschied
- HTML Grundlagen
- Excel

Weiteres

- Grundlagen Netzwerktechnik
- Hardware

Grundlagen

Dezimalzahlen und Hexadezimal zahlen umwandeln

Binärzahl

1	1	1	1	1	1	0	0	1
265	128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1

Hexadezimal

Dezimal-System	Binär-System	Hexadezimal-System
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Beispiel FF50

Stelle	Wert	Potenz (16^x)	Berechnung
F	15	16^3 = 4096	15*4096 = 61440
F	15	16^2 = 256	15*256 = 3840
5	5	16^1 = 16	5*16 = 80
0	0	16^0 = 0	0*16 = 0
		Summe	65360

Daten und Informationen

Unterschied zwischen Daten und Informationen

Begriff	Beschreibung
Daten	Rohwerte, noch ohne Bedeutung - z.B. 43, true, 2025-04-13
Informationen	Bedeutung, die aus den Daten entsteht, wenn sie in Kontext gesetzt werden.

Daten werden zu Information durch:

- Kontext: Woher kommen die Daten? Was bedeuten sie?
- Struktur: Formatierung, Datentypen
- Verarbeitung: Algorithmen, Analysen, Interpretation

Arten von Daten

- Strukturierte Daten: Tabellen, Datenbanken
- Unstrukturierte Daten: Texte, Bilder, Videos
- Halbstrukturtiere Daten: XML, JSON

Datenqualität

- Richtigkeit
- Vollständigkeit
- Aktualität
- Konsistenz

ASCII und Unicode

ASCII

- 7 Bit Zeichencodierung (128 Zeichen)

- Standard in frühen Computern, Betriebssystemen und Netzwerken
- Ursprünglich für Englische Texte Entwickelt

Dezimal	Zeichen	Bedeutung
32	(Space)	Leerzeichen
48-57	0-9	Ziffern
65-90	A-Z	Großbuchstaben
97-122	a-z	Kleinbuchstaben
10	\n	Zeilenumbruch
13	\r	Wagenrücklauf

Begrenzungen:

- Kein Support für Umlaute, Sonderzeichen anderer Sprachen, Emojis etc.
- Nur für US-Englisch praktisch verwendbar

Unicode

- Internationaler Standard zur Darstellung aller Schriftzeichen Weltweit
- Ziel: jeder Buchstabe, jedes Emoji, jedes Symbol bekommt eine eindeutige Nummer (Codepoint)
- Beispiel: U+00E4 = "ä"

Unicode hat mehrere verschiedene Kodierungen (Prinzip wie man Unicode-Zeichen als Bytes speichert)

UTF-8	Am verbreitetsten, variable Länge (1-4 Bytes), kompatibel mit ASCII	Effizient, ASCII Kompatibel
UTF-16	2 oder 4 Byte pro Zeichen	Schneller Zugriff, nicht ASCII Kompatibel
UTF-32	4 Byte pro Zeichen, fixe Länge	Selten, eher für Spezialfälle

- Unterstützt alle Sprachen: z.B. Deutsch (ä, ö, ü), Arabisch, Chinesisch
- Emojis, mathematische Symbole, technische Zeichen etc.
- Standard für Web, APIs, Datenbanken und Dateien

Datentypen

- Integer -> Ganzzahlen
- float -> weniger genau (4 Byte)
- double -> höhere Genauigkeit (8 Byte)
- string -> Zeichenkette

- char -> einzelnes Zeichen
- bool -> Wahrheitswert

Softwarequalitätskriterien

Softwarequalitätskriterien sind Merkmale, die beschreiben wie gut eine Software ihre Aufgaben erfüllt - technisch, funktional und nutzerbezogen.

1. Funktionale Qualität (Erfüllt die Software was sie soll)
   - Funktionalität: Sind alle Anforderungen korrekt umgesetzt?
   - Sicherheit: Ist die Software vor unbefugtem Zugriff geschützt?
   - Zuverlässigkeit: Läuft die Software stabil und korrekt?
2. Nicht funktionale Qualität (Wie gut macht die Software das?)
   - Effizienz: Ist sie schnell und Ressourcenschonend
   - Wartbarkeit: Lässt sich der Code leicht ändern und erweitern?
   - Portabilität: Wie kann man sie leicht auf anderen Systemen ausführen?
   - Testbarkeit: Wie einfach ist automatisiertes oder manuelles Testen?
   - Benutzbarkeit (Usability): Ist die Software einfach und angenehm zu bedienen?
   - Wiederverwendbarkeit: Kann man Teile in anderen Projekten wiederverwenden?

ISO/IEC 25010

Kriterium	Beschreibung
Funktionale Eignung	Erfüllt die Software die spezifizierten Aufgaben richtig?
Zuverlässigkeit	Verfügbarkeit, Fehlertoleranz, Wiederherstellbarkeit
Benutzbarkeit	Verständlichkeit, Bedienbarkeit, Benutzerzufriedenheit
Effizienz	Antwortzeiten, Ressourcenverbrauch
Wartbarkeit	Modifizierbarkeit, Analysierbarkeit, Stabilität bei Änderungen
Übertragbarkeit	Anpassbarkeit an neue Umgebungen, Installierbarkeit
Kompatibilität	Interoperabilität mit anderer Software, Co-Existenz

Beispiel für Banking-App

Qualitätskriterium	Bedeutung für die App
Sicherheit	Keine unberechtigten Überweisungen möglich
Zuverlässigkeit	Kein Absturz bei Last
Usability	Nutzer versteht die App sofort
Wartbarkeit	Neue Funktionalitäten (z.B. Apple Pay) leicht einbaubar
Effizienz	Schnelle Ladezeiten, geringer Akkuverbrauch

Aufgaben

Für jedes Softwarepaket wird ein Pflichtenheft erstellt. Die darin enthaltenen Anforderungsbeschreibungen müssen den folgenden drei Kriterien genügen:

a.) Vollständigkeit, b.) Eindeutigkeit und c.) Testbarkeit

Erläutern Sie diese drei Begriffe.

1. Eine Anforderung ist vollständig, wenn sie alle notwendigen Informationen enthält, damit Entwickler, Tester und andere Beteiligte genau wissen, was umgesetzt werden soll. Es dürfen keine wichtigen Details fehlen. Beispiel: Wenn eine Funktion beschrieben wird, muss auch angegeben sein, welche Eingaben möglich sind, was genau das Ergebnis ist und was bei Fehlern passieren soll.
2. Eine Anforderung ist eindeutig, wenn sie nur eine einzige Interpretation zulässt. Mehrdeutige oder unklarer Formulierungen wie "schnell", "benutzerfreundlich" oder "in der Regel" sind zu vermeiden. Beispiel: Statt "Das System soll schnell starten" sollte stehen: "Das System soll innerhalb von 5 Sekunden nach dem Einschalten betriebsbereit sein"
3. Eine Anforderung ist testbar, wenn man objektiv überprüfen kann, ob sie erfüllt wurde. Dafür müssen klare messbare Kriterien vorliegen. Beispiel: Eine testbare Anforderung könnte lauten: "Das System soll 1000 Anfragen pro Minute ohne Fehlermeldung verarbeiten können".

Welche Konsequenzen drohen bei Verletzung dieser Kriterien?

Fehlerhafte Anforderungen führen fast immer zu höheren Kosten, längerer Projektlaufzeit, schlechter Qualität und unzufriedene Kunden.

Beurteilen Sie folgende Anforderung im Licht der drei Kriterien:

Die Länge des Güterzugs beträgt bei Einsatz einer einzelnen Lokomotive maximal 30 Waggons. Aus Ökonomischen Gründen darf er aber auch nicht zu Kurz sein.

a => Es fehlt ganz klar an Informationen, was ist "zu Kurz"

b => Es ist nicht eindeutig ob sich diese Anforderungen auf die Anzahl der Waggons oder die Länge des Zuges selber beziehen

c => Durch fehlenden Kontext und fehlende Information ist diese Anforderung nur bedingt testbar.

Die zu erstellenden Softwarelösungen sollen effektiv und effizient sein

Grenzen Sie diese Begriffe voneinander ab.

Effektivität beschreibt Ob das Ziel/die Anforderung überhaupt erreicht wird.

Effizienz beschreibt wie gut in dem Falle die Software die Anforderungen ausführt. Effizienz beschreibt dagegen wie

Nennen Sie zwei exemplarische Möglichkeiten, ein Programm effizienter zu gestalten

- Optimierung von Algorithmen => Optimierung der Rechenzeit bei großen Datenmengen
- Reduzierung des Speicherverbrauches => Optimierung der Datenstruktur je nach Größe der Datenmengen

Bei der Verwaltung des Zugaufkommens fallen große Datenmengen an. Ihr Team diskutiert darüber, ob zur Sortierung dieser Daten ein quadratisches oder ein logarithmisches Verfahren vorzuziehen sei. Nennen Sie jeweils einen Vorteil und einen Nachteil quadratischer und logarithmischer Sortierverfahren.

Quadratische Sortierverfahren (z.B. Bubble Sort, Selection Sort):

Vorteil:

- Einfachheit und Implementierung: Quadratische Algorithmen sind einfach zu verstehen und schnell zu Implementieren, da sie keine komplexen Datenstrukturen oder rekursive Aufrufe erfordern. Dies macht sie besonders für kleinere oder weniger komplexe Anwendungen nützlich.

Nachteil

- Schlechte Skalierbarkeit: Quadratische Algorithmen haben eine Zeitkomplexität von $O(n^2)$, was bedeutet, dass die Laufzeit mit zunehmender Datenmenge stark ansteigt. Bei großen Datensätzen wird die Ausführung extrem langsam, was sie bei großen Datenmengen ineffizient macht.

Logarithmische Sortierverfahren (z.B. MergeSort, QuickSort)

Vorteil:

- Hervorragende Skalierbarkeit: Logarithmische Sortierverfahren wie MergeSort und QuickSort haben eine Zeitkomplexität von $O(n \log n)$. Das bedeutet, dass die Laufzeit mit wachsender Datenmenge viel langsamer zunimmt als bei quadratischen Algorithmen. Sie sind daher sehr gut geeignet für Verarbeitung großer Datenmengen.

Nachteil

- Komplexität der Implementierung: Diese Algorithmen sind komplexer zu implementieren und erfordern oft zusätzliche Speicherressourcen (z.B. bei MergeSort). Außerdem können sie in spezifischen Fällen (z.B. bei schlecht verteilten Daten) durch ungünstige Rekursionstiefen langsamer werden.

Um die Codeerstellung im Team zu vereinheitlichen, gelten in Ihrem unternehmen Codierungsrichtlinien.

Für die Schreibweise von Variablennamen gilt die Camel-Case-Notation. Erläutern Sie diesen Begriff anhand eines Beispiels.

CamelCase beschreibt das verwenden von Großbuchstaben für jedes Wort innerhalb einer Variable.

Nennen Sie vier weitere Beispiele für solche Gestaltungsrichtlinien

- Einheitliche Kommentierung und Dokumentation wie z.B. Methoden oder Inline-Kommentare
- Verwendung von Konstanten statt "Magic Numbers". Anstatt feste Werte dierekt im Code zu verwenden, sollen diese in benannten Konstanten gespeichert werden.
- Fehlerbehandlung und Logging. Fehler müssen immer korrekt abgefangen und behandelt werden, und es müssen aussagekräftige Logs erstellt werden, um die Ursachen von Problemen schnell nachvollziehen zu können.
- Vermeidung von Code-Duplizierung, der Code muss so strukturiert sein, dass redundante oder doppelte Logik vermieden wird

Projektmanagement

> Ein Projekt ist eine **einmalige**, **zeitlich befristete** Aufgabe, die darauf abzielt, ein **spezifisches Ziel** zu erreichen. Es unterscheidet sich von routinemäßigen Arbeiten oder Prozessen da es klar definierte **Anfangs- und Endpunkte** sowie spezielle Ziele und eine gewisse **Komplexität** hat. Projekte sind durch eine Abfolge von Handlungsschritten definiert, die auf das entsprechende Ziel ausgerichtet sind. Projekte können in verschiedenen Bereichen vorkommen wie Bau, Forschung, Marketing u.s.w.

Einschaften eines Projekts

1. **Einmaligkeit**: ein Projekt ist in seiner Art einzigartig. Es kann Ähnlichkeiten mit anderen Projekten haben, aber es gibt immer spezifische Anforderungen oder Bedingungen, die es von anderen Projekten unterscheiden.
2. **Zeitliche Begrenzung**: Ein Projekt hat einen definierten Anfang und ein definiertes Ende. Es ist also eine zeitlich begrenzte Aktivität, im Gegensatz zu kontinuierlichen Prozessen. Es endet, sobald das Ziel erreicht wurde oder das Projekt abgebrochen wird.
3. **Zielorientierung**: Jedes Projekt hat ein klares Ziel, dass durch bestimmte Ergebnisse, Produkte oder Dienstleistungen definiert wird. Das Ziel ist spezifisch und Messbar.
4. **Komplexität**: Projekte sind in der Regel komplex, da sie verschiedene Aktivitäten/Handlungsschritten, Ressourcen und Personen umfassen. Sie erfordern oft eine koordinierte Zusammenarbeit verschiedener Abteilungen und Experten.

4-Phasen-Modell

1. **Initiierung**: in dieser Phase wird das Projekt definiert und gestartet. Der Projektleiter erarbeitet eine erste Projektidee, und das Ziel des Projekts wird klar definiert. Die Machbarkeit wird geprüft, oft durch eine Machbarkeitsstudie, und die wichtigsten Stakeholder werden identifiziert. Hier wird auch ein Projektauftrag erstellt und genehmigt, der die grundlegenden Projektziele und Anforderungen festhält.
2. **Planung**: In der Planungsphase erfolgt eine detaillierte Organisation und Vorbereitung. Der Projektzeitplan mit allen Aufgaben, Meilensteinen und Deadlines wird erstellt. Zudem werden benötigte Ressourcen wie Personal, Material und Budget kalkuliert und zugeordnet. Ein zentrales Element ist auch die Risikoanalyse, in der mögliche Risiken und deren Bewältigungsstrategien festgelegt werden. Für eine reibungslose Kommunikation während des Projekts werden klare Kommunikationswege festgelegt.
3. **Durchführung**: in der Durchführung wird der Projektplan umgesetzt. Dies umfasst die Aufgabenverteilung an das Team, die Überwachung des Projektfortschritts und die Anpassung des Projektplans bei Bedarf. Qualitätsmanagement stellt sicher, dass die

Ergebnisse den geforderten Standards entsprechen. Regelmäßige Meetings und Berichte halten das Team auf dem Laufenden und ermöglichen es, frühzeitig auf Herausforderungen zu reagieren.

4. **Abschluss**: zum Ende des Projekts werden die Ergebnisse final geprüft und formell abgenommen. Ein Abschlussbericht fasst den gesamten Projektverlauf zusammen und dokumentiert wichtige Erkenntnisse und Verbesserungspotentiale. Zudem erfolgt eine Nachkalkulation, und die Projektergebnisse werden an den Auftraggeber übergeben. Auch eine Abschlussbesprechung wird abgehalten, um das Projektteam und die Stakeholder über die Projektergebnisse informieren.

Projektinitiierung

> Hauptziel ist die Identifikation aller Projektziele und Anforderungen. Auf ihnen basierend soll eine grobe Entscheidung getroffen werden, ob das Projekt realisierbar und sinnvoll ist. Es werden SMARTe Projektziele definiert und der Grundstein für das Projekt gelegt, indem eine Vision und Richtung für alle Beteiligten geschaffen ist.

Aktivitäten

- **Projektidee und -Zielsetzung:** Entwickeln einer klaren Projektidee und Definition der Projektziele. Hier werden oft SMART-Kriterien
- **Anforderungen des Kunden** ob funktional oder nicht-funktional sind zu identifizieren und zu schärfen.
- **Machbarkeitsanalyse:** Untersuchung der technischen, rechtlichen und wirtschaftlichen Machbarkeit, um die Realisierbarkeit des Projekts zu sichern.
- **Stakeholderanalyse:** Identifikation der wichtigsten Stakeholder, also Personen oder Gruppen, die direkt oder indirekt betroffen sind oder Interesse am Projekt haben.
- **Projektauftrag und Genehmigung:** Erstellung eines Projektauftrags der die Ziele, den Umfang, Ressourcen und den genehmigten Zeitrahmen festhält und die Grundlage für die nächste Phase bildet.

Notationsstandards

Lastenheft

Das **Lastenheft** ist ein zentrales Dokument im Rahmen der Projektplanung und -Durchführung, das die Anforderungen und Erwartungen an ein Projekt aus Sicht des Auftraggebers beschreibt. Es dient als Grundlage für die Kommunikation zwischen Auftraggeber und Auftragnehmer und legt die Ziele, Funktionen und Rahmenbedingungen fest, die erfüllt werden müssen.

In einem Lastenheft werden nicht nur die **spezifischen Anforderungen** an das Produkt oder die Dienstleistung detailliert dargestellt, sondern auch der **Kontext,** in dem das Projekt durchgeführt

wird. Dazu gehören beispielsweise der Einsatzbereich, Zielgruppen und relevante Rahmenbedingungen. Durch die klare Definition der Erwartungen wird sichergestellt, das alle Beteiligten ein gemeinsames Verständnis der Projektziele haben.

Ein gut strukturiertes Lastenheft fördert die Effizienz im Projektverlauf, reduziert Missverständnisse und trägt entscheidend zur Qualität des Endprodukts bei. Es ist daher essenziell, das Lastenheft sorgfältig zu erstellen und regelmäßig zu überprüfen, um den sich möglicherweise ändernden Anforderungen Rechnung zu Tragen. In der Folge wird das Lastenheft oft durch das Pflichtenheft ergänzt, das die spezifischen Umsetzungsschritte und technischen Details aus Sicht des Auftragnehmers festhält.

Projektorganisationsmatrix

Eine **Projektorganisationsmatrix** ist ein Werkzeug im Projektmanagement, das die Verantwortlichkeiten und Rollen von Teammitgliedern für verschiedene Aufgaben in einem Projekt übersichtlich darstellt. Die hilft, Zuständigkeiten klar zu definieren und sicherzustellen, dass jeder weiß, was von ihm oder ihr erwartet wird.

Die Matrix ist in der Regel als Tabelle aufgebaut, wobei:

- Die Spalten die Teammitglieder oder Rollen (z.B. Projektleiter, Entwickler, Marketing, Finanzen) darstellen.
- Die Zeilen die einzelnen Projektaufgabe oder Arbeitspakete realisieren.

In den Zellen der Matrix wird dann für jede Aufgabe und jede Rolle festgelegt, welche Art von Verantwortlichkeit besteht. Ein weit verbreitetes Modell ist das RACI-Modell:

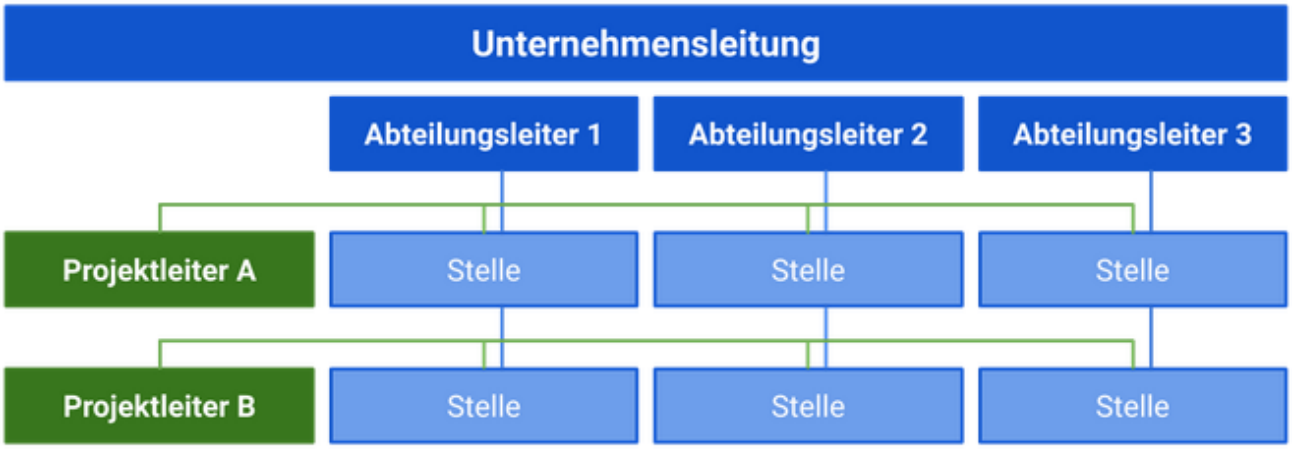
1. Responsible: Die Person, die die Aufgabe ausführt.
2. Accountable: Die Person, die letztendlich für das Ergebnis verantwortlich ist. Jede Aufgabe sollte nur eine "Accountable"-Person haben.
3. Consulted: Personen, die als Experten oder zur Beratung herangezogen werden.
4. Informed: Personen, die über den Fortschritt oder das Ergebnis informiert werden müssen.

Vorteile:

- Klarheit: jeder weiß, wer für was verantwortlich ist, was Missverständnisse und Konflikte verringert.
- Effizienz: Durch die klare Aufteilung der Rollen wird die Zusammenarbeit effektiver.
- Verantwortungsbewusstsein: Teammitglieder können sich klar auf ihre Zuständigkeiten fokussieren und Verantwortung übernehmen

Nachteile:

- Komplexität bei großen Projekten: Bei sehr großen Projekten mit vielen Aufgaben und Teammitgliedern kann die Matrix schnell unübersichtlich werden.
- Flexibilität: Wenn sich Aufgaben oder Rollen ändern, muss muss die Matrix regelmäßig aktualisiert werden.



Use-Case-Diagramm

Ein Use-Case-Diagramm in der Unified Modeling Language beschreibt die funktionalen Anforderungen an ein System aus Sicht der Benutzer. Es zeigt, wie verschiedene Akteure (Benutzer oder andere Systeme) mit dem System interagieren und welche Ziele sie dabei erreichen möchten.

1. Akteure

Ein Akteur ist eine Entität, die mit dem System interagiert, um ein bestimmtes Ziel zu erreichen. Akteure können reale Personen, andere Systeme oder sogar externe Organisationen sein. Sie werden in Use-Case-Diagrammen durch eine Strichfigur symbolisiert und außerhalb des Systems dargestellt. Es gibt verschiedene Akteurtypen:

- Primäre Akteure: Diese haben das Hauptziel, eine bestimmte Funktion des Systems zu nutzen.
- Sekundäre Akteure: Unterstützen den Hauptprozess, haben jedoch nicht das Hauptziel, die Funktion des Systems zu nutzen

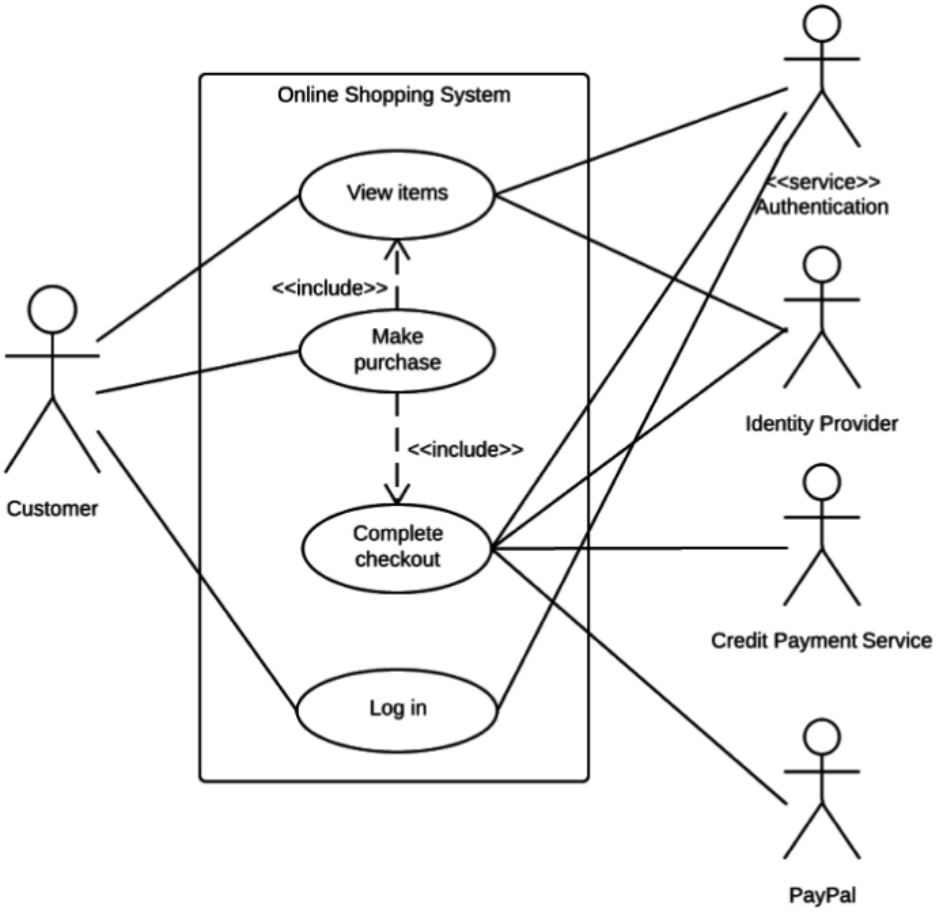
2. Anwendungsfälle (Use Cases)

Ein Anwendungsfall beschreibt eine spezifische Interaktion oder Funktion, die das System für einen Akteur bereitstellt. In einem Diagramm wird ein Use Case als Ellipse dargestellt und gibt einen klaren Einblick in die Anforderungen des Systems aus der Benutzerperspektive. Ein Anwendungsfall umfasst eine Folge von Schritten, die den Benutzer zu einem bestimmten Ergebnis führen, z.B. "Buchung vornehmen" oder "Passwort ändern".

3. Beziehungen zwischen Anwendungsfällen

- Assoziation: Verknüpft einen Akteur mit einem Anwendungsfall, z.B. eine "Kundin", die eine "Bestellung aufgibt"

- Include: Zeigt an, dass ein Use Case einen anderen Use Case zwingend einschließt, z.B. "Bezahlung abwickeln" innerhalb von "Bestellung aufgeben".
- Extend: Beschreibt optionale oder bedingte Erweiterungen eines Use Cases, z.B. "Zusatzleistungen bucen" nur unter bestimmten Bedingungen in "Hotelreservierung vornehmen".
- Generalisierung: Eine Vererbungshierarchie, in der ein spezialisierter Use Case (z.B. "Schnellbuchung") eine Verallgemeinerung (z.B. "Buchung vornehmen") erweitert.



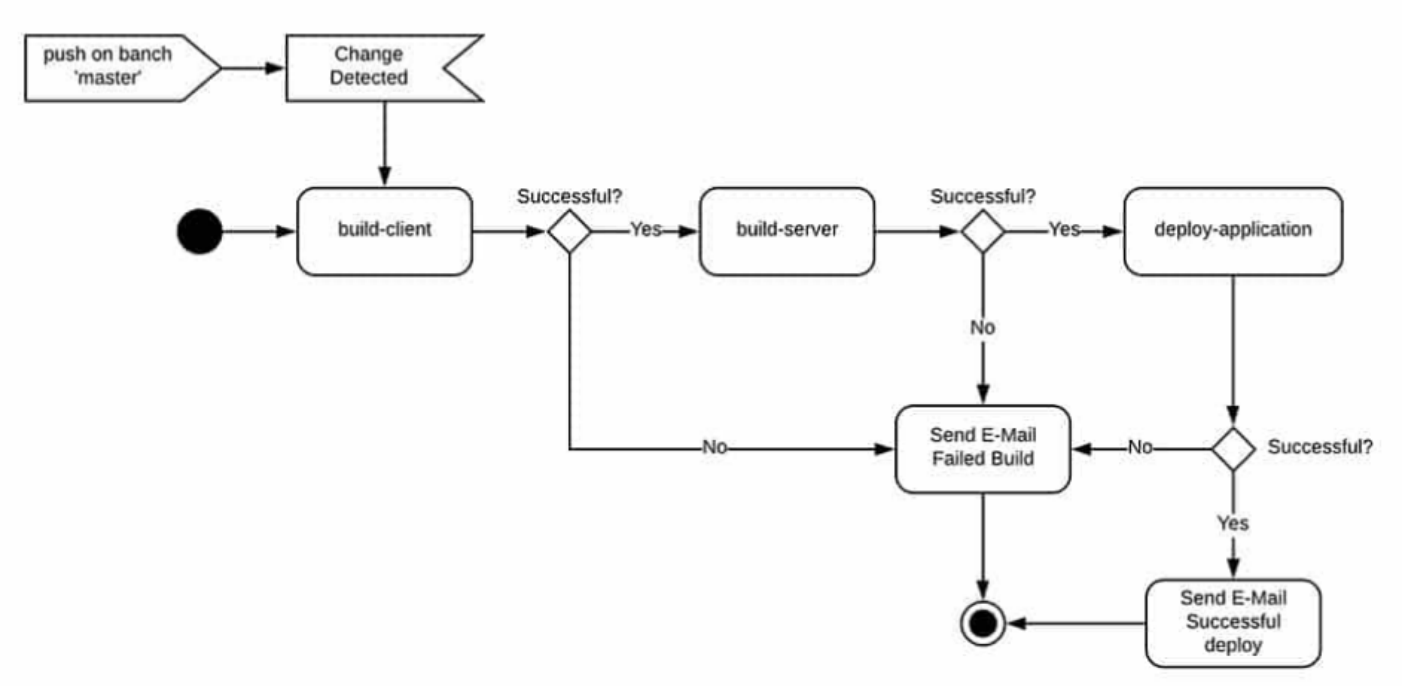
Aktivitätsdiagamm

Ein Aktivitätsdiagramm in UML ist eine grafische Darstellung des dynamischen verhalten eines Systems, die zeigt, wie Prozesse, Abläufe und Workflows innerhalb eines Systems ablaufen. Es beschreibt die logische Reihenfolge von Aktivitäten und dient zur Visualisierung von Prozessen, wie etwa Geschäfts- oder Arbeitsabläufen, und zeigt parallele sowie bedingte Abläufe auf.

1. Aktivitäten:
   - Aktivitäten sind die grundlegenden Schritte oder Aktionen innerhalb eines Prozesses. Sie werden als abgerundete Rechtecke dargestellt und beschreiben einzelne Arbeitsschritte. jede Aktivität entspricht einer spezifischen Aufgabe, wie etwa "Daten abrufen" oder "Bestellung verarbeiten".
2. Kontrollfluss

- Der Kontrollfluss stellt die Reihenfolge zwischen den Aktivitäten dar und wird durch Pfeile dargestellt. Er zeigt, welche Aktivität als nächstes ausgeführt wird und stellt damit den logischen Ablauf des Prozesses sicher.

3. Start und Endknoten:
   - Ein Startknoten (schwarz ausgefüllter Kreis) markiert den Beginn des Prozesses oder der Aktivität.
   - Ein Endknoten (Kreis mit einem schwarzen Punkt in der Mitte) Signalisiert das Ende des Prozesses. In einem Diagramm kann es mehrere Endknoten geben, wenn verschiedene Endpunkte erreicht werden können.
4. Entscheidungsknoten und Verzweigungen:
   - Entscheidungsknoten (dargestellt als Raute) stellen Punkte dar, an denen der Kontrollfluss je nach Bedingung in unterschiedlichen Richtungen verzweigen kann.
   - Nach der Entscheidung folgt der Prozess entweder einem bestimmten Pfad (z.B. "wenn ja dann..") oder einem anderen (z.B. "wenn nein, dann...").
5. Parallelität und Synchronisation:
   - Aktivitätsdiagramme ermöglichen es, parallele Abläufe darzustellen. Diese parallelen Abläufe werden durch einen "Fork-Knoten" (horizontaler oder vertikaler Balken) erzeugt, der den Prozess in mehrere parallele Aktivitäten aufteilt.
   - Der Join-Knoten synchronisiert mehrere parallele Aktivitäten und führt sie wieder in einem einzigen Kontrollfluss zusammen, wenn alle parallelen Aktivitäten abgeschlossen sind.
6. Objektfluss:
   - Ein Objektfluss (oft ein beschrifteter Pfeil) zeigt, dass ein Objekt (z.B. ein Datenwert oder ein Dokument) zwischen Aktivitäten weitergegeben wir. Er hilft dabei, die Datenabhängigkeiten im Ablauf sichtbar zu machen.



User Story

Eine **User Story** ist eine kurze, einfache Beschreibung einer Funktion oder Anforderung aus der Perspektive eines Endbenutzers. Sie wird hauptsächlich in agilen Methoden, wie Scrum oder Kanban verwendet, um die Bedürfnisse und Ziele der Benutzer klar und nachvollziehbar zu dokumentieren. User Stories helfen Entwicklungsteams, die Anforderungen aus Sicht des Endbenutzers zu verstehen und priorisieren.

Eine typische User Story folgt dem Format:

- "Als [Benutzerrolle] möchte ich [Ziel/Wunsch], damit ich [Nutzen/Mehrwert] erreichen kann."
- -> "Als **Kunde** möchte ich meine **Bestellung** verfolgen, damit ich jederzeit über den **Status** informiert bin."

1. **Benutzerrolle:** Stellt dar, wer die Funktion benötigt. Dies könnte ein spezifischer Benutzertyp sein, wie ein "Administrator", ein "Kunde" oder ein "Vertriebspartner". Eine präzise Benutzerrolle hilft dem Team, die Bedürfnisse des Nutzers besser zu verstehen
2. **Ziel/Wunsch:** beschriebt, was der Benutzer mit der Funktion erreichen möchte. Dieser Teil stellt die eigentliche Anforderung dar.
3. **Nutzen/Mehrwert:** Zeigt auf, welchen Vorteil oder Nutzen der Benutzer durch das Erreichen des Ziels erhält. Der Nutzen ist entscheidend, um den Mehrwert der Anforderung zu verdeutlichen.

Eigenschaften guter User Stories (INVEST-Kriterien)

- Independent: Sie sollte unabhängig von anderen Stories sein.
- Negotiable: Sie dient als Diskussionsgrundlage und ist verhandelbar.
- Valuable: Sie sollte dem Nutzer einen erkennbaren Mehrwert bringen.
- Estimable: Sie kann in ihrem Aufwand abgeschätzt werden.
- Small: Sie ist klein und überschaubar, sodass sie in einem Sprint abgeschlossen werden kann.
- Testable: Sie ist testbar, um zu überprüfen ob sie korrekt implementiert wurde.

Projektplanung

Das Ziel der Planungsphase ist es, einen detaillierten Fahrplan für das Projekt zu erstellen, der alle notwendigen Ressourcen, Aufgaben, Verantwortlichkeiten, Risiken und Kommunikationswege umfasst. Diese Phase schafft die Grundlage, um das Projekt systematisch und effizient zu steuern.

- Projektstrukturplan erstellen: Zerlegung des Projekts in kleinere, handhabbare Teilaufgaben oder Arbeitspakete. Dies bietet eine klare Übersicht über die Aufgabenverteilung und dient als Basis für die weitere Planung.
- Entscheidung für ein Vorgehensmodell innerhalb der Durchführung.
- Zeit- und Terminplanung: Erstellung eines Zeitplans, oft in Form eines Gantt-Diagramms oder Netzplans, der alle Meilensteine und Deadlines des Projekts enthält.
- Ressourcenplanung: Bereitstellung und Zuweisung der benötigten Ressourcen, darunter Personal, Materialien und Budget. Die Ressourcen werden auf die verschiedenen

Aufgaben des Projekts abgestimmt. Aus der Verteilung der Ressourcen und der Ablaufplanung des Projektes wird ein Kostenplan erstellt.
- Risikoanalyse und Risikomanagement: Identifikation möglicher Risiken und Entwicklung von Maßnahmen zur Minimierung dieser Risiken. hier wird auch festgelegt, wie auf unvorhergesehene Herausforderungen reagiert werden kann.
- Kommunikationsplanung: Entwicklung des Kommunikationsplans, der die Kommunikationswege und -mittel innerhalb des Teams und mit den Stakeholdern festlegt. Dadurch wird sichergestellt, dass alle Beteiligten regelmäßig informiert sind und Missverständnisse vermieden werden

Pflichtenheft

Das Pflichtenheft ist ein zentrales Dokument im Projektmanagement, das die spezifischen Anforderungen und Lösungen aus Sicht des Auftragnehmers beschreibt. Es basiert auf den Vorgaben des Lastenhefts und konkretisiert, wie die im Lastenheft formulierten Anforderungen technisch und organisatorisch umgesetzt werden sollen.

In einem Pflichtenheft werden detaillierte Informationen zu den Funktionen, Eigenschaften und Schnittstellen des Produkts oder der Dienstleistung festgehalten. Dabei werden nicht nur die gewünschten Ergebnisse definiert, sondern auch die Methoden, Technologien und Ressourcen, die zur Erreichung dieser Ziele eingesetzt werden. Dies umfasst auch Zeitpläne, Kostenrahmen und Qualitätskriterien.

Das Pflichtenheft dient als verbindliche Grundlage für die Projektarbeit und ermöglicht eine klare Kommunikation zwischen Auftragnehmer und Auftraggeber. Durch die Präzise Dokumentation der Anforderungen und Lösungen wird sichergestellt, dass alle Beteiligten ein einheitliches Verständnis der Projektziele haben und Missverständnisse im Verlauf des Projekts minimiert werden.

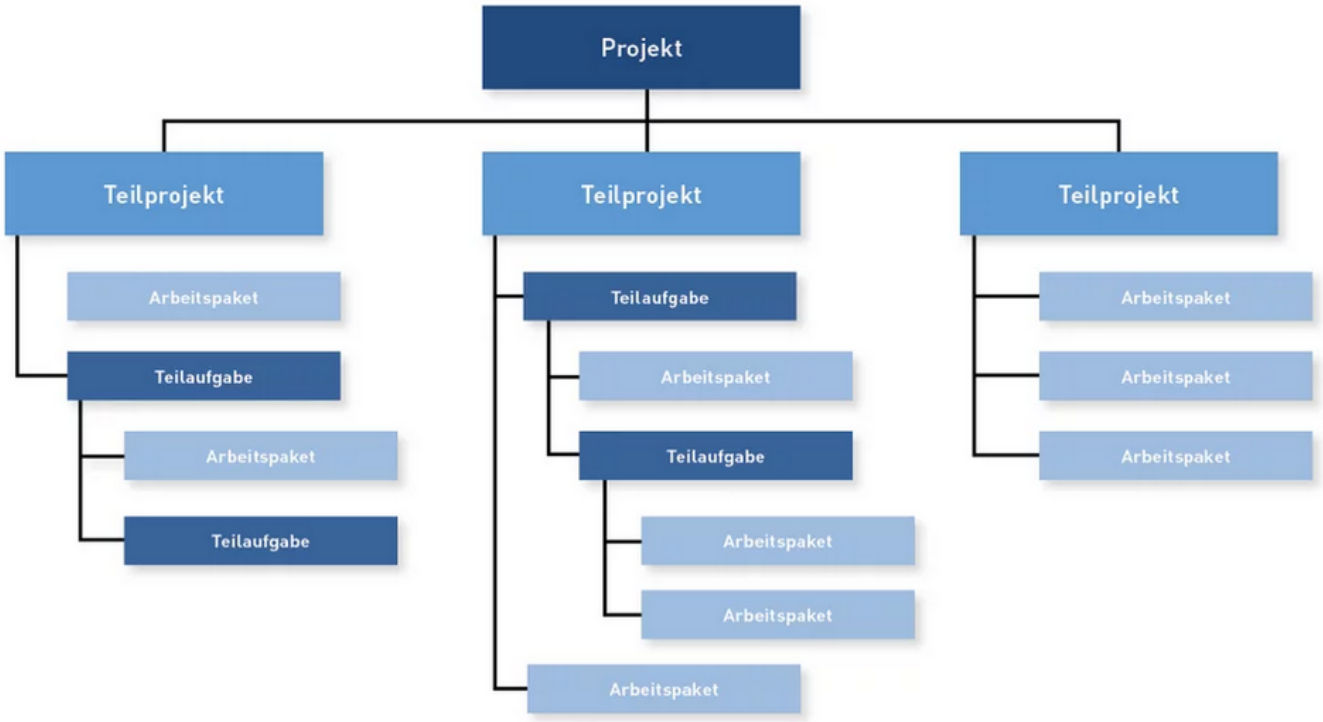
Ein gut ausgearbeitetes Pflichtenheft ist entscheidend für den Erfolg eines Projekts, da es als Referenzpunkt für die Umsetzung, Tests und Abnahmen dient. Es bildet somit die Basis für die Qualitätssicherung und das Risikomanagement und trägt maßgeblich zur Zufriedenheit aller Stakeholder bei. In der dynamischen Projektlandschaft ist es wichtig, das Pflichtenheft regelmäßig zu überprüfen und bei Bedarf anzupassen, um den sich verändernden Anforderungen gerecht zu werden.

Projektstrukturplan

Ein Projektstrukturplan ist ein zentrales Instrument des Projektmanagements und dient dazu, ein Projekt in überschaubare und strukturierte Teilaufgaben oder Arbeitspakete zu gliedern. Durch dieses Strukturierung wird das Projekt klarer und besser planbar, was die Organisation und Kontrolle des Projekts vereinfacht. Der PSP bildet also das gesamte Projekt hierarchisch ab und hilft dabei, die erforderlichen Ressourcen, Verantwortlichkeiten und Abhängigkeiten zu erkennen.

Bestandteile und Aufbau eines Projektstrukturplans

1. **Projektaufteilung** in Ebenen: Der PSP gliedert das Projekt hierarchisch in immer kleinere Einheiten. Die oberste Ebene stellt das Gesamtprojekt dar, die nachfolgenden Ebenen brechen das Projekt weiter in Teilprojekte, Arbeitspakete und Aktivitäten herunter. Jedes Arbeitspaket ist dabei eine eigenständige Einheit und sollte so detailliert wie nötig, aber auch so einfach wie möglich deformiert sein.
2. **Hierarchische Struktur**: Die Struktur ist meistens baumartig, wodurch Abhängigkeiten und die Reihenfolge von Aufgaben sichtbar werden. Das ermöglicht eine einfache Zuordnung und Delegation von Aufgaben an Projektbeteiligte.
3. **Arbeitspakete**: Die kleinsten Einheiten in einem PSP sind die Arbeitspakete. Sie enthalten konkrete Aufgaben, die in sich geschlossen und unabhängig voneinander bearbeitet werden können. Ein Arbeitspaket sollte so definiert sein, dass es eindeutig einem Verantwortlichen zugeordnet und hinsichtlich Zeit und Ressourcen geplant werden kann.
4. **Nummerierung**: Um die Übersichtlichkeit zu verbessern, werden die Elemente im PSP oft nummeriert. So können zum Beispiel Hauptaufgaben mit einer einstelligen Nummer und deren Unteraufgaben mit weiteren Ziffern versehen werden.



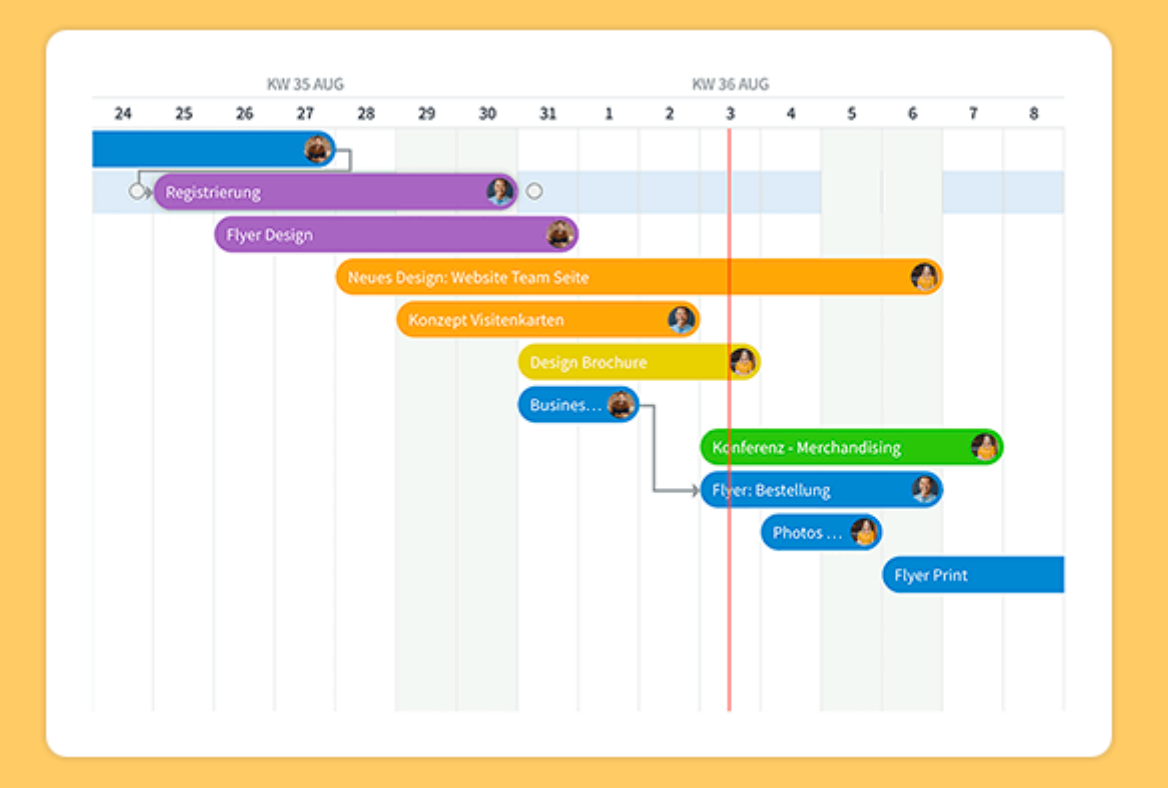
Gantt-Diagramm

Ein Gantt-Diagramm ist eine visuelle Projektmanagement- und Planungshilfe, die häufig verwendet wird, um Aufgaben und deren zeitlichen Verlauf in einem Projekt darzustellen.

Das Gantt-Diagramm ist meist als Balkendiagramm aufgebaut. Dabei zeigt die horizontale Achse den Zeitverlauf (in Tagen, Wochen oder Monaten) und die vertikale Achse listet die einzelnen Aufgaben oder Arbeitsschritte des Projekts auf. Jeder Aufgabe ist ein Balken zugeordnet, der die Dauer der Aufgabe angibt und zeigt, wann sie beginnt und endet. Die Länge des Balkens repräsentiert die geschätzte oder geplante Dauer der jeweiligen Aufgabe.

Wichtige Elemente

- Aufgaben und Arbeitsschritte: Jede Aufgabe wird als separater Balken dargestellt.
- Zeitleiste: Die horizontale Achse zeigt den Zeitraum, in dem das Projekt stattfindet.
- Dauer: Die Länge eines Balkens stellt die Dauer einer Aufgabe dar.
- Abhängigkeiten: Gantt-Diagramme können Abhängigkeiten zwischen Aufgaben darstellen, z.B. wenn eine Aufgabe erst nach Abschluss einer anderen beginnen kann.
- Meilensteine: Besondere Ereignisse oder wichtige Zeitpunkte werden oft als vertikale Linien oder Symbole hervorgehoben.



Netzplan

Ein Netzplan ist eine grafische Darstellung eines Projekts, die alle Aktivitäten und zeitlichen Abhängigkeiten Zeigt. Er wird häufig im Projektmanagement eingesetzt, insbesondere bei der Methode der Netzplantechnik. Netzpläne ermöglichen eine detaillierte Analyse und Planung komplexer Projekte und werden oft bei Projekten mit vielen Abhängigkeiten und parallelen Prozessen verwendet.

Der Netzplan besteht aus Knoten und Kanten:

- Knoten repräsentieren die einzelnen Aufgaben oder Ereignisse.
- Kanten (Linien oder Pfeile) zeigen die Abhängigkeiten und Reihenfolge der Aufgaben an.

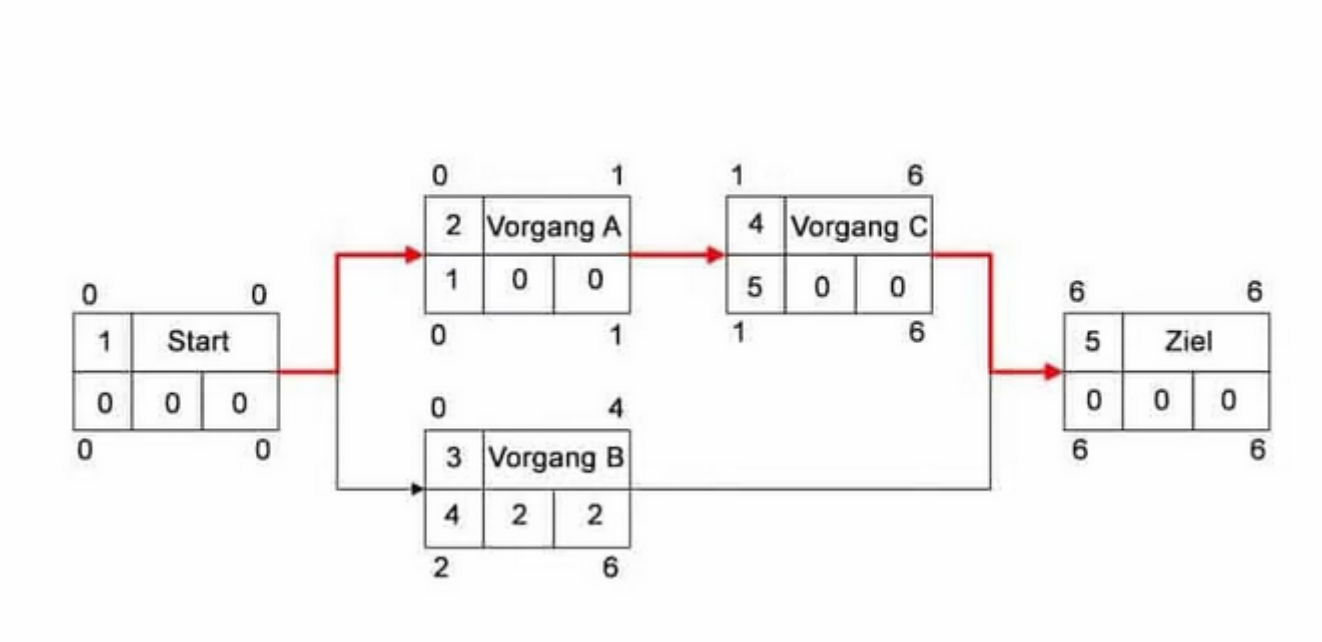
Jede Aktivität wird durch einen Knoten dargestellt und hab typischerweise Dauer, während die Abhängigkeiten anzeigen, welche Aufgaben zuerst abgeschlossen sein müssen, bevor eine neue beginnen kann.

Bestandteile eines Netzplans

1. Vorgänge: Die einzelnen Aktivitäten oder Aufgaben, die im Projekt abgeschlossen werden müssen.
2. Abhängigkeiten: Die logischen Verbindungen zwischen den Vorgängen, die abgeben, welche Vorgänge zuerst abgeschlossen sein müssen.
3. Dauer: Zeitdauer für jeden Vorgang.
4. Zeitplan: Die Berechnung des frühesten und spätesten möglichen Anfangs- und Endzeitpunkts (Vorwärts- und Rückwärtsrechnung)
5. Kritischer Pfad: Die längste Pfad durch das Netz, der die kürzeste mögliche Dauer des Projekts bestimmt. Vorgänge auf dem kritischen Pfad haben Puffer, d.h., Verzögerungen bei diesen Vorgängen verzögern das gesamte Projekt.

Wichtige Begriffe

- Frühester Anfang (FA) und frühestes Ende (FE): Die frühesten Zeitpunkte, zu denen ein Vorgang beginnen bzw. enden kann.
- Spätester Anfang (SA) und spätestens Ende (SE): Die spätesten Zeitpunkte, zu denen ein Vorgang starten bzw. Enden darf, ohne das Projekt zu verzögern.
- Pufferzeiten: Die Zeit, um die ein Vorgang verzögert werden kann ohne das Gesamtprojekt zu verzögern.
- Pufferzeiten: Die Zeit, um die ein Vorgang verzögert werden kann, ohne das Gesamtprojekt zu verzögern. Vorgänge auf dem kritischen Pfad haben keinen Puffer.



Roadmap

Eine Roadmap ist ein strategisches Planungsinstrument, das einen klaren Überblick über die Ziele, Meilensteine und geplanten Aktivitäten eines Projekts oder einer Produktentwicklung über einen bestimmten Zeitraum bietet. Sie ist sowohl für interne Teams als auch für externe Stakeholder von Bedeutung, um den Fortschritt und die Richtung eines Projekts zu kommunizieren.

Roadmaps können in verschiedenen Formaten dargestellt werden, aber sie enthalten typischerweise folgende Elemente:

1. Zeitrahmen: Ein Zeitrahmen, der oft in Quartalen, Monaten oder Jahren unterteilt ist.
2. Ziele und Visionen: Übergeordnete Ziele, die erreicht werden sollen, sowie eine klare Vision des Projekts oder Produkts.
3. Meilensteine: Wichtige Ereignisse oder Abschlussdaten, die signifikante Fortschritte markieren.
4. Aktivitäten und Aufgaben: Konkrete Schritte, die unternommen werden, um die Ziele zu erreichen.
5. Verantwortlichkeiten: Wer für die verschiedenen Aktivitäten verantwortlich ist.
6. Prioritäten: Welche Aufgaben oder Meilensteine vorrangig behandelt werden sollten.

Fokusthemen

Ressourcenplanung:

1. Personelle Ressourcen:
   - Teamzusammensetzung: Identifikation der benötigten Rollen und Kompetenzen (z.B. Projektleiter, Entwickler, Tester, Designer).
   - Verfügbarkeit: Berücksichtigung der Verfügbarkeit der Teammitglieder, insbesondere bei Teilzeitkräften oder dual Studierenden.
2. Technische Ressourcen:
   - Hardware: Notwendige Server, Computer, mobile Endgeräte etc.
   - Software: Entwicklungsumgebungen, Tools für das Projektmanagement, Testsoftware usw.
   - Lizenzen: Verwaltung der benötigten Softwarelizenzen.
3. Finanzielle Ressourcen:
   - Budgetplanung: Ermittlung der Gesamtkosten des Projekts, inklusive Personalkosten, Materialkosten und sonstiger Ausgaben.
   - Finanzierung: Klärung, wie das Projekt finanziert wird (Eigenmittel, Fördermittel etc.).
4. Zeitliche Ressourcen:
   - Zeitrahmen: Einschätzung der benötigten Zeit für die verschiedenen Projektphasen.
   - Deadlines: Festlegung von Meilensteinen zur Überprüfung des Projektfortschritts.

Ablaufplanung:

1. Projektphasen:
   - Initiierung: Definition der Projektziele und -anforderungen.
   - Planung: Detaillierte Planung der Ressourcen und Festlegung der Projektstruktur.
   - Durchführung: Umsetzung der Projektpläne, inklusive Programmierung, Tests und Dokumentation.
   - Abschluss: Abschluss des Projekts, Evaluierung der Ergebnisse und Dokumentation.
2. Meilensteine:
   - Festlegung von wichtigen Punkten im Projektverlauf, an denen bestimmte Ergebnisse oder Fortschritte erreicht werden müssen.
3. Aufgabenverteilung:

- Detaillierte Zuweisung von Aufgaben an die einzelnen Teammitglieder oder Gruppen.
- Nutzung von Projektmanagement-Tools zur Nachverfolgung des Fortschritts und zur Koordination der Zusammenarbeit.

4. Risikomanagement:
   - Identifikation potenzieller Risiken, die den Projektablauf gefährden könnten.
   - Entwicklung von Strategien zur Risikominderung und Notfallplänen.

Dokumentation

- Alle Planungen sollten umfassend dokumentiert werden, um Transparenz zu gewährleisten und als Referenz für zukünftige Projekte zu dienen.
- Aus der Planung aller Ressourcen und Abläufe kann eine Kostenplanung abgeleitet werden. Diese stellt entsprechend die anfallenden Kosten für sämltiche Ressourcen im Projektverlauf dar. Die tatsächlichen Kosten Ende des Projektes können natürlich von den geplanten Kosten abweichen.
- Die Dokumentation sollte auch regelmäßige Updates und Anpassungen während des Projektverlaufs enthalten.

Risikoanalyse:

Eine Risikoanalyse für Projekte ist ein systematischer Prozess, der potenzielle Risiken identifiziert, bewertet und priorisiert, um deren Auswirkungen auf das Projekt zu minimieren oder zu vermeiden. In der IHK-Ausbildung für IT-Berufe ist die Risikoanalyse ein wichtiger Bestandteil des Projektmanagements, da sie hilft, die Unsicherheiten, die bei der Planung und Durchführung von IT-Projekten auftreten können, proaktiv zu managen.
Ziele der Risikoanalyse

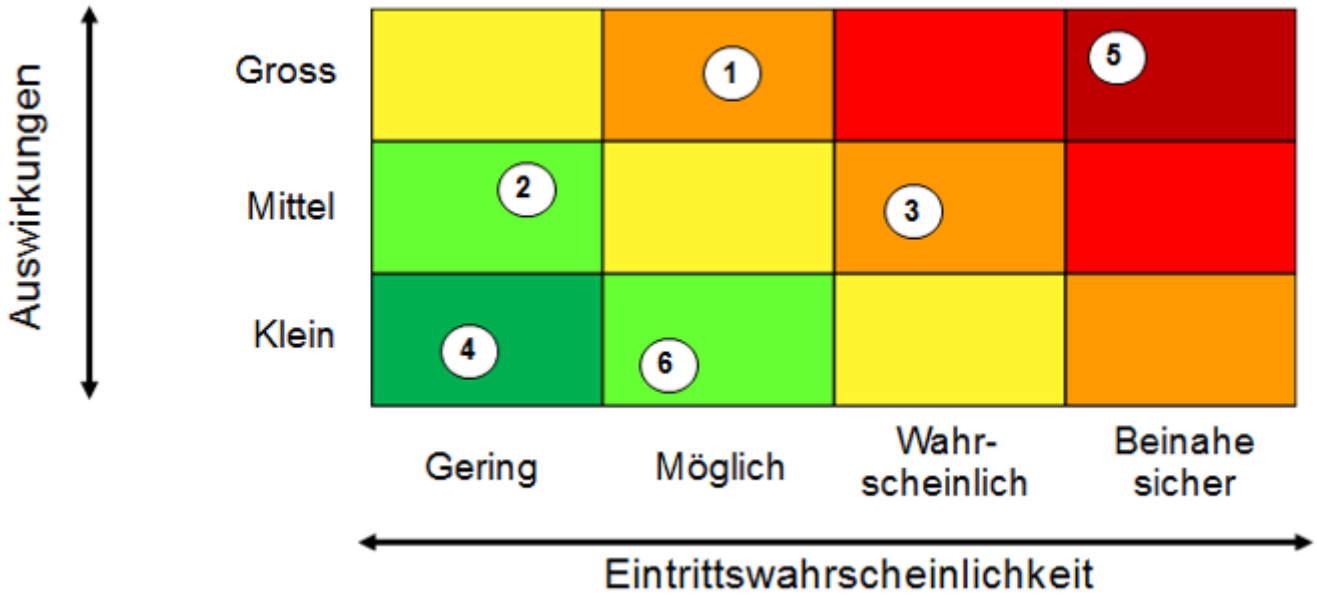
1. Identifikation von Risiken: Frühzeitiges Erkennen von Risiken, die den Projekterfolg gefährden könnten.
2. Bewertung von Risiken: Analyse der Wahrscheinlichkeit und der potenziellen Auswirkungen der identifizierten Risiken.
3. Entwicklung von Maßnahmen: Planung von Strategien zur Risikominderung, -vermeidung oder -akzeptanz.
4. Überwachung von Risiken: Kontinuierliche Beobachtung und Bewertung von Risiken während des Projektverlaufs.

Schritte einer Risikoanalyse

1. Risikoidentifikation:
   - Brainstorming-Methoden, Interviews und Workshops mit dem Projektteam und Stakeholdern werden eingesetzt, um potenzielle Risiken zu erfassen.
   - Risikokategorien können helfen, eine umfassende Identifikation durchzuführen, z. B. technische Risiken, finanzielle Risiken, personelle Risiken, organisatorische Risiken, rechtliche Risiken usw.
2. Risiko-Bewertung:
   - Jedes identifizierte Risiko wird hinsichtlich seiner Wahrscheinlichkeit (wie wahrscheinlich ist es, dass das Risiko eintritt?) und Auswirkung (wie stark würde es das Projekt beeinflussen?) bewertet.

- Oft wird eine Risiko-Matrix verwendet, um eine visuelle Darstellung der Risiken zu erstellen und sie nach ihrer Dringlichkeit zu priorisieren.

3. Risikomanagement-Strategien:
   - Entwicklung von Strategien zur Minimierung oder Vermeidung von Risiken. Dazu gehören:
     - Vermeidung: Änderungen am Projektplan, um das Risiko zu eliminieren.
     - Minderung: Maßnahmen zur Reduzierung der Wahrscheinlichkeit oder der Auswirkungen des Risikos.
     - Übertragung: Risiken an Dritte abgeben, z. B. durch Versicherungen oder Outsourcing.
     - Akzeptanz: Risiken akzeptieren, wenn die Kosten für Maßnahmen zur Risikominderung den potenziellen Verlust übersteigen.
4. Dokumentation und Kommunikation:
   - Alle identifizierten Risiken, Bewertungen und Strategien werden dokumentiert, oft in einem Risikoregister.
   - Regelmäßige Kommunikation der Risikobewertungen und -strategien an das Projektteam und die Stakeholder ist wichtig, um alle Beteiligten informiert zu halten.
5. Überwachung und Kontrolle:
   - Risiken müssen kontinuierlich überwacht werden, um neue Risiken zu identifizieren oder Veränderungen bei bestehenden Risiken zu erkennen.
   - Regelmäßige Überprüfungen und Updates des Risikoregisters sind entscheidend.



Projektdurchführung

Ziele: Das Hauptziel der Durchführungsphase ist die planmäßige Umsetzung aller Aufgaben und Meilensteine im Projekt. Der Fortschritt des Projekts wird in dieser Phase kontinuierlich überwacht und gesteuert, um die gesetzten Ziele (Anforderungen) im Rahmen der vorgegebenen Zeit- und Budgetgrenzen zu erreichen.
Aktivitäten:

- Aufgabenverteilung und Teamentwicklung: Zuweisung spezifischer Aufgaben an Teammitglieder basierend auf deren Fähigkeiten und Verantwortlichkeiten.
- Projektsteuerung und Controlling: Laufende Überwachung des Fortschritts und Vergleich der Ergebnisse mit dem Projektplan. Dies umfasst auch die Erfassung von Abweichungen und die Umsetzung notwendiger Korrekturmaßnahmen.
- Qualitätsmanagement: Sicherstellen, dass alle Projektergebnisse den geforderten Qualitätsstandards entsprechen. Durch regelmäßige Überprüfungen und Tests wird die Qualität der Ergebnisse gesichert.
- Änderungsmanagement: Verwaltung von Änderungen, die sich im Laufe des Projekts ergeben, sei es durch neue Anforderungen oder andere äußere Einflüsse. Dabei wird darauf geachtet, dass die Änderungen dokumentiert und im Projektplan berücksichtigt werden

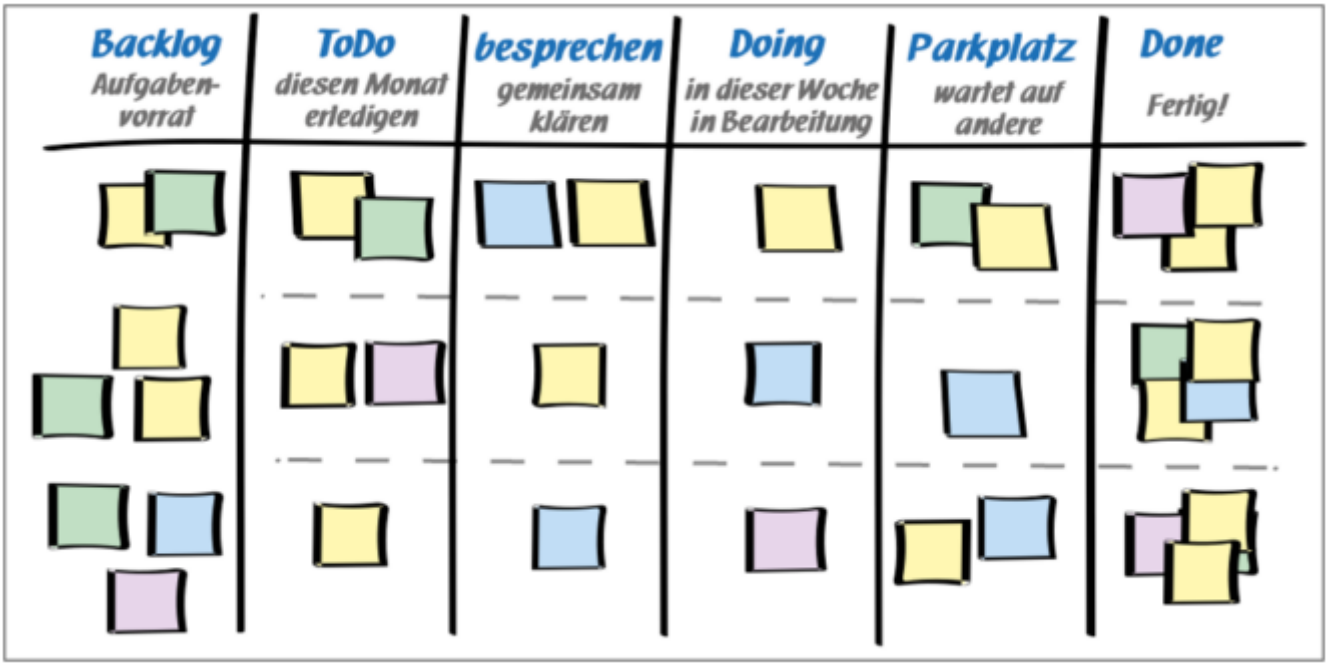
KANBAN-Board

Ein KANBAN-Board ist ein visuelles Werkzeug, das ursprünglich aus der Produktion von Toyota stammt und heute weit verbreitet im Projekt- und Aufgabenmanagement eingesetzt wird. Es hilft dabei, den Workflow von Aufgaben übersichtlich zu gestalten und Engpässe zu identifizieren. Typischerweise besteht ein KANBAN-Board aus Spalten, die den Fortschritt von Aufgaben abbilden, beispielsweise "To Do", "In Progress" und "Done".
Hauptmerkmale eines KANBAN-Boards

1. Visuelle Darstellung des Workflows: Aufgaben werden als Karten (Cards) dargestellt und durchlaufen verschiedene Spalten, die den Fortschritt der Arbeit widerspiegeln. Jede Spalte repräsentiert einen Schritt im Prozess.
2. Fokus auf den Fluss der Arbeit: Ziel ist es, den Arbeitsfluss (Flow) möglichst reibungslos zu gestalten und so Engpässe oder Verzögerungen schnell zu erkennen.
3. WIP-Limits (Work-in-Progress-Limits): Um Überlastungen zu vermeiden, kann die Anzahl der Aufgaben pro Spalte begrenzt werden. Dadurch bleibt der Fokus auf den laufenden Aufgaben, und es wird vermieden, dass zu viele Aufgaben gleichzeitig begonnen werden.
4. Pull-Prinzip: Im KANBAN-System wird eine neue Aufgabe erst dann in Angriff genommen, wenn Kapazität frei ist (Pull-Prinzip), anstatt dass Aufgaben „hineingedrückt“ werden (Push-Prinzip).

Aufbau eines typischen KANBAN-Boards
Ein klassisches KANBAN-Board besteht in der Regel aus diesen Spalten:

- To Do: Alle Aufgaben, die noch nicht begonnen wurden.
- In Progress: Aufgaben, an denen aktiv gearbeitet wird.
- Done: Aufgaben, die abgeschlossen sind.



Ticketsystem

Ein Ticket in einem Ticketsystem ist eine Art digitale Karte oder Datensatz, der eine spezifische Aufgabe, Anfrage, oder ein Problem dokumentiert und verfolgt. Tickets werden genutzt, um Arbeitsaufträge strukturiert zu erfassen, zu priorisieren, zuzuweisen und nachzuverfolgen, bis sie abgeschlossen sind. Sie sind vor allem in IT-Support-, Kundenservice- und Projektmanagement-Prozessen sehr hilfreich.
Hauptmerkmale eines Tickets
Ein Ticket enthält in der Regel folgende Informationen:

1. Titel oder Betreff: Eine kurze Zusammenfassung des Inhalts, wie etwa „Passwort zurücksetzen“ oder „Bug in der Benutzeroberfläche“.
2. Beschreibung: Eine detaillierte Erklärung des Problems oder der Aufgabe, die weitere Hintergrundinformationen und manchmal auch Lösungsansätze enthält.
3. Zuständiger Mitarbeiter: Die Person oder das Team, das für die Bearbeitung des Tickets verantwortlich ist.
4. Priorität: Eine Einstufung, wie dringend das Ticket bearbeitet werden muss (z. B. niedrig, mittel, hoch, kritisch).
5. Status: Der aktuelle Stand des Tickets, wie zum Beispiel „Neu“, „In Bearbeitung“, „Warten auf Feedback“ oder „Abgeschlossen“.
6. Zeitstempel und Historie: Dokumentiert, wann das Ticket erstellt, bearbeitet und abgeschlossen wurde. Dadurch wird der Bearbeitungsverlauf nachvollziehbar.
7. Kommentare und Anhänge: Zusätzliche Informationen, Screenshots oder Dateien können beigefügt werden, um das Ticket besser zu erläutern

Projektabschluss

Ziele: Die Abschlussphase hat das Ziel, das Projekt offiziell zu beenden und die Ergebnisse an den Auftraggeber zu übergeben. Zudem wird das Projekt final bewertet, um Erkenntnisse für zukünftige Projekte zu gewinnen. Mögliche neue Ansätze, welche man in potentiellen Anschlussprojekten

umsetzen kann, werden festgehalten. Neben der Bewertung des Projektergebnisses soll auch die Arbeitsweise im Projekt retrospektivisch bewerten werden, um Verbesserungspotentiale zu erkennen.
Aktivitäten:

- Abnahme und Übergabe: Formelle Abnahme der Projektergebnisse durch den Auftraggeber und Übergabe der Projektdokumentation.
- Abschlussdokumentation: Erstellung eines Abschlussberichts, der alle wichtigen Informationen und Erkenntnisse des Projekts zusammenfasst. Hier werden sowohl Erfolge als auch Herausforderungen dokumentiert.
- Erfahrungsanalyse (Lessons Learned): Analyse des gesamten Projektverlaufs, um wertvolle Erkenntnisse und Verbesserungspotenziale für zukünftige Projekte festzuhalten.
- Auflösung des Projektteams: Offizielle Beendigung des Projekts und Rückführung der Ressourcen, insbesondere der Teammitglieder, in ihre regulären Aufgaben oder andere Projekte.

Retroperspektive

Die Retrospektive dient dazu:

1. Stärken und Schwächen zu reflektieren: Das Team bespricht, was im Sprint gut gelaufen ist und was hätte besser laufen können.
2. Verbesserungsmöglichkeiten zu identifizieren: Auf Basis der Erfahrungen aus dem letzten Sprint werden konkrete Maßnahmen und Ansätze entwickelt, die die Arbeitsweise des Teams verbessern.
3. Teamzusammenhalt und Kommunikation zu fördern: Die Retrospektive bietet eine offene und sichere Umgebung, in der jedes Teammitglied seine Meinung und Ideen äußern kann, ohne Angst vor Kritik zu haben.

Typischer Ablauf einer Retrospektive:
Eine Scrum-Retrospektive besteht normalerweise aus folgenden Schritten:

1. Einstieg und Zielsetzung: Der Scrum Master eröffnet die Retrospektive und erklärt kurz das Ziel, das darin besteht, die Zusammenarbeit und Prozesse zu verbessern.
2. Rückblick auf den Sprint: Das Team reflektiert den vergangenen Sprint und sammelt Feedback zu verschiedenen Aspekten der Arbeit.
3. Identifikation von Verbesserungen: Gemeinsam werden Herausforderungen und Hindernisse/Hemnissen identifiziert. Häufig werden Methoden wie „Start, Stop, Continue“ genutzt, um festzulegen, was neu begonnen, aufgehört oder fortgesetzt werden sollte.
4. Maßnahmen planen: Auf Basis der Erkenntnisse werden konkrete Maßnahmen für den kommenden Sprint festgelegt, mit denen das Team seine Arbeitsweise verbessern kann.
5. Abschluss: Der Scrum Master fasst die Ergebnisse und Maßnahmen zusammen und stellt sicher, dass alle Teammitglieder diese verstanden haben.

Hindernisboard

Ein Hindernisboard (auch „Impediment Board“ genannt) ist ein visuelles Tool, das im agilen Projektmanagement, insbesondere in Scrum, verwendet wird, um Hindernisse oder Blockaden im Arbeitsprozess des Teams zu identifizieren und sichtbar zu machen. Es dient dazu, Probleme, die

die Arbeit des Teams verlangsamen oder behindern, zu dokumentieren und systematisch zu beseitigen.
Zweck eines Hindernisboards:

1. Transparenz schaffen: Durch die Visualisierung von Problemen können alle Teammitglieder, der Scrum Master und andere Beteiligte sofort erkennen, wo es Schwierigkeiten gibt.
2. Blockaden zeitnah lösen: Probleme werden nicht nur identifiziert, sondern können auch priorisiert und effektiv angegangen werden, um die Produktivität des Teams zu steigern.
3. Verantwortlichkeiten klären: Das Board kann Verantwortlichkeiten und Fortschritte bei der Problemlösung aufzeigen, was sicherstellt, dass nichts „liegen bleibt“.

Aufbau eines Hindernisboards
Das Hindernisboard ist typischerweise in Spalten unterteilt, ähnlich einem Kanban-Board, um den Status jedes Hindernisses zu visualisieren:

- Identifiziert: Hier werden neu erkannte Hindernisse aufgenommen. Das kann alles sein, was das Team daran hindert, die Arbeit effizient zu erledigen, wie technische Probleme, Ressourcenkonflikte, fehlende Informationen oder externe Abhängigkeiten.
- In Bearbeitung: Hindernisse, an denen aktuell gearbeitet wird, um sie zu beseitigen.
- Gelöst: Sobald ein Problem beseitigt ist, wird es in diese Spalte verschoben, was dem Team zeigt, dass es die Blockade überwunden hat.

Klassendiagramm und OOP

Klassendiagramm

Was bedeutet überladen?

Beim überladen einer Methode werden mehrere Methoden gleichen Namens innerhalb einer Klasse definiert, die sich durch ihre Parameterlisten unterscheiden.

-> Methode hat den selben Namen, aber verschiedene Parameter!

Was bedeutet überschreiben?

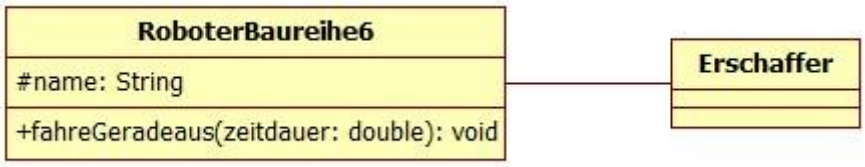
Beim überschreiben einer Methode wird eine Methode der Basisklasse in der abgeleiteten Klasse neu definiert. Die Parameterliste ändert sich dabei nicht. Hier wird die Methode "toString" überschrieben, indem in jeder abgeleiteten Klasse der jeweilige Typ mit angegeben wird.

-> Methode wird in Unterklasse überschrieben!

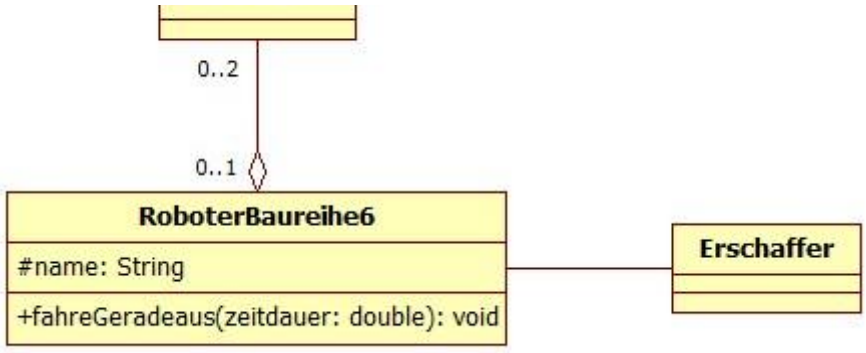
Was bedeuten die Zugriffsparameter

- - : private -> Also nur innerhalb der eigenen Klasse verfügbar
- # : protected -> Sind ausschließlich innerhalb ihrer Klasse und der von ihnen abgeleiteten Klassen verwendbar
- + : public -> Unterliegen keinen Zugriffsbeschränkungen

Was ist eine Assoziation

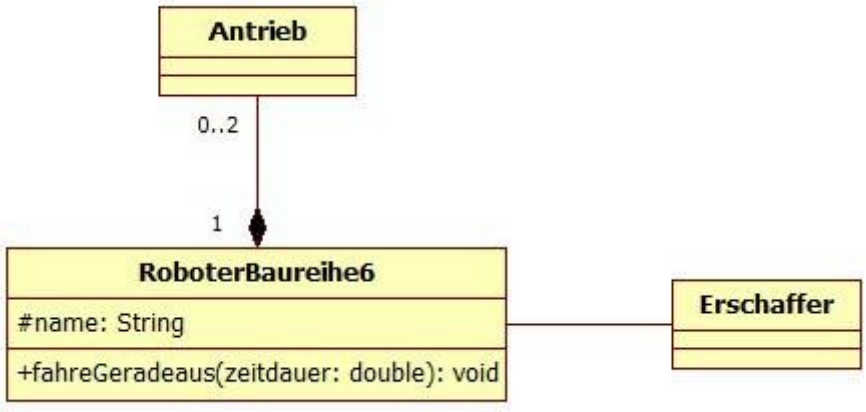


Was ist eine Aggregation

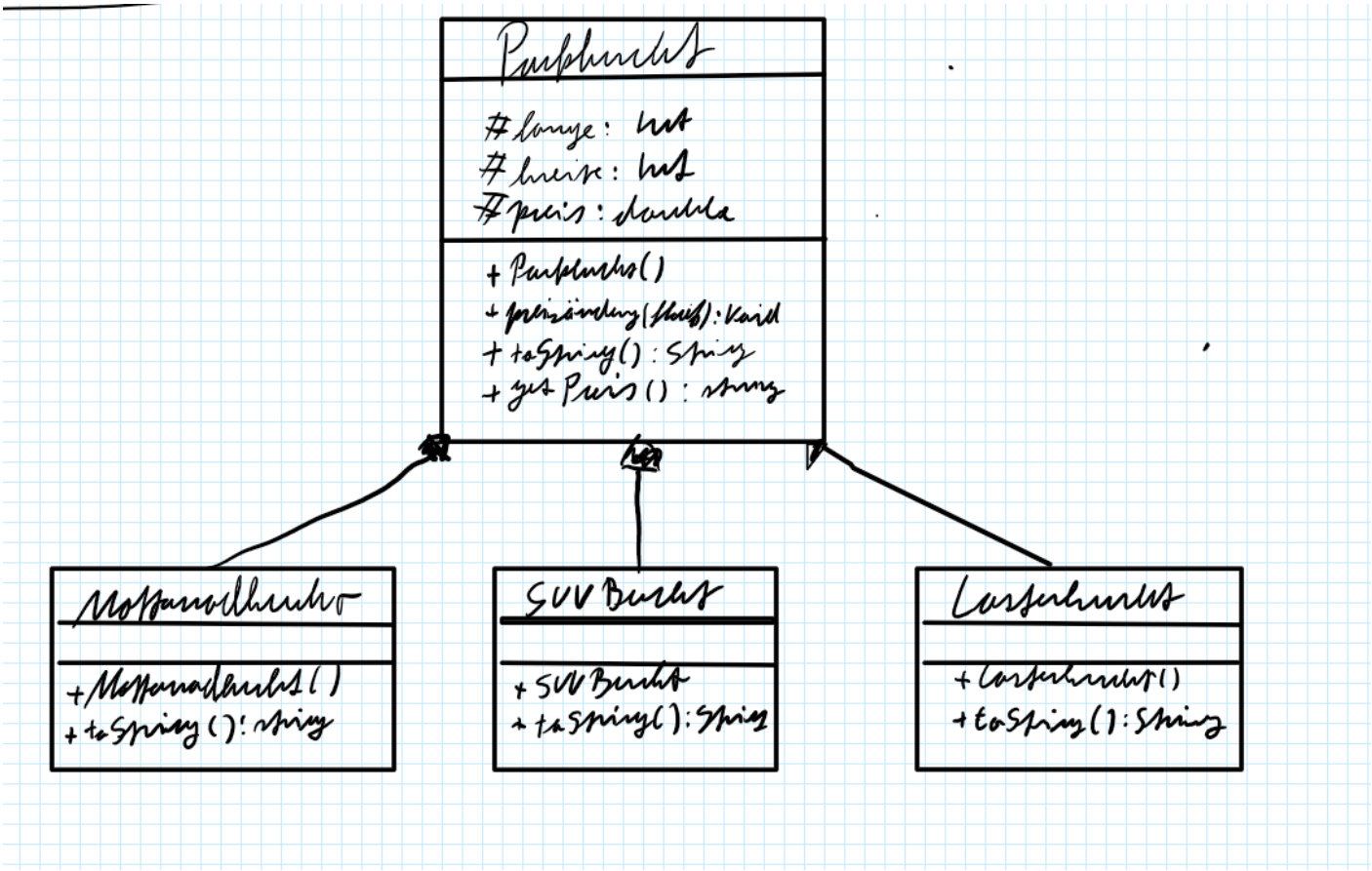


- Wird statt einer Assoziation verwendet, wenn es sich um eine Teil-Ganzes Beziehung handelt
- Zusammensetzung wird Multiplizität genannt

Komposition



- Bei einer Komposition stehen zwei Klassen nicht nur in einer Teil-Ganzes Beziehung sondern das "Teil" existiert ohne das "Ganze" auch nicht. (Existenzabhängigkeit)
- Die Multiplizität muss auf der Seite des Ganzen zwangsläufig 1 sein



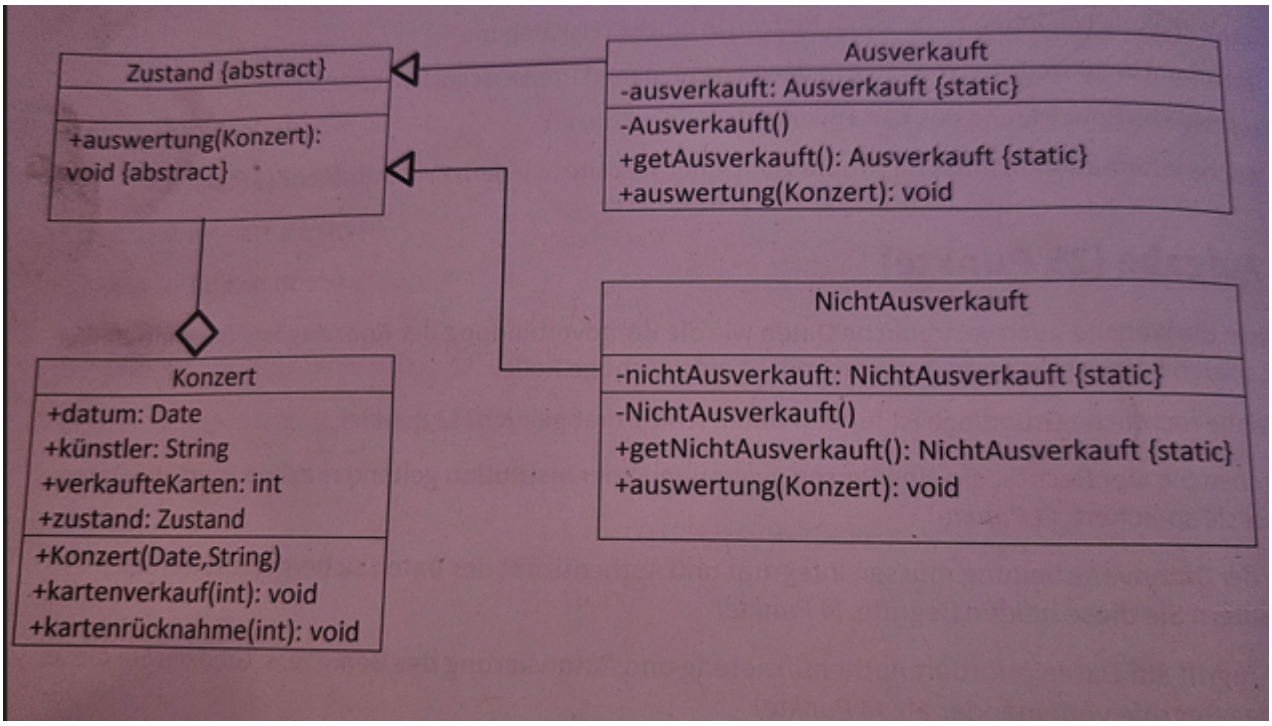
Was bedeutet {abstract} im Klassendiagramm

- Abstrakte Klassen sind nicht vollständig - man kann davon keine Objekte direkt erzeugen
- Sie dienen als Grundlage für andere Klassen (Basisklassen)
- Können abstrakte Methoden enthalten, nur deklariert aber nicht implementiert

Was bedeutet {static} im Klassendiagramm

- Statische Mitglieder (Attribute oder Methoden) gehören zur Klasse, nicht zu einzelnen Objekten
- Man kann sie verwenden um ein Objekt zu erzeugen.

Aufgaben



Die Methode "getAusverkauft" ist folgendermaßen implementiert:

```
public static Ausverkauft getAusverkauft()
{
 falls ausverkauft = null
   ausverkauft = new Ausverkauft()
 rueckgabe ausverkauft
}
```


```
public static Ausverkauft getAusverkauft()
{
 falls ausverkauft = null
   ausverkauft = new Ausverkauft()
 rueckgabe ausverkauft
}
```


Erläutern Sie hieran die Besonderheiten der Singleton-Klasse "Ausverkauft"

- Die Methode sorgt dafür das nur ein einziges Objekt der Klasse Ausverkauft existiert.
- Wenn noch keins da ist, wird es erstellt.
- Danach wird immer dieses eine Objekt zurückgegeben
- -> Das nennt man Singleton-Pattern

Warum ist diese Methode "static"?

- Die Methode kann man ohne Objekt aufrufen, also es muss kein Objekt instanziiert werden
- Sie gehört zur Klasse selbst, nicht zu einem Objekt

```
Ausverkauft a = Ausverkauft.getAusverkauft();
```

- Das ist wichtig, weil man ja noch gar kein Objekt hat, bevor man diese Methode aufruft.

Implementieren Sie einen Konstruktor der Klasse "Konzert", der u.a. die Zahl der verkauften Karten auf 0 und den Zustand auf "nichtAusverkauft" setzt.

```
public Konzert(datum Date, küenstler String)
{
 this.datum = datum,
 this.kuenstler = kuenstler,
 this.verkaufteKarten = 0,
 this.Zustand = NichtAusverkauft.getNichtAusverkauft()
}
```

Implementieren Sie die Methode "auswertung" der Klasse "nichtAusverkauft". Diese prüft, ob für das betreffende Konzert mehr als 1000 Karten verkauft werden wurden. In diesem Fall setzt sie den Zustand des Konzerts auf "Ausverkauft".

```
Public void auswertung(Konzert konzert)
{
 wenn konzert.verkaufteKarten > 1000
  konzert.Zustand = Ausverkauft.GetAusverkauft()
}
```

Implementieren Sie die Methode "kartenverkauf", der die Zahl der verkauften Karten erhöht und anschließend eine Auswertung des Zustands auslöst.

```
public void kartenverkauf(int anzahl)
{
 verkaufteKarten = verkaufteKarten + anzahl
 zustand.auswertung(this) // this, weil wir uns in der Klasse "Konzert" befinden.
}
```

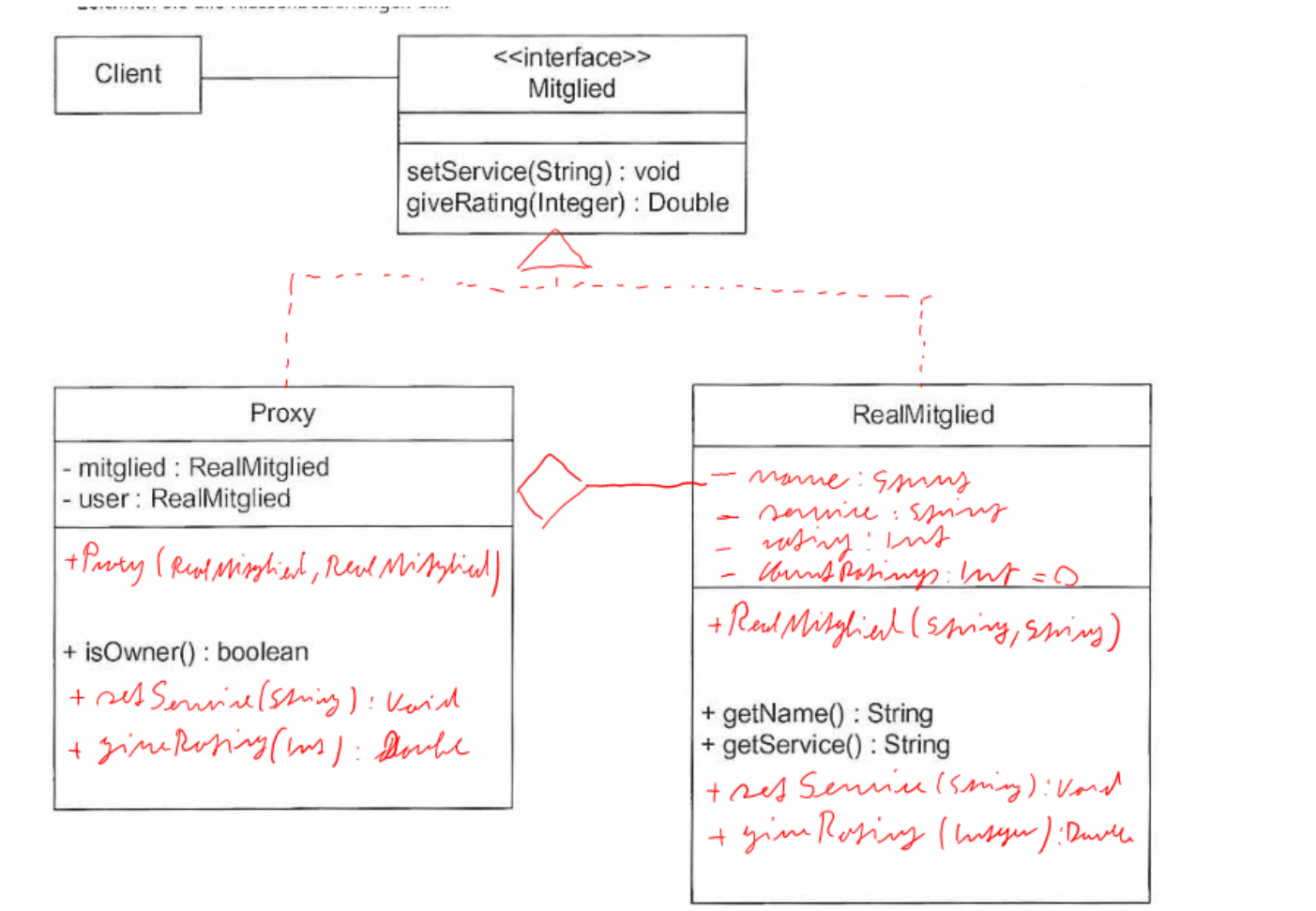
Implementieren Sie die folgende Funktion, die die Anzahl ausverkaufter Konzerte bestimmt.

```
public int anzahlAsuverkaufterKonzerte (Konzert[] liste)
{
```

```
  zaehler = 0
  fuer alle Konzert in liste
    falls konzert.zustand hatTyp Ausverkauft
      zaehler = zaehler + 1
}
```

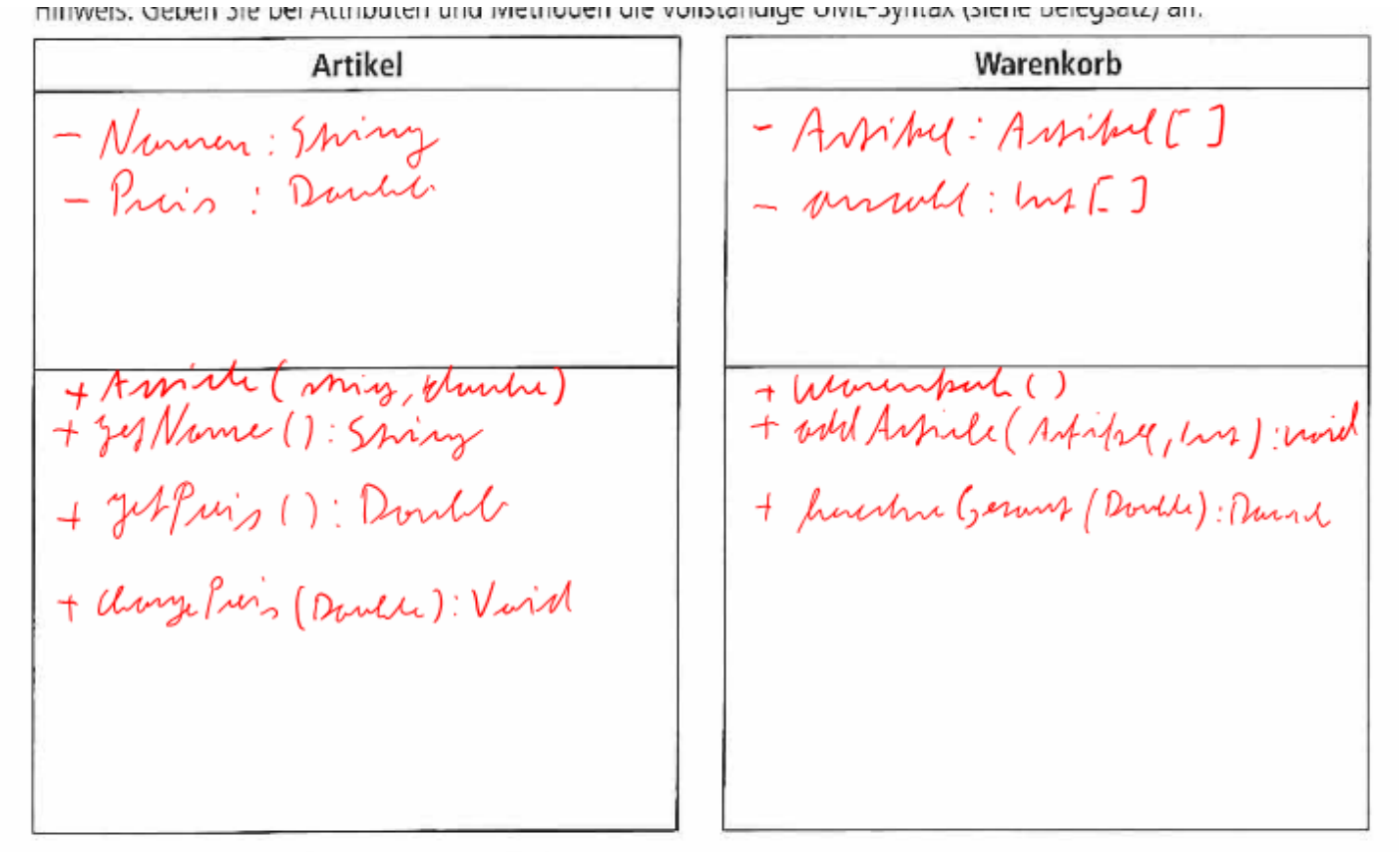
Ergänzen Sie das folgende unvollständige UML-Klassendiagramm nach den folgenden Vorgaben.

- Die Klasse "RealMitglied" soll die nur klasseninternen sichtbaren Instanzvariablen "name", "service", "rating", und "countRatings" beinhalten.
- Der öffentliche Konstruktor der Klasse "RealMitglied" soll zwei Übergabeparameter haben, mit denen "name" und "service" initialisiert werden.
- Der öffentliche Konstruktor der klasse "Proxy" soll zwei "RealMitglied"-Objekte übergeben bekommen und diese mit diese mit den Referenzen "mitglied" und "user" verknüpfen.
- Die Klassen "Proxy" und "RealMitglied" sollen beide das Interface Mitglied implementieren.
- Zeichen Sie alle Klassenbeziehungen ein:



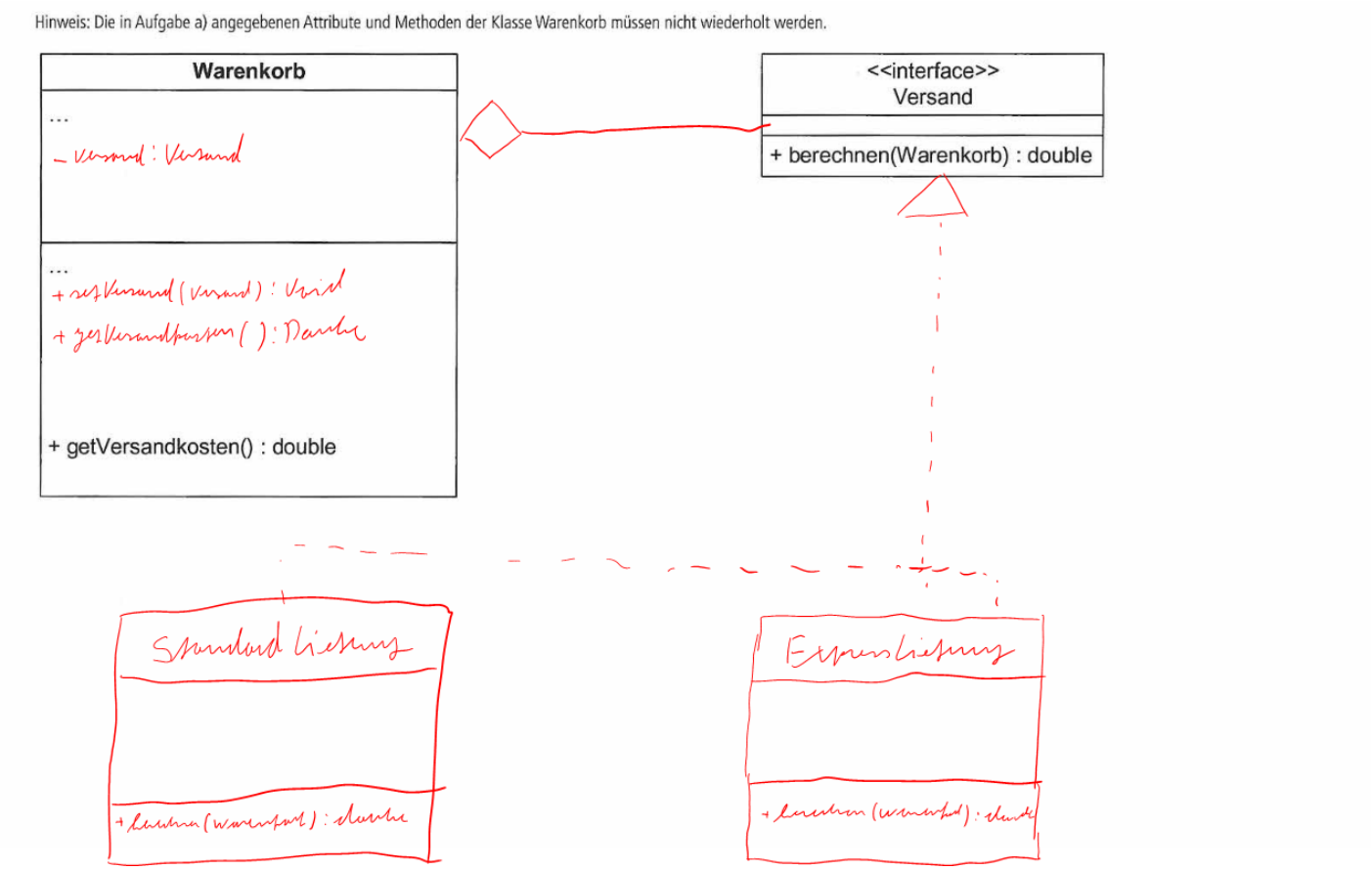
Es ist folgendes Pflichtenheft gegeben:

- ein Artikel hat einen Namen und einen Preis
- Beim Anlegen eines neuen Artikels werden Name und Preis festgelegt
- Der Name eines Artikels kann abgefragt, aber nicht verändert werden.
- Der Preis eines Artikels kann abgefragt und verändert werden.
- Einem "Warenkorb" können verschiedene Artikel in beliebigen Stückzahlen hinzugefügt werden.
- Für den Inhalt des Warenkorbs kann der Gesamtwert berechnet werden.
- Instanzvariablen sind nach außen nicht sichtbar (information hiding/geheimnisprinzip)
- Methoden sollen von überall her aufrufbar sein.



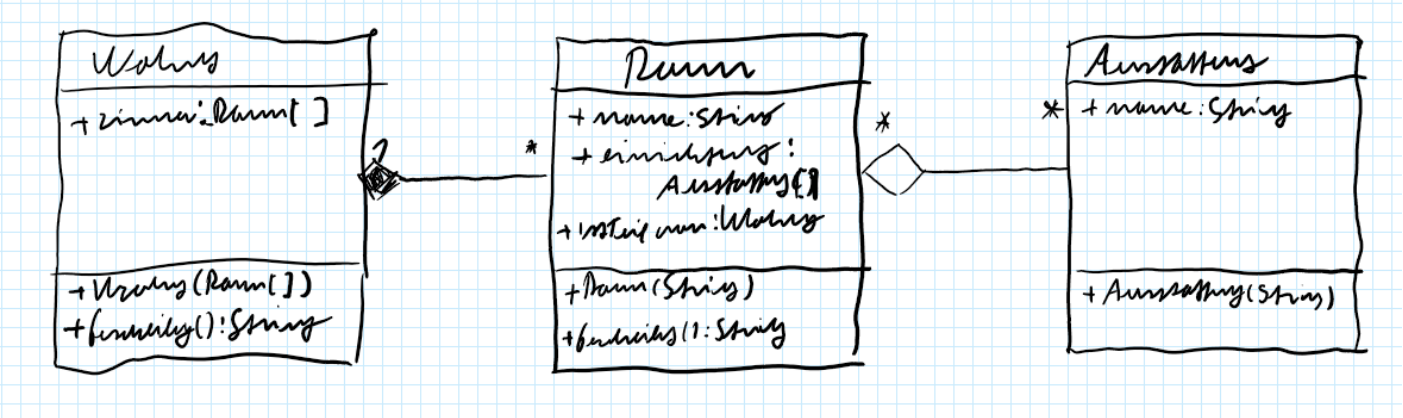
Das Pflichtenheft wird um folgende Punkte erweitert:

- Die Versankosten sollen ebenfalls ermittelt werden.
- Bei Standard-Lieferungen und Express-Lieferungen werden unterschiedliche Berechnungsalgorytmen verwendet.
- Die Berechnungsalgorythmen der Versankosten wechseln häufig, deshalb sollen sie vom **Warenkorb entkoppelt sein**
- Dazu soll die Klasse "Warenkorb" um eine **Referenz "versand" vom Typ des Interdace "Versand",** eine Methode "setVersand" und Setzen der Versandart und eine Methode "getVersandkosten" zur Abfrage der Versandkosten erweitert werden.
- -> Ergänzen Sie das Klassendiagramm!



Implementieren Sie in Pseudocode die Methode "getVersandkosten" der Klasse Warenkorb

```
public double getVersandkosten()
{
  double versankosten = versand.berechnen(this)
  rueckgabe versandkosten
}
```



Erläutern Sie den Unterschied zwischen den Relationen "Aggregation" und "Komposition"

Bei einer Aggregation handelt es sich um eine Teil-Ganzes Beziehung, und bei einer Komposition um eine Abhängigkeitsbeziehung, was bedeutet das eine Klasse nicht ohne die Andere existieren kann. Beide Beziehungen beschreiben Abhängigkeiten voneinander und enthalten Multiplizitäten.

Die Relation "Raum" enthält Ausstattung entspricht einer m:n-Relation in einem Entity-Relationship-Diagramm. Erläutern Sie die unterschiedliche Umsetzung einer m:n Relation in einer Datenbank gegenüber der entsprechenden Relation in einem objektorientierten Modell. (Hinweis: Beziehen Sie sich in ihrer Erläuterung auf die 1. Normalform)

Das ein Raum mehrere Ausstattungsmerkmale haben kann und ein Ausstattungsmerkmal bei mehreren Räumen auftreten kann, wird in einem Klassendiagramm durch listenförmige Attribute dargestellt.

Die erste Normalform verbietet listenförmige Attribute in einer Datenbank. Stattdessen werden m:n-Relationen durch eine Zwischentabelle dargestellt, die die Schlüsselattribute beider Relationspartner miteinander kombiniert.

Die Umsetzung der Relation "Wohnung enthält Raum" ist redundant gestaltet

- **Erläutern Sie dies anhand des Klassendiagramms**
  - Raum enthält das Attribut istTeilVon Wohnung. Das ist eine Dopplung.
- **Nennen Sie je ein Vorteil und einen Nachteil dieser redundanten Speicherung.**
  - Vorteil: Sie ermöglicht eine schnelle Zuordnung zu Wohnung
  - Nachteil: Sie verbraucht Speicherplatz und macht das Modell komplexer

Geben Sie den Quellcode des Konstruktors der Klasse "Wohnung" an, der den Wert des Attributs "zimmer" setzt und für jeden Raum angibt, dass er Teil dieser Wohnung ist.

```
public Wohnung (Raum[] raeume)
{
  zimmer = raeume
  foreach(Raum r in Zimmer)
  {
    r.istTeilvon(this);
  }
}
```

Geben Sie den Quellcode der Methode "beschreibung" der Klasse "Wohnung" an. Die Methode soll für alle Räume der Wohnung deren Beschreibung liefern.

```
public string beschreibung()
{
  string ergebnis = "";
```

```
  foreach(Raum r in Zimmer)
  {
    ergebnis += r.beschreibung();
  }
  return ergebnis;
}
```

Die in Aufgabe 2 genannte Klasse "Raum" verfügt über folgende Methode

```
public string beschreibung()
{
  string ausgabe = name + " verfügt über folgende Ausstattungsmerkmale: ";
  for (int i = 0; i <= einrichtung.size; i++)
  {
    ausgabe += einrichtung[i].name + ", ";
  }
  return ausgabe;
}
```

Beim Aufruf dieser Methode erscheint zunächst die Meldung "NullReferenceException". Nach entsprechender Korrektur des Programms erscheint die Meldung "ArrayOutOfBoundsException".

aa) Erklären Sie den Begriff Exception im Allgemeinen

Eine Exception ist ein Objekt welches beim Debuggen Aufschluss über die Art des Fehlers liefert

ab) Um welche speziellen Exceptions handelt es sich hier?

NullReferenceException gibt an, wenn eine undeklarierte Variable vor der Deklaration verwendet wird

ArrayOutOfBoundsException wird ausgegeben wenn der falsche Index übergeben wird, der die Gesamtgröße des Arrays übersteigt.

ac) Wie kann man ein Programm erweitern, um eine Exception abzufangen?

Mit Try Catch Blöcken kann eine Exeption zur Laufzeit erkannt und ausgegeben werden.

ad) Korrigieren Sie das Programm so, dass beide Exceptions nicht mehr auftreten

```
public string beschreibung()
{
  string ausgabe = this.name + " verfügt über folgende Austattungsmerkmale: ";

  for(int i = 0; i < einrichtung.size; i++)
  {
    ausgabe += einrichtung[i].name + ", ";
  }
  return ausgabe;
}
```

b) Sollte das Attribut "name" der Klasse "Raum" keinen Wert haben, führt das Programm zu einem nicht zufreidenstellenden Ergebnis. Formulieren Sie den Code des Konstruktors der Klasse "Raum" so, dass dieses Problem nicht entsteht

```
Public Raum(name string)
{
  this.name = "Kein Name";
}
```

c) Nennen Sie einen weiteren Grenzfall, der beim Testen des Programms zu beachten ist und der zu einem nicht zufriedenstellenden Ergebnis führt

Es sollte in Betracht gezogen werden, dass Raum keinen Namen haben könnte.

Die Kfz-Kennzeichen werden in einem Array gespeichert. Zur Suche nach einem bestimmten Kennzeichen soll die folgende Funktion verwendet werden:

```
int suchen(String suchwert, String[] parkliste, int startposition, int endposition)
{
  int mitte=(startposition+endposition)/2;
  falls suchwert < parkliste[mitte]
    rueckgabe suchen(suchwer, parkliste, startposition, mitte+1)
  falls suchwert > parliste[mitte]
    rueckgabe suchen(suchwert, parkliste, mitte+1, endposition)
}
```

Schreiben Sie in Pseudocode einen Algorithmus, der ein Arrayelement an der Stelle i löscht. Die länge des Arrays beträgt parkliste.laenge

Weitere OOP Begrifflichkeiten

- **Garbage Collector** -> Räumt nicht mehr verwendete Objekte aus dem Speicher.
- **Getter und Setter** -> Dienen zum Zugriff auf private Attribute einer Klasse und ermöglichen Kontrolle beim Lesen und Ändern der Daten
- **Konstruktor** -> Wird aufgerufen wenn ein Objekt neu erzeugt wird. Kann Parameter annehmen, um das Objekt gleich richtig zu initialisieren
- **this** -> Verweist auf das aktuelle Objekt, innerhalb der Klasse
- **Polymorphie** -> Objekte können unterschiedliches Verhakten Zeigen, obwohl sie denselben Typ haben.
- **interface** -> Eine Schnittstelle, die vorgibt was eine Klasse tun soll, Implementierung erfolgt in Klassen
- **Abstrakte Klasse** -> Kann nicht dierekt instanziert werdne und dient als gemeinsame Oberklasse mit fertiger funktionalität
- **Kapselung** -> Daten werden versteckt (private) und nur über Getter/Setter zugänglich gemacht.

Pseudocode, PAP, Struktogramm

Vorgehensweise

- Aufgabenstellung lesen & verstehen
  - Was ist der Zweck des Programms?
  - Welche Eingaben, Verarbeitungen und Ausgaben gibt es?
- Eingaben Identifizieren
  - Was bekommt die Funktion als Parameter oder von außen übergeben?
- Verarbeitung logisch zerlegen
  - Rechenschritte, Bedingungen, Schleifen? Was passiert wann?
  - Gibt es Rechenvorschriften wie "abwechselnd mit 1 und 3 gewichten"? Notieren!
- Ergebnis/Ausgabe festlegen
  - Was soll am Ende zurückgegeben oder angezeigt werden?

Aufgaben

Für den Onlineversand sollen die Weinflaschen mit Barcode-Aufkleber versehen werden, der nähere Informationen zum Wein enthält. Der Barcode besteht aus zehn Ziffern.

123-456-78-9-p

- Ziffern 1 bis 3: Schlüssel für Region
- Ziffern 4 bis 6: Schlüssel für Rebsorte
- Ziffern 7 - 8: Jahrgang
- Ziffer 9: Geschmacksangabe (lieblich, halbtrocken, trocken ...)
- Dazu kommt an der 10. Stelle eine Prüfziffer p.

Die Prüfziffer soll nach folgender Beschreibung errechnet werden:

- Die einzelnen Ziffern werden alternierend gewichtet von Links nach Rechts mit 1 und 3:
  - Ziffer_1 x 1, Ziffer_2 x 3 .... Ziffer_9 * 1
- Die 9 gewichteten Produkte werden addiert.
- Die Prüfziffer ist die Differenz der Summe zum nächstkleineren Vielfachen von 10 (modulo 10)

```
function ermittlePrüfziffer(int[] barcode)
{
```

```
  int pruefziffer = 0

  für i = 0; i < barcode.laenge; i++
  {
    wenn (i % 2 = 0)
    {
      pruefziffer += barcode[i] * 3
    } sonst
    {
      pruefziffer += barcode[i] * 1
    }
  }

  pruefziffer = (10 - (pruefziffer % 10)) % 10

  rueckgabe pruefziffer
}
```

Alle aktuell vorhandenen Barcodes der Weine und deren Jahresabsatz sind in einer zweidimensionalen Tabelle "Absatz" in folgender Form gespeichert:

Region	Rebsorte	Jahrgang	Geschmacksrichtung	Absatz in Stk.
123	456	78	9	46
333	125	20	4	998
...	...	...		

Erstellen Sie auf der Folgeseite eine Funktion "sucheTopseller", die für ein übergebendes Kriterium (0 = Region; 1 = Rebsorte, 2 = Jahrgang; 3 = Geschmacksrichtung) und einen entsprechenden Vorgabewert (z.B. 123 für eine bestimmte Region) den umsatzstärksten Wein ermittelt und den Barcode dieses Weines als Zeichenkette ohne Prüfziffer zurückliefert. Das zweidimensionale Array "Absatz" steht in der Funktion "sucheTopseller" zur Verfügung. Gehen Sie davon aus, dass alle Weine einen unterschiedlichen, positiven Absatz haben.

Hinweis: Für das Zusammenfügen von Zeichenketten kann der + Operator verwendet werden. Gemischte Ausdrücke vom Typ String und Ganzzahl sind möglich.

```
FUNKTION sucheTopseller(int kriterium, int vorgabe) : STRING
  int barcode = 0;
  int maxAbsatz = -1;
  int maxIndex = -1
```

```
  VON i = 0 BIS Absatz.laenge
    WENN Absatz[i][kriterium] == vorgabe und Absatz[i][4] > maxAbsatz  //
      maxAbsatz = Absatz[i][4]
      maxIndex = i
    ENDE WENN
  ENDE VON

  barcode = barcode + Ansatz[maxIndex][i]
  VON i = 1 BIS 3
    barcode = barcode + "-" + Absatz[maxIndex][i]
  ENDE VON
  RUECKGABE barcode
ENDE FUNKTION
```

Für die Helferlein e.V. soll eine Prozedur entwickelt werden, welche die erbrachten Leistungen und die Aufsummierten Entgelte auflistet. Die Daten sind in einem Journal gespeichert.

Journal (Beispiel)

Datum	MitgliedID	LeistungID	AnzahlStunfrn
01.04.2021	100062	100076	2
11.04.2021	100062	100076	3
10.04.2021	100062	500123	1
13.04.2021	201235	200234	1
14.04.2021	201235	200234	1
07.04.2021	201235	200356	1

Das Journal ist nach MitgliedID und bei gleicher MitgliedID nach LeistungID sortiert.

Die Ausgabeliste soll wie folgt aufgebaut sein:

Nr	MitgliedID	Name	Vorname	LeistungID	Leistung	AnzahlStunden	Stundensatz	Gesamt
1	100062	Clausen	Jens	100076	Gießen	5	6,00	30,00
2	100062	Clausen	Jens	500123	Umgraben	1	6,00	6,00
							Summe	36,00
1	201235	Rader	Sabrina	200234	Hausputz	2	6,00	12,00
2	201235	Rader	Sabrina	200356	Einkauf	1	6,00	6,00

Nr	MitgliedID	Name	Vorname	LeistungID	Leistung	AnzahlStunden	Stundensatz	Gesamt
							Summe	18,00
							Gesamtsumme	54,00

Folgende Funktionen sollen verwendet werden:

hole_satz() : String	Liest den nächsten Datensatz der Journal-Tabelle in eine Zeichenkette ein. Kann kein Satz mehr gelesen werden, liefert die Funktion den String "".
lese_m_id(satz:string):Integer	Ermittelt die MitgliedID aus Satz
lese_l_id(satz:string):Interger	Ermittelt die LeistungsID aus Satz
lese_anz_std(satz: string):Integer	Ermittelt die Anzahl der Stunden aus Satz
schreibe_kopf()	Schreibt die Kopfzeile der Positionen-Tabelle
schreibe_daten (nr: Integer, mitgliedid: Integer, leistungid: Integer, anzahlstunden: Integer, stundensatz: Double, summe: Double)	Schreibt eine Datenzeile in der geforderten Darstellung. Name, Vorname und Leistung werden automatisch aus mitgliedid und leistungid ermittelt.
schreibe_summe(summe: Double)	Schreibt die (berechnete) Summe für ein Mitglied
schreibe_gsumme(gsumme: Double)	Schreibt die (berechnete) Gesamtsumme des Journals

Entwickelt Sie einen Algoritmus für die Prozedur erstelle_liste(stundensatz: Double), der die Liste entsprechend der Vorgabe schreibt und dazu die entsprechenden Werte berechnet. Verwendete Variablen müssen nicht deklariert werden.

```
erstelle_liste(stundensatz: Double)

  printKopf()
  datensatz = hole_satz()
  zaehler = 0
  gesamtsumme = 0;

  SOLANGE datensatz != "" Dann
     mitglied = lese_m_id(datensatz)
     summeMitglied = 0
```

```
    SOLANGE nächsterDatensatz != "" UND lese_m_id(datensatz) = mitglied
        leistung = lese_l_id(datensatz)
        stunden = 0;



        WENN lese_l_id(datensatz) = lese_l_id(nächsterDatensatz) DANN
            summeLeistung += lese_anz_stunden(datensatz) + lese_anz_stunden(nächsterDatensatz) *
stundensatz

      schreibe_daten(zaehler, lese_m_id(datensatz), lease_l_id(datensatz), lease_anz_stunden(datensatz),
stundensatz, summeLeistung)
      schreibe_summe(summe)

        SONST
            summeLeistung += lese_anz_stunden(datensatz) + lese_anz_stunden(nächsterDatensatz) *
stundensatz
            gesamtsumme = gesamtsumme + summeleistung
        ENDE WENN

        datensatz = naechsterSatz



        WENN get_m_id(datensatz) != get_m_id(naechsterDatensatz)
```

Eine geplante Stored Procedure soll Folgendes leisten:

1. Zunächst werden alle Ferienwohnungen im selben Ort ermittelt und in einer Tabelle gespeichert
2. Jede Wohnung aus dieser Tabelle wird anschließend darauf überprüft, ob die Bettenzahl mindestens so groß wie bei der stornierten Wohnung ist und der Preis höchstens um 10% abweicht
3. Eine Wohnung, die diese Kriterien nicht erfüllt, wird aus der Tabelle gestrichen
4. Sollten anschließend mehr als 3 Wohnungen übrigbleiben, werden alle bis auf die drei preiswertesten gelöscht. (Hier ist der betreffende Preis zu ermitteln.)

5. Sollte keine Wohnung übrig bleiben, werden alle Wohnungen aus demselben Land wie die stornierte Wohnung ermittelt und für diese Schritt 2-4 wiederholt.
6. Sollte es keine Wohnung in demselben Land mehr geben, erfolgt eine Misserfolgsmeldung

Zeichnen Sie für diese Stored Procedure ein Struktogramm

Datenbanken

Inhalt

Eine Datenbank ist ein strukturiertes System zur Speicherung, Verwaltung und Abfrage von Daten. Typischerweise arbeitet man mit relationalen Datenbanken, bei denen die Daten in Tabellen organisiert sind.

Diese Tabellen stehen oft in Beziehungen zueinander - daher der Name "relationale Datenbank"

Normalisierung

Normalisierung ist ein Prozess in der Datenbankmodellierung, bei dem man Tabellen in sinnvolle, saubere Strukturen bringt, um:

- Redundanzen zu vermeiden
- Dateiinkonsistenzen zu verhindern
- Flexibilität und Wartbarkeit zu erhöhen

Dazu gibt es verschiedene Normalformen, die man Schritt für Schritt durchläuft.

Normalformen

Erste Normalform

Alle Attribute sind atomar - aber nicht teilbar.

Beispiel (nicht 1NF):

Kunde	Telefonnummern
Anna	0123,0456

Problem: Mehrere Telefonnummern in einem Feld -> nicht atomar

In die 1NF umgewandelt

Kunde	Telefonnummer
Anna	0123
Anna	0456

Zweite Normalform

1. Muss in 1NF sein
2. Keine partiellen Abhängigkeiten vom Primärschlüssel (gilt nur bei zusammengesetzten Schlüsseln)

Beispiel (2NF nicht erfüllt):

Student_ID	Student_Name	Kurs_ID	Kurs_Name	Dozent	Raum
1	Anna Müller	A1	Mathe	Dr. Koch	R101
2	Anna Müller	B2	Physik	Dr. Meyer	R102
3	Ben Schulze	A1	Mathe	Dr. Koch	R101

- Tabelle muss 1NF sein -> erfüllt
- Alle Nicht-Schlüsselattribute müssen voll funktional abhängig vom gesamten Primärschlüssel sein

Problem:

- Primärschlüssel = (Student_ID, Kurs_ID)
- -> Aber Student_Name hängt nur von Student_ID ab, nicht vom Kurs!
- -> Verstoß gegen 2NF

Lösung: Aufteilen in zwei Tabellen

Studenten

Student_ID	Student_Name
1	Anna Müller
2	Ben Schulze

Kursbelegung

Student_ID	Kurs_ID
1	A1
1	B2
2	A1

Kurse

Kurs_ID	Kurs_Name	Dozent	Raum
A1	Mathe	Dr. Koch	R101
B2	Physik	Dr. Meyer	R102

Dritte Normalform

1. Muss in 2NF sein
2. Keine transitiven Abhängigkeiten (Nicht-Schlüsselattribute dürfen nicht voneinander Abhängen)

In der Kurs-Tabelle: Raum hängt eigentlich vom Dozent ab ab (z.B. Dr. Koch unterrichtet immer in Raum R101) -> transitives Abhängigkeit.

Lösung: Dozenten in eine eigene Tabelle auslagern

Kurse

Kurs_ID	Kurs_Name	Dozent_ID
A1	Mathe	D1
B2	Physik	D2

Dozenten

Dozent_ID	Dozent	Raum
D1	Dr. Koch	R101
D2	Dr. Meyer	R102

Aufgaben

Die Datenbank, in der Angebote und Buchungen gespeichert werden sollen, muss die folgenden Anforderungen erfüllen:

- Jede Wohnung wird nur von einem Anbieter bereitgestellt
- Für eine Wohnung werden zu verschiedenen Buchungszeiten verschiedene Preise verlangt
- Ein Kunde kann mehrere Wohnungen buchen
- Zu jeder Buchung gehören An- und Abreisedatum
- PLZ und Ort der Kunden und Anbieter werden in einer gemeinsamen Tabelle angelegt

Geben Sie ein Datenbankschema in der 3. Normalform an, das die genannten Anforderungen erfüllt. Außer den angegebenen Attributen brauchen Sie nur die Schlüsselattribute sowie den Namen der Kunden und Anbieter anzugeben. Kennzeichnen Sie Primär- und Fremdschlüssel mit PK bzw. FK.

Wohnung

ID (PK)	Anbieter (FK)

Anbieter

ID (PK)	Name	Ort (FK)

Kunde

ID (PK)	Name	Ort (FK)

Preis

Wohnung (FK)	Wohnung (FK)	Beginn	Ende	Preis

Buchung

Kunde (FK)	Wohnung (FK)	Abreisedatum	Anreisedatum

Ort

ID (PK)	PLZ	Ort

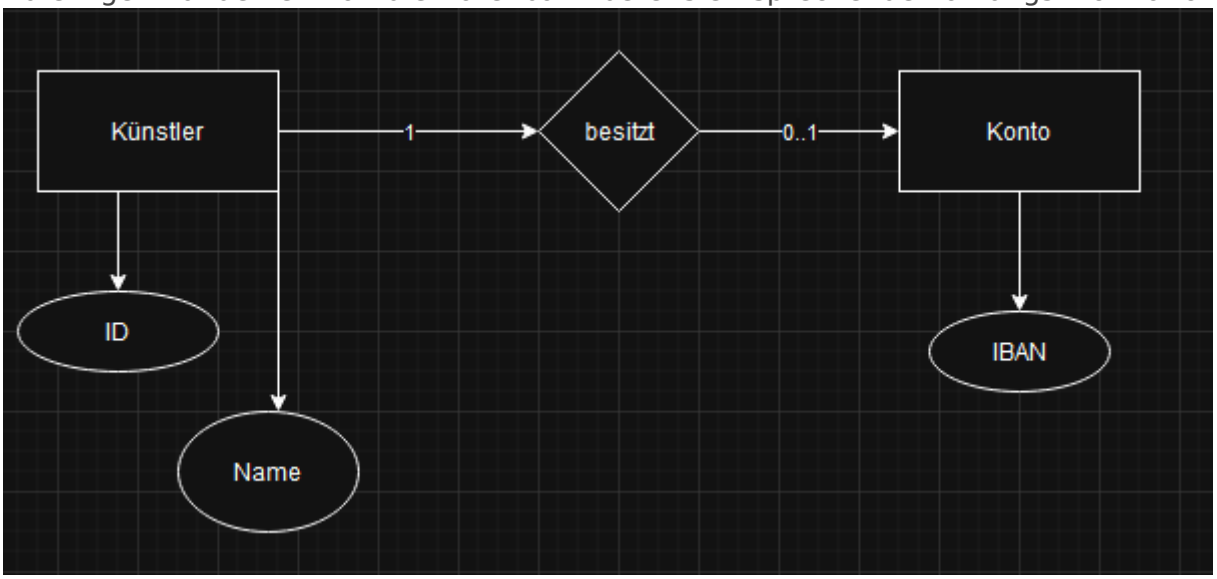
Begründen Sie, warum die Adresse des Kunden gemäß der 1. Normalform nicht nur als ein einziges Attribut dargestellt werden sollte.

Die gesamte Adresse in einem Feld wäre nicht Atomar. Dieses Problem löst die erste Normalform. Es sind Eingabefehler bei der Adresse nicht auszuschließen.

Erklären Sie, warum bei der Implementierung der Datenbank für die An- und Abreise der Datentyp "DATE" anstelle eines Zeichenkettentyps verwendet werden sollte.

Der Datentyp stellt sicher, das sich nur Datumswerte im einheitlichen Format innerhalb dieses Feldes befinden. Über DATE ist eine chronologische Sortierfolge definiert, über Zeichenketten eine alphabetische. Dies führt bei Vergleichen und Sortierungen zu unterschiedlichen Ergebnissen.

Zu einigen Kunden enthält die Datenbank bereits entsprechende Zahlungsinformationem:



Eine solche Relation kann in einer relationalen Datenbank wahlweise in einer oder in zwei Tabellen dargestellt werden. Geben Sie beide Formen an.

2NF

ID (PK, FK)	Name	IBAN

3NF

Konto

ID (PK)	Name	KontoID (FK)

Konto

KundenID(FK)	IBAN

Welche Vor- oder Nachteile hat die Darstellung als getrennte Tabellen bei einer 0...1 Relation im Hinblick auf folgende Aspekte

- Speicherplatz
- Performant
- Datenschutz
- Datensicherheit

Speicherplatz:

- Nachteil: Zusätzliche Tabelle erforderlich.

Performanz:

- Nachteil: Zusätzliche Verknüpfung der Tabellen mit Abfragen erforderlich

Datenschutz:

- Vorteil: Separate Kontotabelle kann mit anderen Zugriffsrechten ausgestattet werden. Dazu dürfte es aber kein FK in der Kundentabelle geben, da dieser Aufschluss über die Kontodaten bieten würde.

Datensicherheit:

- Vor- und Nachteile bei der Datensicherheit: höherer Speicherbedarf auch bei Datensicherungen, aber besserer Zugriffsschutz möglich.

Was ist bei einer SQL-Anweisung zu beachten, die aus zwei getrennten Tabellen die Namen aller Kunden und, falls vorhanden, deren Konto ermitteln soll?

Eine solche SQL-Anweisung muss JOIN-Anweisungen (OUTER) enthalten, um die Tabellen miteinander zu Verknüpfen über den Fremdschlüssel

Kunden wechseln gelegentlich ihre Kontoverbindungen. Geben Sie an, wie sich das ER-Modell ändert, wenn auch frühere Kontoverbindungen weiter gespeichert werden sollen.

- Die 0...1 Relation ändert sich in eine 1:n Relation
- Konto wird um ein Attribut erweitert, der den Aktiven Datensatz markiert.

Zunächst haben nicht alle User der Datenbank dieselben Rechte. Erläutern Sie, welche Zugriffsrechte bei einer Datenbank vergeben werden können

Es können Lese, Lösch und Schreibberechtigungen, sowie den Zugriff auf bestimmte Tabellen eingeschränkt werden. Gleiches Gillt für Prozeduren oder Views.

Alle Daten werden unter der Berücksichtigung der referentiellen Integrität eingepflegt. Erklären Sie dieses Prinzip und seine Auswirkung auf die Dateneingabe.

Das Prinzip der referentiellen Integrität verlangt, dass es zu jedem Fremdschlüsselwert, der in eine Datentabelle eingepflegt wird, bereits einen Primärschlüsselwert in der entsprechenden Mastertabelle gibt. Dies führt dazu, dass zunächst die Datensätze der Mastertabelle erstellt werden müssen und anschließend der Detailtabellen.

Manche Kunden Löschen ihre Konten wieder. Erläutern Sie welche Maßnahmen ergriffen werden müssen, um Löschanomalien zu vermeiden.

Es müssen alle Datensätze auch aus allen weiteren Tabellen mithilfe deren Primär- oder Fremdschlüsseln gelöscht werden, sodass keine Referenz mehr besteht.

Auch die verwendeten COMMIT- und REVOKE-Anweisungen tragen zur Sicherheit bei. Erläutern Sie dies.

COMMIT und REVOKE bewirken die sog. Transaktionssicherheit. Darunter versteht ,an, dass bestimmte Anweisungen nur zusammen oder gar nicht ausgeführt werden dürfen. Sollte also die Verarbeitung mitten in einer Transaktion abbrechen, müssen die bereits ausgeführten Transaktionen mittels REVOKE wieder rückgängig gemacht werden.

Alle SQL-Anweisungen sind sorgfältig getestet worden. Beispielweise wurde für die folgende Anweisung, die alle Wohnungen in Italien ermitteln soll, zur Laufzeit als fehlerhaft erkannt:

SELECT * FROM Wohnung WHERE Ort = (SELECT ID FROM Ort WHERE Land="Italien")

Beschreiben Sie den Unterschied zwischen einem Laufzeitfehler und einem syntaktischen Fehler.

Ein syntaktischer Fehler wird durch den Interpreter bzw. Compiler entdeckt und führt dazu, dass die betroffene Anweisung gar nicht erst ausgeführt werden kann. Bei einem Laufzeitfehler dagegen wird eine Anweisung zunächst ausgeführt, die Ausführung scheitert aber, z.B. weil die vorgefundenen Daten nicht zu den in einer Anweisungen getroffenen Annahmen passen.

Der Laufzeitfehler tritt nur auf, wenn die Datenbank mehrere italienische Wohnungen enthält. Was bedeutet das für die Entwicklung entsprechender Testfälle?

Bei der Entwicklung von Testfällen müssen insbesondere Grenzwerte Beachtet werden, z.B. dass tz demselben Land kein, nur ein oder mehrere Datensätze vorhanden sind.

Worin liegt der Fehler?

Es muss IN statt = verwendet werden.

Dieselbe Abfrage kann auch folgendermaßen formuliert werden. Warum entsteht in der umformulierten Fassung kein Laufzeitfehler mehr?

SELECT * FROM Wohnung, Ort WHERE Wohnung.Ort = Ort.ID AND Land = "Italien";

Hier wird die Spalte Ort korrekt mit der Ortstabelle verknüpft, sodass für jede Wohnung genau ein Datensatz in der Ortstabelle zugeordnet ist.

Erläutern Sie den Unterschied zwischen einer Stored Procedure und einem Trigger

Eine Stored Procedure ist eine Prozedur die manuell ausgeführt werden muss, ein Trigger hingegen wird bei jedem Insert, Update oder Delete in eine bestimmte Tabelle ausgeführt.

Die ID der stornierten Wohnung wird der Stored Procedure als Parameter übergeben

1. **Erläutern Sie den Unterschied zwischen fakultativen und obligatorischen Parametern**
   - Fakultative Parameter sind Optional und können bei der Ausführung weggelassen werden je nach Anwendungsfall, Obligatorische Parameter müssen bei jeder Ausführung der Prozedur übergeben werden
2. **Erläutern Sie den unterschied zwischen Parametern und lokalen Variablen**
   - Parameter werden einer Prozedur oder Funktion übergeben, Variablen sind im Quelltext deklariert und können nur im Quellcode selbst bearbeitet werden