Fachhochschule Köln
Cologne University of Applied Sciences

Softwaretechnik

Fomuso Ekellem

WS 2011/12

Weiteren Verlauf der Vorlesung

- **31.10.2011(2 Std) OO Vorgehensmodelle, UML, Teamarbeit, Gruppenbildung,.**
- **07.11.2011(2,5Std) Projektvorstellung, Planungsphase**
- **14.11.2011(2 Std) Angebotspräsentation, Planungsphase**

- **28.11.2011(4 Std) Definitionsphase und Entwurfsphase**
- **05.12.2011(4 Std) Implementationsphase**
- **12.12.2011(4 Std) Test-, Abnahme-, und Einführungsphase**
- **19.12.2011(4 Std) Wartung- und Pflegephase**

- **16.01.2012(2 Std) Produkte und Recht**
- **23.01.2012(4 Std) Projektvorführung**
- **30.01.2012(4 Std) Dokumentationsvorführung**

Inhalt

- **Kurze Frage-Runde(Übung 1)**
- **Kurze Einführung in der OOP**
- **Objektorientierte Abstraktion**
- **Objektorientiertes Vorgehensmodelle**
- **UML Diagramme**
- **Projektpraktikum**
- **Teamarbeit**
- **Gruppenbildung****

Kurze Einführung in der OOP

- Objektorientierte Programmierung ist ein Programmierungsparadigma.
- Paradigma: Ein Beispiel, das als Muster oder Modell dient.
- Vier vorwiegend bekannte Haupttypen:
  - Prozedurale/Imperative
  - Logische
  - Funktionale
  - Objektorientierte
- Programme werden aus verschiedenen Objekten aufgebaut.
- Ein Objekt repräsentiert ein Individuum, identifizierbaren Artikel, Einheit, oder Rechtsträger, entweder real oder abstrakt, mit einer klar definierten Rolle bei der Problem-Domäne.
  - Materielle Dinge - wie ein Auto, Drucker, ..., Rollen - als Mitarbeiter, Chef, ..., Vorfälle - wie Flug-, Überlauf-, ..., Interaktionen - als Vertrag, den Verkauf, ..., Technische Daten - wie Farbe, Form, ...

Kurze Einführung in der OOP

- **Grundkonzept**
  - Identifikation von Objekten und Implementierung von Objekt-Typen in Klassen.
  - Zuweisung von Aufgaben zu diesen Objekten (Methoden).
  - Erzeugte Objekte von Klassen, kommunizieren mit anderen Objekten gleichen Typs durch Senden von Nachrichten über Methoden.
- Hier stehen die Daten (Eigenschaften) und nicht die Funktionen oder Prozeduren des Programms im Vordergrund .
- Die Daten werden in Objekten gekapselt (Information hiding), die auch über Funktionalitäten verfügen, um die Daten zu ändern.
- In anderen Worten: Ein Objekt hat die Verantwortung, zu wissen und die Verantwortung zu tun.
- In diesem Zusammenhang spricht man jedoch nicht von Funktionen, sondern von Methoden.

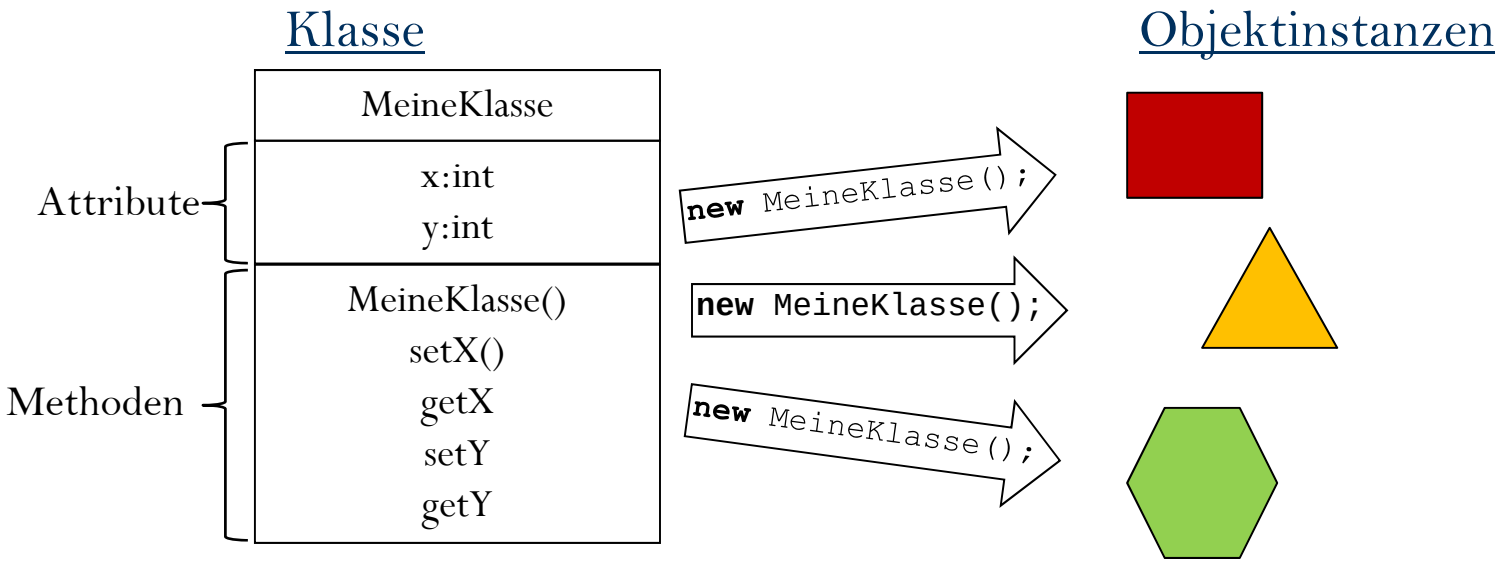
Kurze Einführung in der OOP

Warum Objekte?

- OOP erfüllt **mit der Unterstützung** von **Modularisierung**, **Wiederverwendbarkeit**, **Erweiterbarkeit**, **Abstraktion** und **Kapselung von Objekten** die zentralen Anforderungen, um die immer komplexere Anwendungslandschaft beherrschbar zu machen.
- **Modularisierung -** große Software-Projekte lassen sich in kleinere Stücke teilen.
- **Wiederverwendbarkeit -** Programme können aus bereits geschrieben Softwarekomponenten zusammengesetzt werden.
- **Erweiterbarkeit -** Neue Software-Komponenten können geschrieben oder aus bestehenden weiterentwickelt werden.
- Objektdaten **Abstraktion** bezeichnet den Prozess, eine Darstellung wesentlichen Attribute oder Merkmale herauszufinden und die unwesentlichen wegzulassen.
- **Datenkapselung**: Daten (Attribute) sind nicht sichtbar aber Methoden zur Manipulation der Attribute sind sichtbar. So kann nichts „Unsinniges“ mit den Daten passieren.

Kurze Einführung in der OOP

Objekte basieren auf Klassen, die einen Bauplan für ein Objekt und dessen Attribute(Daten) und Methoden festlegen.



Klassen fassen gleichartige Objekte zusammen. Objekte haben eindeutige Werte.

Kurze Einführung in der OOP

Zusammenfassung:

- Objekte = Daten + Methoden
- Daten (Attribute) = Eigenschaften
- Methode = Operationen rund um das Objekt
- Klassen (Objektkapsel) = Program-Bauplan (Implementierte Objekt in einer Programmiersprache)

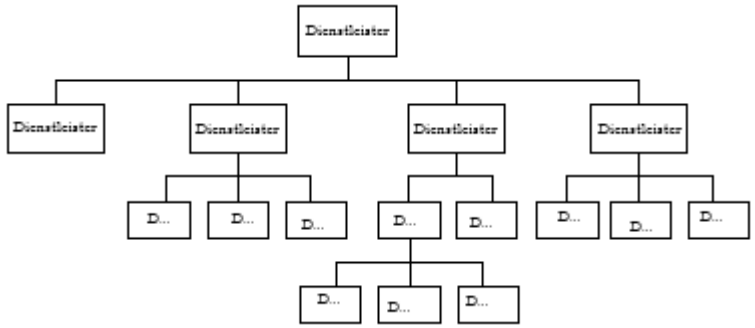
Objektorientierte Abstraktion

- Die Software auf Rechnern sind immer Abstraktionen realer Dinge
- Die Stellen Modelle der Wirklichkeit dar :
  - Zwei Abstraktionsbereiche: Operationen(Aktionen) und Daten
  - Aktivitäten im System: Folge von Operationen
- Daten sind Simulationen von konkreten oder abstrakten Dingen der Realität, über die Informationen gespeichert werden müssen
- Beide Arten der Abstraktion müssen in der Implementierungsumgebung abgebildet werden
- Abstraktion in andere Worte:
  - Verallgemeinerung,
  - Absehen vom Besonderen,
  - Modellbildung
- Man unterscheidet grob:
  - Funktionale Abstraktion und
  - Datenabstraktion

Objektorientierte Abstraktion

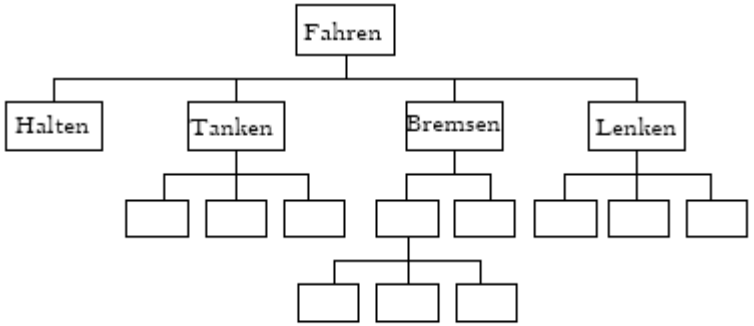
Funktionale Abstraktion:

- Versucht man ein mittelgroßes Programm zu schreiben, so versucht man das Programm in seiner Ganzheit in jedem Detail zu überblicken.
  - Sukzessive Zerlegung der Gesamtfunktion eines Programms in Teilaufgaben (*Top-Down Design)*
  - Die Zerlegung ist beendet, wenn die verbleibende Teilaufgabe leicht zu implementieren ist (*Konkretisierung*)
  - Jede Ebene verlässt sich auf die Dienstleistungen der niedrigeren Ebene
  - Funktionen als Dienstleister
  - Jedes Kästchen enthält eine Funktion mit starkem inneren Zusammenhalt: Eine genau definierte Operation.



Objektorientierte Abstraktion

- Beispiel



- Hierarchie wird gehalten
- Entwurfsphase
  - Beim Entwurf müssen Abstraktion und ihr Gebrauch in der Definition genau beschrieben werden
  - Bei Anwendung der Abstraktion erfolgt Informationsaustausch mit der „Umgebung“ über Ein- und Ausgabe-Parameter
  - Wie die Abstraktionen implementiert sind, ist für den Nutzer nicht wesentlich

Objektorientierte Abstraktion

Daten Abstraktion

- Objekte“ sind Gegenstand der Datenabstraktion.
  - Daten werden in Strukturen abgebildet und können durch bestimmte Operationen manipuliert werden
  - Schwerpunkt der Datenabstraktion sind die Operationen, die auf die Daten angewendet werden sollen
  - Details der Implementierung (der innerer Aufbau der Strukturen) bleiben wieder verborgen
  - Die Daten sind gekapselt.

Objektorientiertes Vorgehensmodelle

- OOP setzt Objektorientierte Analyse (OOA) und Objektorientierte Design(OOD) um in eine Implementierung(Programmierung)
- In OOA ist Wichtig: Identifikation der relevante Teile und Umsetzung der Requirement in Objektmodelle.
- OOD erweitert die Ergebnisse von OOA mit dem Fokus auf eine geplante Implementierung und deren Strukturierung.
- Zentrales Basiskonzept aller objektorientierten Analysemethoden ist ein objektorientiertes Klassenmodell
  - Klassen als gemeinsame Kapselungseinheit von Daten und Funktionen
  - Abbildung von Struktur und Verhalten einer Klasse durch Attribute und Operationen
- Ein wesentliches Merkmal ist wie bei allen Vorgehensmodellen, das Systemmodul entsteht in einem kontinuierlichen Prozess der schrittweise Verfeinerung (Iterativ-Inkrementelles Vorgehen)

Objektorientiertes Vorgehensmodelle

- Voraussetzung für die Durchführung des Vorgehensmodels:
  - Es liegt eine vollständige Anforderungsspezifikation(Lastenheft, Pflichtenheft, Systemglossar, Anforderung-Repository) vor
  - Die Anforderungen an die extern sichtbare Systemfunktionalität sind bereits in Form von Anwendungsfällen(Use Cases) beschrieben
- **Eigenschaften**
  - Anwendungsfallgetrieben, Architektur- und komponentenzentriert, Iterativ und Inkrementell

Objektorientiertes Vorgehensmodelle

Anwendungsfallgetrieben

- Funktionale Anforderungen an das zu entwickelnde System werden in Anwendungsfälle zerlegen
  - Aus Sicht der Anwender
  - Bilden Grundlage und Checks für alle weiteren Schritte
- Anwendungsfälle können priorisiert werden bzgl. der Reihenfolge der Umsetzung

Objektorientiertes Vorgehensmodelle

Architektur- und komponentenzentriert

- Eigenschaften und Besonderheiten einer vorhandenen Anwendungsarchitektur müssen berücksichtigt werden.
- Die Anwendungsarchitektur bestimmt, welche Artefakte überhaupt zu entwickeln sind.
- Beispiel einer Anwendungsarchitektur ist die Verwendung eines Anwendungs-Frameworks
- Systeme können in Komponenten und Subsysteme gegliedert sein. Auch das ist zu berücksichtigen
- Die Vorgehensweise sollte also die Erarbeitung der durch die Architektur vorgegebenen Artefakte optimal unterstützen

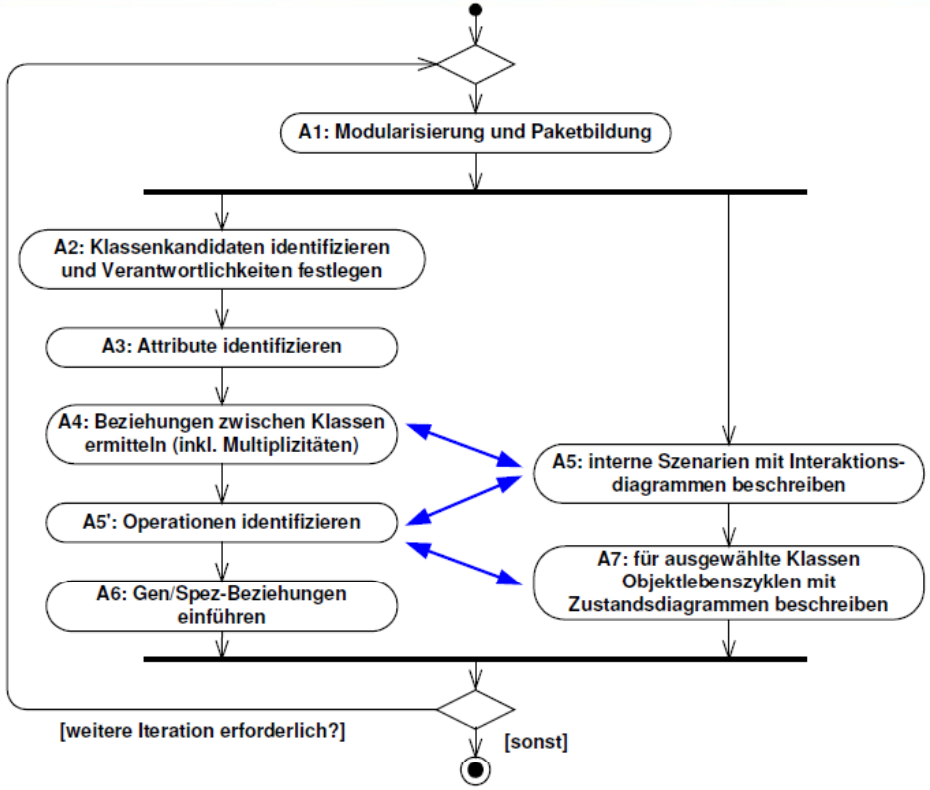
Objektorientiertes Vorgehensmodelle

Iterativ und Inkrementell

- Das Model wird in mehreren Iterationsschritten(nach und ggf. nebeneinander) erstellt und dabei permanent verbessert und verfeinert.
  - Zunächst Anwendungsfälle: Grobe Anforderungsbeschreibung.
  - Dann ggf. Bildung von Subsystemen
    - Subsysteme können von separaten Teams realisiert werden
  - Teilergebnisse der Teams müssen regelmäßig synchronisiert werden
    - Zwischenergebnisse werden in Reviews validiert und verifiziert
  - Jedes Team legt zu festgelegten Zeiten Ergebnisse vor:
    - Gliederung des Entwicklungsprozesses jedes Teams in Iterationen
    - Jede Iteration liefert ein Teilergebnis
- Inkrementell: Gesamtfunktionalität wächst mit jedem Schritt
- Jedes Modell-Inkrement wird vor seiner Freigabe geeigneten qualitätssichernden Maßnahmen unterzogen.

Objektorientiertes Vorgehensmodelle

- Ein einfaches Vorgehensmodell: *In der LifeCycle werden wir mehr sehen…*



UML Diagramme

Objektorientierte Modelle beschrieben durch UML

- UML (Unified Modeling Language) ist eine standardisierte Sprache und Notation zur Darstellung, Konstruktion, Dokumentation und Spezifikation von (objektorientierten) Modellen.
- UML verwendet verschiedene Diagrammtypen zur Darstellung unterschiedlicher Aspekte eines Systems
  - Statische Aspekte
  - Dynamische Aspekte
  - Benutzeranforderungen an das System
  - Implementierungsbezogene Aspekte

UML Diagramme

Objektorientierte Modelle beschrieben durch UML

- Modellierung der statischen Aspekte
  - **Klassen-**und **Objektdiagramme**: Darstellung von Klassen, Objekten und deren statischen Beziehungen
- Modellierung der dynamischen Aspekte
  - Verhaltensdiagramme: Beschreiben das Verhalten von Objekten des Systems
    - **Sequenzdiagramme:** Darstellung von Nachrichtenfolgen zwischen Objekten einer oder mehrerer Klassen
    - **Kollaborationsdiagramme:** Beschreiben das Zusammenwirken von Objekten beim Ausführen von Operationen
    - **Zustandsdiagramme:** Darstellung der Zustände und Zustandsübergänge der Objekte einer Klasse

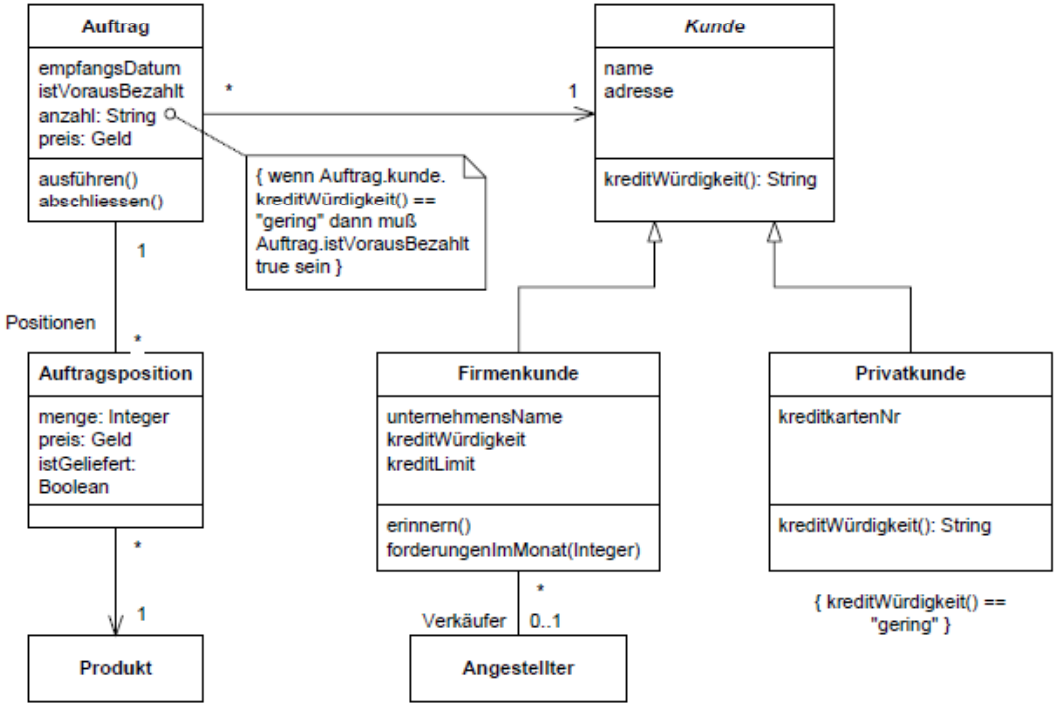
UML Diagramme

Objektorientierte Modelle beschrieben durch UML

- Modellierung der funktionalen Anforderungen an das System
  - **Anwendungsfalldiagramme**: Darstellung der Interaktionen zwischen externen Objekten (Akteuren) und dem System
- Modellierung implementierungsbezogener Aspekte (**Komponentendiagramme, Einsatzdiagramme**)

UML Diagramme

- **Beispiel UML Struktur**



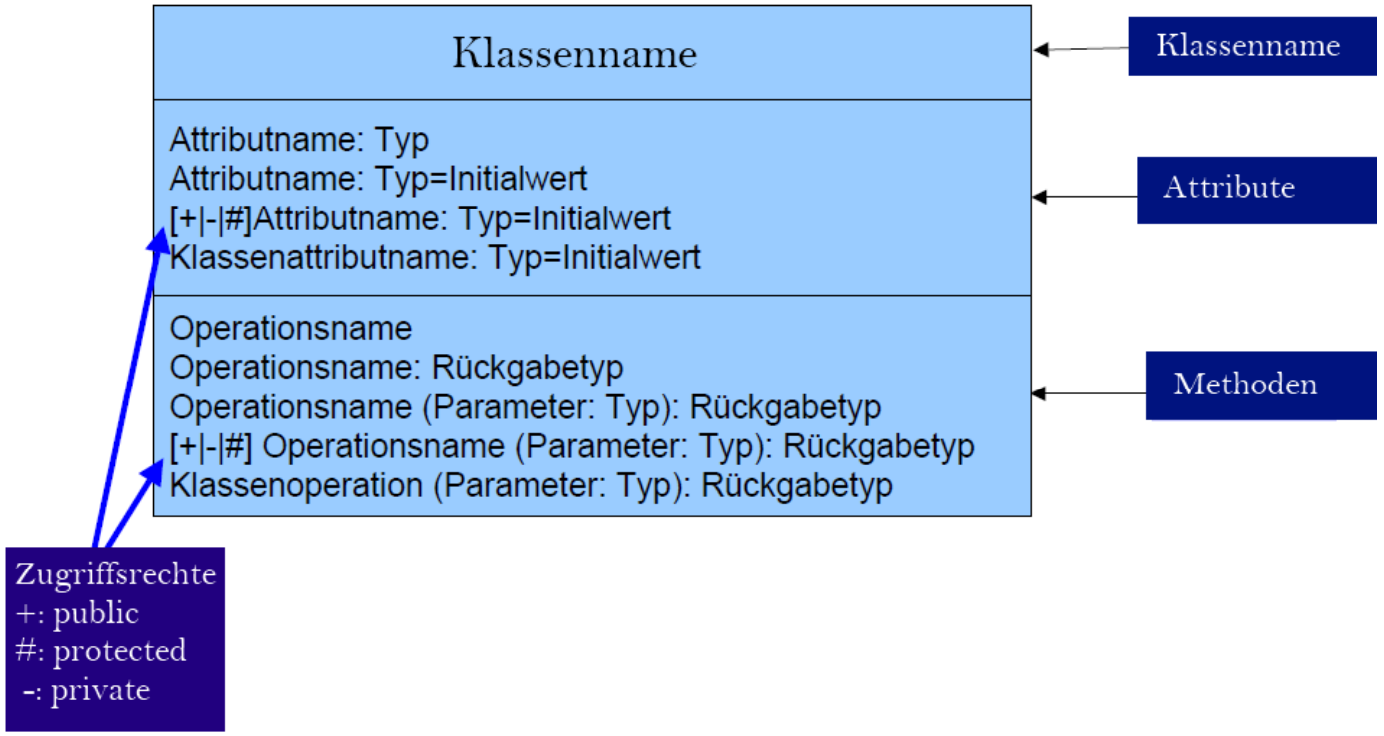
UML Diagramme

Klassendiagramme

- Erstellung von Klassendiagrammen ist zentrale Technik innerhalb aller objektorientierten Methoden
- Für eine Klassendiagramme gibt es unendlich viele Objekt-Diagramme
  - Klassen haben Attribute und Objekte haben Werte.
- Jede Objektinstanz hat grundsätzlich eine eigene Identität!
- Klassen werden durch Rechtecke dargestellt
- Klassenname muss eindeutig sein

UML Diagramme

- **Klassendiagramme**



UML Diagramme

Beziehungen:

- Beziehungen zwischen Klassen und Objekte
- Beziehungen zwischen Klassen
- Beziehungen zwischen Objekte
- *Wir werden in der Vorgehensphasen (LifeCycle) detailliert sehen…*

Projektpraktikum

- Bildung von Teams(3 Gruppe), die ein Software selbständig bearbeiten.
- Teams als kleine Firma(Auftragnehmer)

  Fomuso Ekellem als Kunde(Autraggeber)
  - Anforderungsermittlung mit Kunde
  - Angebotserstellung und Präsentation
  - Termingerechte Fertistellung des Software_Produkts (9 Wochen)
- Ziele:
  - Vorbereitung der Studierenden auf die industrielle Praxis
  - Fachkenntnise vertiefen
  - Praktische Erfahrung sammeln in (Teamarbeit, methodische Softwareentwicklung, Projektmanagement, Konfigurationsmanagement und Qualitätssicherung)

Projektpraktikum

- **Teamarbeit**
  - Erlernen undErleben von Teamarbeit im Studium
  - Treffen gemeinsamer Entscheidungen
  - Umgang mit Kritik
  - Selbstständige Teamorganisation
    - Teamleiter
    - Projektmanager
    - Entwickler
    - Qualitätssicherer
    - Konfigurationsmanger
  - Kommunikation im Team
  - Unterschiedlicher Kenntnisstand der Teammitglieder
  - Unterschiedliche Arbeitsweisen

Projektpraktikum

Softwareentwicklung

- **Auswahl von Vorgehensmodell**
- **Analyse (Objektorientierte Analyse) und Entwurf (Objektorientierte Design) in UML-Notation**
- **Implementierung (OOP) in C++**

Projektpraktikum

Projektmanagement

- Projektplanung
- Projektkontrolle
- Präsentation
  - Angebotspräsentation
  - Projektvorführung
  - Dokumentation Vorführung

Projektpraktikum

Konfigurationsmanagement

- Einweisung der Teammitglieder
- Schnittstellenverwaltung

Projektpraktikum

Qualitätssicherung

- Reviews
- Komponenten und Systemprüfungen
  - Planung
  - Durchführung
  - Auswertung
  - Dokumentation
- Konstruktive Qualitätssicherungsmaßnahmen

Projektpraktikum

- **Die Risiken und Chancen immer vor Augen führen**

Chancen	Risiken
• Benötigtes Wissen kann durch Kombination unterschiedlicher Kompetenzen erreicht werden • Komplexe und dynamische Aufgaben können besser gelöst werden • Gemeinsame Entscheidungen werden besser akzeptiert • Bei schwierigen Aufgaben kann soziale Unterstützung gegeben werden • Arbeitsmotivation und Engagement von Mitarbeitern kann gesteigert werden • Arbeitszufriedenheit kann erhöht und Stress reduziert werden	• Gruppe kann mehr Wert auf Harmonie als auf kritische Auseinandersetzung legen • Gruppe kann sich gegen externe Informationen und Einflüsse abschotten • Soziale Anforderungen an die Gruppenmitglieder können zu hoch sein • Zu viel Zeit und Energie kann für die Koordination der Arbeit benötigt werden • Destruktive Konflikte können innerhalb der Gruppe und mit anderen organisationalen Einheiten entstehen • Mitglieder können die Lust verlieren