universität innsbruck

Objektorientierung

Programmiermethodik

Lukas Kaltenbrunner, Simon Priller
Universität Innsbruck

Motivation

- Was wurde bisher besprochen?
  - Datentypen, Anweisungen, Arrays, Grundlagen der Methoden
- Was kann man damit machen?
  - Algorithmen implementieren
  - Kleinere Programme schreiben
- Große Programme?
  - Ja, aber nicht sinnvoll
  - Problem der Komplexität
- Komplexität
  - Schwierigkeiten
    - Komplizierte Algorithmen
    - Komplexe Software, d.h. viele aber meist einfache Funktionen
    - Umfangreicher Sachverhalt, der abgebildet werden muss
  - Probleme bei der Softwareentwicklung
    - Viele Funktionen müssen organisiert werden
    - (Großes) Team muss organisiert werden
    - Anforderungen an die Software können sich ändern

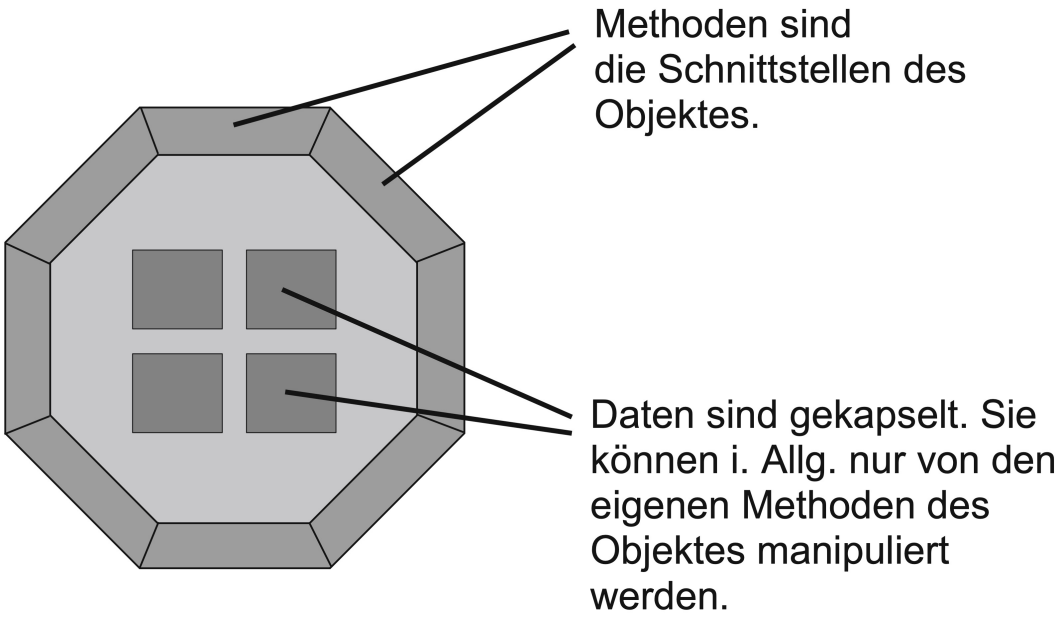
Komplexität in den Griff bekommen

- Beachten von verschiedenen Prinzipien
  - Ein abgeschlossener Codeteil (Modul) sollte nur eine Verantwortung (Aufgabe) haben.
  - Wiederholungen vermeiden → DRY-Prinzip!
  - Trennung der Schnittstelle von der Implementierung!
  - Testbarkeit
  - etc.
- Techniken der Objektorientierung unterstützen diese Prinzipien.
- **ABER:**
  - Objektorientierung führt nicht automatisch zu besserer Software.
  - Objektorientierung unterstützt auch nicht alle Prinzipien vollständig!

Objekt (1)

- Eigenschaften von Objekten:
  - Identität
    - Jedes Objekt hat eine unveränderliche Identität.
    - Nachrichten an Objekte werden über ihre Identität geschickt.
  - Zustand
    - Der Zustand eine Objekts setzt sich aus der aktuellen Belegung der Datenfelder zusammen.
  - Verhalten
    - Das Verhalten eines Objekts gibt an, wie sich ein Objekt beim Empfangen einer Nachricht verhält.
- Ein Objekt ist ein Verbund von Operationen die sich einen Zustand teilen.
  - Operationen bestimmen auf welche Nachrichten ein Objekt reagieren kann.
  - Direkter Zugriff auf den Zustand ist von außerhalb nicht möglich.
  - Der Zustand wird durch Variablen repräsentiert (= Objektvariablen).
  - Der Zustand kann nur von den Operationen des Objekts verändert werden.
  - Durch die Operationen wird das Verhalten des Objekts beschrieben.

Objekt (2)



- Objekte gehen einen Kontrakt ein, der die Rahmenbedingungen beim Aufruf einer Operation regelt.
- Die Rahmenbedingungen sind:
  - Vorbedingungen
  - Nachbedingungen
  - Invarianten

Abbildung übernommen aus „Java als erste Programmiersprache“, Goll & Heinisch

Objektorientierung

- Objektorientierung
  - Ist ein Programmierparadigma für die Analyse, den Entwurf und die Implementierung von objektorientierten Systemen.
  - Objektorientierte Programmierung erlaubt eine natürliche Modellierung vieler Problemstellungen.
  - Die objektorientierten Prinzipien können durch eine objektorientierte Implementierung in unterschiedlichem Maß eingehalten werden.
- Grundprinzipien der Objektorientierung
  - Kapselung
  - Polymorphie
  - Vererbung

Objektorientierte Konzepte

- Klassenkonzept:
  - Deklaration von Klassen
  - Klassen beschreiben Eigenschaften von Objekten dieser Klasse
- Prototypkonzept:
  - Beschreibung einzelner Objekte
  - Neue Objekte werden durch Klonen und Abändern bestehender Objekte erzeugt

Klassenkonzept

Klassen

- Ein Objekt ist bei objektorientierten Programmiersprachen, welche auf dem Klassenkonzept basieren, immer zumindest einer konkreten Klasse zugeordnet.
- Eine Klasse:
  - Beschreibt Gemeinsamkeiten (Eigenschaften und Operationen) von Objekten
  - Definiert einen Datentyp
  - Dient als Konstruktionsplan für Objekte des Typs der Klasse
  - Ist ein elementares Modellierungswerkzeug
- Konzepte objektorientierter Programmiersprachen mit Klassenkonzept:
  - Abstraktion
  - Kapselung
  - Beziehungen
  - Polymorphie

Abstraktion

- Der Prozess der Abstraktion hilft essenzielle Details herauszuheben, unwichtige Details zu ignorieren und die Komplexität der resultierenden Programme zu reduzieren.
- Trennung zwischen Konzept und Umsetzung
- Realisiert durch Klassen und Objekte
- Objekte sind tatsächlich existierende Dinge aus der Domäne des Programms
- Klassen sind Beschreibungen eines oder mehrerer ähnlicher Objekte. Sie beschreiben mindestens:
  - Wie ist ein Objekt der Klasse zu bedienen?
  - Welche Eigenschaften hat ein Objekt der Klasse?
  - Wie verhält sich ein Objekt der Klasse?
  - Wie wird ein Objekt der Klasse hergestellt?

Kapselung

- Variablen und Methoden werden in einer logischen Einheit, dem Objekt, gekapselt.
- Daten und Methoden gehören zum Objekt
  - Daten gehören explizit einem Objekt
  - Methoden repräsentieren das Verhalten des Objekts
  - Methoden sollten die einzige Möglichkeit sein um mit dem Objekt interagieren zu können
  - Direkter Zugriff auf Daten ist nicht erlaubt.
- Kapselung hilft dabei
  - den Aufwand bei Änderungen der zugrundeliegenden Datenstruktur eines Objekts gering zu halten.
  - die Daten konsistent zu halten, da ein direkter Zugriff auf die Daten nicht erlaubt ist.

Beziehungen

- Vererbung („is-a”-Beziehung)
  - Beziehung zwischen ähnlichen Klassen
    - Abbildung von Beziehungen zwischen Klassen durch Hierarchien aus Klassen und Unterklassen.
    - Objekte der Hierarchie teilen die Spezifikation bzw. Spezifikation und Implementierung gewisser Operationen.
- Aggregation und Komposition („part-of”-Beziehung)
  - Zusammensetzung eines Objekts aus anderen Objekten.
  - Komposition bezeichnet die strenge Form der Aggregation auf Grund einer existenziellen Abhängigkeit.
- Verwendungs- und Aufrufbeziehungen
  - Verwendung anderer Klassen bzw. Objekte als
    - Temporäre Variable
    - Typ eines formalen Parameters

Polymorphie

- Polymorphie = Vielgestaltigkeit
- Die Vielgestaltigkeit bezieht sich bei Programmiersprachen auf Variablen und Methoden.
  - Eine Variable kann Objekte unterschiedlicher Klassen aufnehmen.
  - Eine Methode kann mit aktuellen Parametern, die unterschiedliche Typen aufweisen, aufgerufen werden.
- Verschiedene Formen der Polymorphie
  - Universelle Polymorphie
    - Generizität
    - Untertypen
  - Ad-hoc Polymorphie
    - Überladen
    - Typumwandlung

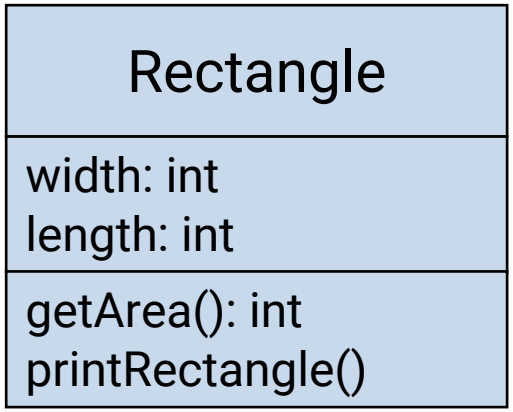
Klassen & Objekte

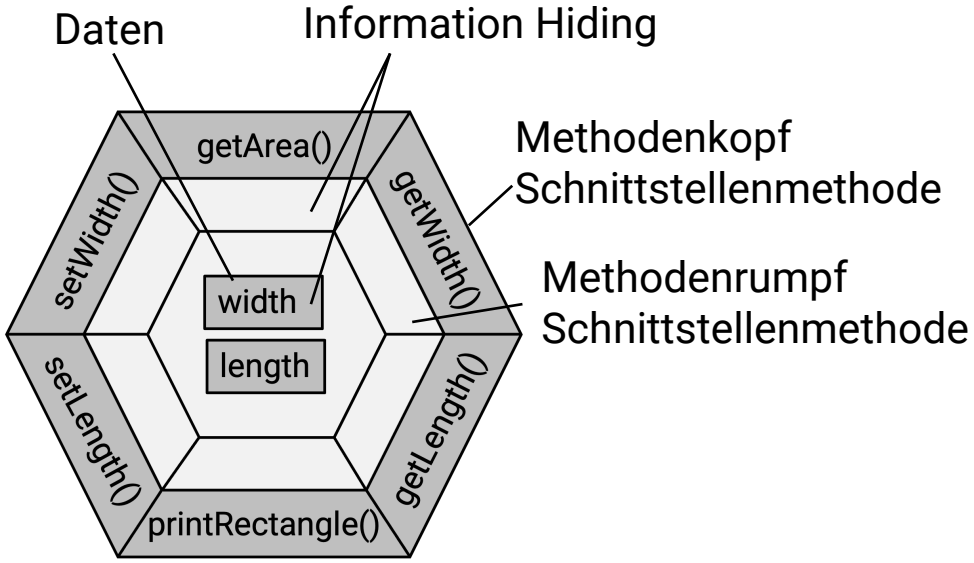
Klassen und Objekte (1)

- Objekte sind Exemplare der Klassen zu denen sie gehören.
- Es können mehrere Exemplare derselben Klasse existieren.
- Objekte können Exemplare mehrerer Klassen sein.
- In den meisten Programmiersprachen sind Objekte nur Exemplare einer Klasse.
- Jedes Exemplar hat einen eigenen Zustand.
- Exemplare teilen die Implementierung der Operationen.
- Das Verhalten von Objekten derselben Klasse kann sich aufgrund des internen Zustands unterscheiden.
- Interaktion mit einem Objekt kann nur über die bereitgestellten Operationen erfolgen (=Kapselung, engl. encapsulation)

Klassen und Objekte (2)

- Beispiel: `Rectangle`-Objekt

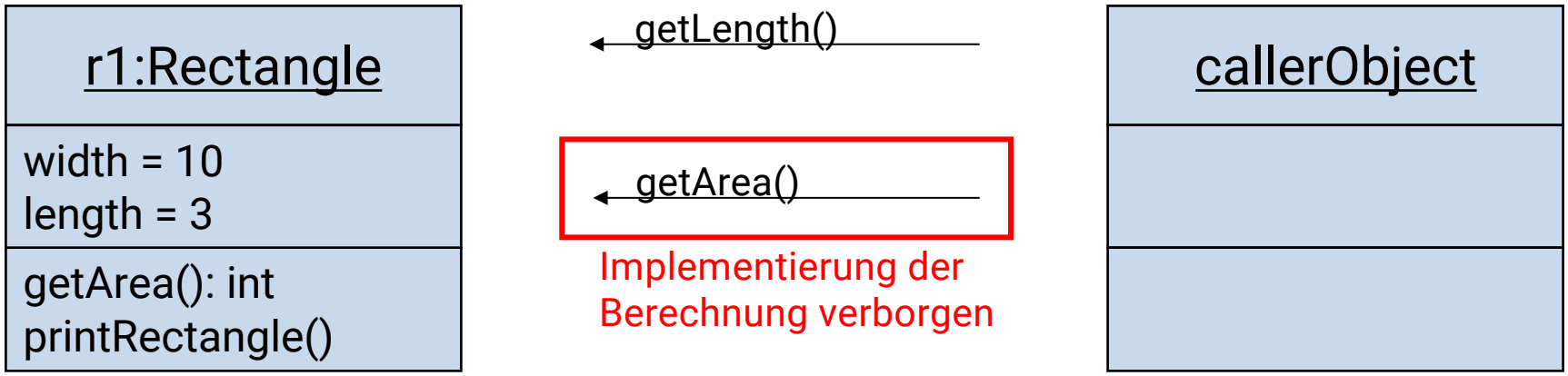




- Geheimnisprinzip (information hiding)
  - Die Daten einer Kapsel, die Rümpfe der Schnittstellenmethoden und die Hilfsmethoden sollen nach außen nicht direkt sichtbar sein.

Abbildung angelehnt an „Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik“, Goll

Kollaboration von Objekten



Durch die Kapselung verschmelzen die Daten und Methoden eines Objektes (externe Sicht).

Typsysteme

Typsysteme

- Klassen legen den Typ für ihre Exemplare fest.
  - Das Typsystem kann dadurch Eigenschafen von Objekten und auf sie anwendbare Operationen erkennen.
- Das Typsystem ist Bestandteil der Umsetzung einer Programmiersprache oder ihres Laufzeitsystems.
- Das Typsystem ordnet jedem Ausdruck einen Typ zu.
- Überprüfung von Ausdrücken auf Verträglichkeit mit dem Typsystem. Beispielsweise:
  - Ist die vorliegende Operation auf dem Objekt erlaubt?
  - Kann das Objekt der vorliegenden Variable zugewiesen werden?
- Unterscheidung bei Typsystemen
  - Statisches vs. dynamisches Typsystem
  - Starkes vs. schwaches Typsystem
  - Namensbasiertes vs. strukturbasiertes Typsystem
  - …

Statisches Typsystem

- Eigenschaften
  - Typ von Variablen und Parametern wird im Quelltext deklariert.
  - Schränkt ein, welche Objekte einer Variable zugewiesen werden können.
  - Compiler erkennt schon eine unpassende Zuweisung oder unpassenden Aufruf einer Operation.
- Vorteile
  - Programmstruktur ist übersichtlicher.
  - Entwicklungsumgebungen können mehr Unterstützung anbieten.
  - Fehler können früher erkannt werden.
  - Kompilierte Anwendungen können besser optimiert werden.
- Beispiele: Java, C++, C#

Dynamisches Typsystem

- Eigenschaften
  - Variablen sind keinem deklarierten Typ zugeordnet.
  - Variablen können beliebige Objekte referenzieren.
  - Ob eine Operation auf ein Objekt erlaubt ist, wird zur Laufzeit überprüft.
- Vorteile
  - Flexibilität
  - Keine Notwendigkeit einer expliziten Typumwandlung!
- Klassen und Typen sind entkoppelt.
- Beispiele: Smalltalk, Ruby, Python, PHP, JavaScript

Stark und schwach typisierte Programmiersprachen

- Stark typisiert
  - Überwacht Zugriff auf Objekte
  - Variable kann nur auf Objekt verweisen, welches die Spezifikation ihres Typs erfüllt.
- Schwach typisiert
  - Keine Überwachung
  - Variable kann auch auf Objekt verweisen, welches die Spezifikation ihres Typs nicht erfüllt.

Quellen

- Bernhard Lahres, Gregor Rayman, Stefan Strich: **Objektorientierte Programmierung: Das umfassende Handbuch**, Rheinwerk Verlag, 5. Auflage, 2021
- Joachim Goll, Cornelia Heinisch: **Java als erste Programmiersprache**, Springer Vieweg, 8. Auflage, 2016
- Guido Krüger, Heiko Hansen: **Handbuch der Java-Programmierung**, Addison Wesley, 7. Auflage, 2011
- Christian Ullenboom: **Java ist auch eine Insel: Einführung, Ausbildung, Praxis**, Rheinwerk Verlag, 16. Auflage, 2022
- Joachim Goll: **Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik**, Springer Vieweg, 2018