Handout
- Objektorientierung Java –

Allgemein:

- Wiederverwendbarkeit spielt große Rolle
- !! schauen, was gibt es schon (Bibliotheken, …) → erst Google befragen, dann selbst implementieren
- Klassen bestehen aus
    - Attributen (was das Objekt hat)
    - Operationen (was das Objekt kann)

Objekte:

- Erzeugen von Objekten mit Befehl *new*
- Konstruktoren, etc. analog c
- gibt Grabage-Collector → gibt automatisch Objekte wieder frei
- this (analog wie c) Zugriff auf Objektvariablen

Eigenschaft	Sieht eigene Klasse	Sieht Klasse im gleichen Paket	Sieht Unterklasse im anderen Paket	Sieht Klasse in anderem Paket
public	ja	ja	ja	ja
Protected	ja	ja	ja	nein
paketsichtbar	ja	ja	nein	nein
private	ja	nein	nein	nein

Statische Eigenschaften:

- statische Methoden (static) → Werte aus Parametern arbeiten nicht mit Objektzustand → Konstanten
    - http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_003.htm (5.3.1 nachlesen)
- Zugriff über Klassenname nicht Referenzvariable
    - http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_003.htm (Beispiel 5.3.3)
- „Shared Memory“ → mehrere Objekte der gleichen Klasse greifen auf Konstanten zu
- public static final <Datentyp> <name> = <Wert> -- konstante Variable

Beziehung zwischen Objekten:

- http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_007.htm
    - Durchlesen → mit Beispielen besser verständlich

Vererbung:

- Schlüsselwort extends → Unterklasse
- alle sichtbaren Eigenschaften der Oberklasse an Unterklasse vererbt
- Ober- und Unterklasse im gleichen Paket → erbt auch paketsichtbaren Eigenschaften

- protected: an alle Unterklassen vererbt + Klassen im gleichen Paket können alle protected-Eigenschaften sehen
    - public > protected > paketsichtbar > private
- Konstruktoren werden nicht vererbt! → Abhilfe mit super() – erste Zeile im Konstruktor – ABER: automatisch vom Compiler, also optional
    - http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_008.htm (5.8.6 Beispiele anschauen)

Typen in Hierarchien:

- http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_009.htm (selbst Lesen; ist zu viel zum Zusammenfassen)

Methoden überschreiben:

- Unterklassen können Methoden der Oberklasse überschreiben → selber Name, Parameter und Datentyp
- Code dann modifizierbar(z.B. Verbesserung durch Nutzung Eigenschaften, die nicht in Oberklasse bekannt)
- Attribute können nicht überschrieben werden!!!! nur überlagert

Schnittstellen:

- Mehrfachvererbung in Java nicht möglich → Interfaces
- Unterklasse kann dann Oberklassen und beliebig viele Schnittstellen beziehen
- Schnittstelle hat keine Implementierung → deklariert nur den Kopf einer Methode (kein Konstruktor!!)
- *Hinweis Der Name einer Schnittstelle endet oft auf -ble (Accessible, Adjustable, Runnable). Er beginnt üblicherweise nicht mit einem Präfix wie »I«, obwohl die Eclipse-Entwickler diese Namenskonvention nutzen.*
- Schlüsselwort implements: public class xyz implements uvw
- Polymorphiebeispiel anschauen: http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_013.htm#mjfac44aa7eeb458fddc00bde8886194c9 (5.13.4)

!!! Ich lege euch sehr ans Herz Kapitel 5 aus dem Buch komplett nochmal selbst durchzulesen !!!

http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_05_001.htm#mj005cd8e604aefc3ae72b92880fcee5c6