LEHR- UND FORSCHUNGSEINHEIT
PROGRAMMIERUNG &
SOFTWARETECHNIK

Softwaretechnik 2015/2016

PST Lehrstuhl

Prof. Dr. Matthias Hölzl

Joschka Rinke

AGENDA

Übung 12:
21.01.2016

- **Schon zur Klausur angemeldet?**
- **Fragen**
- **Besprechung Blatt11**

Aufgabe 1 – Objektorientiertes Design II

Aus statischen & dynamischen Teilmodell Objektentwurf ermitteln.

Statisches Analysemodell wird erweitert → Objektentwurf:

- **Operationen hinzufügen**
- **Assoziationen ausrichten**
- **Zugriffsrechte bestimmen**
- **Mehrfachvererbung auflösen**
- **Wiederverwendung von Klassen**
- **Algorithmen beschreiben**

Aufgabe 1 – Objektorientiertes Design II

Operationen hinzufügen:

- **Für jede empfangene Nachricht Operation hinzufügen (Sequenzdiagramm)**
- **Für jedes Call-Event & jede lokale Operation eine Operation hinzufügen (Zustandsdiagramm)**
- **Benutzerdefinierte Konstruktoren hinzufügen**
- **Zugriffsoperationen hinzufügen**

Assoziationen ausrichten:

- **Wird Assoziation nur in eine Richtung durchlaufen, dann ausrichten**

Zugriffsrechte:

- **Attribute & Rollennamen sollten nicht öffentlich sein**

Mehrfachvererbung auflösen

- **Notwendig, wenn Zielsprache Mehrfachvererbung nicht unterstützt**

Wiederverwendung von Klassen:

- **Evtl. Spezialisierung notwendig**
- **Kapselungsprinzip beachten**

(nicht-triviale) Algorithmen beschreiben

- **Aktivitätsdiagramm oder**
- **Pseudocode**

Aufgabe 1 – Objektorientiertes Design II

Eventuell zusätzliche Elemente in den Klassen → vgl. Operationen hinzufügen

- **Benutzerdefinierte Konstruktoren**
- **Zugriffsoperationen**

Aufgabe 1 – Objektorientiertes Design II

socrative by MasteryConnect

Was könnte das Diamond-Problem sein?

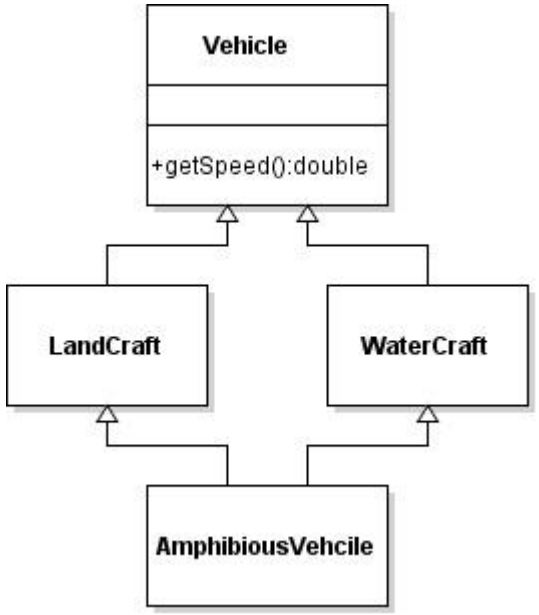
- **Das Verhalten einer Klasse, die auf unterschiedlichen Vererbungspfaden von ein und derselben Klasse erbt ist nicht immer klar**
- **Eine Klasse, die von mehreren Klassen erbt kann gleichnamige Attribute besitzen wodurch nicht immer eindeutig ist, welches Attribut wann gemeint ist**
- **Für eine Klasse die mehrere Oberklassen hat ist in der Klassenhierarchie unklar welche Klasse Vaterklasse ist**
- **Die Kinder einer Oberklasse können nicht von einander erben**

Aufgabe 1 – Objektorientiertes Design II

Diamond-Problem:
Klasse erbt auf zwei unterschiedlichen Vererbungspfaden von ein und derselben Basisklasse

➜ Höchstgeschwindigkeit variiert je nach Terrain, welche Rückgabe macht Sinn?

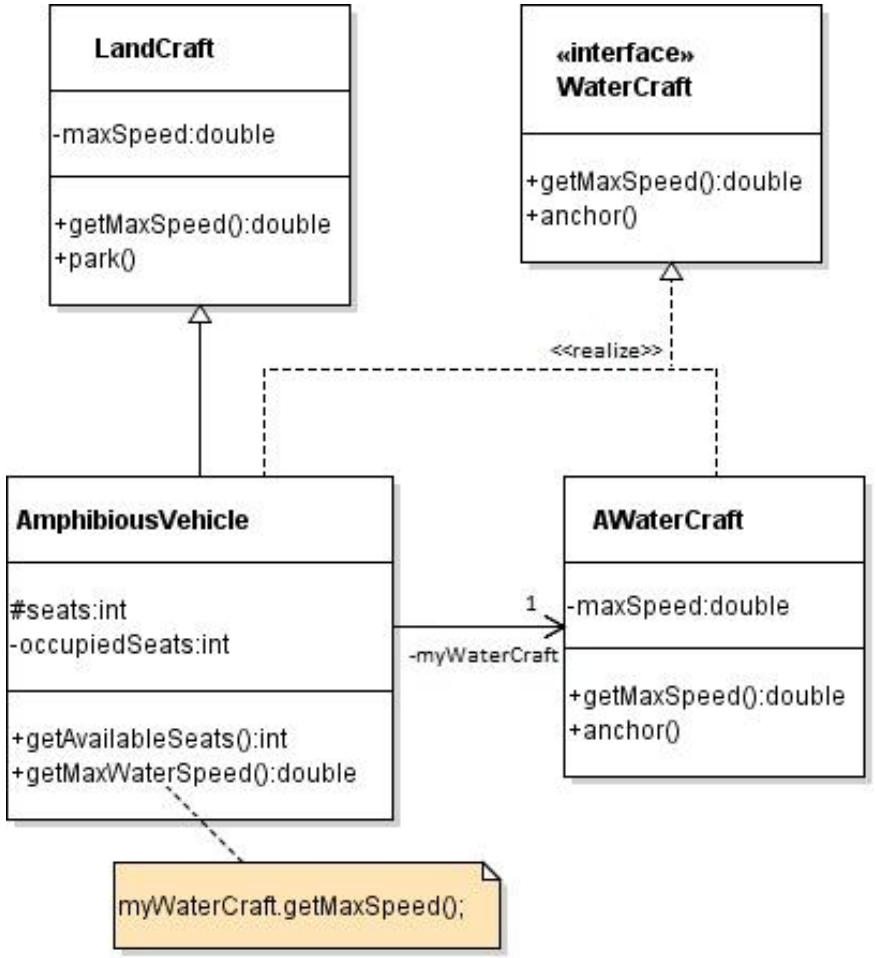
Mehrfachvererbung auflösen:
➜ Interfaces verwenden



Aufgabe 1 – Objektorientiertes Design II

Mehrfachvererbung in Java → Auflösen durch Verwendung von Interfaces

1. **Interface definieren**
2. **Interface realisieren durch: Erbende & implementierende Klasse**
3. **Assoziation herstellen Erbend → Realisierend**
4. **Operation hinzufügen (in erbender Klasse)**



Aufgabe 1 – Objektorientiertes Design II

Warum ist Wiederverwendung durch Delegation sinnvoll?
Wenn möglich vorhandene Klassen wiederverwenden, dabei könnte das Kapselungsprinzip verletzt werden (Zugriffsoperationen werden nicht benötigt)
→ Wiederverwendung durch Delegation

d.h. keine Spezialisierung erstellen, sondern Assoziation hinzufügen

Manchmal ist es sinnvoll der delegierenden Operation
das aufrufende Objekt als Übergabeparameter zu übergeben

Aufgabe 1 – Objektorientiertes Design II

Warum ist es manchmal sinnvoll der delegierenden Operation das aufrufende Objekt als Übergabeparameter zu übergeben?

- **Dann ist immer klar wieso die Operation ausgeführt wird**
- **Dann hat das Objekt zu dem delegiert wird eventuell benötigte Informationen des Objekts, dass die Operation angestoßen hat**
- **Dann kann man leicht zurückverfolgen welches Objekt eine Operation angestoßen hat**
- **Das macht überhaupt keinen Sinn**

Aufgabe 2 – Realisierung von Zustandsdiagrammen

- **Prozedurgesteuert: Ereignisse durch modale Dialoge steuern**
  - **— nur für Objekte an der Systemgrenze möglich**
  - **— nicht sehr flexibel**
- **Fallunterscheidung: ENUM für Zustände**
  - **— schlechte Erweiterbarkeit bzgl. neuer Zustände (neue Fälle in jeder zustandsabhängigen Operation)**
  - **+ gute Erweiterbarkeit bzgl. neuer Operationen**
- **Zustandsobjekte: Objekte der Klasse mit Zustandsobjekten verbunden**
  - **— schlechte Erweiterbarkeit bzgl. neuer Operationen**
  - **+ gute Erweiterbarkeit bzgl. neuer Zustände (neue Klasse)**
- **Zustandsmaschine: Größen aus Zustandsdiagramm in Klassen darstellen**
  - **— komplex zu implementieren**
  - **+ sehr variabel**

Aufgabe 2 – Realisierung von Zustandsdiagrammen

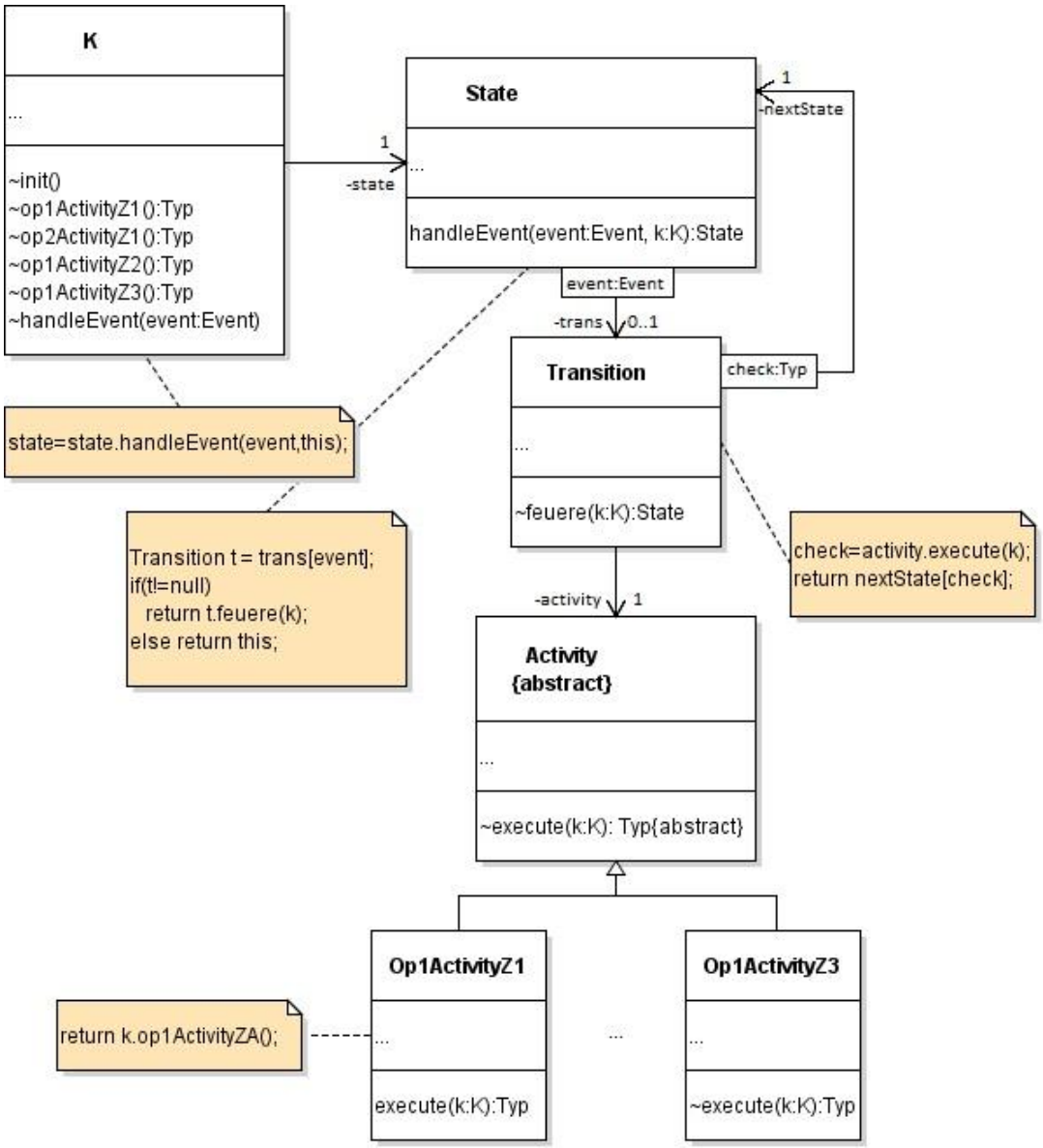
Zustandsmaschine erstellen:
Klassen für alle Größen aus Zustandsdiagramm, d.h.:

- **Klasse K inklusive Operationen op1ActivityZ1(), …, op1ActivityZ3()**
- **Klasse State**
- **Klasse Transition**
- **Abstrakte Klasse Activity**
- **Klassen Op1ActivityZ1 … Op1ActivityZ3**

Zusätzlich nach dem Schema aus der VL:

1. **Assoziationen hinzufügen und ausrichten (immer gleich)**
2. **Bedingungen hinzufügen**
3. **Restliche Operationen hinzufügen**

Aufgabe 2 – Realisierung von Zustandsdiagrammen



Aufgabe 3 – Anbindung an relationale DB

Vererbung in relationaler Datenbank:

- **Tabelle pro Klasse:**
  ggf. müssen mehrere Tabellen berücksichtig werden
- **Tabellen der Unterklassen enthalten geerbte Attribute:**
  Bei Änderungen an der Oberklasse müssen auch Tabellen der Unterklassen geändert werden
- **Tabelle für alle Ober-/Unterklassen:**
  evtl. müssen Nullwerte eingetragen werden

Aufgabe 3 – Anbindung an relationale DB

Assoziation abbilden:
Primärschlüssel von CarPool in Vehicle

Vererbung abbilden:
Unterschiedliche Varianten, also z.B. alles in eine Tabelle schreiben

carpoolKey	country	address
…	…	…

vehicleKey	numberOfWheels	ps	carpoolKey	numberOfSeats	type	color
…	…	…	…	…	…	…

Aufgabe 3 – Anbindung an relationale DB

Verbindung mit DB herstellen:

1. **JDBC Treiber laden:**

```
static final String JDBC_DRIVER = "imaginary.mysql.jdbc.Driver";
//…
Class.forName(JDBC_DRIVER);
```

2. **Verbindung herstellen:**

```
Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);
```

3. **Anfrage formulieren:**

```
Statement stmt = conn.createStatement();
String sql = "SELECT id FROM CarPool";
ResultSet rs = stmt.executeQuery(sql);
```

4. **Attribute extrahieren:**

```
while(rs.next()){
  int id = rs.getInt("id");
  //...
}
```

Aufgabe 3 – Anbindung an relationale DB

socrative by MasteryConnect

Welches Pattern kommt zum Einsatz um die Treiber zu laden?

- **Reflection**
- **Template**
- **Observer**
- **Visitor**

Reflection →Class.*forName(…);*