# Software Composition Paradigms

## Sommersemester 2015

Radu Muschevici

Software Engineering Group, Department of Computer Science

TECHNISCHE
UNIVERSITÄT
DARMSTADT

2015-06-30

# Cohesion

- How can code cohesion be improved in the following Java class `Flight` that is part of a larger model of an airline system?

```
class Flight {
    FlightNr flightNr;
    Airport departure;
    Airport destination;
    AircraftType aircraft;
    SeatMap seatMap;

    public FlightNr getFlightNr() {...}
    public Airport getDeparture() {...}
    public Airport getDestination() {...}
    public AircraftType getAircraft() {...}
    public SeatMap getSeatMap() {...}
}
```

# Subtyping

- Explain the **substitution principle** in the context of subtyping in object-oriented languages and give an example.

# Prototype-based Programming

- In the absence of classes and class inheritance, what is a mechanism for reusing code in a prototype-based language like Self? Describe briefly how the mechanism works.

# Mixins and Traits

- Scala linearisation exercise (see lecture)

# Mixins and Traits

- Define the **flattening** property of traits.

# Mixins and Traits

- Describe how **super** calls are resolved in the context of mixins in the Scala language. Contrast with **super** calls in Java in the context of single inheritance.

# Aspect-Oriented Programming

- Explain the following sentence:
  “Aspects make quantified statements about the behavior of programs.”

# Aspect-Oriented Programming

- A pointcut is a set of join points where advice can be inserted. Describe the kind of join points captured by the AspectJ pointcut:

```
call(void *.set*(..));
```

# Aspect-Oriented Programming

Complete the following sentence by inserting one of the choices given below.

Aspects are well-modularised ….

1. interfaces to data or behaviour
2. cross-tree constraints
3. cross-cutting concerns