Software Composition Paradigms

Sommersemester 2015

Radu Muschevici

Software Engineering Group, Department of Computer Science

TECHNISCHE
UNIVERSITÄT
DARMSTADT

2015-06-30 & 2015-07-07

Important: Exam Room Assignment

Updated on 9 July 2015:

Everybody please go to room **S101/A1 (Audimax)**.
Address: Karolinenplatz 5, 64289 Darmstadt

(There is no need to split up the class between two rooms anymore.)

- Reminder: the SCP exam is on **Wednesday, 15 July, starting at 19.00h** (duration: 90 minutes).
- The exam is **closed book**.
- Research well in advance how to find the building and room for the exam!

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

Metaprogramming & Reflection

- An computational system uses a class-based object-oriented language to specify the state and behaviour of its programs. Your task is to **design an object-oriented meta system** for this computational base system. How will your meta system represent classes of the base system? Give a brief justification.

Metaprogramming & Reflection

- In a reflective system, a meta program manipulates and modifies the base program. What can you say about the relation between base program and meta program?

Context-Oriented Programming

Consider the following context-oriented program excerpt.

```
class Hello {
    String msg() { return("Hello "); }
    layer A { String msg() { return(proceed() + "beautiful "); } }
    layer B { String msg() { return(proceed() + "wonderful "); } }
    layer C { String msg() { return("Good day world"); } }
    layer D { String msg() { return(proceed() + "world "); } }
}
```

What is the output of each of the print statements below?

```
Hello obj = new Hello();

with (A) { with (B) { with (D) { println(obj.msg()); } } }

with (B) { with (D) { with (E) { println(obj.msg()); } } }

with (A) { with (B) { with (C) { println(obj.msg()); } } }
```

Feature-Oriented Software Product Lines

- Software product line engineering distinguishes two engineering processes: **domain engineering** and **application engineering**. Describe the activities that each process entails.

Feature-Oriented Software Product Lines

- What is a **feature** in the context of Feature-Oriented Software Product Lines?

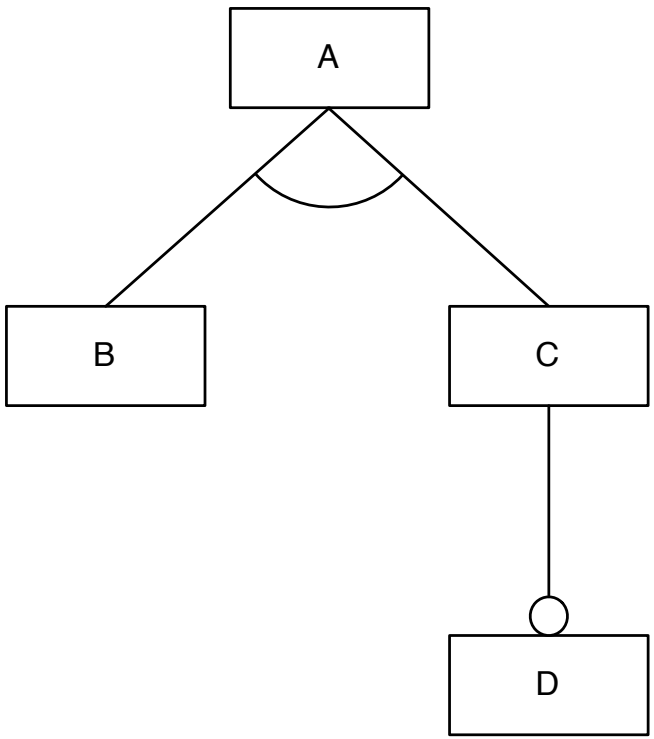
- What is the purpose of a **feature model**?

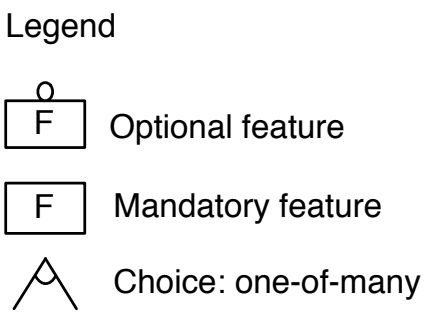
Feature-Oriented Programming

- Feature-Oriented Software Development aims to establish **feature traceability** between problem and solution space. Explain the reasons and benefits of this approach.

Feature-Oriented Programming

Consider the following feature diagram. List **all valid feature selections** defined by the underlying feature model.





Feature-Oriented Programming

- In FOP, a feature module corresponds to a feature and vice-versa. Why is this 1:1 mapping sometimes too rigid (not flexible enough)?

Delta-Oriented Programming

- **Deltas** in Delta-Oriented Programming and **feature modules** in Feature-Oriented Programming are similar concepts to encapsulate the behaviour of features. Explain the differences.

Delta-Oriented Programming

Consider a program in the ABS language. The two deltas `D1` and `D2` implement features F1 and F2 respectively. When both features `F1` and `F2` are selected, a conflict arises. This conflict is resolved by application of a third delta, `D3`. **Complete the following product line configuration to make sure that:**

- Delta `D1` is applied whenever feature `F1` is selected.
- Delta `D2` is applied whenever feature `F2` is selected.
- Delta `D3` is applied whenever both features `F1` and `F2` are selected.
- Delta `D3` is always applied after applying `D1` and `D2`.

```
productline MySPL;
features F1, F2;
delta D1 ...
delta D2 ...
delta D3 ...
```

Important: Exam Room Assignment

Updated on 9 July 2015:

Everybody please go to room **S101/A1 (Audimax)**.
Address: Karolinenplatz 5, 64289 Darmstadt

(There is no need to split up the class between two rooms anymore.)

- Reminder: the SCP exam is on **Wednesday, 15 July, starting at 19.00h** (duration: 90 minutes).
- The exam is **closed book**.
- Research well in advance how to find the building and room for the exam!