Lehrstuhl für Informatik 3
RWTH Aachen
Univ.-Prof. Dr.-Ing. Manfred Nagl,
Christian Fuß, Bodo Kraft

Modellierung von Software-Architekturen
Übung zur Vorlesung
SS 2006
11.05.2006

Übungsblatt 05

Aufgabe 11 (Textuelle Konsistenzbedingungen)

In der Vorlesung wurden Konsistenzbedingungen für Softwarearchitekturen eingeführt, die allein auf der Ebene der Architekturdiagramme nachprüfbar sind. Dies sind die Regeln 1 bis 9 in der Tabelle 4.37 im Buch.

Formulieren Sie diese so um, dass Sie sich nicht auf die Architekturdiagramme, sondern auf die textuelle Architekturnotation beziehen!

Aufgabe 12 *(Graphische Konsistenzbedingungen)*

In Abbildung 2-A.1 ist ein fehlerhaftes Architekturdiagramm angegeben, d.h. darin werden einige der o.g. Konsistenzbedingungen verletzt. Identifizieren Sie die Stellen, an denen Verletzungen bestehen, und erläutern Sie jeweils kurz, wieso welche Konsistenzbedingung verletzt ist.

Aufgabe 13 *(Vererbungshierarchie)*

In Abbildung 3-A.1 sehen Sie den Quellcode einer Java-Klassenhierarchie, spielen Sie System und schreiben Sie hinter jede System.out.println()-Zeile die Ausgabe. (Versuchen Sie es mal ohne das Programm abzutippen ;-))

Abgabe: Donnerstag, 18.05.2006

Sie können Ihre Lösung zum obigen Termin in Papierform in der Vorlesung, der Übung oder in elektronischer Form per eMail an pig@i3.informatik.rwth-aachen.de abgeben.
Bitte vermerken Sie in jedem Fall die Namen und Matrikelnummern aller beteiligten Personen (maximal 3), beim Versenden per eMail auch alle eMail-Adressen. Abgaben in elektronischer Form können ausschließlich in den Dateiformaten *Plain-Text oder PDF* erfolgen.

Aktuelle Informationen zur Vorlesung finden Sie auf den Webseiten des Lehrstuhls unter
http://www-i3.informatik.rwth-aachen.de.

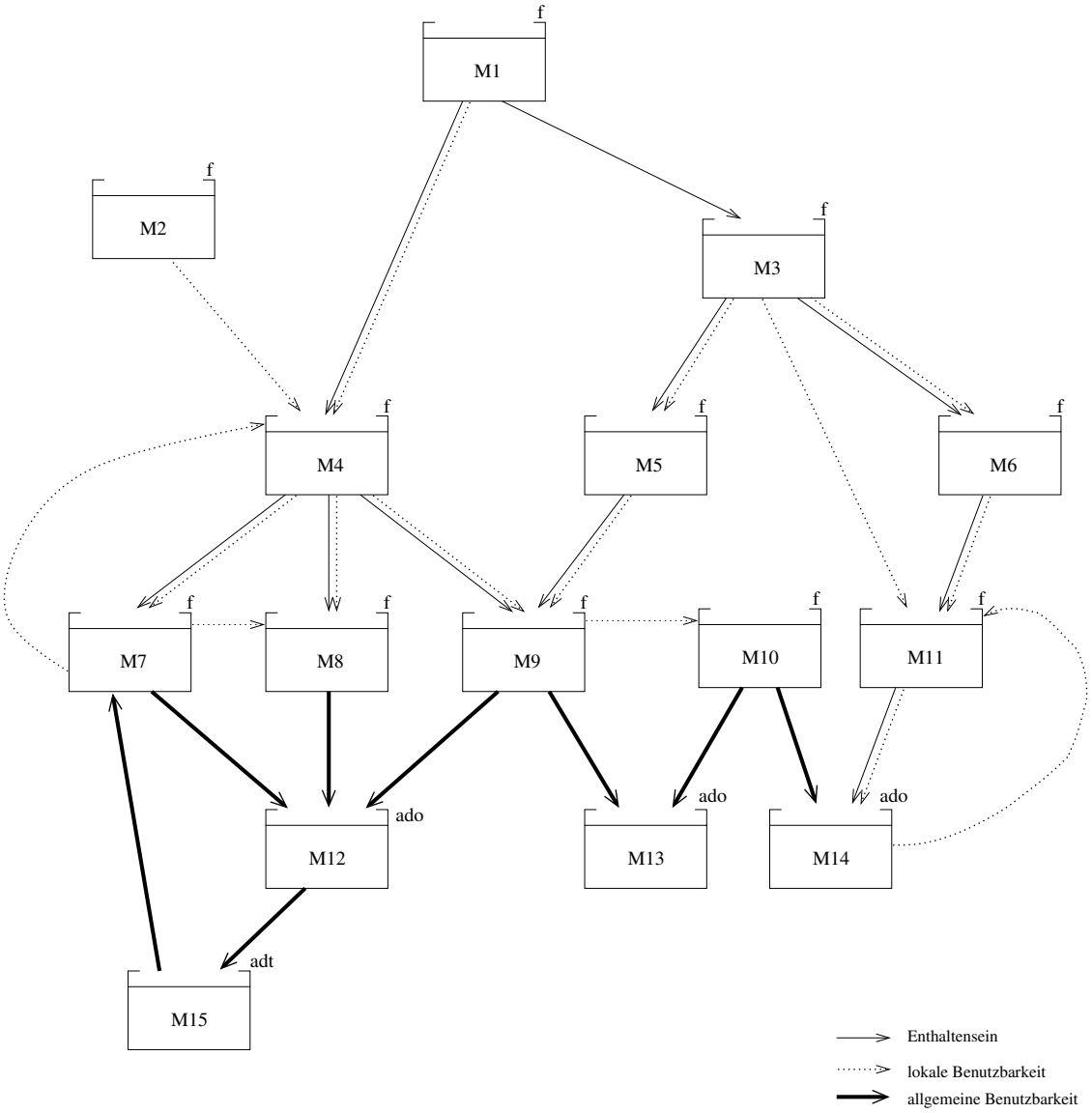

Abbildung 2-A.1: Fehlerhaftes Architekturdiagramm

```
public class Main {
    public static void main(String[] args) {
        Root l1 = new Level1(), l2 = new Level2(), l3 = new Level3();
        System.out.println("l1.getId(): " + l1.getId());
        System.out.println("l2.getId(): " + l2.getId());
        System.out.println("l3.getId(): " + l3.getId());

        System.out.println("l1.getFinalId(): " + l1.getFinalId());
        System.out.println("l2.getFinalId(): " + l2.getFinalId());
        System.out.println("l3.getFinalId(): " + l3.getFinalId());

        System.out.println("l1.getOverriddenId(): " + l1.getOverriddenId());
        System.out.println("l2.getOverriddenId(): " + l2.getOverriddenId());
        System.out.println("l3.getOverriddenId(): " + l3.getOverriddenId());
    }
}

class Root {
    protected String myId;

    public String getId() {
        return this.myId;
    }

    final public String getFinalId() {
        return getOverriddenId();
    }

    String getOverriddenId() {
        return "Root";
    }
}

class Level1 extends Root {
    {
        myId = "Level1";
    }
}

class Level2 extends Level1 {
    {
        myId = "Level2";
    }

    public String getId() {
        return "MyLevel2";
    }
}

class Level3 extends Level2 {
    {
        myId = "Level3";
    }

    String getOverriddenId() {
        return super.getOverriddenId() + "Level3";
    }
}
```

Abbildung 3-A.1: Vererbungshierarchie