

## FS 2015 - Software Engineering 2 – Prüfung vom 11. August 2015

Name:

Umfang/Erwartung:

- Die Prüfung umfasst 129 Punkte.
- Wir erwarten, dass Sie Aufgaben im Umfang von 120 Punkten lösen (ca. 1 Punkt/Minute Zeitbudget).
- Ca. 90% von 120 Punkten, also ca. 110 Punkte ergeben die Note 6.

Hilfsmittel :

- Folienkopien mit Vorlesungsnotizen ergänzt. Dazu darf ein Inhaltsverzeichnis angefertigt werden.
- Eigene Zusammenfassung der Vorlesung.
- **Keine alten Prüfungen und keine Übungen!**
- 1 A4-Seiten Zusammenfassung pro eLearning Album
- Keinerlei elektronische Hilfsmittel, wie Handys, Kameras, Tablets!

Eine gute Prüfung wünschen Marcel Huber, Daniel Keller, Farhad Mehta und Hans Rudin!

Nr.	Aufgabe	Punkte	erreicht
1	Projektplanung	10	4
2	Project Automation	8	6 1/2
3	Test Driven Development	6	5
4	Software Engineering Practices	10	7
5	Error Handling Design	10	6
6	Design by Contract	10	7
7	Software Architektur	10	4
8	Code Smells	12	6
9	Refactoring mit Refactoring to Patterns	11	3
10	Metriken	9	5
11	Reviews	5	3
12	Aufwandschätzung	2	0
13	Profiling, Performance	4	0
14	Agile Software Entwicklung	12	0
15	Funktionale Programmierung	6	0
16	Software Tatort	4	
	Total:	129	58 1/2
	Note:= 1 + 5*Punkte/ca. 110 <i>105</i>		3.76 => 4

## Aufgabe 1: Projektplanung (10 Punkte)

- a) (1P) Warum heisst es bei der Folie "Beispiel-Projekt: Zeitaufteilung" (Folie 6 in der Vorlesung Projektplanung) "Achtung: Prozent der Zeit  $\neq$  Prozent der Kosten"? d.h. warum sind die beiden Prozentzahlen allermeistens NICHT gleich?

Meistens wird in kurzer Zeit schon recht viel erreicht.  
80-20-Regel

- b) (2P) Use Cases können gut für das Abstecken des Funktionsumfanges (Definition of Scope) eingesetzt werden. Wann und wie machen Sie das am besten?

In der Elaboration des Projekts. Am besten die 3 Arten von Use Cases verwenden

used

- c) (3P) Welche der folgenden Tätigkeiten gehören/gehören nicht zur Vorbereitung auf den Checkpunkt 'End of Elaboration'? Bitte kreuzen Sie auf jeder Zeile das entsprechende Antwortfeld an. Jede richtige Antwort gibt einen halben Punkt, falsche Antworten geben 0P und keinen Abzug. Am Schluss werden Ihnen 3 Punkte abgezogen, weil man ja mit Raten 50% der Antworten (=3P) richtig hätte.

Wenn Sie beim Ausfüllen unsicher sind oder denken, dass für Ihre Antwort eine Anmerkung oder Randbedingung nötig ist, dann schreiben Sie eine Bemerkung dazu.

Tätigkeit	Gehört zu End of Elab.	Gehört NICHT zu End of Elab.
Einrichten der Entwicklungs- und Testumgebung		
Aufsetzen des Software Architecture Documents	X	
Erstellen eines Domainmodells	X	
Schreiben von Unit Tests		X
Usability Testing	X	
Performance Testing		
Erstellen eines Projektplanes		
GUI Entwürfe	X	
Demo von Prototyp(en)	X	
Aufsetzen von Remine/JIRA Tickets	X	
Schreiben von Use Cases	X	
Definition von Interfaces für grössere Subsysteme		

d) (2P) Geben Sie bitte zwei gute Beispiele für die Definition von nicht-funktionalen Anforderungen.

- Performance : Serverresponse Time bei Suche nach Produkt XY ✓  
dauert nicht länger als 50ms.
- Mengengerüst : Auf dem Server können 200 Besucher gleichzeitig sein. ✓

e) (1P) Auf Folie 18 heisst es "Vereinfacht ausgedrückt: - Use Cases = Dynamik - Domain-Modell = Persistenz". Machen Sie ein Beispiel, wo diese Vereinfachung nicht oder nicht ganz zutrifft.

f) (1P) Wozu dient ein Use Case Diagramm? Was kann man daraus gut herauslesen?

- geeignet für eine Übersicht verschiedener Operationen .
- Wie hängen verschiedene Abläufe zusammen? .

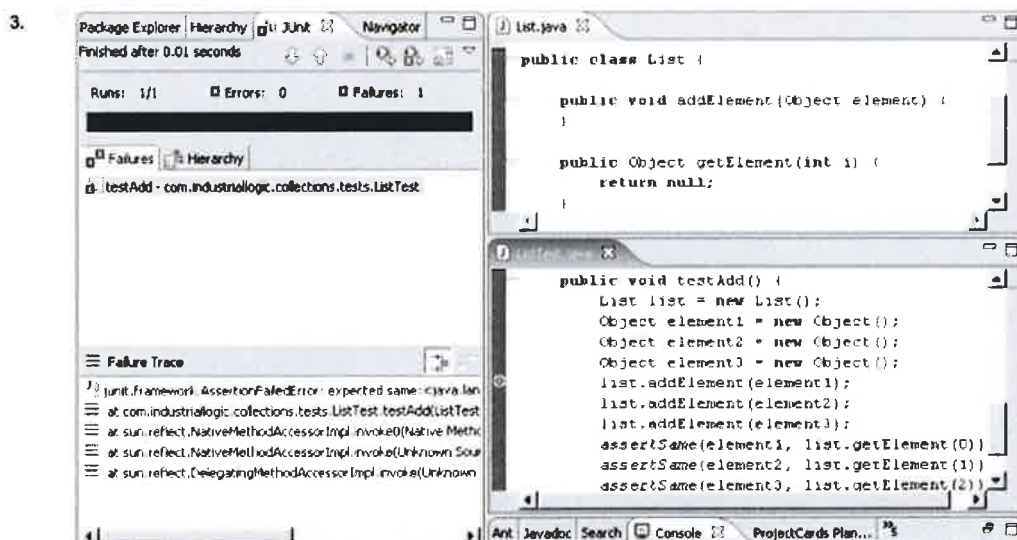
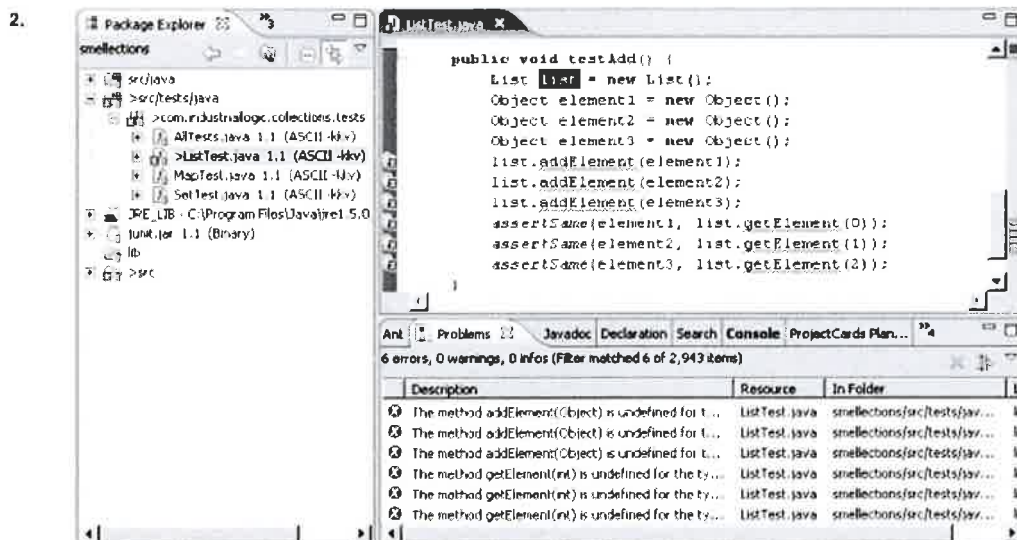
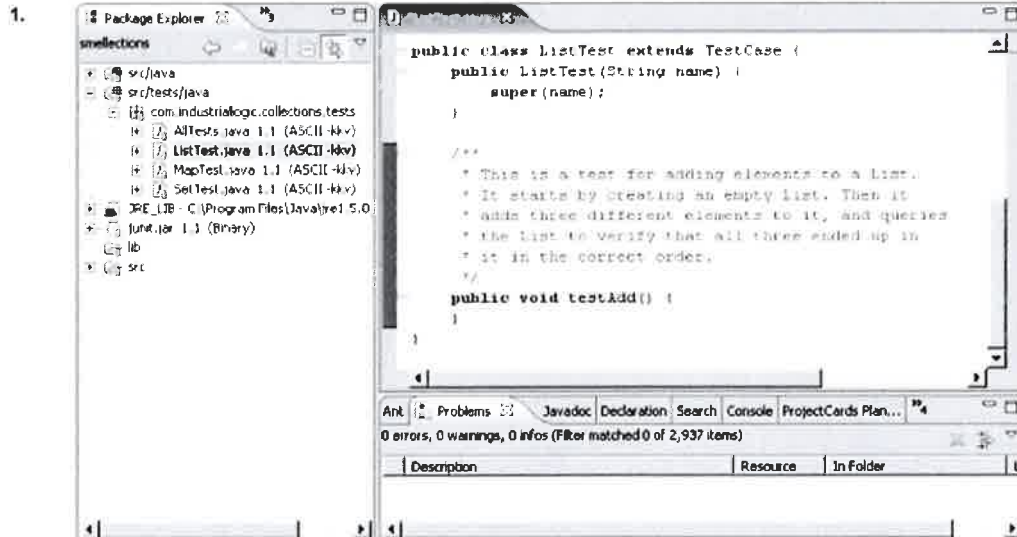
## Aufgabe 2: Projektautomatisierung und Continuous Integration (8 Punkte)

Schreiben Sie in kurzen Sätzen oder Stichworten mindestens **vier** Risiken auf, welche durch den Einsatz von Continuous Integration reduziert oder ausgeschlossen werden können. Nennen Sie dann zu jedem dieser Punkte mindestens eine Massnahme im Sinne von Continuous Integration, um das Risiko zu minimieren oder sogar auszuschliessen.

Risiko (je 1P.)	Massnahme/Umsetzung (je 1P.)
- Doppelten Code vorhanden ✓	- VersionControll einsetzen - Bsp. <del>Git</del> GitHub
- Programm nur auf „meinem“ Computer lauffar ✓	- CI Tool einsetzen um immer einen lauff Build zu haben - z.B. Jenkins ✓
- Fehler zu spät erkennen ✓	- Automatisierte Tests und Buildserver verwenden - Bsp. Jenkins ✓
- Projekt nicht sichtbar für alle ✓	- GitHub einsetzen, damit jeder Zugriff auf den Sourcecode hat. ~ 1/2

## Aufgabe 3: Test Driven Development (6 Punkte)

- a) (3P) Der erste Schritt im Refactoring-Zyklus ist der Schritt RED. Welche der folgenden Abbildungen repräsentiert diesen Schritt am besten (1). Erklären Sie wieso (2).



Es wurde ein Test geschrieben der kompiliert, jedoch fehlschlägt, da die zu testenden Funktionen noch nicht implementiert sind.

b) (2P) Wie lange sollte der Schritt GREEN im TDD-Zyklus ungefähr dauern (1) und wieso (1).

- Sollte nicht allzu lange dauern, da der Code so programmiert wird, dass der Test dazu nicht fehlschlägt. <sup>- d.h.</sup>
- Es wird kein grosser Wert auf Code-Design gelegt.

! ✓

c) (1P) Werden im Schritt Refactor produktiver Code oder Testcode verbessert?

- Ab und zu beides
- Meistens jedoch nur der Produktive Code

✓

## Aufgabe 4: Software Engineering Practices (10 Punkte)

- a) (6P) Halten Sie die untenstehenden Aussagen für gute Anforderungen? Geben Sie jeweils kurz Ihre Überlegungen bzw. Bedenken an.

„Die Applikation muss für Sehbehinderten auch benutzbar sein“

Gute Anforderung (Ja/Nein):

Nein ✓

Grund:

was genau b.

~ ok.

„Bei der Entwicklung wird SCRUM angewendet“

Gute Anforderung (Ja/Nein):

Ja X

Grund:

Dies ist ein Design Entscheid. welcher am Anfang des Projekts festgelegt wird. ... und..

„Das System hat ein Verfügbarkeitsziel von 95%“

Gute Anforderung (Ja/Nein):

Ja

Grund:

Eine klar definierte Zahl, welche messbar ist.



- b) (2P) Nennen Sie die zwei wichtigsten Gründe, warum die Erhebung von nicht-funktionalen Anforderungen (NFA) wichtig ist.

Grund 1:

Um zu definieren, für welche Last ein System ausgelegt ist.  
Damit der Benutzer zufrieden ist.

Grund 2:

Um sich abzusichern und dem Kunden zu zeigen was „abgemacht“ wurde.

! Es gibt wichtigere Gründe die  
im Vorlesung besprochen worden sind

~

- c) (2P) Bei welchem Prozentsatz an Test-Coverage (d.h. Kodezeilen, die aus den Tests erreichbar sind) kann man sicher sein, dass der Code fehlerfrei ist? Begründen Sie Ihre Antwort.

Prozentsatz:

—

Grund:

Man kann dies nicht sagen. Auch 100% Test Coverage bedeutet noch lange nicht das der Code fehlerfrei ist.

✓



## Aufgabe 5: Error Handling Design (10 Punkte)

a) (8P) Bitte vervollständigen Sie folgende Tabelle:

System	Fehler- behandlungs- technik K=Konservativ O=Optimistisch	Begründung und Bemerkungen
Internet- Suchmaschine (Beispiel)	O	Verfügbarkeit wichtig. Ungültige Sucheingaben sollen das System nicht ausser Betrieb nehmen können.
Eisenbahn- Stellwerk	K ✓	Kann zu Personenunfällen führen falls Fehler nicht korrekt behandelt ✓
Präsenzkontroll- System für eine Schule	O ✓	Verfügbarkeit wichtig, falls jemand einen Fehler verursacht dürfen die anderen dadurch nicht verhindert werden ✓
Email Newsletter Verwaltungs- Software	O ✓	Nicht sicherheitskritisch falls ein Newsletter mal nicht verschickt werden kann. Verfügbarkeit... ok
E-Banking System	O ✗	Muss hochverfügbar sein, da viele Leute das System benötigen und Fristen einhalten müssen. (Rechnungen bezahlen) .. Sicherheit? ✗

b) (2P) In einem Projekt wurde Code mit Assertionen (en: Assertions) versehen. Die Assertionen bleiben bei der produktiven Ausführung sowie beim Testen eingeschaltet. Der Kunde hat nun beschwert, dass die Anwendung zu langsam läuft. Es wird vorgeschlagen, die Assertionen beim produktiven Code auszuschalten. Finden Sie das eine gute Idee? Begründen Sie Ihre Antwort.

Vorschlag ist gut (Ja/Nein):

Ja

Grund:

Assertions sollten kein produktiver Code enthalten. Sie dienen lediglich für den Programmierer als Absicherung.

## Aufgabe 6: Design by Contract (10 Punkte)

- a) (10P) Gegeben ist ein Interface „Set“, welches Operationen auf Mengen spezifizieren soll. Ihre Aufgabe ist es, dieses mit Preconditions, Postconditions und Invarianten zu ergänzen. In der Notation sind Sie frei. Falsche oder überflüssige Aussagen geben Abzug.

**Wichtig:** Eine Menge darf **kein null** enthalten!

**Hinweis:** Bitte sehen Sie sich das gesamte Interface an, bevor Sie die Aufgabe lösen.

```
public interface Set<E> {
```

Interface Invariant:

```
@inv contains(null) == false  
@inv size() >= 0 ①
```

Contract für Methode „contains()“:

```
// Prüft das Vorhandenseins eines Elements in der Menge  
boolean contains(E e);
```

Contract für Methode „size()“:

```
// Gibt der Anzahl Elemente in der Menge aus.  
int size();
```

Contract für Methode „insert()“:

@pre !contains(E e) -- 34  
@post size() == size()@pre  
@post contains(E e) - ①

// Fügt ein Element in die Menge ein.  
void insert(E e);

Contract für Methode „remove()“:

@pre contains(E e) == true -- 34  
@post contains(E e) == false ①  
@post size() == size()@pre - 1 .. nur wahr falls ✓ ①

// Entfernt ein Element aus der Menge.  
void remove(E e);

Contract für Methode „removeAll()“:

@post size() == 0 - ①

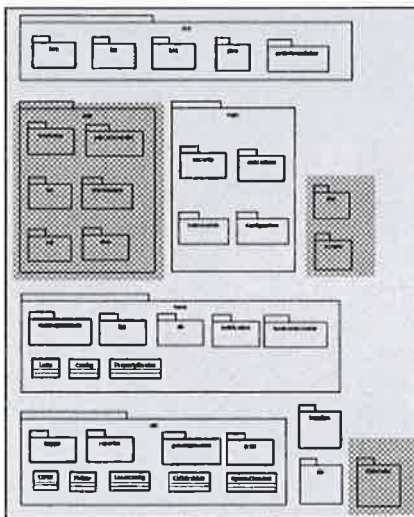
// Entfernt alle Elemente aus der Menge.  
void removeAll();

}

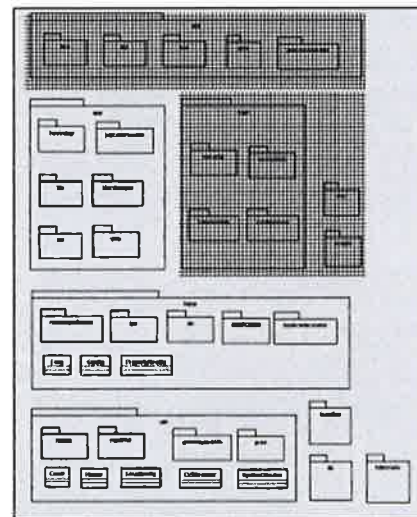
## Aufgabe 7: Software Architektur (10 Punkte)

- a) (5P) Wo welcher Code läuft: Zeichnen Sie ein beispielhaftes Deployment-Diagramm für die Situation, wenn der Code von diesem Schichtendiagramm (das Schichtendiagramm ist für Client und Server dasselbe) auf **3 Tiers** verteilt wird. (siehe auch Folie 8 der Vorlesung Software-Architektur, es ist dieselbe Information, nur in Farbe). Die vier Schichten sind (von oben nach unten): GUI / Logik / high-level Services / Libraries. Die Package-Namen sind nicht lesbar, Sie dürfen Ihre eigenen Annahmen treffen. (s. Bild für Positionierung)

### Clients

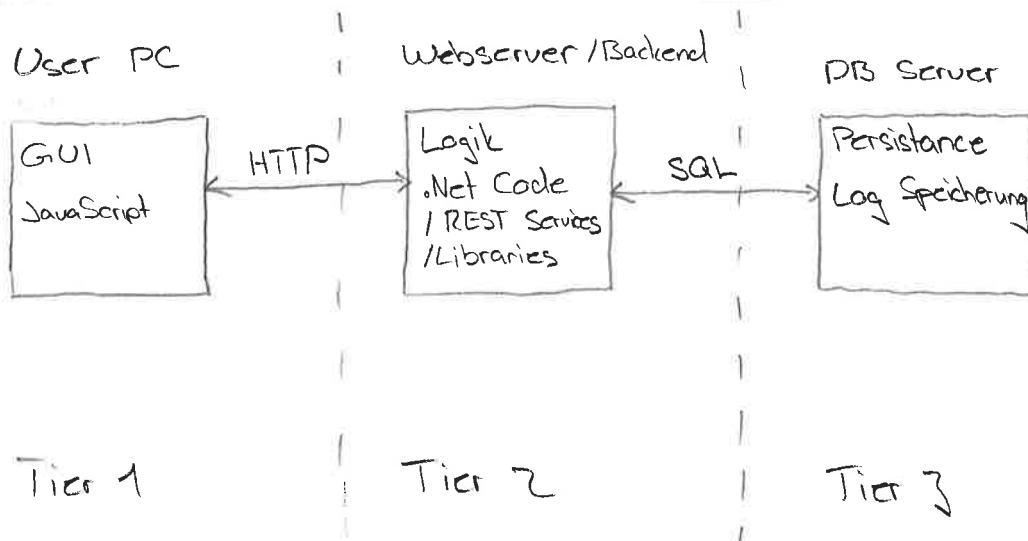


### Server



 =läuft *nicht* auf Clients

 =läuft *nicht* auf Server



3.5

b) (3P) Zeichnen Sie (informell, mit Kreisen und Pfeilen, aber bitte fein und deutlich, wenn möglich in anderer Farbe) in den obigen Diagrammen ein, welche Stücke des Codes wo laufen; mit anderen Worten: verbinden Sie das Schichtendiagramm mit dem Deployment-Diagramm, oder nochmals anders erklärt: ‚welche Stücke des Codes‘ im Schichtendiagramm verbinden mit dem ‚wo‘, nämlich den entsprechenden Teilen des Deployment-Diagramms.

c) (2P) Wo kann es in einem SW-System unter Umständen zu einer Verletzung der "Don't Repeat Yourself" Regel (DRY) kommen?

c1) Machen Sie bitte ein Beispiel für ein kleines SW-System:

Ein bestimmtes Stück Code welches einfach zu verstehen ist,  
wird mit einem Kommentar ergänzt.

c2) Machen Sie bitte ein (möglichst typisches) Beispiel für ein sehr grosses SW-System:

Zwei grosse Funktionen, die fast dasselbe machen

## Aufgabe 8: Code Smells (12 Punkte)

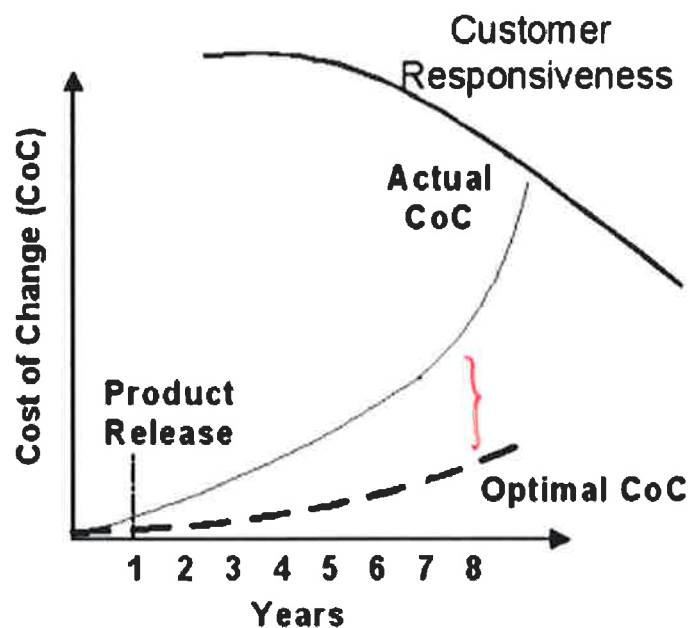
a) (1P) Zeichnen Sie in folgender Figur *Technical Debt* ein:

### Decreasing Customer Responsiveness

**Your responsiveness to Customer needs DECREASES as the Cost of Change increases!**

**MISMANAGEMENT of Technical Debt is what gets you into this mess.**

Image courtesy of Jim Highsmith

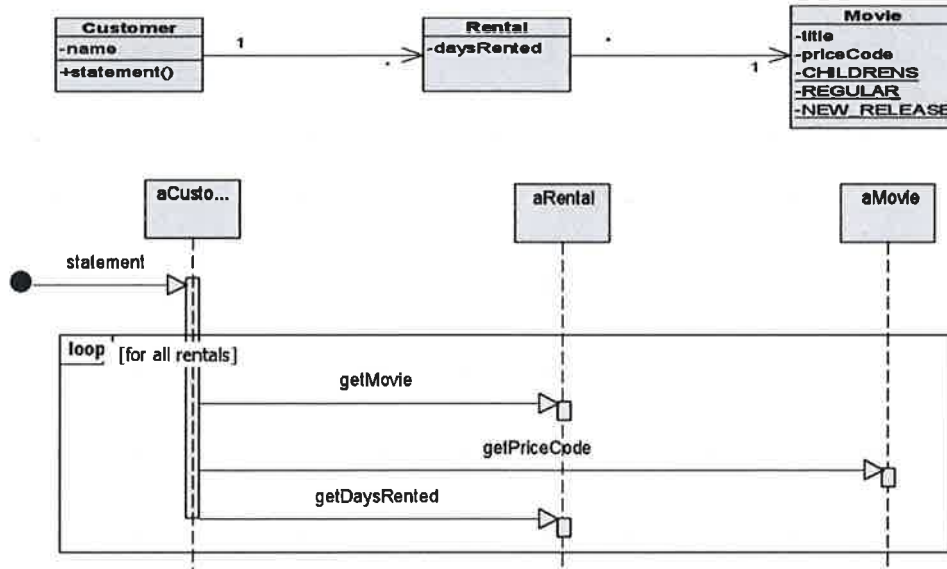


b) (3P) Folgendes Codefragment kommt mehrfach in ähnlicher Form in einem Programm vor, die Klassen- und Variablennamen sind dabei verkürzt, da sie nichts mit der Frage zu tun haben. Klar hier liegt auch der Code Smell *Duplicated Code* vor. Welcher andere Code Smell liegt hier auch noch vor? Mit welchem GRASP Pattern kann man ihn beheben und geben Sie dafür eine Begründung.

```

if (x instanceof B1) {
    ...
} else if (x instanceof B1) {
    ...
} else if (x instanceof B2) {
    ...
} else if (x instanceof B3) {
    ...
}
  
```

- c) (4P) Die folgenden UML-Diagramme zeigen die Klassenstruktur und die Objektinteraktionen der Methode `statement()`, mit welcher der totale Preis aller Film-Ausleihen eines Kunden berechnet wird. Der Preis einer Ausleihe ist abhängig von der Ausleihdauer und dem Preiscode eines Films.



Welche beiden Code Smells können Sie in der Methode `statement()` erkennen? Bitte geben Sie eine Begründung (ohne Begründung keine Punkte).

Feature Envy: Klasse **Customer** interessiert sich für die „Movies“ und deren Price Code. Sollte von „Rental“ übernommen werden. 2

- d) (2P) Eine Software ist in einem git-Repository unter Konfigurationsverwaltung und mit Unit-Tests gut abgedeckt. Beim Vornehmen von Codeänderungen erkennen Sie ein Stück „Dead Code“. Was machen Sie und warum?

- Einen neuen Branch anlegen um den anderen Code im konsistenten Zustand zu halten
- Code im Branch bearbeiten
- Rücksprache mit anderen Teammitgliedern ob der Code wirklich nicht mehr benötigt wird

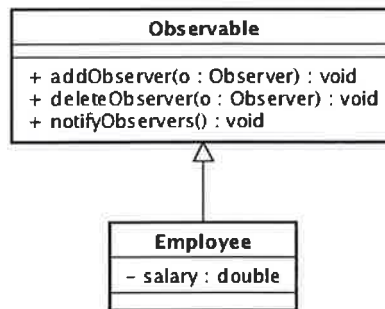
- e) (2P) Eine Methode legt ein Zwischenresultat einer Berechnung in einer Instanzvariable ab. Nachdem die Methode zu Ende gelaufen ist, hat der Wert der Instanzvariable keine Bedeutung mehr. Wie heisst der vorliegende Code Smell? Worin liegt das Problem?

Temporary Field: Macht den Code schwieriger zu verstehen

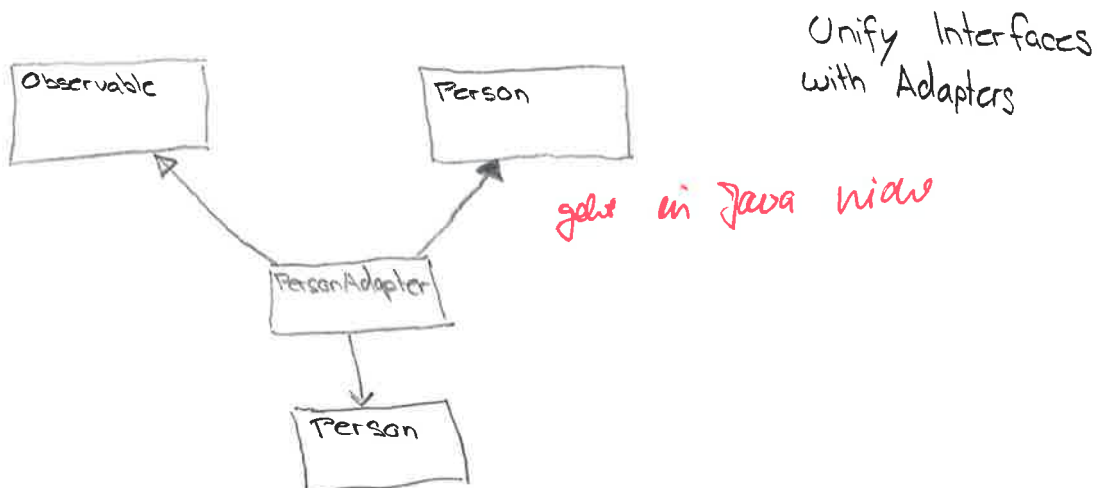


## Aufgabe 9: Refactoring mit Refactoring to Patterns (11 Punkte)

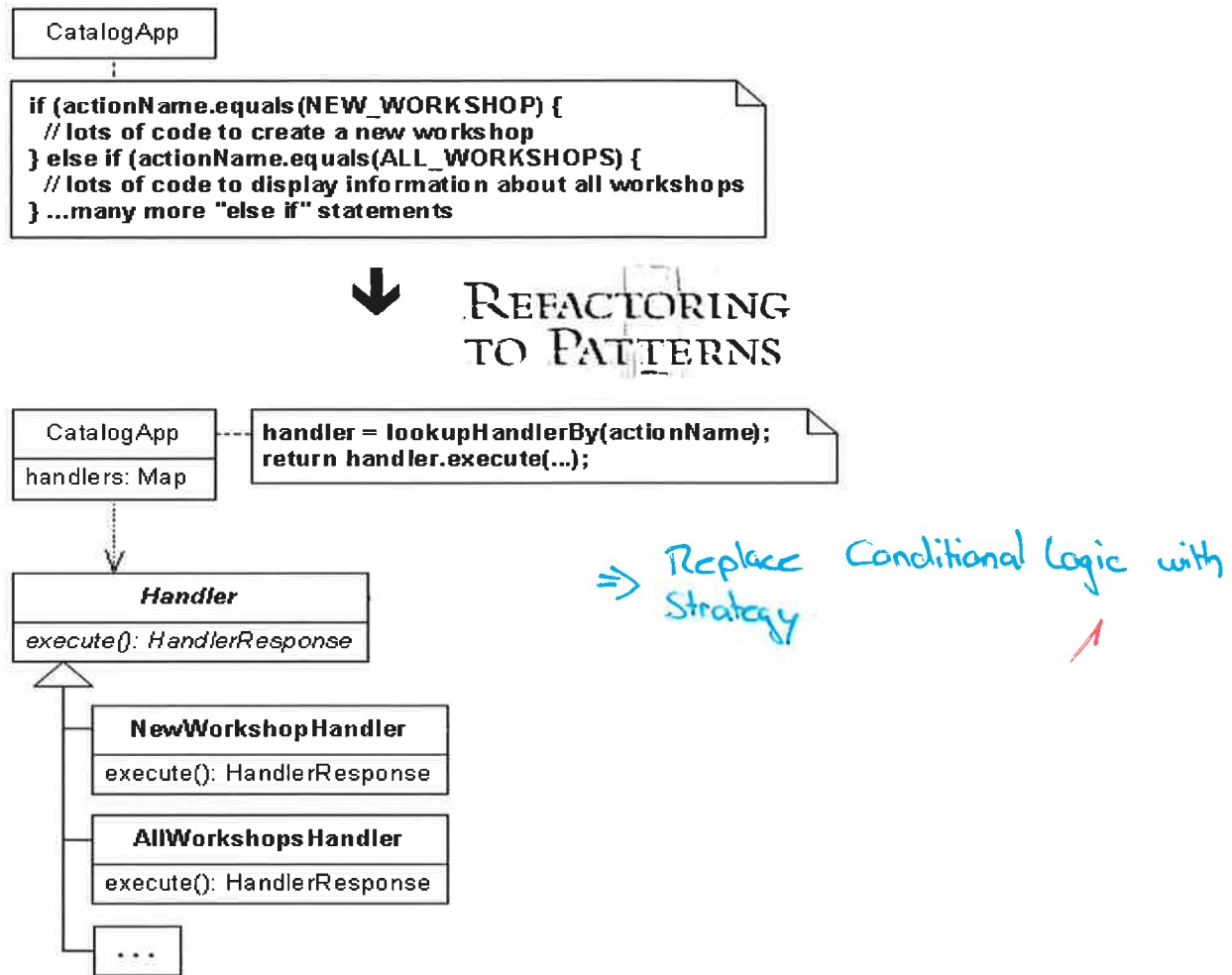
- a) (1P) Wie gehen Sie vor, wenn Sie eine grössere Refactoring-Aufgabe angehen müssen?
- *Piecemeal Refactoring: Nach dem „Divide and Conquer“ Prinzip vorgehen*
  - *Am Anfang müssen automatisierte Test korrekt ablaufen. Nach dem Refactoring sollten sie ebenfalls noch funktionieren.*
- b) (2P) Eine schwer verständliche *Long Method* soll durch mehrere Refactorings übersichtlicher und besser verständlicher werden. Was müssen Sie dabei beachten? Dafür gibt es auch ein *Refactoring to Pattern*, wie heisst dieses?
- *Funktionen sollten die gleichen sein nach Refactoring / Tests müssen immer noch funktionieren*
  - *In kleinen Schritten vorgehen / Divide-by-Conquer-Prinzip*
  - *R-t-P: Replace Conditional Logic with Strategy*
- c) (3P) Eine Klasse *Employee* wird in Java gemäss folgendem Klassendiagramm als *Observable* verwendet:



Nun möchte man *Employee* von einer Klasse *Person* ableiten, aber *Employee* gleichzeitig als *Observable* verwenden und dabei natürlich die Funktionalität, welche die Klasse *Observable* bietet, weiter verwenden. Welches Refactoring kann man dafür verwenden? Wie sieht das resultierende Klassendiagramm aus?



d) (3P) Welches Refactoring to Patterns wird hier gezeigt? Zeichnen Sie die Rollen des Patterns ein:



e) (2P) Mit dem Refactoring to Pattern *Replace Conditional Logic with Strategy* wird oft ein anderes Refactoring to Pattern eingesetzt. Wie heisst dieses Refactoring to Pattern? Erklären Sie wofür (nur Punkte mit Erklärung).

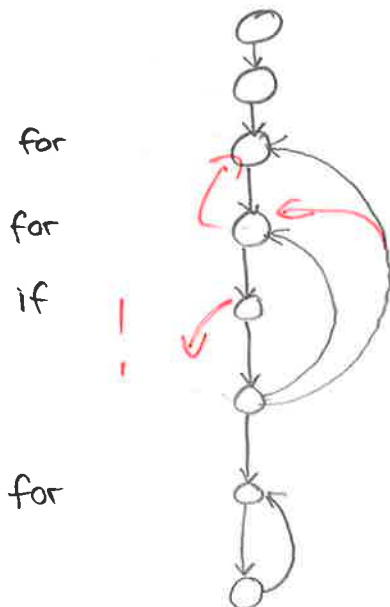
## Aufgabe 10: Metriken (9 Punkte)

- a) (4P) Zeichnen Sie den Kontrollflussgraph der main()-Methode des folgenden Programmes und berechnen Sie aus dem Kontrollflussgraph die Komplexitätszahl nach McCabe:

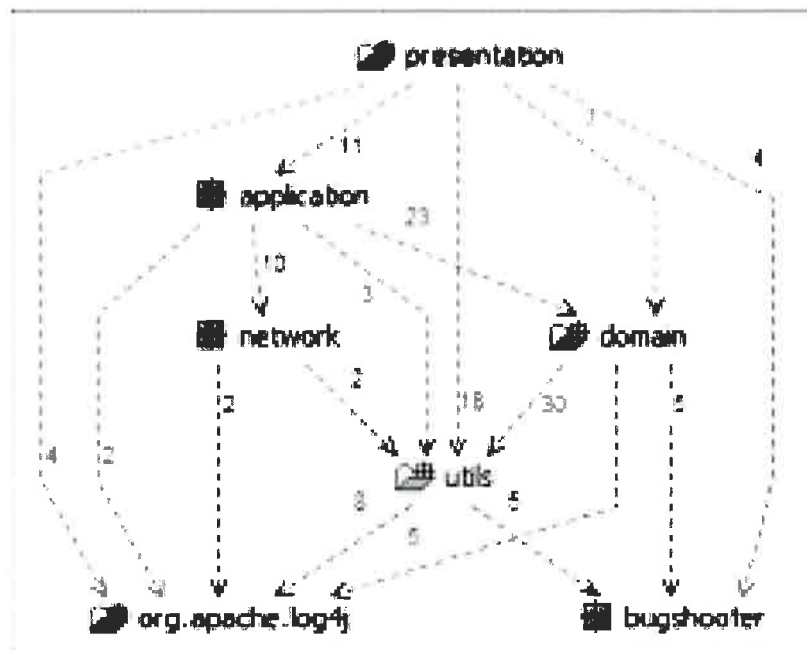
```

public class Sort {
    static int[] array = { 7, 3, 10, 1, -5, 11, 6 };
    public static void main(String[] args) {
        int outerIndex;
        int innerIndex;
        for (outerIndex = array.length - 1; outerIndex > 1; outerIndex--)
            for (innerIndex = 0; innerIndex < outerIndex; innerIndex++)
                if (array[innerIndex] > array[innerIndex + 1]) {
                    int temp = array[innerIndex];
                    array[innerIndex] = array[innerIndex + 1];
                    array[innerIndex + 1] = temp;
                }
        for (int i = 0; i < array.length; i++)
            System.out.println(array[i]);
    }
}
    
```

M McCabe: Anzahl FOR + IF + 1 = 5 1



- b) (5P) Die folgende Abbildung zeigt eine Strukturanalyse einer Applikation mit Structure 101, mit STAN würde es praktisch gleich aussehen.



Um was für ein UML-Element handelt es sich bei den gestrichelten Pfeilen?

Abhängigkeiten

✓

Was bedeutet die Zahl beim gestrichelten Pfeil?

wieviele "Referenzen" innerhalb des Package bestehen zum anderen.

✓

Was kann der gestrichelte Pfeil im Code genau alles bedeuten?

- Vererbungen

✓

- Instanzen

✓

## Aufgabe 11: Reviews (5 Punkte)

- a) (1P) Unit Testing: Test-Abdeckung-Tools wie EMMA oder Cobertura liefern genaue und nützliche Zahlen. Was sind aber ihre Schwächen? Wo sind die Tools nicht so stark?

Man muss selber etwas von Software Engineering verstehen  
- Aufwändig zu konfigurieren

- b) (1P) Wie begründen Sie den Nutzen eines Architektur-Reviews?

Jeder versteht danach den Architekturstil des Projekts. Alle sprechen vom gleichen

- c) (1P) Sie wollen einen Usability Test machen. Wann setzen Sie diesen Test am besten an? Begründen Sie bitte Ihre Antwort.

- In der Elaboration  
- Somit kann man bei End of Elab fast sicher sein, den Kunden korrekt verstanden zu haben.

- d) (1P) Nennen Sie zwei statische Code-Analyse-Tools

STAN, Eclemma

- e) (1P) Statische Code-Analyse-Tools finden viele potentielle Fehler. Welche Fehler, welche schlechte Code-Qualität können diese Tools NICHT aufdecken?

## Aufgabe 12: Aufwandschätzung (2 Punkte)

- a) (1P) Nehmen wir an, Sie wollen Aufwand/Kostenschätzungen in Ihrer Firma richtig machen. Sie wollen ein Berechnungs-Modell einsetzen (z.B. COCOMO). Wie kann man die Parameter dieses Modell richtig kalibrieren?
  
  
  
  
  
  
  
  
  
  
- b) (1P) Nehmen wir an, Ihre Organisation arbeitet mit Scrum & Story Points und kennt die aufgewendeten Arbeitsstunden im Detail. Wie ist der Zusammenhang zwischen geleisteten Arbeitsstunden und Story Points?

## Aufgabe 13: Profiling, Performance (4 Punkte)

- a) (2P) Beschreiben Sie mit ein paar Worten, was ein Profiler macht, z.B. der Java Profiler in NetBeans.
  
  
  
  
  
  
  
  
  
  
- b) (1P) Was ist der Grund, dass in der Übungsanleitung zum Profiling explizit darauf aufmerksam gemacht wurde, man solle einen „Ausgangspunkt festlegen (establish a baseline)“ ?
  
  
  
  
  
  
  
  
  
  
- c) (1P) Wie kann man ein langlebiges Programm, z.B. einen Webshop, immer wieder einmal (zumindest nach jedem Release) testen, ob die Performance noch gleich bzw. genügend ist? oder ob man z.B. durch den neuen Release nicht langsamer geworden ist, was die Kundenakzeptanz und damit den Umsatz verschlechtern würde.

## Aufgabe 14: Agile Software Entwicklung (12 Punkte)

a) (1P) Wer entscheidet in Scrum über den Umfang der in einem Sprint realisierten Funktionalität?

b) (1P) Was sagt der Scrum Guide zu User Stories?

c) (3P) In einem Agilen Projekt werden sowohl Use Cases als auch User Stories verwendet.  
1. Wie sind die Multiplizitäten zwischen diesen beiden Konzepten? Bitte Erklärung zur Antwort.



Wozu verwendet man wohl beides?

d) (2P) Vergleichen Sie Scrum mit dem RUP bezüglich der unterstützten Disciplines.

e) (2P) Wieso wird der Aufwand für User Stories in der Regel nicht in Stunden sondern in Story Points geschätzt?

f) (3P) In agilen Projekten werden Aufwände im Team geschätzt. Nennen Sie dafür genau drei Vorteile?



## Aufgabe 15: Funktionale Programmierung (6 Punkte)

In der Vorlesung haben wir einen algebraischen Datentyp für Listen definiert:


```
type 'a list =  
  | Nil  
  | Cons of 'a * 'a list;;
```

Als Beispiel, haben wir eine Funktion „length“ definiert, die die Länge einer Liste berechnet:


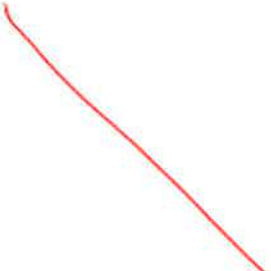
```
let rec length = function  
  | Nil -> 0  
  | Cons(hd,t1) -> 1 + length t1  
;;
```

In dieser Aufgabe müssen Sie weitere Funktionen über Listen im funktionalen Stil definieren. Benutzen Sie bitte die gleiche Syntax wie bei der Definition von „length“ oben.

- a) (3P) Definieren Sie eine Funktion „contains“, die das Vorhandenseins eines Elements in einer Liste überprüft. Z.B. `contains 2 (Cons(1,Cons(2, Nil))) = true`; `contains 3 Nil = false`.



- b) (3P) Definieren Sie ein Funktion „remove“, die alle Vorkommnisse eines Elements aus einer Liste entfernt. Z.B. `remove 2 (Cons(1,Cons(2, Nil))) = (Cons(1,Nil))`



## Aufgabe 16: Software Tatort (4 Punkte)

- a) (4P) In der Vorlesung haben wir den Heartbleed-Bug vertieft angeschaut. Schlagen Sie zwei Massnahmen vor, die ähnliche Software-Fehler in der Zukunft verhindern könnten. Begründen Sie Ihre Antwort.

Massnahme 1:

Testing Code-Review ✓

Begründung für Massnahme 1:



Massnahme 2:

Systematische Angriffssuche - OK

Begründung für Massnahme 2:

