Methods in Software Engineering Practice Exam

Summer Term 2021 • Prof. Dr. Gidon Ernst • LMU Munich

Before the exam

- turn off your mobile phone and smartwatch now and store them away,
  if we catch you with a turned-on phone or similar we must regard it as cheating
- put any bags and jackets in the row in front of you

- write down your name and matriculation number

  Name:

  Matriculation Number:

When the exam time starts

- check that all 14 pages are included (the last page is spare)

Rules

- blue or black pen (not erasable, no pencil, no green/red),
- ok: mask, drinks, snack, ruler
- not ok: own paper, *any* other material, including:
  notes, books, calculators, dictionary

Time 90 minutes

Language

- you can answer both in English and/or German
- please ask if words or sentences are unclear

Grading ☐ please *do not* grade this exam (“entwerten”)

Exercise	1	2	3	4	5	6	Σ
Points	of 15	of 6	of 15	of 15	of 9	of 15	of 75

Remarks

- the exam is subject to copyright and may not be distributed outside of this lecture
- this practice exam corresponds exactly to exam #1 held on July 28, 2021
- errata: some typos, marks for exercise 5/6 distributed differently than indicated

1 Functional and Imperative Programs (15 points)

Evaluation of an expression e with respect to a substitution σ is defined by:

$$\text{eval}(k, \sigma) = k \quad \text{if } k \text{ is a constant} \tag{1}$$

$$\text{eval}(x, \sigma) = \sigma(x) \quad \text{for a variable } x \in \text{dom}(\sigma) \tag{2}$$

$$\text{eval}(e_1 + e_2, \sigma) = v_1 + v_2 \quad \text{where } v_1 = \text{eval}(e_1, \sigma) \text{ and } v_2 = \text{eval}(e_2, \sigma) \text{ (similarly for multiplication } *) \tag{3}$$

$$\text{eval}(f(e_1, \ldots, e_n), \sigma) = \text{eval}(e, \sigma') \quad \text{where } v_i = \text{eval}(e_i, \sigma) \text{ for } i = 1, \ldots, n \text{ and } \sigma' = [x_1 \mapsto v_1, \ldots, x_n \mapsto v_n] \text{ where } f(x_1, \ldots, x_n) = e \text{ is a definition} \tag{4}$$

a) Given this definition and substitution (7 points)

$$square(x) = x * x \qquad \sigma = [x \mapsto 3, y \mapsto 4]$$

the goal of this exercise is the evaluation of the expression

$$square(x + 7)$$

using the intermediate steps outlined below:

- in each step, write the *resulting value*
- check, which of the above rules (1)–(4) defines the respective evaluation step
- complete the definition of σ' such that it fits into the evaluation

evaluation	resulting value	rule
$\text{eval}(x, \sigma) =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)
$\text{eval}(7, \sigma) =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)
$\text{eval}(x + 7, \sigma) =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)
$\text{eval}(x, \sigma') =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)
where $\sigma' =$		(look ahead where σ' is needed)
$\text{eval}(x * x, \sigma') =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)
$\text{eval}(square(x + 7), \sigma) =$		☐ (1) ☐ (2) ☐ (3) ☐ (4)

b) Define two new evaluation rules for the two constructors of the inductive data type of lists. As a hint, think about what the *values* of this data type are. Your definition should satisfy:

$$\begin{aligned}&\mathrm{eval}(\texttt{cons}(x + y, w), [x \mapsto 1, y \mapsto 2, z \mapsto \texttt{cons}(5, \texttt{nil})])\\ &\quad = \texttt{cons}(3, \texttt{cons}(5, \texttt{nil}))\end{aligned}$$

$\mathrm{eval}(\texttt{nil}, \sigma) =$ (1 points)

$\mathrm{eval}(\texttt{cons}(e_1, e_2), \sigma) =$ (3 points)

c) What is problematic with the evaluation of expressions that use of this definition? (1 points)

$$f(x) = f(x + 1) + 1$$

Which concept, in addition to eval, did we introduce to address this problem? (1 points)
Check the correct box and complete the following sentence with a reference to this concept:

For expression $f(e)$ and any substitution σ, there is ☐ no ☐ some number n

such that _____ holds, if f is defined as shown above.

d) For each starting configuration given below, write down the configuration resulting from the **first execution step** of the imperative program wrt. the respective substitution. The different lines are independent from each other.

(2 points)

starting configuration	$\longrightarrow$	configuration resulting from first step
$\langle\, [x \mapsto 1, y \mapsto 2] \mid x := x + 1 \,\rangle$		
$\langle\, [x \mapsto -1] \mid \texttt{if}\ x > 0\ \texttt{then}\ y := 1\ \texttt{else}\ y := -1 \,\rangle$		

2 Propositions as Types (6 points)

As a reminder, we have the following inductive data types that can represent logical conjunction $A \wedge B$ and disjunction $A \vee B$, as well as function types $A \to B$ that correspond to implication.

$$\texttt{data Pair}\ A\ B = (\mathit{fst} : A, \mathit{snd} : B)$$
$$\texttt{data Either}\ A\ B = \texttt{left}(a : A) \mid \texttt{right}(b : B)$$

Complete the missing parts below (type, function definition, formula). For example:

-) formula $A \implies A$
type $f : A \to A$
definition $f(x) = x$

a) formula (1 points)

type (1 points)

definition $f(x, y) = x$

b) formula $A \wedge A \implies B \vee A$

type (1 points)

definition (1 points)

c) formula $(A \implies B) \wedge A \implies B$

type (1 points)

definition (1 points)

3 Component Design (15 points)

a) Which technology can be used to document the operations of a web-service in order to make it easy for third-party developers to use this interface? (1 points)

b) Consider a public transport company that operates a subway network. There are stations and numbered subway lines, and trains running on a schedule that depends on the current day of the week. We are concerned with the design of a web-service that is accessible to third-party web-sites or apps.

Specify the interface of the operation for the following requirements in terms its **parameter(s)** and **return value(s)**. Briefly describe in your answer how the underlined concepts are represented in the public data model (i.e., the associated types and valid values).

i. `GetStations`: Query the stations at which a particular subway line stops. (4 points)

ii. `FindRoute`: At a particular departure time on a given day of the week, returns a route between two stations. Note, it may be necessary to change train lines. (6 points)

c) A component called `BusyLogic` in your system requires access to a database connection, which is already implemented as a class `Database`.

In this exercise, we explore different potential solutions.

i. Draw a UML class diagram for a design based on *inheritance* (1 points)
(no attributes/methods, only the mentioned classes plus relevant associations)

ii. Draw a UML class diagram for a design based on *composition* (1 points)
(no attributes/methods, only the mentioned classes plus relevant associations)

iii. Suppose that `BusyLogic` should in addition use another component for logging, which is implemented in class `DebugLog`. *Check* which applies and *explain briefly*.
There are different valid reasons (e.g., specific to Java or wrt. a particular principle).

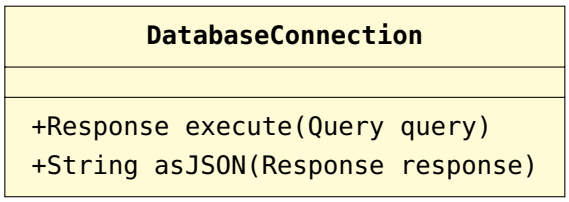
The design based on ☐ inheritance ☐ composition is better,
because: (1 points)

iv. Suppose there should now be support for choosing *at runtime* (e.g. via a configuration option) between three implementations of databases: `SQLDatabase`, `TextFileDatabase`, and `InMemoryDatabase`. *Check* which applies and *explain briefly*.

The design based on ☐ inheritance ☐ composition is better,
because: (1 points)

4 SOLID Principles (15 points)

a) Consider the following UML class diagram



What is the conceptual role of objects of this class? Check the correct box. (1 points)

☐ data ☐ an algorithm ☐ a component

Name the SOLID principle that violated here. (2 points)

Briefly explain why this is the case and describe an idea to improve the design.

b) Complete the definition: (1 points)

A class invariant is a property/assertion over the attributes of a class

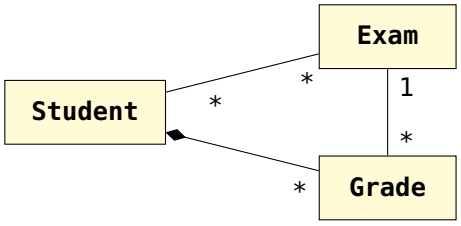
- that is

  _____

- and that is

  _____

c) Invariants may refer to multiple classes at the same time. Describe informally an *invariant* for the UML diagram shown below, expressing that the three associations are consistent with each other. (2 points)



d) Consider two Java classes, **A** and **B** that both implement a common interface `IntSet`, which should behave like a set data structure (think of `java.util.Set`).

```java
class A extends IntSet {
  boolean elems[] = new boolean[](10);

  void add(int x) {
    elems[x%100] = true;
  }

  boolean contains(int x) {
    return elems[x%100];
  }

  ...
}
```

```java
class B extends IntSet {
  int elems[] = new int[](10);
  int size    = 0;

  void add(int x) {
    elems[size++] = x;
  }

  boolean contains(int x) {
    for(int i=0; i<size; i++) {
      if(elems[i] == x) return true;
      else return false;
    }
  }
}
```

Name the SOLID principle that is violated here. (1 points)
Concrete violations are asked for in subtask **e)**

What does this principle require from the *execution histories* of components? (1 points)

What does this principle guarantee regarding the *composition* of components? (1 points)

e) There are multiple distinct violations in the code from **d)** of the principle discussed above. Provide *histories*, which uncover these. Write *one event per line*. Both can be solved with *three or even fewer* events, but it is ok if you use more.

Your histories should end in a mismatch in the result of **A** and **B**, where results are one of: return values of an operation resp. nothing (—) in case of `void`, or an exception. You do not need to include a call to a constructor as the first event.

i. Provide a history, in which the problem is in class **A**. (3 points)

step	operation name	parameter value	result for **A**	result for **B**
1.				
2.				
3.				
⋮				

ii. Provide a history, in which the problem is in class **B**. (3 points)

step	operation name	parameter value	result for **A**	result for **B**
1.				
2.				
3.				
⋮				

5 Modern Software Development (9 points)

a) Name **three** key tools of modern software development that were discussed in the lecture and that are integrated in a platform like `GitHub`, and **describe** them briefly:

i. (1 points)

ii. (1 points)

iii. (1 points)

b) Describe the following concepts

A patch is: (1 points)

A branch is: (1 points)

c) Graphically sketch an abstract `git` history that can potentially lead to a merge conflict, and indicate the point where it can occur (no need to give concrete files/changes). (2 points)

d) Name a goal and a metric relevant in testing:

goal: (1 points)

metric: (1 points)

e) Name **two** advantages of testing in general over formal proofs/static analysis

i. (1 points)

ii. (1 points)

6 Unit and Property-based Testing (15 points)

a) When compared to property-based testing, name

one advantage of unit-testing: (1 points)

one drawback of unit-testing: (1 points)

b) You are given a Java method with a documentation comment as follows:

```java
// return the smallest index i such that data[i] == elem
// if none exists, return data.length
int firstIndexOf(int elem, int[] data);
```

Specify **four** Unit-Tests in terms of inputs and expected result (a value or an exception).
Cover as many different cases as possible. (4 points)
You can write arrays values with curly braces, e.g. `{1,2,3}`

	`elem`	`data`	expected result
1.			
2.			
3.			
4.			

c) You are given a Java interface `List` with some comments as follows:

```
interface List {
  // return the number of elements stored in the list
  int  length();
  // return the element at given index, where 0 <= index and index < length()
  Elem get(int index);
  // return a *new* list that has the same elements but in reverse order
  List reverse();
}
```

i. Write a property-based test for `reverse`, following the idea of *idempotency*, i.e., reversing twice gives back the original list. Rely on `assertEquals` and `@ForAll`. (2 points)

```
@Property
void checkReverse1(                                                    )
{









}
```

ii. Write another property-based test for `reverse`, that checks a *single* pair of elements in the original and reversed list, for a particular index that *should be given as a random parameter*. Remember to consider valid indices only. (5 points)

```
@Property
void checkReverse2(                                                    )
{









}
```

Extra Page

If you use this page, please

- strike through those parts and solution attempts that should not be graded
- place a short note on the exercise sheet: “see extra page” or similar
- more paper is available on request, please always return *all* sheets