Software Engineering: Tutorial 11

David Voigt

January 20th, 2022

Agenda

1. **No** discussion of homework :(
2. Exam Preview
3. UML
4. Exercises

Any questions?

Exam Preview

1. Trying solving the exam preview in the next 15 minutes
2. Afterwards, we will discuss the solutions

Grading Scheme

Paper: https://dl.acm.org/doi/pdf/10.1145/1189136.1189164

- Three options for each question:
  - Not answering at all
  - Mark exactly one answer
  - Mark more than one answer

$$S(k,a,c) = \begin{cases} 0 & a = 0 \lor a = k \\ \log\left(\frac{k}{a}\right) & a > 0 \land c = 1 \\ -\frac{a}{k-a}\log\left(\frac{k}{a}\right) & a > 0 \land c = 0 \end{cases}$$

- $k :=$ number of possible answers, $a :=$ number of marked answers, $c :=$ whether the correct answers has been marked
- 0 points for not answering
- Partial points if the correct answer is among the marked ones
- Negative points if the correct answer has not been marked

Answers

Question 1

```
assert(distinct(List()) == List())
assert(distinct(List(1)) == List(1))
assert(distinct(List(1, 1)) == List(1))
assert(distinct(List(1, 1, 2, 3, 3)) == List(1, 2, 3))
assert(distinct(List(1, 1, 2, 2, 2, 3, 3, 3, 3)) == List(1, 2, 3))
assert(distinct(List(3, 3, 2, 1, 1)) == List(3, 2, 1))
```

Question 2

- Software elements may be freely combined with each other in possibly new environments
- Directly connected to reusability
- Well-defined and well-designed interfaces are essential for composability
- Example: Unix shell commands

Questions 3

UML is a formal, graphical modeling language

Questions 4

A magic number is a number literal directly used in the code

Recap: UML

- While being also used informally, UML is a formal, graphical modeling language first
- One may distinguish between two types of diagram types
  - **Structural Diagrams**: Describe entities with static relations to one another
  - (**Behavioral Diagrams**: Describe dynamic information flow)

Object Diagrams

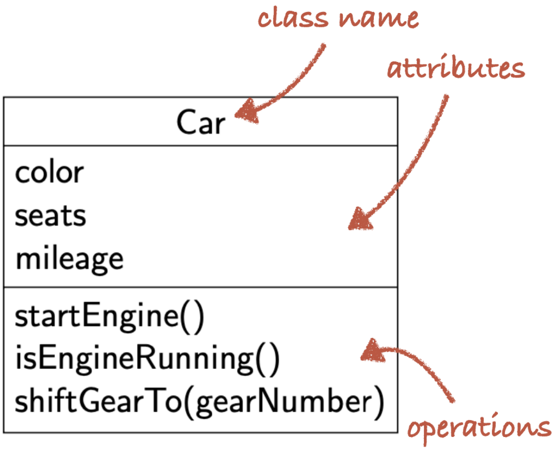

Figure 1: Brachthäuser, “10. Design Principles”, Software Engineering 2022/2023

Class Diagrams

Lines express associations between classes

- $\rightarrow$: simple association
- $\dashrightarrow$: no ownership
- $\Diamond$: Weak ownership (**aggregation**)
- $\blacklozenge$: Strong ownership (**composition**)

Associations

- `1`: associated with exactly one instance
- `*`: associated with arbitrary many instances
- `0,1`: associated with zero or one instances
- `1..*`: associated with at least one instance

Generalization

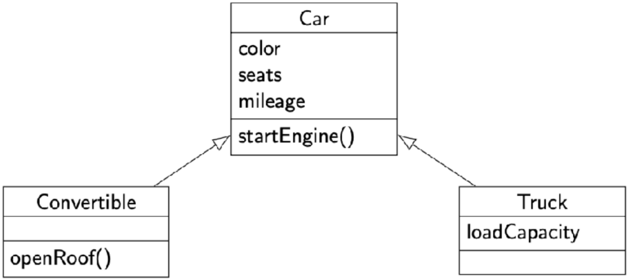

Figure 2: Brachthäuser, “10. Design Principles”, Software Engineering 2022/2023

Interfaces

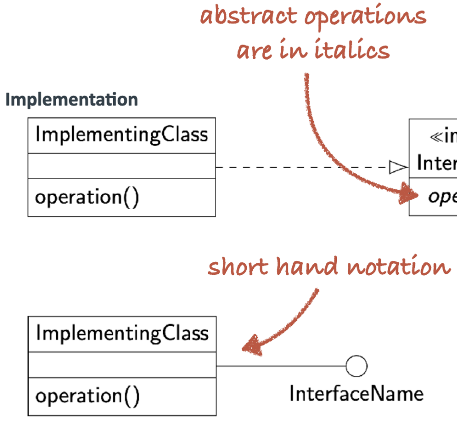

Figure 3: Brachthäuser, “10. Design Principles”, Software Engineering 2022/2023

Exercises

https://github.com/se-tuebingen-exercises/tut7-exercise11