Software Engineering: Tutorial 8

David Voigt

December 16th, 2022

Common Mistakes: 1

```
1  def getAnswer(bigPack: Pack, smallPack: Pack): ProblemAnswer = {
2      var result = ProblemAnswer.Small
3      if (bigPack.pricePerLiter() <= smallPack.pricePerLiter()) {
4          result = ProblemAnswer.Large
5      }
6      return result
7  }
```

Common Mistakes: 1

```
1  def getAnswer(bigPack: Pack, smallPack: Pack): ProblemAnswer =
2      if (bigPack.pricePerLiter() <= smallPack.pricePerLiter())
3          ProblemAnswer.Large
4      else
5          ProblemAnswer.Small
```

Common Mistakes: 2

```scala
1   try {
2       throw Exception("The input has the wrong format")
3   } catch {
4       case e: Exception => println("The input format is invalid")
5   }
```

Common Mistakes: 2

```scala
1  try {
2      throw Exception("The input has the wrong format")
3  } catch {
4      case e: Exception => println("The input format is invalid")
5  }
```

- Either the current state is really **exceptional** and we terminate the current computation or the computation may continue and we just print a warning
- Here the computation is not terminated and the exception is just used for immediately printing something to the screen
- Use exceptions if something **exceptional** occurs to prematurely exit a computation with a descriptive error message

```
if c then true
else false
```

```
if c then true
else false
```

but this is just

```
c
```

Revision: Interfaces

What is an interface?

Revision: Interfaces

What is an interface?

User Programs
Using types and operations
specified in interfaces.

Implementation
Implementing the specified operations.

Interfaces
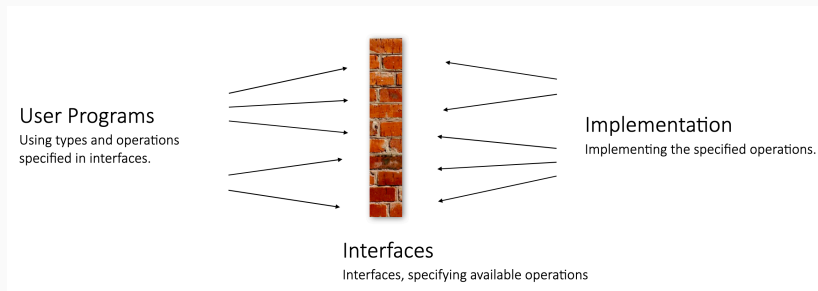Interfaces, specifying available operations

Figure 1: Brachthäuser, J. (2022). "Interface visualization". Slides Software Engineering (WS22)

- **API**: **A**pplication **P**rogramming **I**nterface describes how two services communicate
- Hides the actual implementation from the user

Design by Contract

Precondition

- Guarantees for the **input** as requirement for the user
- To be keept as weak as possible for maximum reuse

Postcondition

- Guarantees for the **output** as requirement for the implementor
- To be keept as strong as possible
- Tradeoff reusability with maintainability

Invariants

- Aspects left unchanged by usage

```
def head[T](xs: List[T]): T
```

Preconditions, Postconditions, Invariant?

Design by Contract: Example

```
def head[T](xs: List[T]): T
```

Preconditions, Postconditions, Invariant?

Preconditions
- The list is non-empty

Postconditions
- The return value is the first element in the list

Invariants
- The given list is unchanged