Software Engineering: Tutorial 12

David Voigt

January 27th, 2023

Agenda

1. Homework 10 discussion
2. Recap: **SOLID** principles
3. Exercises

Homework 10: Writing very narrow generators

```
val words: Gen[String] = Gen.oneOf(
  "Lagerregal", "aba", "a", "oTtO"
)
property("Palindrom") {
    forAll(words) { (word: String) =>
    assertEq(isPalindrome(word), isPalindrome(word.reverse))
}
}
```

- This is not general generator. It only generates some handpicked samples.
- This does not leverage the benefits of property tests

```scala
val randomString = Gen.alphaStr

property("if string is palindrome, then its equal to its reversion") {
  forAll(randomString) { (s: String) =>
    assert(if (isPalindrome(s) {
      s.toLowerCase() == s.reverse.toLowerCase()
    } else {
      s.toLowerCase() != s.reverse.toLowerCase()
    })
  }
}
```

SOLID

- Heuristics to identify good designs
- State desirable properties of software designs

Single Responsibility Principle (SRP)

A component (class, function, module, . . .) should only have one reason to change.

- A class `HttpRequest` is responsible to know the requested URL
- Changes in the requirements (problem space) leads to changes in the implementation (solution space)
- An entity should only have **one responsibility** and thus only **one reason** to change
- This simplifies adding new features

Open-Closed Principal (ODP)

*A component should be **open for extension**, but closed **for modifications**.*

- **Extension** means extending the behavior of a module.
  - Changing requirements can be solved by extending the module
- **Modification** means changing the source code of a module.
  - No need for changing the source code of a module to support changed requirements

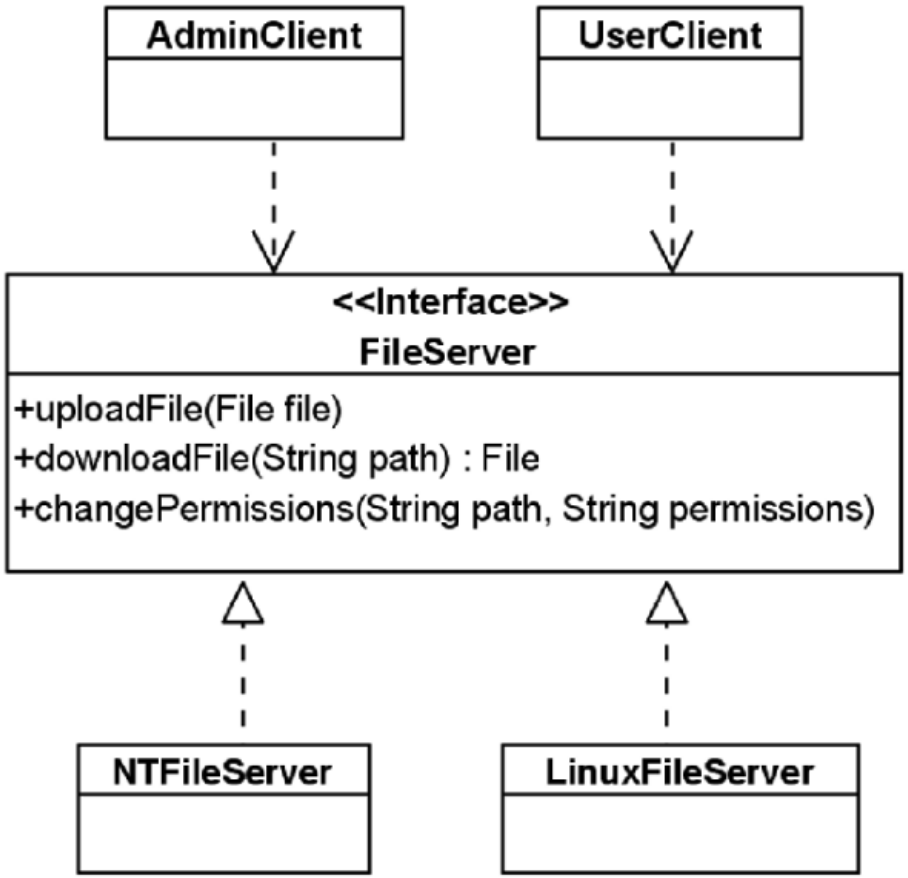
Liskov-Substitution Principle (LSP)

> *Subtypes must be **behaviorally substitutable** for their base types*

Let $P \in Ps$ be an observable property, such that $P(x)$ holds for all objects x of type T. Then a subtype $S <: T$ conforms to LSP with respect to Ps and T, if $P(y)$ is true for all objects y of type S.

Interface Segregation Principle (ISP)

Clients should not be forced to depend on methods that they do not use.



Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules.
Both should depend on abstractions.

- High level design decisions should influence low level implementation, not the other way around!
- We want to be able to switch lower levels and **change** implementation details, **without affecting the higher levels**.

Exercises

Design Principles: Part 1

What SOLID design principle(s) the alternative design adheres to?

Design Principles: Part 1

What SOLID design principle(s) the alternative design adheres to?

- **Dependency Inversion Principle**
  - An abstraction (Item interface) is introduced between the high-level classes (Shelf) and low-level classes (Book & DVD) that changes the direction of the dependency
- **Open-Closed Principle**
  - To extend the application, e.g. to use other "items", simply add another concrete implementation of the Item interface.
  - The extention (adding a new Item) will not require any changes to the classes already exsiting in the model.

Design Principles: Part 2

Is this a reasonable design? Take the SOLID principles into account!

How would you improve the design?

Design Principles: Part 2

Is this a reasonable design? Take the SOLID principles into account! How would you improve the design?

- This design goes against **Liskov Substitution Principle (LSP)**
- Many of `Board`'s methods, such as the given `getTile`, are designed to work with two dimension, not three.
- They lose their context/meaning when it comes to three dimensions
- Client code attempting to use the `3DBoard` class as its base class `Board` would fail

Analysis and Design

What classes will be in your “model” (for example a UML class diagram?) Only name the classes.

Analysis and Design

What classes will be in your "model" (for example a UML class diagram?) Only name the classes.

- Professor, Announcement, Student and Feed as classes

User Stories - As a student I want to read the feed so I can see important announcements - As a professor I want to see who has access to the feed I post my announcements on