Winter Semester 17/18

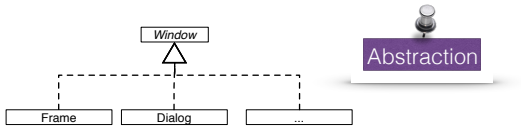
Software Engineering Design & Construction

Dr. Michael Eichberg
Fachgebiet Softwaretechnik
Technische Universität Darmstadt

Bridge Pattern

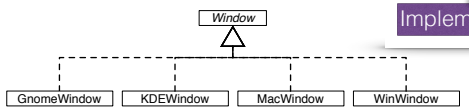
Motivation by Example

We want to provide different types of windows:



Abstraction

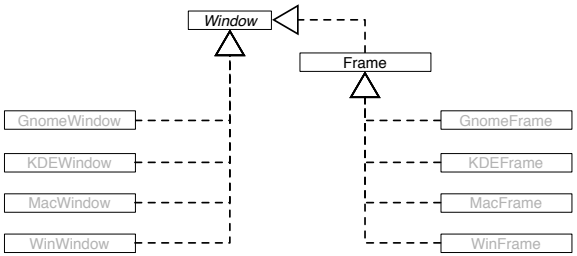
We want to support multiple operating systems:



Implementation

Motivation by Example

Two dimensions of variability!



Can you imagine a better solution?

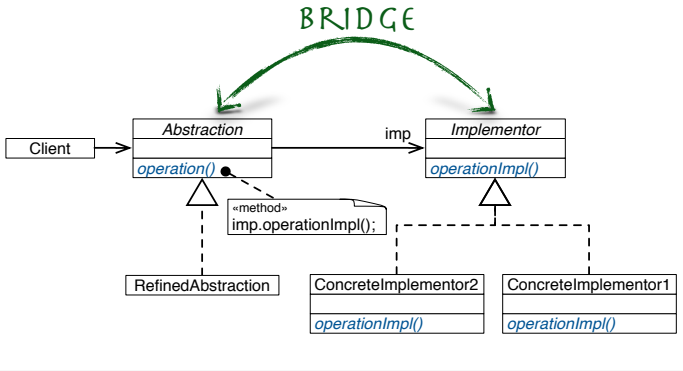
Several problems:

- Implementation (e.g., `GnomeWindow`) bound to abstraction (`Window`)
- Code duplication and proliferation of classes

The Bridge Design Pattern

Decouple an **abstraction** from its **implementation**.
So that the two can vary independently.

Bridge Design Pattern - Structure



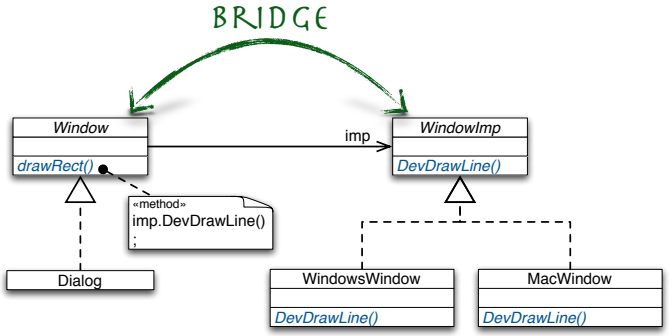
Combine inheritance and object composition.

Combine inheritance and object composition:

- Use inheritance to model variations of the abstraction.
- Use object composition to abstract from implementation variations.

The interface defines **all** methods the client will ever use. I.e., when designing the Implementor interface we have to foresee the needs of all future clients and the capabilities of future subclasses.

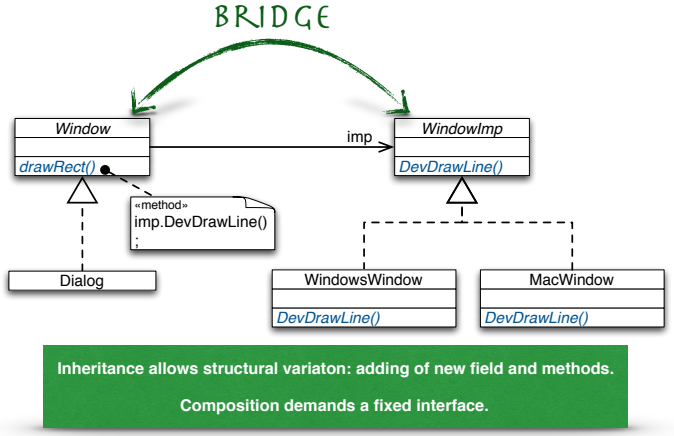
Bridge Design Pattern - Illustrated



The Rationale Underlying the Solution:

- Object composition and inheritance provide different trade-offs for expressing variations.
- Object composition is used to implement dynamic variations with a fixed interface.
  Implementation variations are more of this kind; although not always…
- For static variations inheritance is preferred, because it supports structural variations.
  Abstraction variations are mostly static.
  They often imply variation of structure.

Bridge Design Pattern - Illustrated



Advantages

Decoupling interface and implementation:

- Implementation can be configured at run-time.
- The implementation in use is hidden inside the abstraction.

Improved extensibility:

- Both abstractions and their implementations become independently extensible by subclassing without a class proliferation.
- Different abstractions and implementations can be combined.

Takeaway

- The Bridge Pattern instructs to use object composition to bridge between two inheritance hierarchies when you need to combine two kinds of variations of an object type.
- The Bridge Pattern allows to vary an abstraction and its implementation independently of each other.
- Works well as long as there is no dependency between the implementation on abstraction variations, i.e., if they do not vary co-variantly.