Bridge

Bridge

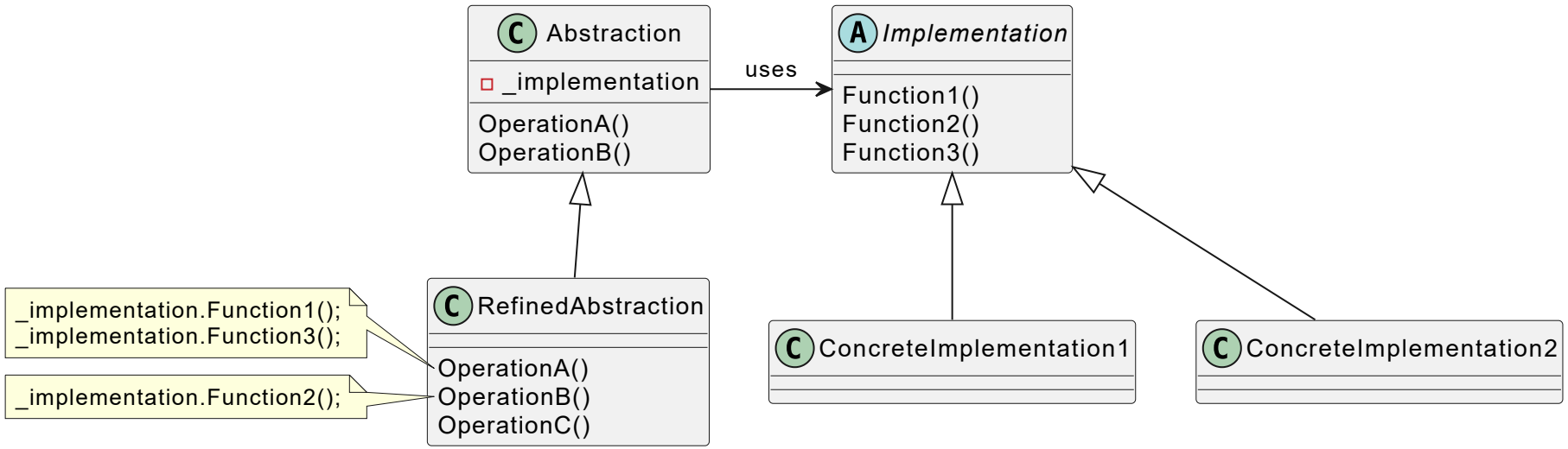
- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*

Bridge

- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*
- hence the name Bridge - because it looks like a bridge (😇)

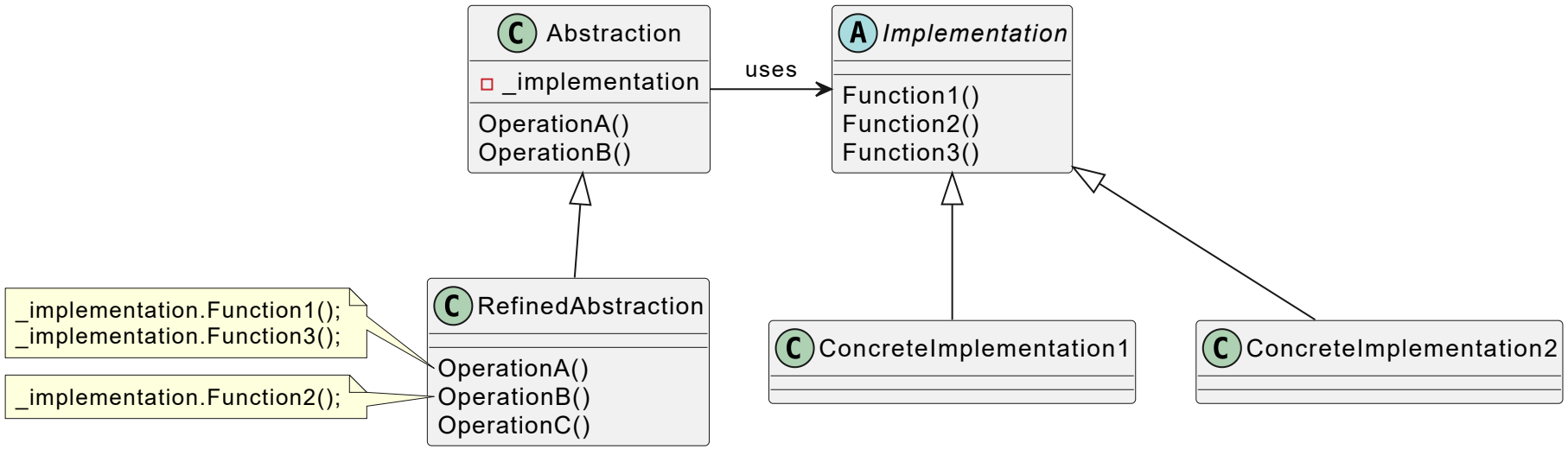
Bridge

- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*
- hence the name Bridge - because it looks like a bridge (😇)
- we could say: it's bridging something that is platform specific (implementation) from something that is platform independent (abstraction)



Bridge

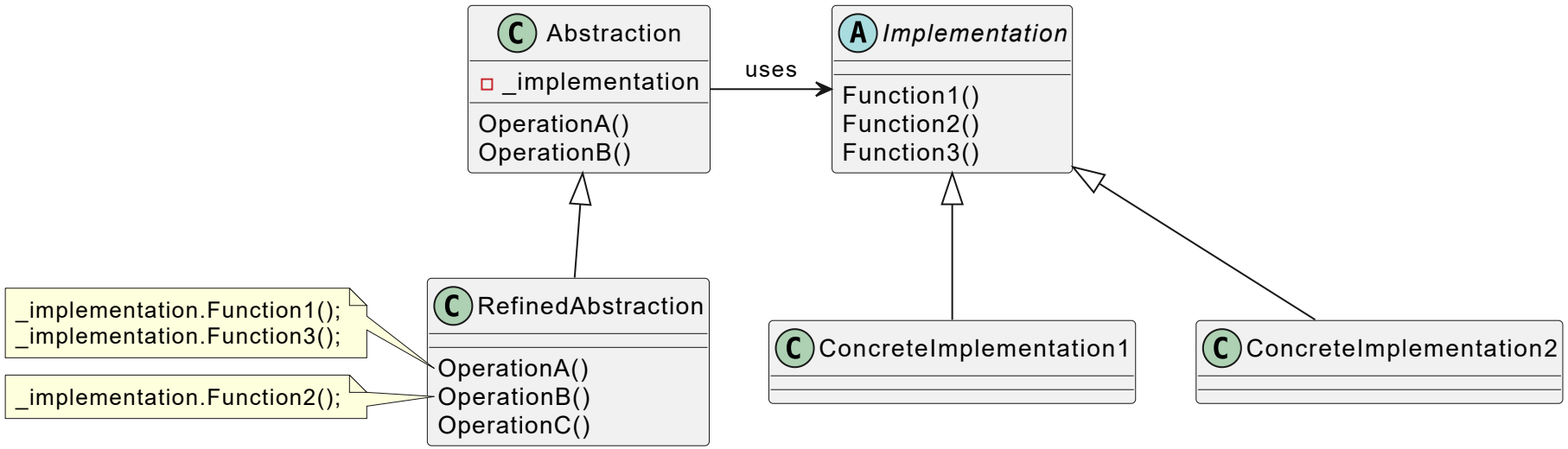
- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*
- hence the name Bridge - because it looks like a bridge (😇)
- we could say: it's bridging something that is platform specific (implementation) from something that is platform independent (abstraction)



- *Refined abstractions* use *Conrete Implementation* calling their methods (not necessarily one-to-one!)

Bridge

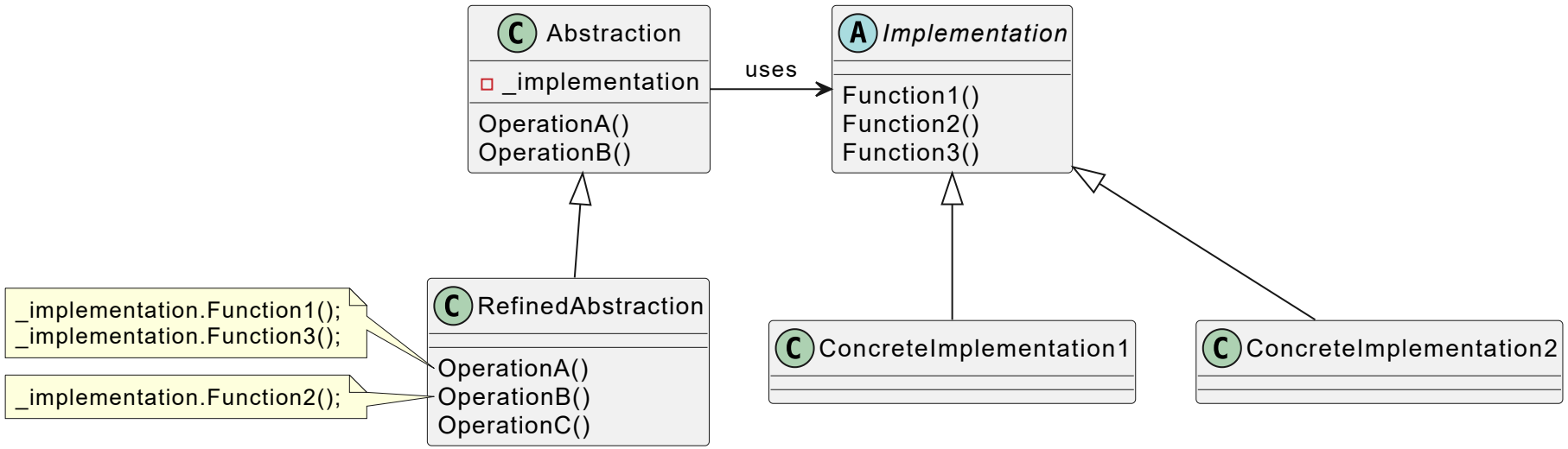
- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*
- hence the name Bridge - because it looks like a bridge (😇)
- we could say: it's bridging something that is platform specific (implementation) from something that is platform independent (abstraction)



- *Refined abstractions* use *Conrete Implementation* calling their methods (not necessarily one-to-one!)
- *Refined abstractions* may extend the interface of *Abstraction*

Bridge

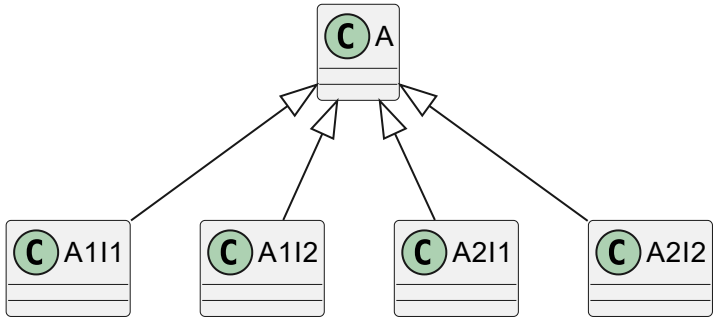
- GoF: *"decouple an abstraction from its implementation so that the two can vary independently"*
- hence the name Bridge - because it looks like a bridge (😇)
- we could say: it's bridging something that is platform specific (implementation) from something that is platform independent (abstraction)

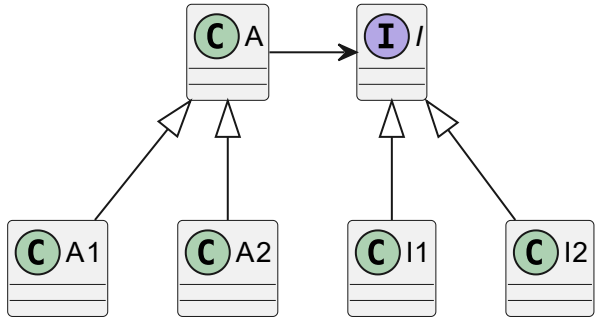


- *Refined abstractions* use *Conrete Implementation* calling their methods (not necessarily one-to-one!)
- *Refined abstractions* may extend the interface of *Abstraction*
- (typically) *Implementation* provides only primitive operations and *Abstraction* defines higher-level operations based on those primitives

Bridge

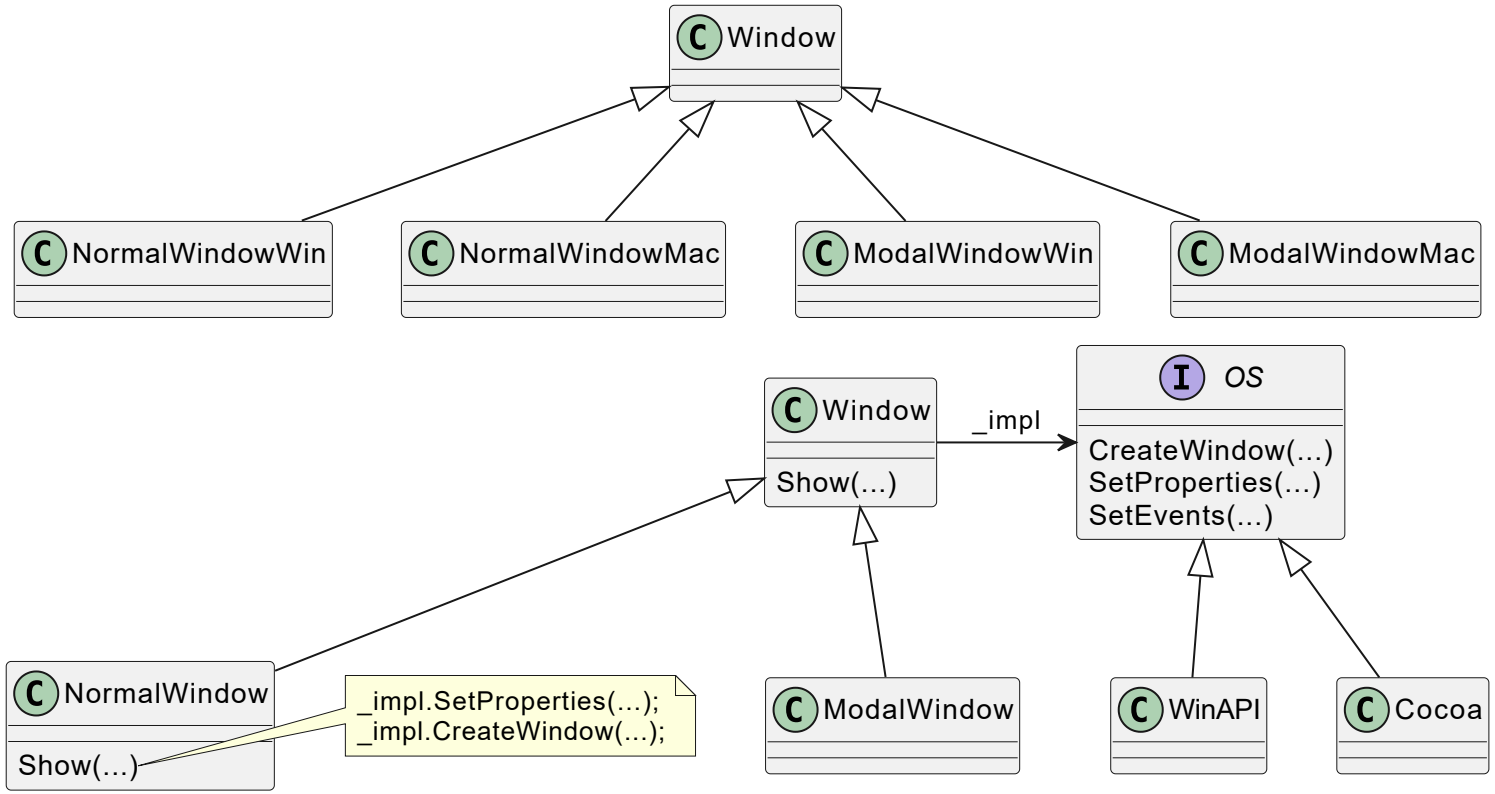
- *"When an abstraction can have one of several possible implementations, the usual way to accommodate them is to use inheritance"*
- *"Clients should be able to create a window without committing to a concrete implementation"*



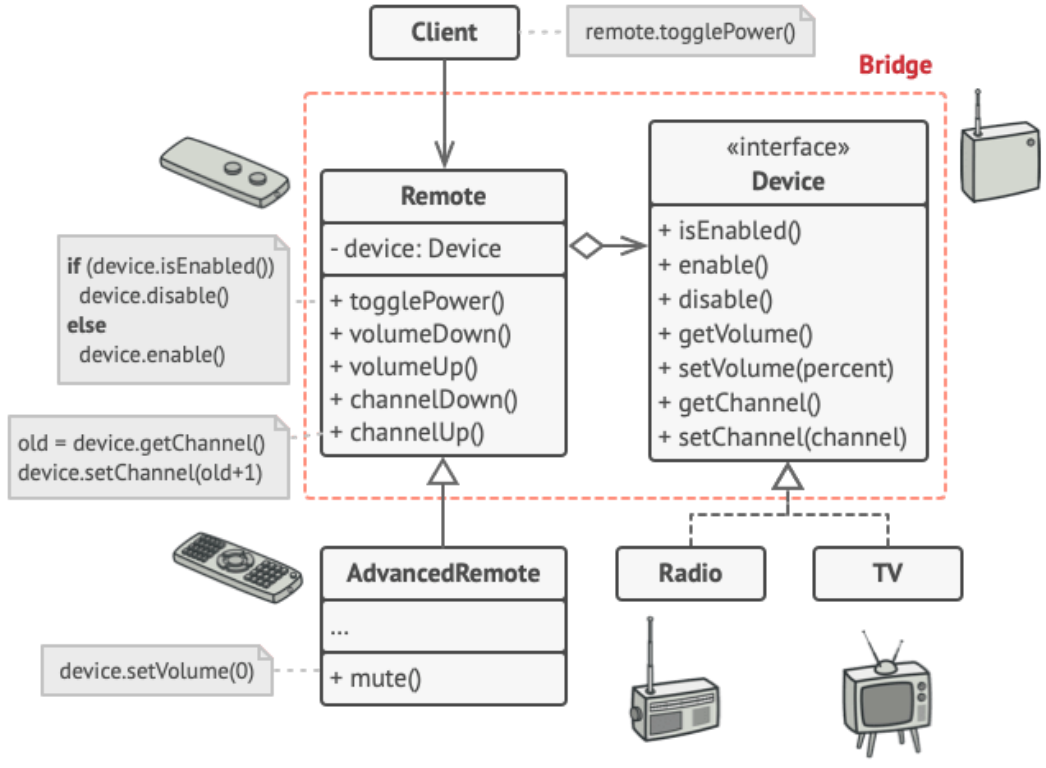


Bridge

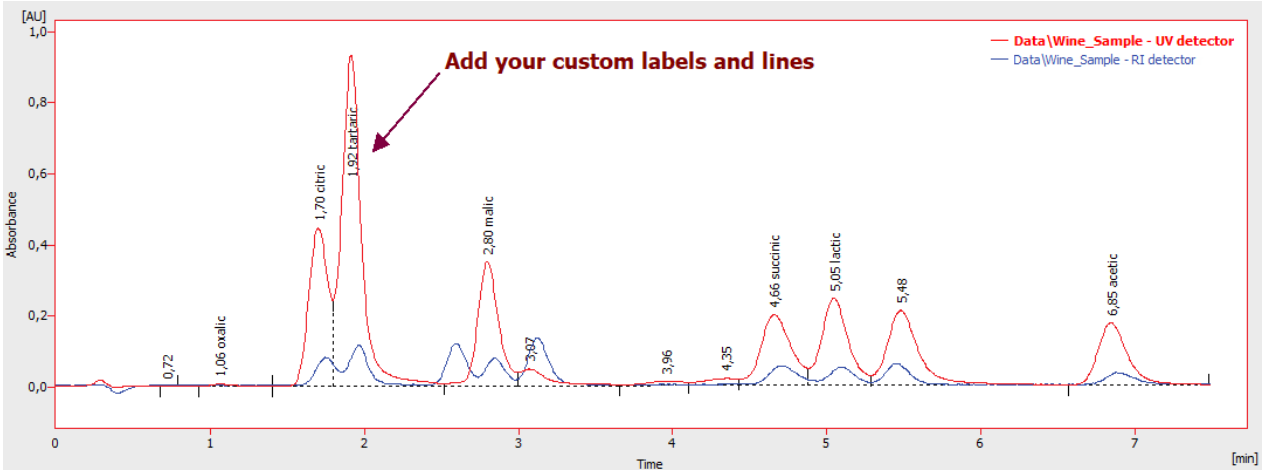
- *"When an abstraction can have one of several possible implementations, the usual way to accommodate them is to use inheritance"*
- *"Clients should be able to create a window without committing to a concrete implementation"*



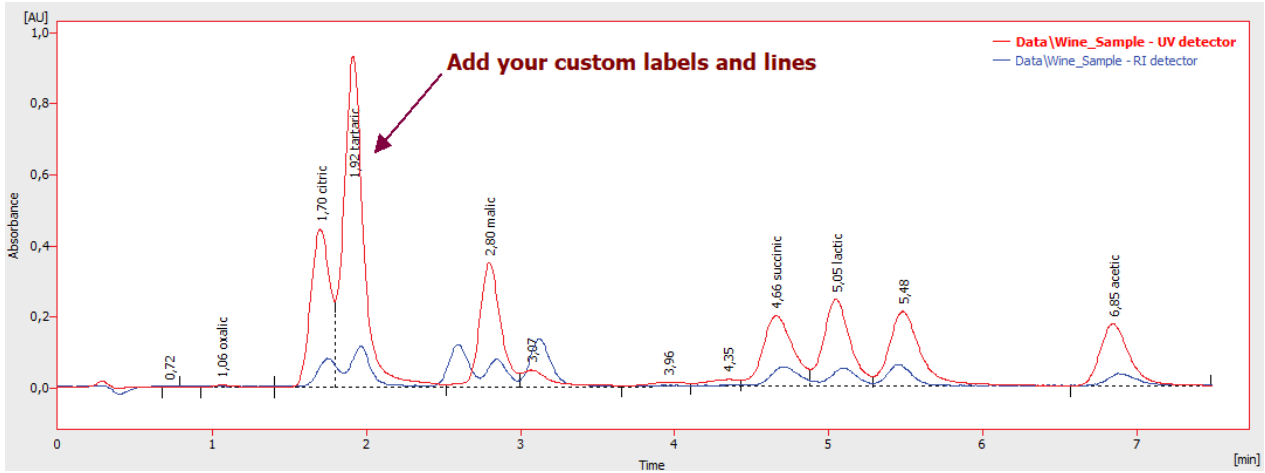
Bridge - example #1

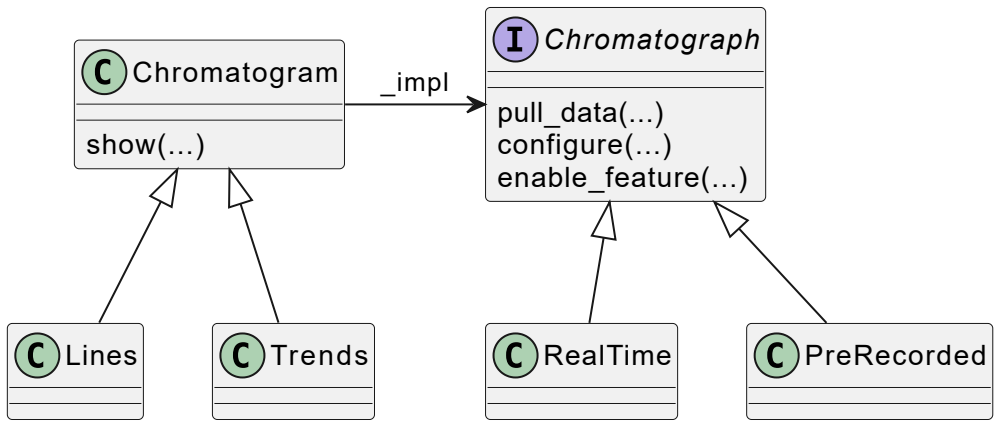


Bridge - example #1

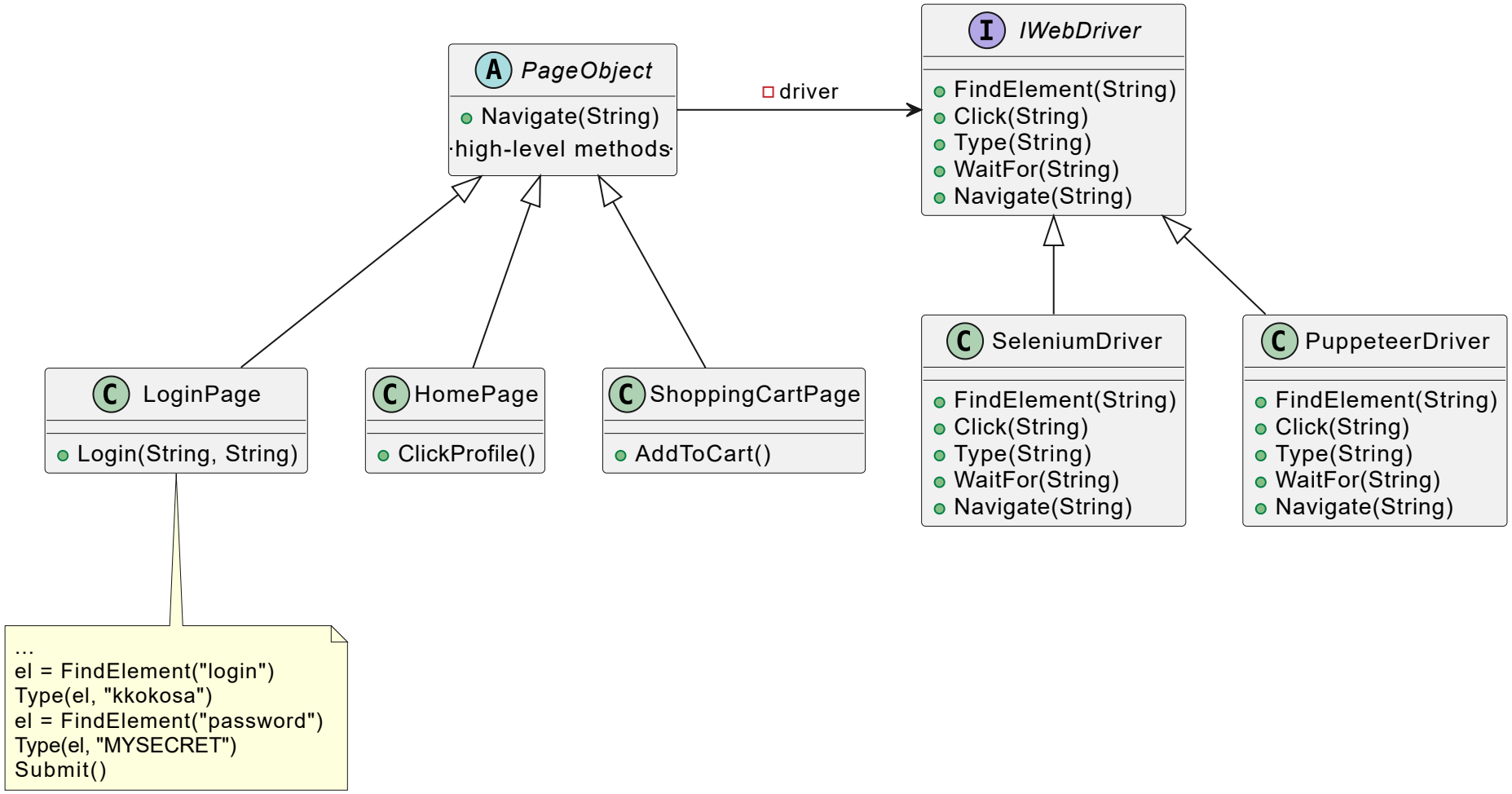


Bridge - example #1





Bridge - example #2



Bridge - example #3

- we send a message/abstraction (short, long) through various channels/implementations (e-mail, SMS)

Bridge

- when we have abstractions on both sides - *Bridge*, and when on one side - *Strategy*
- special feature that differs from *Strategy pattern* - use/combination of methods with Implementation