الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

المحاضرة الرابعة

كلية الهندسة المعلوماتية

مقرر تصميم نظم البرمجيات

Design Patterns 1: Strategy Pattern

د. رياض سنبل

Design Patterns

- Designing object-oriented software is hard, and designing reusable object-oriented software is even harder.
- You also want to avoid redesign, or at least minimize it
- **design pattern**:
  A standard **solution** to a common software **problem** in a context.
  - describes a recurring software structure or idiom
  - is abstract from any particular programming language
  - identifies classes and their roles in the solution to a problem

Design Patterns

- **Creational Patterns** *(abstracting the object-instantiation process)*

  Factory Method
  Abstract Factory
  Singleton
  Builder
  Prototype

- **Structural Patterns** *(how objects/classes can be combined)*

  Adapter
  Bridge
  Composite
  Decorator
  Facade
  Flyweight
  Proxy

- **Behavioral Patterns** *(communication between objects)*

  Command
  Interpreter
  Iterator
  Mediator
  Observer
  State
  Strategy
  Chain of Responsibility
  Visitor
  Template Method

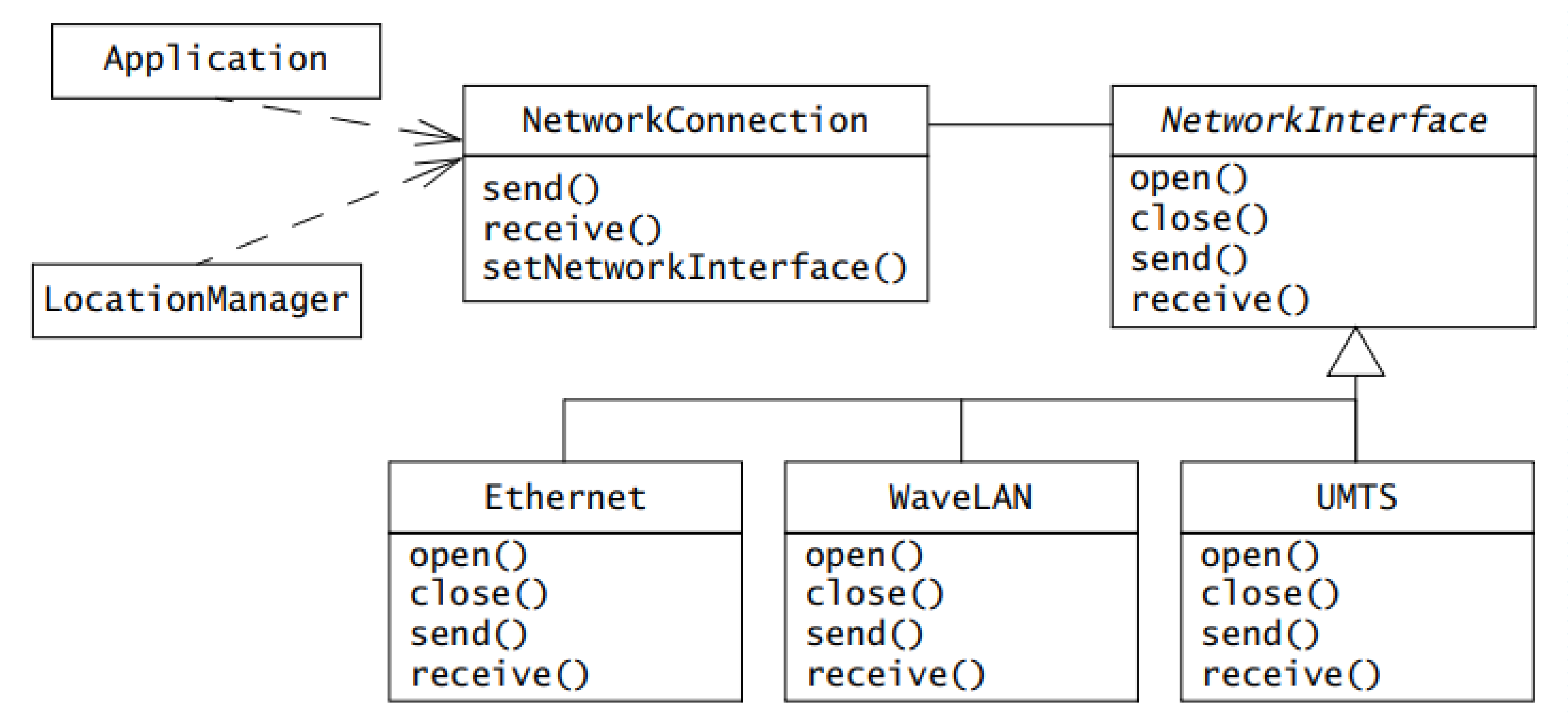
Strategy (or Policy) Pattern

Example

- Consider a mobile application running on a wearable computer that uses different networks :
  - *A car mechanic using the wearable computer to access repair manuals and maintenance records for the vehicle under repair.*
  - *The wearable computer should operate in the shop with access to a local wireless network as well as on the roadside using a third-generation mobile phone network, such as UMTS.*
  - *When updating or configuring the mobile application, a system administrator should be able to use the wearable computer with access to a wired network such as Ethernet.*

- This means that the mobile application needs to deal with different types of networks as it **switches between networks dynamically**, based on factors such as location and network costs.
- Assume that during the system design of this application, we identify the dynamic switching between wired and wireless networks as a critical design goal.
- Furthermore we want to be able to deal with future network protocols without having to recompile the application

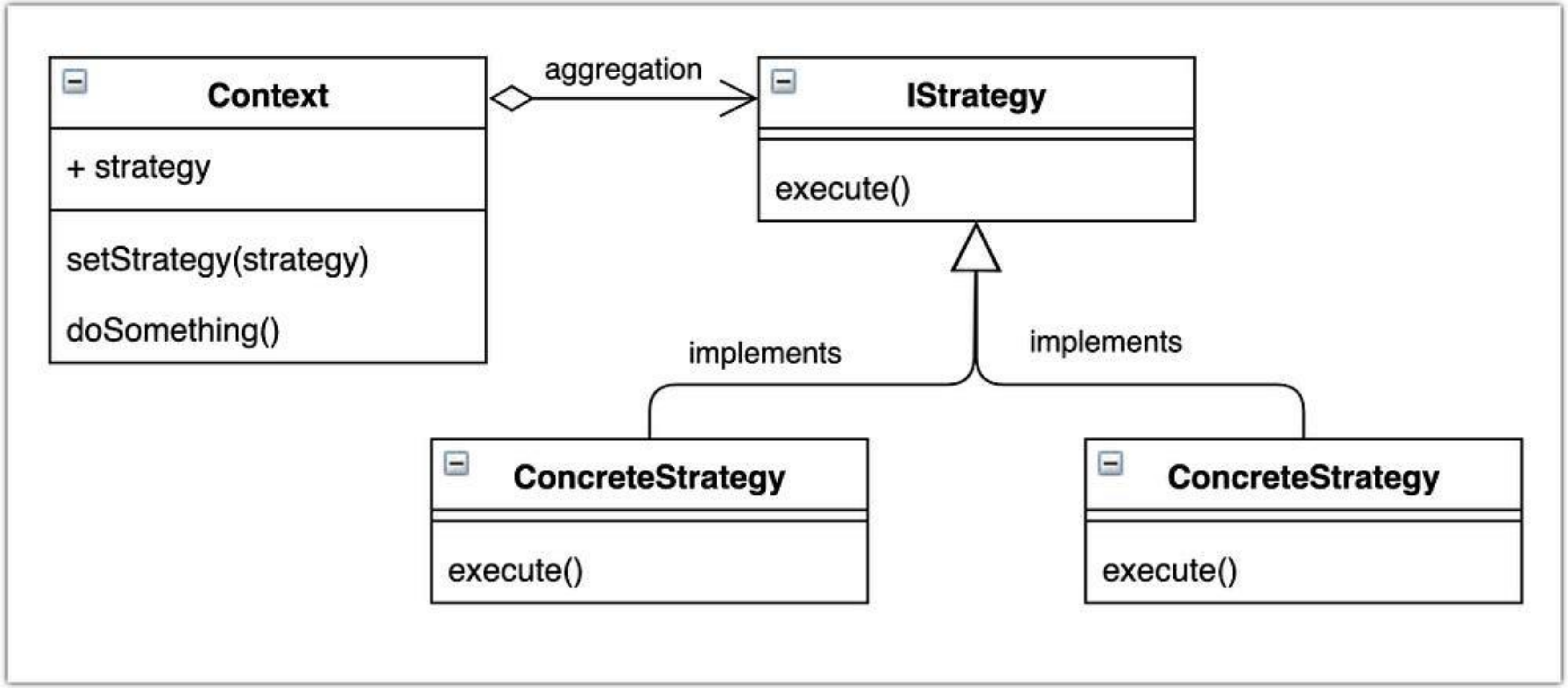
Final Solution



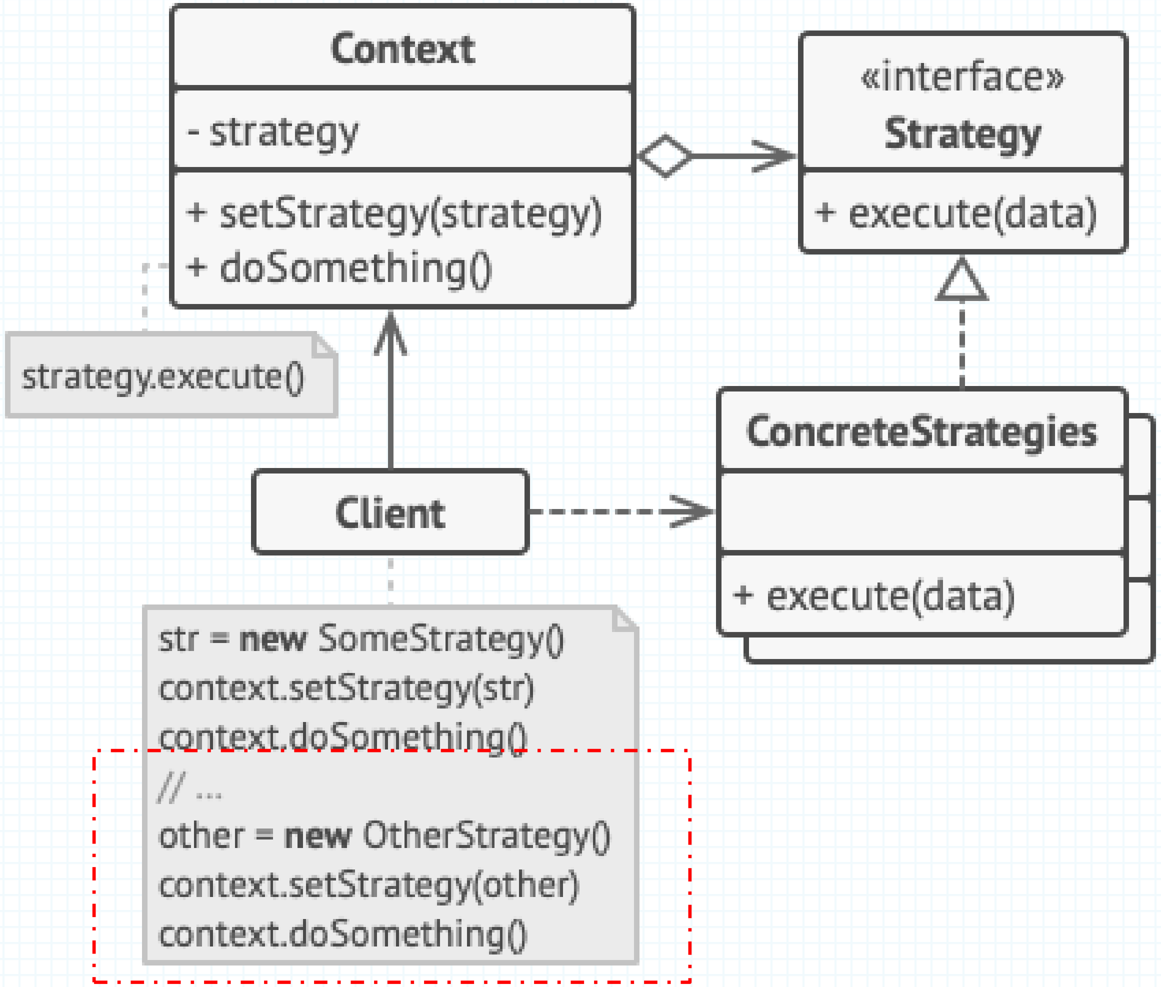
Strategy Pattern

- The intent of the Strategy Pattern is to:
  - (1) Define a family of algorithms, policies, logics, etc
  - (2) Encapsulate each one,
  - (3) Make them interchangeable within that family
- Strategy Pattern lets the algorithm **vary independently from the clients** that use it
- The Strategy Pattern enables an algorithm's behavior to be **selected at runtime**

The structure



The structure



```
str = new SomeStrategy()
context.setStrategy(str)
context.doSomething()
// ...
other = new OtherStrategy()
context.setStrategy(other)
context.doSomething()
```