Strategy Pattern

Behavioural Pattern

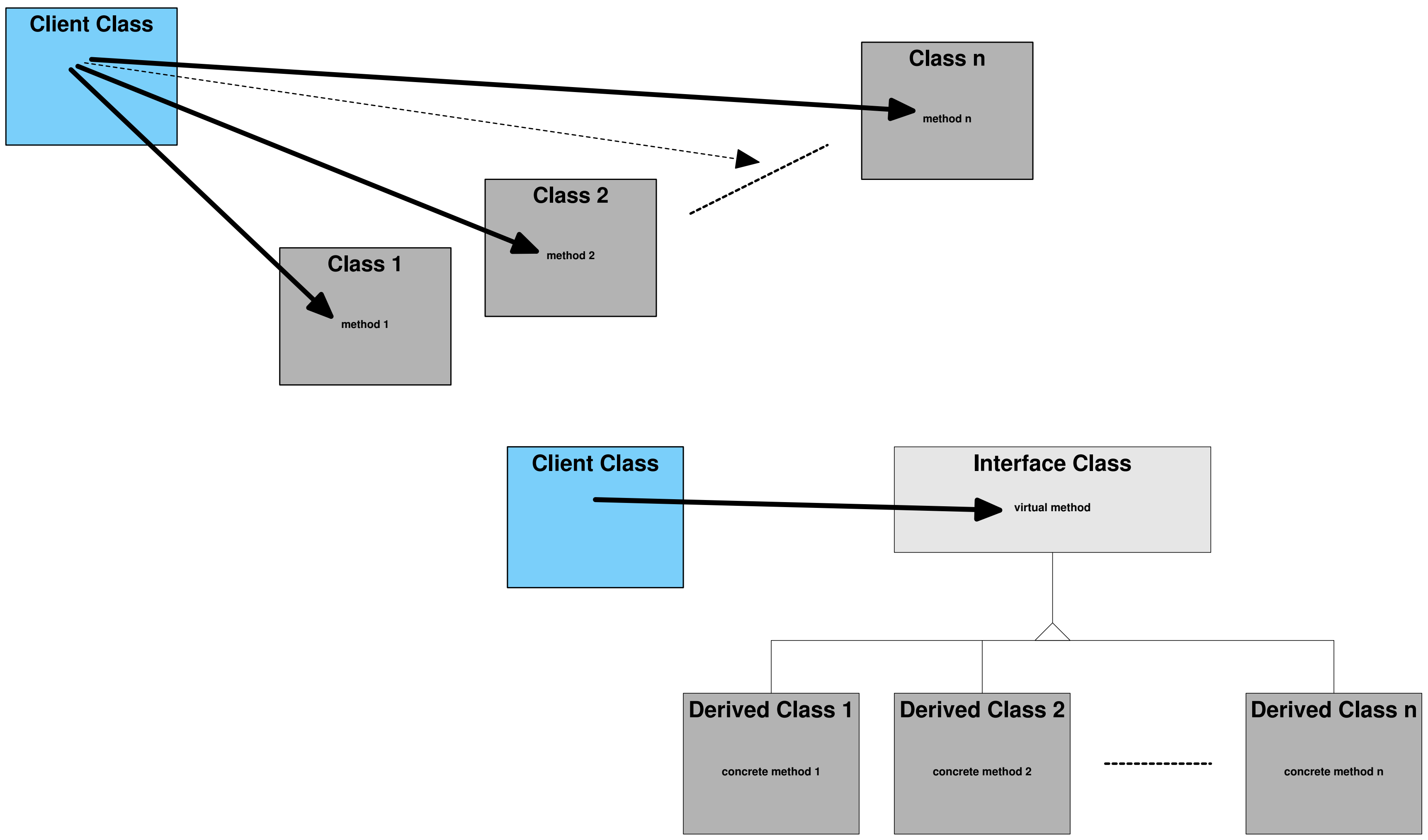
Intent

- To implement different algorithms to solve the same problem and to be able to swap between these algorithms at runtime.

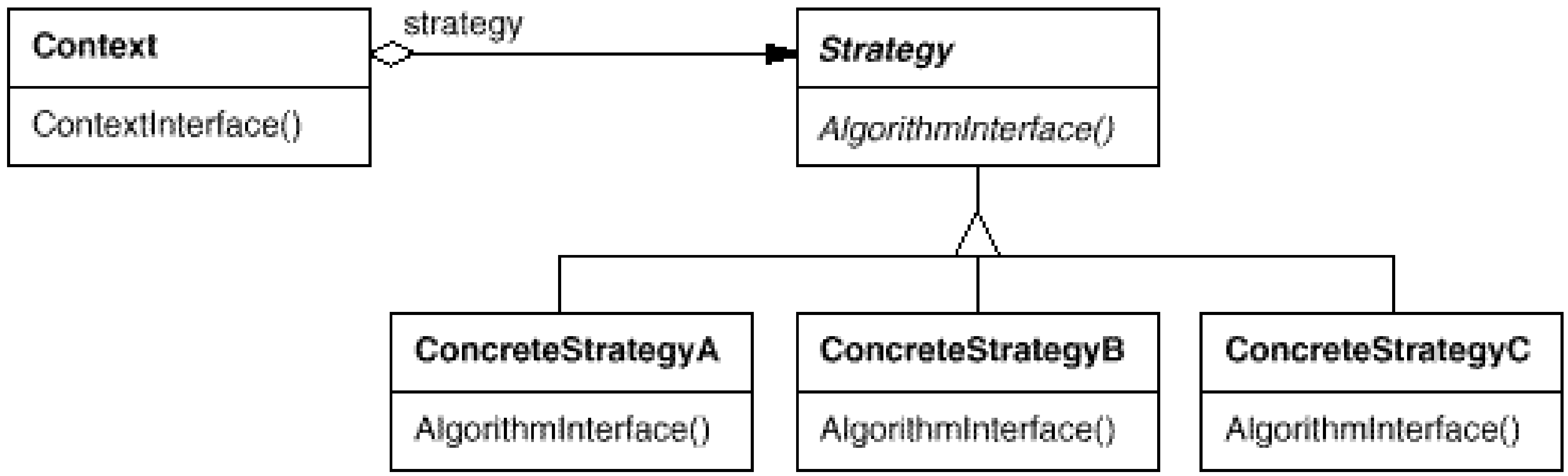
- To consolidate scattered conditional behaviour by applying polymorphism.

- To control coupling

Controlling coupling



Strategy Pattern

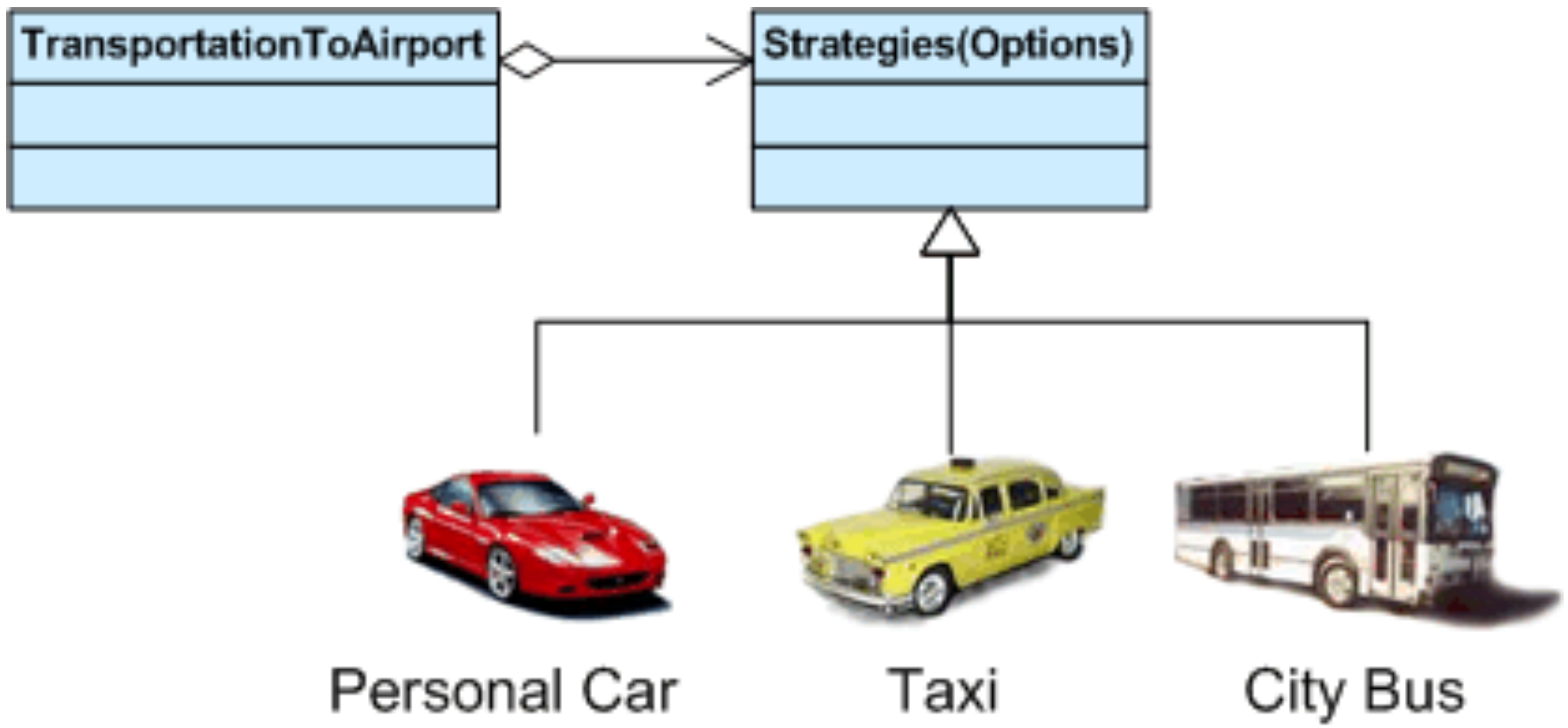


Participants

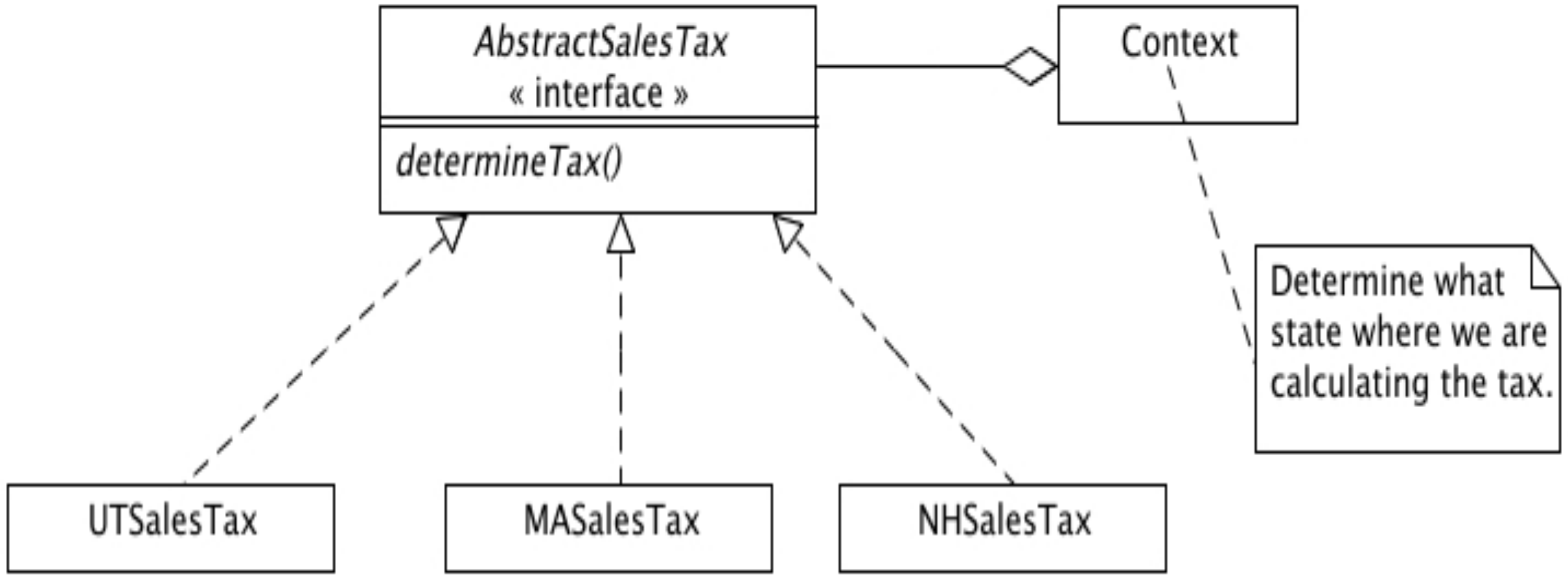
- Context
  - The class that uses different strategies
- Strategy
  - The common interface for different strategies
- Concrete Strategy
  - Implementation of a strategy

Example

- A Strategy include a decision structure to decide on what strategy is to be used.



Example


AbstractSalesTax
« interface »
determineTax()
Context
Determine what state where we are calculating the tax.
UTSalesTax
MASalesTax
NHSalesTax

Ways of Coupling between the Context and the Strategy Interface

- Pass the data that has to be operated on to the algorithm via a parameter
- Pass a pointer the data that has to be operated on to the algorithm via a parameter
- Pass a pointer to the Context to the algorithm and allow the algorithm to manipulate the data directly.
- Strategy store a permanent reference to the Context and manipulate the data directly.

Disadvantages of Strategy

- High coupling between Strategy and Context
  - Interface must cater for all that all strategies need and therefore, is not small.
  - Not all strategies use the whole interface resulting in parameters initialised an never used
- When strategies are shared between different clients at the same time there is a risk of unwanted side-effects.

Factory Method vs Strategy

- Both use Polymorphism to implement variation
  - Factory Method – to create the correct object
  - Strategy- to execute the correct algorithm

Template Method vs Strategy

- Template Method manipulates generic objects while Strategy manipulates defined objects generically.
- Method of varying behaviour:
  - Template: Inheritance; - changes part of algorithm
  - Strategy: Delegation; - select a whole algorithm
- Examples:
  - Template Method: Quiksort Persons/Numbers/etc.
  - Strategy: Quicksort/merge-sort/etc. numbers

Strategy vs State

- Same structure and same techniques to achieve their respective goals
- Different intents
  - Strategy is about applying different algorithms to achieve a fixed outcome
  - State is about achieving different outcomes based on the current state.

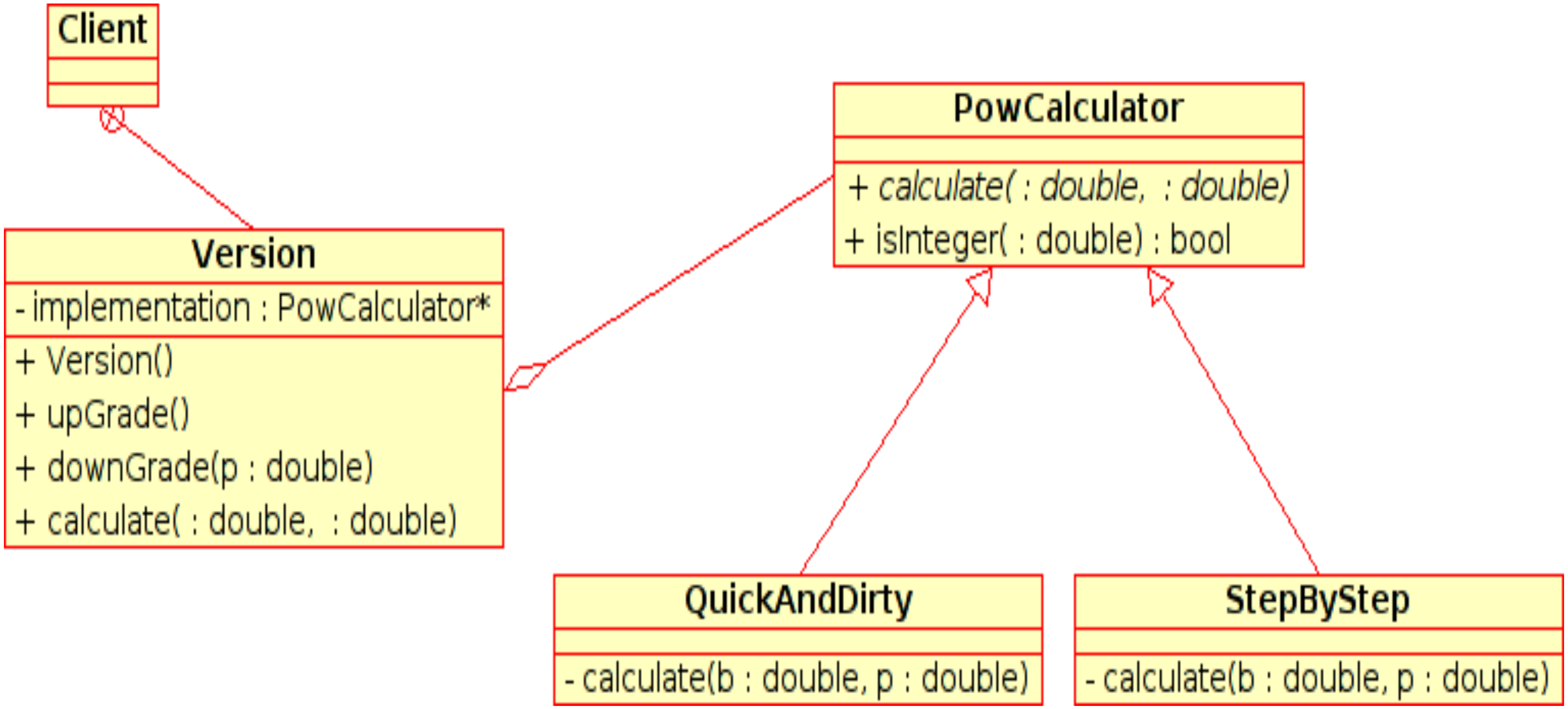
Flyweight and Strategy

- Strategy objects often makes good Flyweights

Lecture Example

An application that decide at runtime what strategy should be used for calculating x^y.

UML Class Diagram



Client
Version
- implementation : PowCalculator*
+ Version()
+ upGrade()
+ downGrade(p : double)
+ calculate(: double, : double)
PowCalculator
+ *calculate(: double, : double)*
+ isInteger(: double) : bool
QuickAndDirty
- calculate(b : double, p : double)
StepByStep
- calculate(b : double, p : double)

Participants

- Context = Version
  - Uses the different strategies
- Strategy = PowCalculator
  - Interface to the concrete stratgies
- Concrete Strategies: QuickAndDirty, StepByStep
  - Exponentiation with pow -function
  - Exponentiation with repeated multiplication