Strategy:
A Behavioral Design Pattern

1

What is a Design Pattern?

A design pattern deals with interactions between individual software components

They are recurring solutions to common problems of design!

2

Types of Design Patterns

- Creational
- Structural
- Behavorial
  - **Strategy** is a behavioral design pattern
    - Also know **Policy pattern**

3

Strategy

In a Strategy design pattern, you will:

- Define a family of algorithms
- Encapsulate each one
- Make them interchangeable

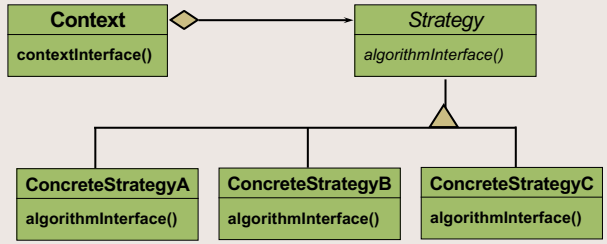
4

You should use Strategy when:

- You have code with a lot of algorithms
- You want to use these algorithms at different times
- You have algorithm(s) that use data the client should not know about

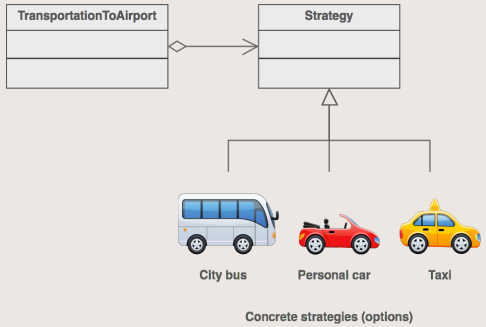
5

Strategy Class Diagram



6

Example



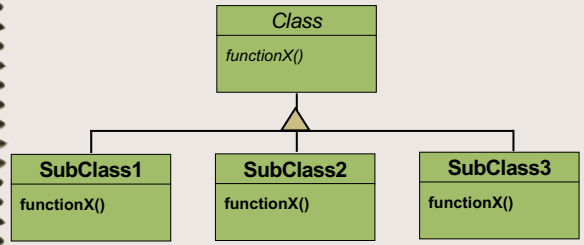
7

Strategy vs. Subclassing

- Strategy can be used in place of subclassing
- Strategy is more dynamic
- Multiple strategies can be mixed in any combination where subclassing would be difficult

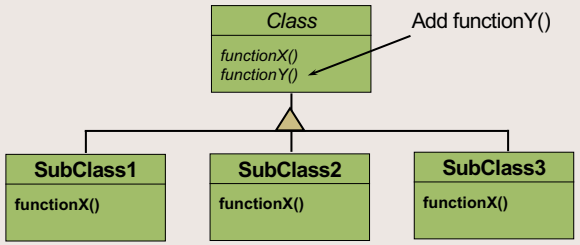
8

Subclassing



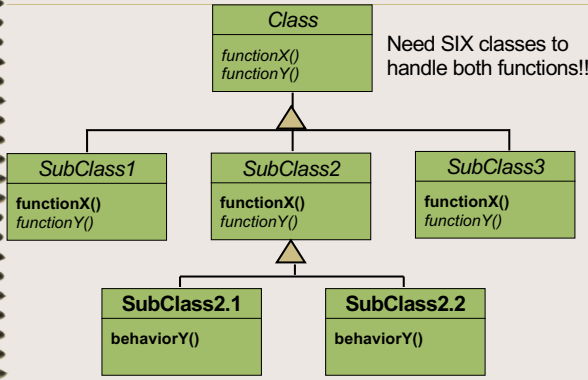
9

Add a function



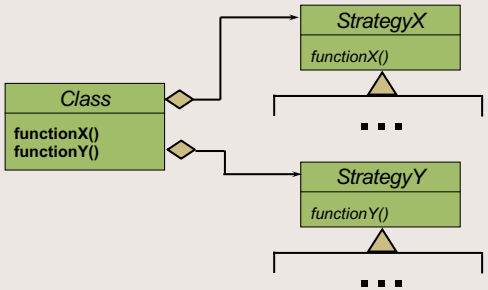
10

What happens?



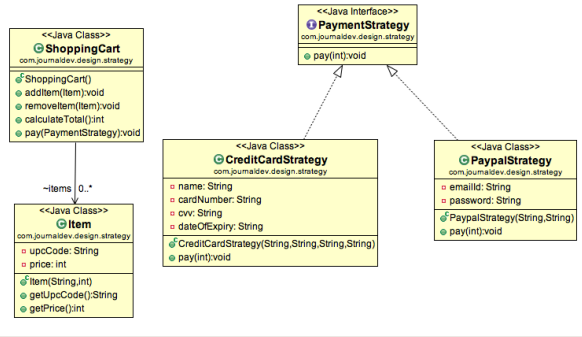
11

Strategy makes this easy!



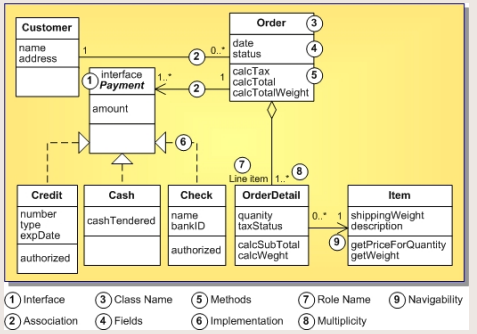
12

Example: Payment strategy



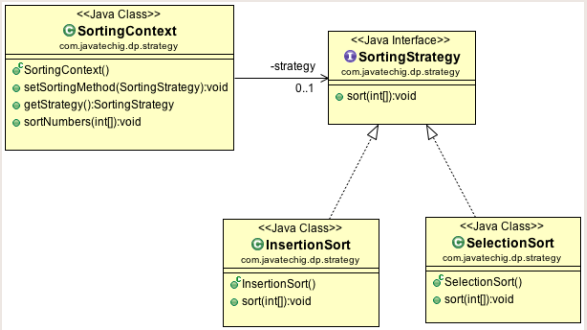
13

Payment strategy: More…



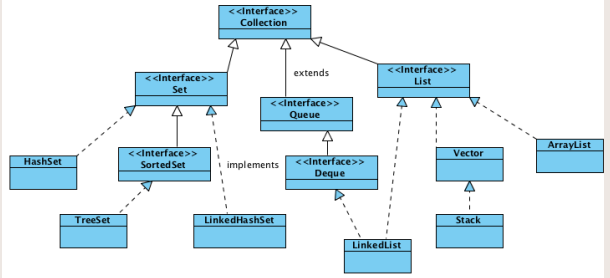
14

Example: Sort strategy



15

Example: Java API collection Strategy for sorting???



16

Benefits of Strategy

- Eliminates conditional statements
  - Can be more efficient than case statements
- Choice of implementation
  - Client can choose among different implementations with different space and time trade-offs

17

Benefits of Strategy

- Families of related algorithms
- Alternative to subclassing
  - This lets you vary the algorithm dynamically, which makes it easier to change and extend
  - You also avoid complex inheritance structures

18

Drawbacks of Strategy

- Clients must be aware of different strategies
  - Clients must know how strategies differ so it can select the appropriate one
- Communication overhead between strategy and context
  - Sometimes the context will create and initialize parameters that are never used

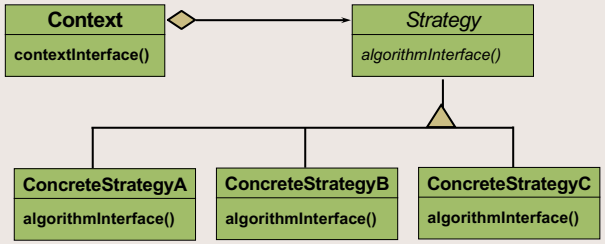
19

Drawbacks of Strategy

- Increased number of objects
  - if the algorithm differences are simple, the extra classes add extra complexity

20

Implementation Issues



21

Implementation Issues

- Concrete Strategy needs efficient access to data
- Should you use a template?
- Should you make strategy objects optional?

22

Thank you to

Provider of the PowerPoint graphs:

http://vik.ktu.lt/moduliai/

23

What have we learned today about strategy?

24