Design Pattern

(OOProg chapter 3)

Peter Levinsky, IT Roskilde

20.10.2024

Design Pattern – Categories

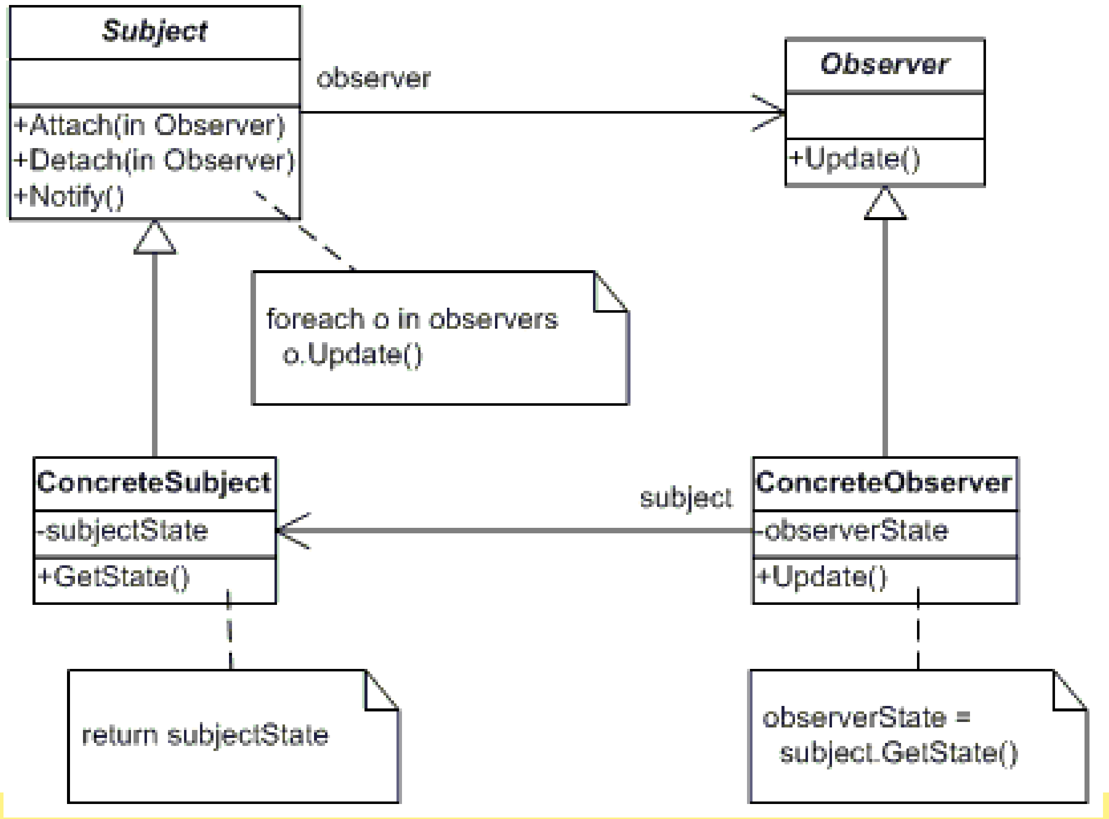
- **Creational Patterns**
  - Factory, Abstract Factory, Singleton …
- **Structural Patterns**
  - Adaptor, Proxy, Facade, Decorator …
- **Behavioral Patterns**
  - Observer, Template, Strategy, State …
- **Concurrency patterns**
  - Monitor, Lock, Thread Pool

Design Pattern – Behavioural Patterns

- **Observer**

  Problem: How to handle different kinds of subscriber objects are interested in the state changes or events of a publisher object

- Solution:



Design Pattern – Behavioural Patterns

- **Observer** - the C# way of doing it

The one that Observe

```
…
XX x = new XX();

// Register as observer
x.PropertyChanged += Update;

….
protected void Update(object sender,
                      PropertyChangedEventArgs arg)
{
. . .
}
```

To be Observered

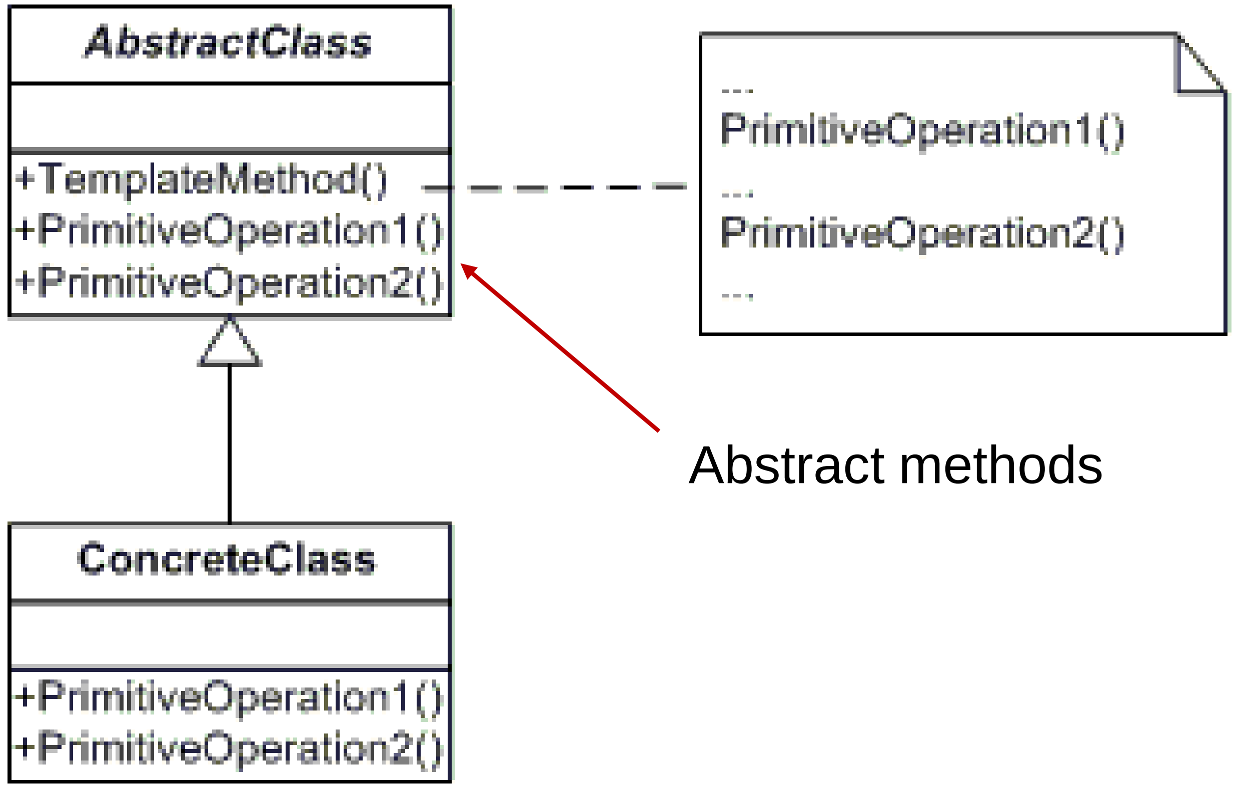
```
Class XX : INotifyPropertyChanged
{
. . .
// Attach, Deattach
public event PropertyChangedEventHandler PropertyChanged;

// notify
protected virtual void OnPropertyChanged(string propertyName)
{
   PropertyChanged?.Invoke(this,
                           new PropertyChangedEventArgs(propertyName));
}
}
```

Design Pattern – Behavioural Patterns

- **Template** **(seen at the TCP server generalisation)**
  Problem: How to reuse a skeleton of an algorithm in an operation

- Solution:

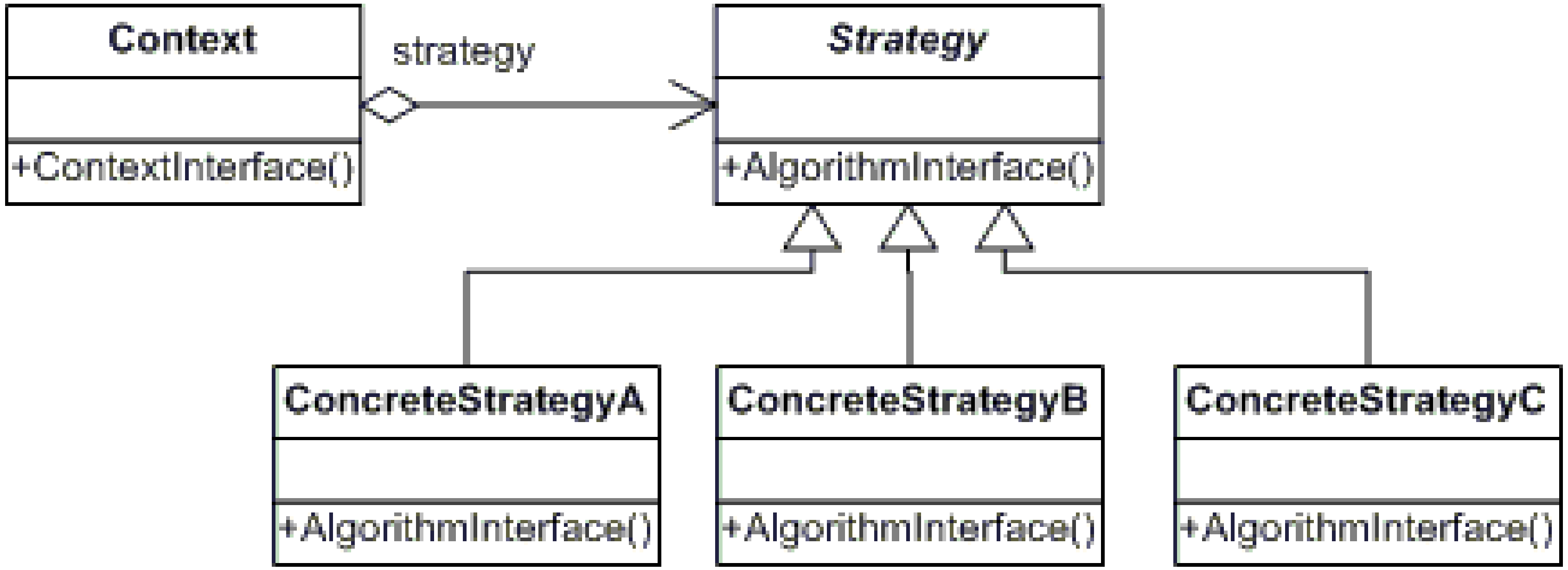


Abstract methods

Design Pattern – Behavioural Patterns

- **Strategy**
  Problem: How to interchange part of algorithm dynamically

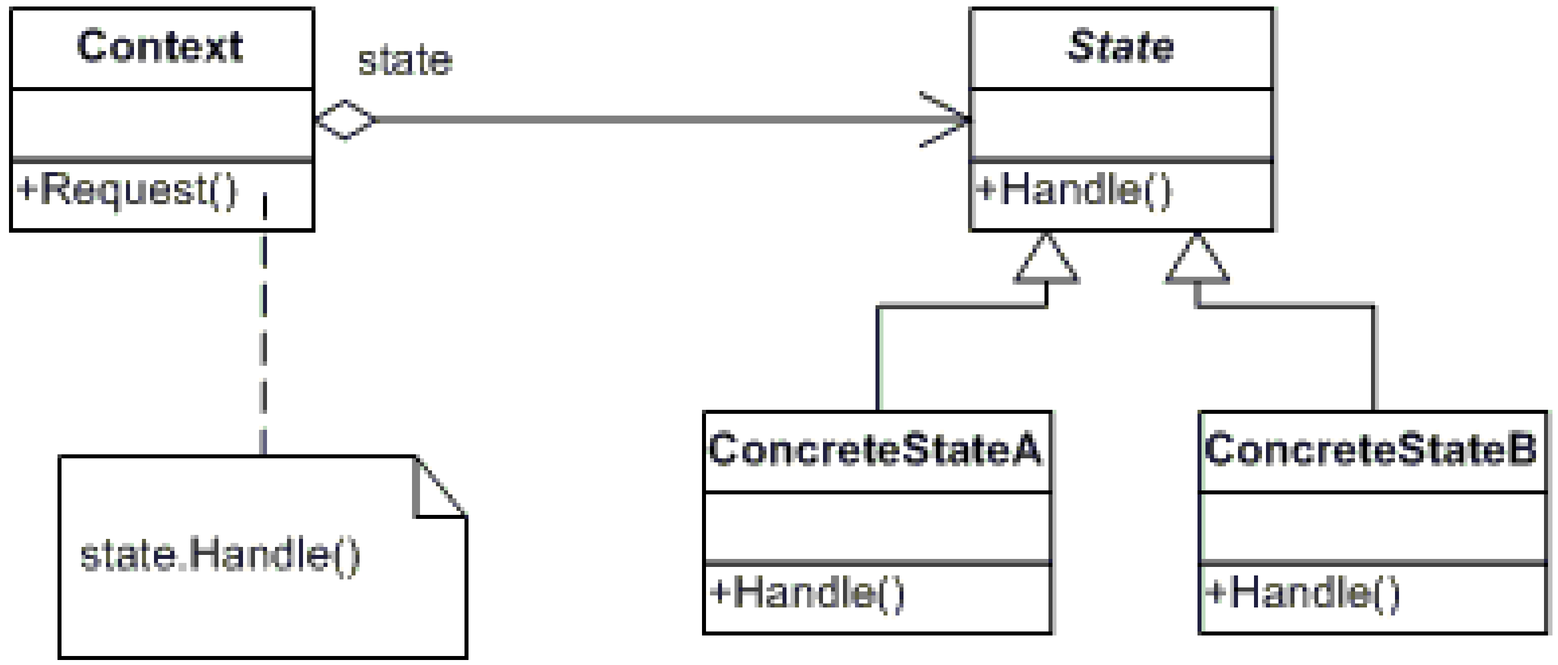
- Solution:



Design Pattern – Behavioural Patterns

- **State (seen at the snake game)**
  Problem: How to Allow an object to alter its behaviour when its internal state changes

- Solution:



Demo

- Demo of Observer, Template og Strategy

- Training: Exercises: 3.7 (Composite), 3.9 (Strategy)

- ***And of course the Mandatory Assignment***