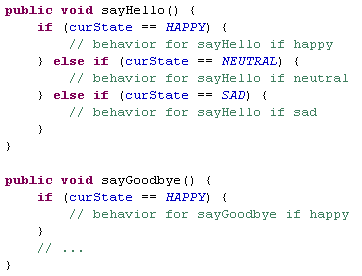
CS1 Task 12: State Pattern

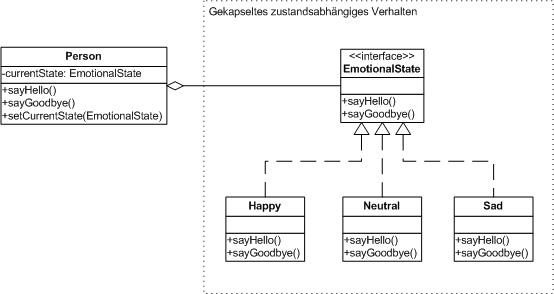
# Konzept

Das State Pattern ermöglicht die elegante Modellierung von zustandsabhängigen Verhalten eines Objekts. D.h., dass das Objekt sein äusseres Verhalten zur Laufzeit in Abhängigkeit von seinem internen Zustand ändern soll.

Ohne Pattern würde die Überprüfung des Zustandes und die Ausgabe des unterschiedlichen Verhaltens etwa folgendermassen implementiert werden:



Beim State Pattern werden die einzelnen Zustände in separate Klassen ausgelagert. Die Klassen enthalten die Definition über das Verhalten des Objekts in einem bestimmten Zustand. Werden die Methoden "sayHello" und "sayGoodbye" der Person aufgerufen, liefern sie die entsprechenden Rückgabewerte, welche in den einzelnen Zustandsklassen definiert sind, zurück.



|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# Vor-/Nachteile

## Vorteile

* Verständlichkeit:  
  Die Auslagerung jedes einzelnen Zustandes in eine eigene Klasse ist gut verständlich und übersichtlich.
* Erweiterbarkeit:  
  neue Zustände und neues Verhalten können auf einfache Art und Weise hinzugefügt werden.
* Trennung der Verantwortlichkeiten:  
  zustandsspezifisches Verhalten ist in mehrere Zustandsklassen aufgeteilt und somit die Verantwortlichkeiten klar getrennt.
* Wiederverwendbarkeit:  
  Zustände können auch von anderen Objekten verwendet werden.

## Nachteile

* Erhöhte Klassenanzahl
* Erhöhter Implementationsaufwand

# Anwendungsfälle allgemein

Das State Pattern findet in folgenden Fällen Anwendung:

* Ein Objekt soll sein äusseres Verhalten zur Laufzeit aufgrund seines inneren Zustands ändern können.
* Ein Objekt besitzt eine Reihe von Methoden ähnlicher Struktur, die sich aus immer gleichen Anweisungen zusammensetzen.
* Beispiele:
  + Werkzeuge in Zeichnungstools (Zustand der Zeichenfläche/Editors)
  + Stoppuhr
  + usw.

# Beispiel mobile Applikation "Apollo"

In unserer mobilen Applikation wird die Menüsteuerung mittels State Pattern realisiert. Auf der Einstiegsseite stehen verschiedene Module der Applikation zur Verfügung. Via Buttonklick gelangt man zu den Unterseiten der einzelnen Module. Jede Unterseite beinhaltet einen "back"-Button, mit welchem man auf die vorherige Seite zurückspringen kann.

init()

init()

|  |  |  |
| --- | --- | --- |
|  |  |  |

back()

back()

Beim Klick auf einen Menüpunkt ruft die init()-Methode diejenige Unterseite auf, welche im entsprechenden Zustand definiert ist. Beim Starten der Applikation ist als Defaultzustand "MainMenuState" gesetzt.

Beim Klick auf den back-Button und dem damit verbundenen Aufruf der back()-Methode, wird diejenige übergeordnete Seite aufgerufen, welche im momentan aktiven Status definiert ist.

Bei jedem Aufruf der init()- oder back()-Methode wird der Zustand der nachfolgenden Seite gesetzt.

