State Pattern Aufgabe

:

Der folgende UML-Zustandsautomat modelliert ein Garagentor, das neben den gewöhnlichen Zuständen `<<offen>>` und `<<geschlossen>>` auch über einen weiteren Zustand `<<abgesperrt>>` verfügt. Abb. 1 zeigt alle möglichen Zustände sowie deren mögliche Folgezustände.

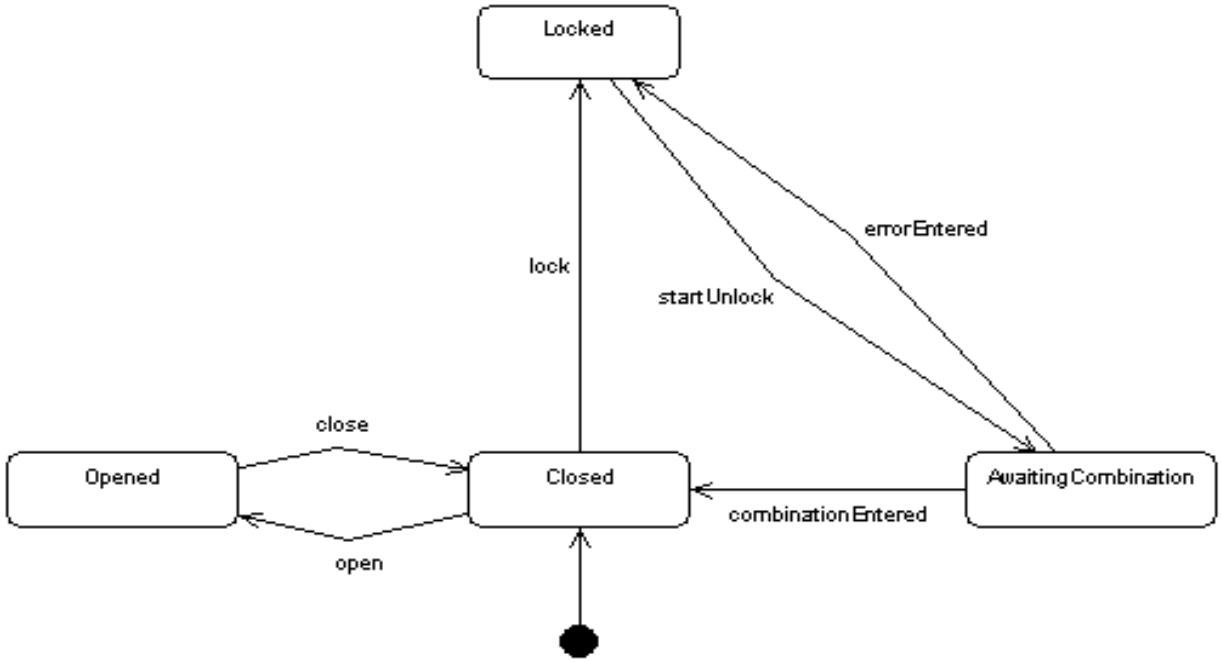

Abb. 1: Zustandsautomat für ein Garagentor

Um eine übersichtliche und flexible Architektur zu erreichen, ist es üblich, dass alle Zustände in eine eigene Klasse überführt werden, die allerdings alle von einer gemeinsamen abstrakten Klasse abgeleitet sind, oder ein gemeinsames Interface implementieren. Ob letztendlich eine abstrakte Klasse oder aber ein Interface verwendet wird, sind „Feindesign"-Implementierungsvarianten. In Java sind grundsätzlich beide Varianten möglich.

Abb. 2 zeigt eine mögliche Klassen-Struktur:

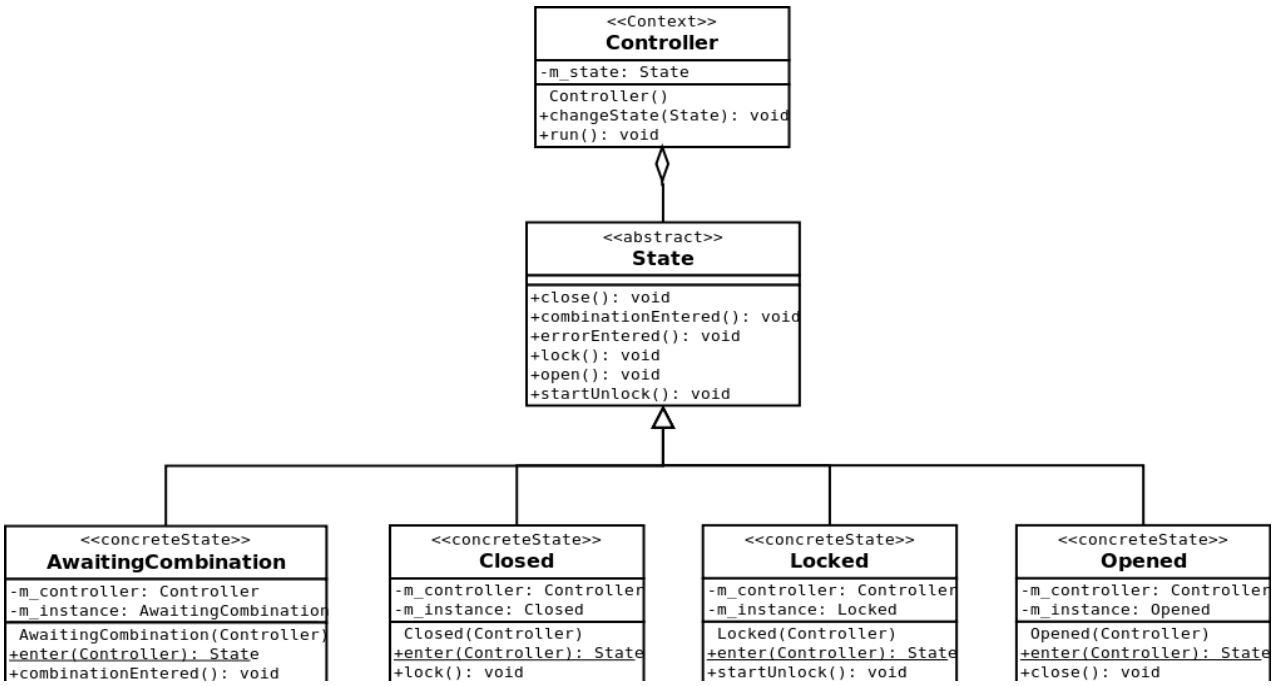

Abb. 2: Mögliche UML Klassenstruktur für State Pattern

Aus dem Zustandsautomaten und dem UML Klassendiagramm ist folgendes ersichtlich:

- Jeder Zustand wird durch eine Klasse repräsentiert
- Kanten werden zu Methoden, wobei jeder Zustand nur seine ausgehenden Kanten implementiert

Jetzt die eigentliche Aufgabe:

1. Implementieren Sie den Zustandsautomaten aus Abb. 1 gemäß State Pattern.
2. Schreiben Sie eine Klasse, die das umgesetzte Garagentor testet

Lösungshinweis:

Denken Sie kurz nach und betrachten nochmals Abb 2.: In jedem **konkretem** Zustandsautomaten gibt es jeden Zustand **nur genau einmal**. Dies kann man durch das Singleton Pattern realisieren, das in einer `enter()` Methode eines Zustand-Objektes implementiert wird. Durch diese Methode wird also erreicht, dass es genau eine Instanz eines Zustandes gibt. Dadurch bleiben die Instanzvariablen des Zustandsobjektes erhalten, wenn der Zustand mehrfach erreicht wird.

`enter()`–Methode aller <<concreteState>> Klassen:

```
public static State enter(Controller c) {

 if(m_instance == null) m_instance = new <<concreteStateName>>(c);

 return m_instance;
}
```

`<<concreteStateName>>` ist durch den entsprechenden Klassennamen zu ersetzen, z.B. `Opened`