Um die verschiedenen Zustände des Controllers abzubilden haben wir ein State-Pattern eingesetzt. Der Check In Controller hält dabei ein Status-Objekt. Die abstrakte Implementierung implementiert alle möglichen Methoden, wirft dabei aber nur eine IllegalStateException. Die konkreten Status-Klassen überschreiben die Methoden die für den aktuellen Status relevant sind und implementieren sie dabei korrekt.

Die IllegalStateExceptions sind dabei Hilfen bei der Implementierung und sollten bei korrekter Verwendung des Controllers nicht mehr im Betrieb des Programms auftreten.

Das Status Objekt hält immer eine Referenz auf den Controller, dadurch wird ermöglicht, dass die einzelnen Status sich selbst austauschen können und zwischengespeicherte Daten auch beim Übergang von einem Zustand in den anderen erhalten werden können, wenn sie im Controller gespeichert werden.

Da jederzeit nur ein Check In Vorgang bei einer Installation des Programms erfolgen kann haben wir ein Singleton-Pattern für den Check In Controller eingesetzt. Dadurch wird gewährleistet, dass nur ein Controller zu einem bestimmten Zeitpunkt verfügbar sein kann und gleichzeitig können wir, anders als bei statischer Implementierung der Klassen, objektorientiert programmieren.

Man kann also mit Vererbung arbeiten und die Methoden auf Objekten und nicht auf Klassen aufrufen, was eine eventuelle Umstellung auf mehrere Instanzen merklich vereinfachen würde.