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

Die „IllegalStateExceptions“ sind dabei Hilfen bei der Entwicklung und sollten bei korrekter Verwendung des Controllers zur Laufzeit nicht mehr ausgelöst werden.

Das Status Objekt hält immer eine Referenz auf den Controller, wodurch es ermöglicht wird, 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 dem Vererbungskonzept arbeiten und die Methoden auf Objekten und nicht auf Klassen aufrufen, was eine eventuelle Umstellung auf mehrere Instanzen merklich vereinfachen würde.