Methoden des Software Engineering

WS 2012/13, Prof. Dr. Prehofer, Christian Kroiß

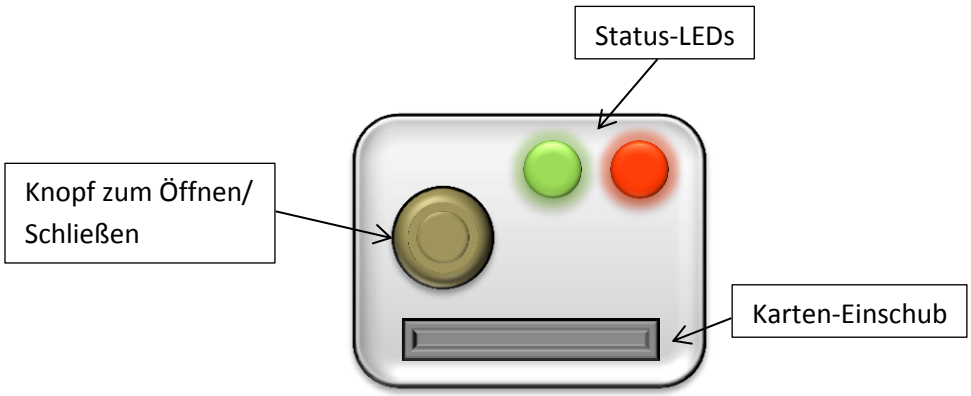
Diese Hausaufgabe wird nicht bewertet und muss daher nicht abgegeben werden! Eine Musterlösung wird in der Übung am 15.1. präsentiert und anschließend online verfügbar gemacht.

Übung 9 – Hausaufgabe

Laden Sie die Datei `homework_u9.zip` von der Homepage der Vorlesung herunter und importieren Sie das darin enthaltene Projekt in Ihren Eclipse-Workspace. Das Projekt enthält bereits alle für die Aufgabe benötigten Bibliotheken und Klassen.

Für diese Aufgabe soll ein etwas abgewandelter Ausschnitt aus der elektronischen Schließanlage vom Anfang der Vorlesung betrachtet werden.

Das Schloss, bzw. dessen Benutzerschnittstelle, sei für das aktuelle Szenario wie folgt aufgebaut:



Um ein Schloss bedienen zu können, muss eine berechtigte Karte im Karten-Einschub eingeschoben sein. Ist dies der Fall, dann kann durch einen Knopf auf dem Schloss-Panel das Schloss geöffnet bzw. geschlossen werden. Das Schloss bietet eine grüne und eine rote LED, die anzeigen, ob die eingelegte Karte korrekt gelesen wurde und berechtigt ist.

Das Schloss wird durch ein vom Hersteller bereitgestellten Treiber ins System eingebunden. Dieser Treiber implementiert das Interface `ILock`, das eine Methode `setLockController()` anbietet, mit dem der sogenannte *Lock Controller* des Systems mit dem Schloss-Treiber verbunden werden kann.

Das Schloss selbst übernimmt das Lesen der Kartennummer von der eingelegten Karte, falls ein Lesefehler auftritt, wird ein Fehler durch die Rote LED angezeigt und keine Daten werden an das System übertragen. Ist das Lesen erfolgreich, so wird die Kartennummer an den Lock Controller übermittelt. Der Lock Controller überprüft daraufhin die Berechtigung der eingelegten Karte und steuert durch die Methoden des Interfaces `ILock` das Verhalten des Schlosses.

Etwas detaillierter sei folgende Spezifikation für die Klasse `LockController` gegeben:

1. Beim Aufruf der Methode `LockController.cardInserted(ILock lock,String cardNumber)` wird per Datenbank-Abfrage über ein Data Access Object der Klasse `CardDAO` überprüft, ob die angegebene Kartennummer einer Karte entspricht, die für das angegebene Schloss freigeschaltet ist.
    a. Falls die Berechtigung nicht gegeben ist, dann wird die rote LED am Schloss eingeschaltet und der Zugriff wird verweigert. Dies ist in folgenden Fällen der Fall:
        i. Es kann in der Datenbank keine Karte mit der gelesenen Nummer gefunden werden.
        ii. Es wird eine Karte mit der gelesenen Nummer gefunden, aber diese hat keine Berechtigung für das Schloss, an dem die Karte eingelesen wurde.
        iii. Es tritt bei der Abfrage ein Datenbank-Fehler auf (eine `SQLException` wird geworfen).
    b. Falls die Karte berechtigt ist, wird der Zugriff gewährt und die grüne LED wird aktiviert.
2. Wenn der Knopf an einem Schloss gedrückt wird, dann überprüft der Controller, ob die richtige Karte im Slot dieses Schlosses eingelegt ist.
    a. Falls ja wird das Schloss entriegelt wenn es vorher verriegelt war bzw. verriegelt wenn es vorher entriegelt war.
    b. Falls nein passiert gar nichts, d.h. der Knopfdruck wird ignoriert.
3. Wenn eine Karte aus dem Slot eines Schlosses entfernt wird, dann erlischt an diesem Schloss die aktive LED.

Aufgabe:

Schreiben Sie eine JUnit-Testklasse, die die Klasse `LockController` testet. Lesen Sie dazu die Spezifikation genau durch und schreiben Sie Testmethoden für alle relevanten Fälle. Setzen Sie das Framework EasyMock ein, um Mock Objects für die Interfaces `ICardDao` und `ILock` zu erzeugen. Verwenden Sie anschließend diese Mock Objects, um das Verhalten von `LockController` zu testen.

Die Klasse `LockController` enthält einen Fehler, der zu einer Sicherheitslücke führt. Schreiben Sie einen Test, der diesen aufdeckt, und erklären Sie den Fehler in einem Kommentar zu diesem Test. Beheben Sie anschließend den Fehler im Code der Klasse `LockController`.

Hinweis: berücksichtigen Sie, dass der Lock Controller normalerweise nicht nur ein Schloss verwaltet.