VU Softwarequalitätssicherung – Beispiel 3 CashMachine

Spezifikation

Dieses Dokument spezifiziert Anforderungen eines Softwaresystems für einen Bankomat.

1. Anforderungen

Es handelt sich hier um einen sehr einfachen Bankomat, der mit einer Bankomatkarte, einem numerischen Tastenfeld von 0 bis 9, Tasten zur Wahl der Banknoten und den zwei Knöpfen „Eingabe“ und „Abbruch“ zu bedienen ist. Auf der Bankomatkarte ist nur die Kontonummer gespeichert. Ausserdem besitzt der Bankomat ein Fach für die Geldausgabe und ein Display.

Der Bankomat verfügt über 200, 100, 50, 20 und 10 Euro Banknoten. Da es verschiedene Typen von Bankomaten gibt, ist die maximal Kapazität nicht näher spezifiziert. Jedoch sollte jeder Automat in der Lage sein, ein Gesamtkapital von 10.000 Euro zu halten. Der Kunde kann auch auswählen, welche Banknoten ausgezahlt werden. Wenn keine Auswahl stattfindet, unterliegt die Auszahlung folgenden Regeln:

- *Regel: solange Betrag > x Euro → gegebene Noten ausgeben und von Betrag abziehen*
- solange > 500 Euro → 2 x 200
- wenn <= 500 und solange > 200 Euro → 1 x 100, 4 x 20
- wenn <= 200 und solange > 110 Euro → 1 x 50, 3 x 20
- wenn <= 110 und solange > 50 Euro → 1 x 50
- ansonsten 10 Euro Noten

Sollten bestimmte Typen von Noten nicht verfügbar sein, sollte dies am Display angezeigt werden. Durch den Abbruch-Knopf wird der Bankomat wieder in den Ausgangszustand gesetzt und die Bankomatkarte wird ausgeworfen.

Ein PIN-Code darf maximal zweimal hintereinander falsch eingegeben werden. Nach drei hintereinander, fehlgeschlagenen Versuchen den PIN-Code für eine bestimmte Bankomatkarte einzugeben, wird diese Karte gesperrt und damit bei erneuten Einsteckversuchen vom Automaten nicht akzeptiert.

Folgende reale Elemente der Domäne werden vernachlässigt:

- Sicherheitskamera
- Anderer Sicherheitsaspekte, wie z.B. Verschlüsselung der PIN-Codes
- Interner Log vom Bankomaten

2. Java Interface

Das Java-Interface des Softwaresystems findet sich unter src/main/java/cashmachine/api/CashMachine.java und beinhaltet eine genaue Spezifikation der aufrufbaren Methoden.

3. Implementation / Source Code

Die vorhandene Software besteht aus folgenden Dateien:

- **Account.java**
  - Kapselt die Daten eines Bankkontos wie Kontonummer (accountId), PIN, Saldo (balance) und Limit beim Abheben.
- **AccountService.java**
  - Interface, das die Anbindung an das interne Netzwerk darstellt um Accountdaten abzufragen, Accounts sperren zu lassen, ... Wird von der CashMachineImpl.java benötigt.
- **Bill.java**
  - Enum, welches vordefinierte Werte für Banknoten angibt.
- **CashMachine.java**
  - Interface/Spezifikation des Bankomaten.
- **CashMachineException.java**
  - Eine „Checked"-Exception, die bei einem echten Fehler geworfen wird. Echte Fehler sind keine Betriebsfehler! Betriebsfehler treten voraussichtlich während des normalen Betriebs auf und können als Benutzerfehler angesehen werden, wie z.B. die Eingabe eines falschen PIN-Codes. Für beide Fehlerarten soll auf dem Display eine sinnvolle Meldung angezeigt werden.
- **CashMachineImpl.java**
  - Diese Klasse verwaltet jegliche Daten über den aktuellen Zustand des Systems, wie den Füllstand von Banknoten sowie die aktuell eingegebene Bankomatkarte und die momentan angezeigte Nachricht auf dem Display. Jegliche Interaktion, die der Bankomat anbietet, soll über diese Klasse möglich sein.
- **MachineConnector.java**
  - Interface, das die Anbindung an die mechanischen Teile der Maschine darstellt, wie z.B. Öffnen des Geldausgabefaches. Wird von der CashMachineImpl.java benötigt.

Um ein besseres Verständnis über die Verwendung des Bankomaten zu erlangen, sind hier die Standardeinstellungen nach der Instanziierung sowie der Ablauf eines Abhebevorganges beschrieben. Diese Erklärung soll auch helfen Testfälle implementieren zu können.

3.1 Standardeinstellungen

Der Bankomat ist mit den bei der Instanziierung angegebenen Banknoten befüllt und wartet auf die Eingabe einer Kontonummer. Wird die Kontonummer eingegeben, also eine Bankomatkarte eingeführt, wartet der Bankomat auf die Eingabe eines PIN-Codes. Die Kontonummer muss vorhanden sein und darf nicht gesperrt sein. Das Fach zur Geldentnahme (moneyTray) ist geschlossen.

3.2 Ablauf eines Abhebevorgang

1. Die Bankomatkarte wird eingeführt (setAccountId()) und die Kontonummer wird automatisch ausgelesen. Dieser Vorgang kann beim Testen vernachlässigt werden. Nachdem die Kontonummer gesetzt wurde, kann der PIN-Code eingegeben werden (setPin()).
2. Wenn der richtige PIN-Code eingegeben wird, sollte der Kunde in der Lage sein eines seiner Konten, für das er zeichnungsberechtigt ist, auszuwählen (setWithdrawalAccountId()).
3. Nach der Auswahl eines Kontos kann davon Geld abgebucht werden (withdraw()).
4. Läuft der Abbuchungsvorgang korrekt durch, kann der Automat Geld auswerfen (dispenseAmount(), MachineConnector.dispenseBill()). Dieser Vorgang hat je nach aktuellem Zustand des Automaten eine andere Meldung auf dem Display (getMessage()) zufolge. Außerdem kann der Kunde auswählen, welche Banknoten ausgezahlt werden. Wie bestimmte Noten ausgewählt werden, soll beim Testen vernachlässigt werden.

5. Verfügt der Bankomat über genügend Geld bzw. die richtigen Banknoten, wird der Geldbetrag ausbezahlt (isMoneyTrayOpen(), MachineConnector.openMoneyTray()).
6. Nachdem der Kunde das Geld entnimmt, wird das Geldentnahmefach wieder geschlossen (onMoneyTaken()) und der Bankomat kehrt in den Ausgangszustand zurück.

3.3 Fehlerbehandlung

- Fehler, die im laufenden Betrieb durch den Kunden auftreten können (z.B. falscher PIN-Code), sollen mit einem Rückgabewert von false der zuständigen Methoden behandelt werden.
- Fehler, die durch eine falsche Verwendung der Spezifikation des Interfaces auftreten (z.B. falsche Reihenfolge der Methoden, nicht beachtete Parameter-Einschränkungen), sollten durch eine CashMachineException signalisiert werden.
- Meldungen werden auf dem Display des Automaten angezeigt. Mögliche Ausgaben sind:
  - Ausser Betrieb! Kein Geld vorhanden!
    - Falls keine einzige Banknote mehr im Bankomat vorhanden ist.
  - Betrag muss ein Vielfaches von 10 sein!
    - Falls der Betrag beim Abbuchungsvorgang nicht durch 10 teilbar ist (withdraw()).
  - Limit ueberschritten!
    - Falls beim Abbuchungsvorgang der Betrag das Limit übersteigt (withdraw()).
  - Saldo ueberschritten!
    - Falls beim Abbuchungsvorgang der Betrag den Saldo übersteigt (withdraw()).
  - Bankomatkarte einfuehren.
    - Die Willkommensnachricht.
  - Bankomatkarte einfuehren. Nur 200 10 Euro Noten verfuegbar.
    - Willkommensnachricht, falls nur mehr 200 und 10 Euro Noten verfügbar sind. Andere Varianten je nach Notenstand möglich.
  - Auszahlungssumme und Banknotensumme stimmen nicht ueberein!
    - Wenn die gewaehlten Banknoten nicht mit der gewählten Auszahlungsumme überein stimmen (dispenseAmount()).
  - Nicht genug Banknoten zur Verfuegung um die Abhebung durchzufuehren!
    - Falls bei der Auszahlung (dispenseAmount()) nicht genügend Banknoten verfuegbar sind.
  - Waehlen Sie ein Abbuchungskonto.
    - Wird angezeigt nachdem der korrekte Pin eingegeben wurde (setPin()).
  - Waehlen Sie einen Betrag.
    - Wird angezeigt nachdem ein Abbuchungskonto ausgewählt wurde (setWithdrawalAccountId()).

3.4 Allgemeine Hinweise

Die vorhandene Implementierung enthält keine main() Methode. Der Bankomat wird durch seinen Konstruktur erzeugt und unter anderem mit den dispenseAmount(), getMessage(), setPin() Methoden verwendet. Die JUnit Testfälle spielen die Rolle des Clients, der diese Methoden aufruft.

Der Bankomat benötigt bei der Instanziierung Objekte, die die Interfaces AccountService und MachineConnector implementieren. Der AccountService stellt die Verbindung zum internen Bank-Netzwerk, der MachineConnector die Verbindung zu den physikalischen Teilen des Bankomats her. Beide Interfaces werden mit Hilfe eines Mocking Frameworks (mockito) simuliert.

4. Test-Case Skeleton

Um die Implementierung gegen die Spezifikation zu prüfen, sollen Testfälle mit JUnit programmiert werden. Bestimmte Testfälle können natürlich fehlschlagen (im Eclipse JUnit Plugin als blau bzw. rot markiert).

Erwartete Fehlerfälle sollten so implementiert werden, dass der JUnit Test auch erfolgreich (als grün markiert) ist.

Die JUnit Testklasse bietet bereits vordefinierte Methoden, die einen Bankomat mit zwei Accounts instanziiert. Weiters sind auch andere sogenannte Creation-Methods (create*()) zum Instanziieren von Bankomatobjekten vorhanden. Sie können natürlich auch eigene Creation-Methods schreiben. Das benötigte AccountService Interface wurde bereits so gemockt, das es sich für den Account 1 richtig verhält.

4.1 Überprüfbarkeit des Bankomats

Grundsätzlich sind die Rückgabewerte der Methoden, geworfene Exception bzw. die Message am Display zu überprüfen. Zudem kann der Füllstand des Bankomaten abgefragt werden bzw. der eigentliche Vorgang des Abbuchens eines Betrags von einem Konto überprüft werden.

4.2 Vorschläge für Testmethoden

- setPin_invalidPin_returnsFalse
- withdraw_amountExceedsLimit_returnsFalse
- ....