

Testbericht & Abgabedatei

3b) Äquivalenzklassen: Für die Methode `CashMachine.withdraw()`

A1: accountId not valid

A2: accountId is valid

WAZERO1: withdrawAmount <= 0

WAZERO2: withdrawAmount > 0,

WADIV1: withdrawAmount % 10 != 0

WADIV2: withdrawAmount % 10 == 0

WABAL1: withdrawAmount > user_account_balance

WABAL2: withdrawAmount <= user_account_balance

WALIM1: withdrawAmount > user_account_limit

WALIM2: withdrawAmount <= user_account_limit

Ungültig:

- (A1, WAZERO1, WADIV1, WABAL1, WALIM1)
- (A1, WAZERO1, WADIV1, WABAL1, WALIM2)
- (A1, WAZERO1, WADIV1, WABAL1, WALIM1)
- (A1, WAZERO1, WADIV1, WABAL2, WALIM2)
- (A1, WAZERO1, WADIV2, WABAL1, WALIM1)
- (A1, WAZERO1, WADIV2, WABAL1, WALIM2)
- (A1, WAZERO1, WADIV2, WABAL1, WALIM1)
- (A1, WAZERO1, WADIV2, WABAL2, WALIM2)
- (A1, WAZERO2, WADIV1, WABAL1, WALIM1)
- (A1, WAZERO2, WADIV1, WABAL1, WALIM2)
- (A1, WAZERO2, WADIV1, WABAL1, WALIM1)
- (A1, WAZERO2, WADIV1, WABAL2, WALIM2)
- (A1, WAZERO2, WADIV2, WABAL1, WALIM1)
- (A1, WAZERO2, WADIV2, WABAL1, WALIM2)
- (A1, WAZERO2, WADIV2, WABAL1, WALIM1)
- (A1, WAZERO2, WADIV2, WABAL2, WALIM2)
- (A2, WAZERO1, WADIV1, WABAL1, WALIM1)
- (A2, WAZERO1, WADIV1, WABAL1, WALIM2)
- (A2, WAZERO1, WADIV1, WABAL1, WALIM1)
- (A2, WAZERO1, WADIV1, WABAL2, WALIM2)
- (A2, WAZERO1, WADIV2, WABAL1, WALIM1)
- (A2, WAZERO1, WADIV2, WABAL1, WALIM2)
- (A2, WAZERO1, WADIV2, WABAL1, WALIM1)
- (A2, WAZERO1, WADIV2, WABAL2, WALIM2)
- (A2, WAZERO2, WADIV1, WABAL1, WALIM1)

- (A2, WAZERO2, WADIV1, WABAL1, WALIM2)
- (A2, WAZERO2, WADIV1, WABAL1, WALIM1)
- (A2, WAZERO2, WADIV1, WABAL2, WALIM2)
- (A2, WAZERO2, WADIV2, WABAL1, WALIM1)
- (A2, WAZERO2, WADIV2, WABAL1, WALIM2)
- (A2, WAZERO2, WADIV2, WABAL1, WALIM1)

Gültig:

- A2, WAZERO2, WADIV2, WABAL2, WALIM2

Grenzwertanalyse:

- withdrawAmount == 0
- withdrawAmount == user_account_balance
- withdrawAmount == user_account_limit

3c) Black-Box Testfälle: Vorbedingung für alle Fälle: Klasse lässt sich kompilieren und instanzieren.

Lfd Nr.	Typ	Beschreibung Testfall	Eingabewerte	Aktionen	Erwartetes Ergebnis	Tatsächliches Ergebnis	System auswirkungen	Test bestanden? (j/n)
0	NF	getMessage_shouldReturnWelcomeMessage: Der Bankomat sollte zu Beginn eine Willkommensnachricht zeigen.	-	Keine ausser Anzeigenachricht ueberpruefen.	GetMessage() gibt Bankomatkarte einfuehren! zurueck.	Erwartetes Ergebnis.	KEINE	Ja
1	NF	setAccountId_shouldBeValid: Testkarte sollte gültig sein.	Account: AC1_ID	Gültige Kartennummer eingeben.	Karte wird gültig.	Erwartetes Ergebnis.	KEINE	Ja
2	NF	setAccountId_shouldBeInvalid: Testkarte sollte ungültig sein.	Account: INVALID_ID	Ungültige Kartennummer eingeben.	Karte ist ungültig.	Erwartetes Ergebnis.	KEINE	Ja
3	FF	setAccountId_shouldThrow: Eingabe zweier Karten sollte Exception werfen.	Account: AC1_ID	Gültige Kartennummer zwei mal hintereinander eingeben.	Exception wird geworfen.	Erwartetes Ergebnis.	KEINE	Ja
4	NF	setPin_shouldBeValid: Pin für Karte sollte gültig sein.	Account: AC1_ID Pin: AC1_ID	Gültige Kartennummer und gültigen Pin eingeben.	Pin ist gültig.	Erwartetes Ergebnis.	KEINE	Ja
5	NF	setPin_shouldBeInvalid: Pin für Karte sollte ungültig sein.	Account: AC1_ID Pin: AC1_ID + 1	Gültige Kartennummer und ungültigen Pin eingeben.	Pin ist ungültig.	Erwartetes Ergebnis.	KEINE	Ja
6	FF	setPin_shouldThrow: Pineingabe ohne Karteneingabe sollte Exception werfen.	Pin: AC1_ID	Pin eingeben.	Exception wird geworfen.	Erwartetes Ergebnis.	KEINE	Ja
7	NF	isCardValid_shouldBeValid: Testkarte sollte gültig sein.	Account: AC1_ID	Gültige Kartennummer eingeben.	Karte ist gültig und Methode <i>isCardValid()</i> ist wahr.	Erwartetes Ergebnis.	KEINE	Ja
8	NF	dispenseAmount_should_open_money Tray: Geldausgabe sollte das Geldausgabefach öffnen.	Account: AC1_ID Pin: AC1_ID Abbuchungskonto: automatisch das erste Geldbetrag: 10→	Gültige Kartennummer, Pin, Abbuchungskonto und Geldbetrag eingeben. Geldausgabefach überprüfen.	Geld wird vom Konto abgeboben und ausgegeben. Geldausgabefach wird geöffnet. <i>MachineConnector.openMoneyTray()</i> wird dabei aufgerufen werden.	Erwartetes Ergebnis.	KEINE	Ja
9	NF	dispenseMoney: Beim Abheben von 660→ sollten die jeweils lt. Spezifikation richtigen Geldscheine ausgegeben werden.	Account: AC1_ID Pin: AC1_ID Abbuchungskonto: automatisch das erste Geldbetrag: 660→	Gültige Kartennummer, Pin, Abbuchungskonto (automatisch das erste) und Geldbetrag (660→) eingeben.	Anzahl der jeweiligen Geldscheine im Automaten entsprechen jenen der unabhängigen Zählung durch die Testmethode.	Anzahl der Geldscheine im Automaten weichen ab!	KEINE	Nein

10	NF	setPin_shouldBanCard: Nach drei Falscheingaben des Pins soll die Karte automatisch gesperrt werden.	Account: AC1_ID Pin: AC1_ID + 1	Gültige Kartennummer wird eingegeben. Falscher Pin wird drei mal hintereinander eingegeben.	Karte wird durch den Aufruf der Funktion <i>AccountService.banAccount()</i> automatisch gesperrt. Zudem soll die Funktion <i>AccountService.isValidAccountld()</i> genau einmal, die Funktion <i>AccountService.validatePin()</i> genau drei mal aufgerufen werden.	Funktion <i>AccountService.banAccount()</i> wird nicht aufgerufen!!	KEINE	Nein
----	----	---	------------------------------------	---	--	--	-------	-------------

3e) Fehlerstatistik: Fassen Sie die Ergebnisse Ihrer durchgeführten Testläufe in folgender Tabelle zusammen und kommentieren Sie diese im Anschluss.

Testfälle	Black-Box Tests	
	Bestanden	Nicht bestanden
Anzahl getesteter Normalfälle (NF)	6	2
Anzahl getesteter Sonderfälle (SF)	0	0
Anzahl getesteter Fehlerfälle (FF)	2	0
Summe (aller Testfälle)	8	2

Kommentieren Sie die gesammelten Ergebnisse und beantworten Sie die Fragen aus der Angabe:

• Begutachten Sie die Abdeckung der Methoden `setAccountId()`, `setPin()` und `withdraw()` und wählen Sie die Methode mit der geringsten Testabdeckung. Beantworten Sie zu dieser Methode folgende Fragen:

• Wieviele Verzweigungen (Branches) werden noch nicht abgedeckt?

50% (bzw. 2 von 4)

• Wieviele Tests würden Sie benötigen um eine 100% Statement bzw. Branch Abdeckung zu erreichen?

2 weitere Tests

• Wären diese zusätzlichen Tests um 100% Abdeckung zu erzielen Black-Box oder White-Box Tests? Warum?

Black-Box Tests, da sie hauptsächlich das Verhalten der Spezifikation testen.

• Ist es sinnvoll eine 100% Code Abdeckung anzustreben?

Nein, da 100% meist gar nicht erreicht werden kann. Dies tritt insbesondere beim Testen von Methoden auf, bei der mehrere interne verschachtelt Methoden aufgerufen werden. Oft überprüfen diese interne Methoden den Status interner Variablen, die sie verwenden. Durch den verschachtelten Aufruf, kann jedoch nur der erste fehlerhafte Status einer solchen Variable überprüft werden kann. Zudem können Variablen von einander abhängen. Auch hier ist keine 100% Abdeckung möglich.

• Nach welchen Kriterien haben Sie Ihre Testfälle gewählt? Z.B. Requirement-based testing, Risk-based testing, ...

Requirement-based testing

• Sind alle Ihre implementierten Tests Unit-Tests? Was für Kriterien haben Unit-Tests?

Ja, denn...

• sie sind automatisiert und wiederholbar.

• sie sind von einander unabhängig.

• sie testen relevanten Code.

• überprüfen genau eine Eigenschaft der Spezifikation.

• sind verständlich und kurz.

3f) Test Driven Development (TDD): Tragen Sie die Reihenfolge der Umsetzung hier ein (Funktionalität, Tests und das beheben von Fehlern).

Funktionalität / Test	Beschreibung
Test X und Test Y	Tests laufen, aber schlagen fehl.
Implementierung X	Test X funktioniert.
Implementierung Y	Test Y funktioniert, Test X schlägt fehl
Fehler in Impl. X behoben	Test X funktioniert wieder.
Tests <i>isValidAccountId_shouldBeValid()</i> und <i>isValidAccountId_InvalidId()</i>	Tests laufen aber schlagen fehl.
Implementierung Klasse <i>SAS_Account</i> (<i>SAS_Account(...)</i> , <i>getAccount()</i> , <i>setBanned(...)</i> , <i>getBanned()</i> , <i>SimpleAccountService(...)</i> und <i>isValidAccountId(...)</i>)	Tests <i>isValidAccountId_shouldBeValid()</i> und <i>isValidAccountId_InvalidId()</i> funktionieren.
Tests <i>banAccount_shouldGetBanned()</i> und <i>banAccount_InvalidAccount()</i>	Tests laufen aber schlagen fehl.
Implementierung <i>banAccount(...)</i>	Tests <i>banAccount_shouldGetBanned()</i> und <i>banAccount_InvalidAccount()</i> funktionieren.
Tests <i>validatePin_shouldBeValid()</i> , <i>validatePin_InvalidPin()</i> und <i>validatePin_InvalidAccount()</i>	Tests laufen aber schlagen fehl.
Implementierung <i>validatePinSAS(...)</i> und <i>validatePin(...)</i>	Tests <i>validatePin_shouldBeValid()</i> , <i>validatePin_InvalidPin()</i> und <i>validatePin_InvalidAccount()</i> funktionieren.
Tests <i>getWithdrawalAccount_shouldBeValid()</i> , <i>getWithdrawalAccount_invalidPin()</i> und <i>getWithdrawalAccount_invalidAccount()</i>	Tests laufen aber schlagen fehl.
Implementierung <i>getWithdrawalSASAccount(...)</i> und <i>getWithdrawalAccount(...)</i>	Tests <i>getWithdrawalAccount_shouldBeValid()</i> , <i>getWithdrawalAccount_invalidPin()</i> , <i>getWithdrawalAccount_invalidAccount()</i> .
Tests <i>withdraw_shouldBeValid()</i> , <i>withdraw_invalidAccount()</i> , <i>withdraw_invalidPin()</i> , <i>withdraw_invalidWithdrawalAccount()</i> , <i>withdraw_invalidAmount()</i> und <i>withdraw_amountTooHigh()</i>	Tests laufen aber schlagen fehl.
Implementierung <i>SAS_Account::withdraw(...)</i> und <i>withdraw(...)</i>	Tests <i>withdraw_invalidAccount()</i> , <i>withdraw_invalidPin()</i> , <i>withdraw_invalidWithdrawalAccount()</i> , <i>withdraw_invalidAmount()</i> , <i>withdraw_amountTooHigh()</i> funktionieren, Test <i>withdraw_shouldBeValid()</i> schlägt fehl.
Fehler in Implementierung <i>SAS_Account::withdraw(...)</i> und <i>SAS_Account::revertWithdraw(...)</i> behoben	Test <i>withdraw_shouldBeValid()</i> funktioniert.
Tests <i>revertWithdraw_shouldBeValid()</i> , <i>revertWithdraw_invalidAccount()</i> , <i>revertWithdraw_invalidPin()</i> , <i>revertWithdraw_invalidTransid()</i> , <i>revertWithdraw_useTransidTwice()</i> und <i>revertWithdraw_usWrongTransids()</i>	Tests laufen aber schlagen fehl.
Implementierung <i>SAS_Account::revertWithdraw(...)</i> und <i>revertWithdraw()</i>	Tests <i>revertWithdraw_shouldBeValid()</i> , <i>revertWithdraw_invalidAccount()</i> , <i>revertWithdraw_invalidPin()</i> , <i>revertWithdraw_invalidTransid()</i> , <i>revertWithdraw_useTransidTwice()</i> und <i>revertWithdraw_usWrongTransids()</i> funktionieren.