ATM Dokumentation

Die Panzerknacker

Die Panzerknacker

Inhaltsverzeichnis

1. ATM Dokumentation Startseite

1.1 Abstract

abstract text

1.2 Das Team

Wir sind die Panzerknacker.

Mitglied	Spezialisierung
Michél Franz	UX
Juri Kaemper	Text & QS
Christian Lopéz	Programmierung
Felix Möhler	Requirements Engineering
Julian Thiele	UML/Kollab.-Werkzeug, Entwicklungsumgebung

2. Anforderungsdokumentation

2.1 Produktvision und Produktziele

2.1.1 Produktvision

Eine regionale Bank hat unser externes Software-Entwicklerteam für einen Auftrag eingestellt. Bei dem uns übertragenem Projekt handelt es sich um die fehlerhafte Software einer ATM (Automated Teller Machine) zu deutsch Bankautomat. Der bereits existente Programmcode wurde von einem externen Unternehmen entwickelt, so dass der Kunde kein Expertenwissen zum Programm verfügt, außerdem fehlt auch die Dokumentation vollständig.

Um dem Bankunternehmen nun die Verwendung des Systems zu ermöglichen, muss das Programm komplett überarbeitet werden, darüber hinaus soll eine detaillierte Dokumentation (vollständig in deutsch) für die Bank erstellt werden. Das fehlerfreie Programm mit den bereits integrierten Features und einer strukturierten Dokumentation ist unser Basisfaktor. Das Programm ist für die Bankautomaten der Bank in Deutschland vorgesehen. Die Dokumentation soll die Entwicklung sowie die Funktionen der Software zusammenfassen und den zuständigen Mitarbeiter verständlich machen.

2.1.2 Produktziele

Die Aufgabe unseres Teams ist es den bereits vorhandenen Code so zu überarbeiten, dass dieser voll funktionsfähig ist und eine sichere Laufzeit gewährleistet werden kann. Zur Entwicklung der Software ist eine vollständig deutsche Dokumentation vorgesehen mit **Anforderungs**-, **Architektur**-, **Test**-, **Abnahme**-, **Benutzer**-, **Projekt**-, und **Codedokumentation**.

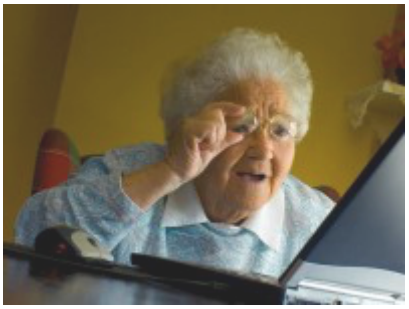
2.2 Rollen und Personas

2.2.1 Rollen

Rollen	Beschreibung
Benutzer	Die Benutzer sind Kunden der Bank, die den Geldautomaten zur Verfügung stellt
Administrator	Administratoren des Bankautomatensystems, die Verwaltungsrechte über alle Benutzer besitzen

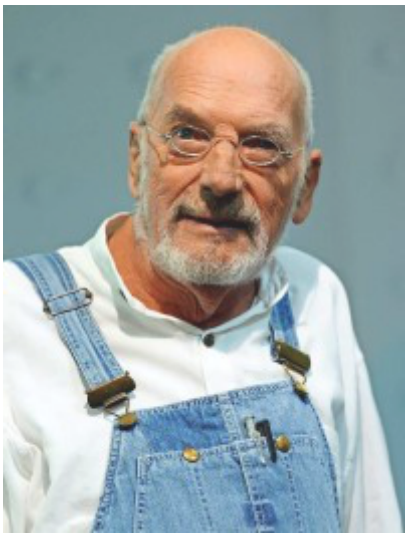
2.2.2 Personas

Gertrude Gabel



Rolle	Benutzer
Alter	65
Geschlecht	weiblich
Tätigkeit	Rentnerin
Familienstand	verheiratet
Bildung	Mittelschule
Computerkenntnisse	Keine
Interessen und Hobbies	Wandern, Kaffee trinken
Einstellung zum Produkt	"Eine tolle Maschine, tut was sie soll"
Wünsche	Einfache Bedienung, wenig zum Merken

Peter Lustig



Rolle	Benutzer
Alter	38
Geschlecht	männlich
Tätigkeit	Handwerker
Familienstand	verheiratet
Bildung	Realschule
Computerkenntnisse	Grundkenntnisse
Interessen und Hobbies	Autos, Actionfilme, Fahrradfahren
Einstellung zum Produkt	"Hoffentlich werden die neuen Geldautomaten besser"
Wünsche	Nützliche Funktionen, Schnelle Bedienbarkeit

Andy Auman

Rolle	Administrator
Alter	29
Geschlecht	männlich
Tätigkeit	Systemadministrator
Familienstand	ledig
Bildung	Abitur
Computerkenntnisse	Fachkenntnisse
Interessen und Hobbies	Programmierung, Netzwerke, Gaming
Einstellung zum Produkt	""
Wünsche	Viele Funktionen, Wenig Konfigurationsaufwand

Mathias Jung

Rolle	Benutzer
Alter	19
Geschlecht	männlich
Tätigkeit	Student
Familienstand	ledig
Bildung	Abitur
Computerkenntnisse	Grundkenntnisse
Interessen und Hobbies	BWL / Wirtschaft
Einstellung zum Produkt	""
Wünsche	Schnelle und Einfache Transaktionen

2.3 User Stories

Als **[Rolle]** möchte ich **[Ziel/Wunsch]**, um **[Nutzen]**

1. Als **Benutzer** möchte ich **verschiedene Geldbeträge eingeben**, um diese abzuheben
2. Als **Benutzer** möchte ich **sehen, wie viel Geld auf meinem Konto** ist, um zu wissen, wie viel ich noch abheben kann
3. Als **Benutzer** möchte ich eine **maximal Debit Betrag pro Tag festlegen** können, um bei Diebstahl den Verlust zu minimieren
4. Als **Benutzer** möchte ich eine **vierstellige Pin zu meiner Karte eingeben** müssen, um Gelddiebstahl von meinem Konto zu vermeiden
5. Als **Benutzer** möchte ich die **Ziffern meiner Pin ändern** können, um sie mir besser merken zu können
6. Als **Benutzer** möchte ich die **Länge meiner Pin ändern** können, um die Sicherheit zu verbessern
7. Als **Benutzer** möchte ich eine **Stückelung auswählen** können, um gewünschte Scheine zu erhalten
8. Als **Benutzer** möchte ich mich **auf meinem Konto einloggen** können, um getätigte Transaktionen zu sehen
9. Als **Mitglied einer anderen Bank** möchte ich **gegen Gebühren Geld abheben** können, um örtlich flexibel zu sein
10. Als **Administrator** der Bank möchte ich eine **vollständige und detaillierte Dokumentation**, um im Fehlerfall schnell handeln zu können

2.4 Aufgaben

- Anfertigen einer Ist-Dokumentation des Codes
- Funktionen aus User Stories implementieren
- Codeverbesserungen in Delta-Dokumentation beschreiben
- Anfertigen einer Anforderungsdokumentation
- Anfertigen einer Systemdokumentation
- Anfertigen einer Testdokumentation
- Anfertigen einer Abnahmedokumentation
- Anfertigen einer Benutzerdokumentation
- Anfertigen einer Projektdokumentation

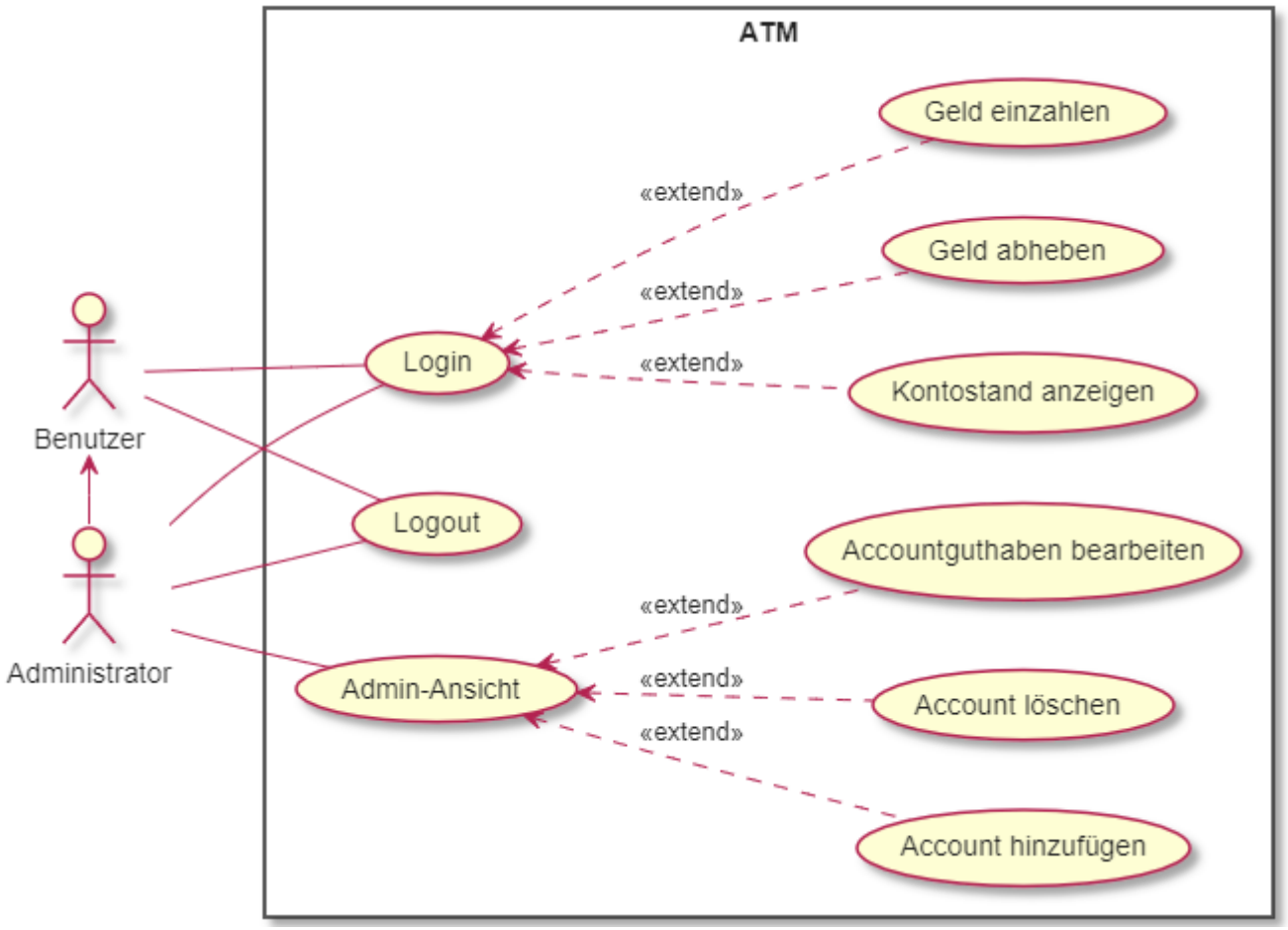
2.5 Begriffslexikon

Begriff	Bedeutung	Beschreibung
Cash Dispenser	Bargeld im ATM-Dispenser	-
Deposit Slot	Geldfach zum Ein- und Auszahlen	-
Balance	Ist-Saldo auf einem Account	-
Withdrawal	Geld abheben	-
Account Pin	Geheimpin eines Accounts (unique)	-
Account number	Nummer eines Accounts (unique)	-
Credit	Gutschrift	-
Debit	Maximale Auszahlung pro Tag	-

2.6 Mengengerüst

Bezeichnung	Beschreibung	Menge	Einheit
Pin	Stellenanzahl der Pin	4	Stellen
Geldautomaten	Anzahl Geldautomaten in Aschaffenburg	43	Stück
Debit	Maximale Auszahlung pro Tag	1000	Euro
Nutzer	Maximale Nutzer gleichzeitig	1	Person
Nutzer	Maximal registrierte Nutzer		
Transaktion	Maximale Transaktion pro Minute		

2.7 Use Cases



3. Architekturdokumentation

3.1 Beschreibung der Systemarchitektur

3.1.1 Priorisierung der nicht funktionalen Anforderungen

Beschreibt wie gut ein System/Produkt eine bestimmte Funktion erfüllt

- Gute Benutzerfreundlichkeit und Bedienbarkeit
- Hohe Performance bei Operationen wie Guthaben abrufen, einzahlen und auszahlen
- Kurze Start-Zeit (Account-Initialisierung)

3.1.2 Architekturprinzipien

Nach welchen Kriterien soll das System in Komponenten unterteilt werden? Welche Aspekte sollen in Komponenten zusammengefasst werden? Welche Dienstleistungen sollen Komponenten nach außen an ihrer Schnittstelle anbieten, welche Aspekte müssen geschützt sein? Wie sollen die Komponenten miteinander interagieren? Wie sollen Komponenten strukturiert und verfeinert werden?

3.1.3 Schnittstellen

- UI mit den Java-Swing GUI Bibliotheken
- `KeypadListener.java` für Kommunikationsschnittstelle zwischen dem Tastenfeld und dem Bildschirm Objekt
- `ATMListener.java` ist die Schnittstelle zum Haupt-ATM-Objekt, in der Aktionen, wie ein Wechsel in einen anderen Modus oder das Betätigen der Enter-Taste behandelt werden

3.1.4 Big Picture der Systemarchitektur

3.2 Systementwurf

3.2.1 Systemdekomposition

3.2.2 Designalternativen und –Entscheidungen

3.2.3 Cross-Cutting-Concerns, NFRs

3.3 Mensch-Maschine-Schnittstelle

3.3.1 Anforderungen an die Mensch-Maschine-Schnittstelle

Die Mensch-Maschine-Schnittstelle, oder auch Benutzerschnittstelle, bezieht sich auf die Kommunikation zwischen einem Nutzer (Mensch) und dem Geldautomaten (Maschine). Der Mensch gibt mit seinen Aktoren (Händen) eine Eingabe-Information an die Peripherieeinheiten des Geldautomaten, welche eine digitale Information an die Recheneinheit des Geldautomaten weiterleiten. Die von der Recheneinheit entgegengenommene Information wird mittels der aufgespielten Software verarbeitet und eine Ausgabe-Information wird erzeugt. Die Recheneinheit steuert digital die Peripherieeinheiten des Geldautomaten an, welche eine

optische (Bildschirm-Ausgabe) und mechanische Ausgabe Information (Geldauszahlung) erzeugen. Die Rückgabe-Informationen werden vom Menschen visuell (Bildschirm-Information) und haptisch (Annahme des ausgezahlten Geldes) verarbeitet.

Ein-/Ausgabe	Mensch Schnittstelle	Hardware Schnittstelle	Software Schnittstelle
Eingabe	Hände	Encrypting PIN Pad	Tastenabfrage
	Augen	ID-Kartenleser, Softkeys oder Touchscreen	Touchbildschirm Abfrage
Ausgabe	Hände	Bildschirm	Grafikausgabe
	Augen	Auszahlmodul	Peripherie Ansteuerung

3.3.2 Gestaltungsprinzipien und Style-Guide

Gestaltungsprinzipien

Gesetze:

1. Gesetz der Nähe
   • Logisch zusammengehörige Informationen werden auch örtlich zusammen gruppiert. Unterschiede in der Hierarchie o. ä. werden durch räumliche Trennung realisiert.
2. Gesetz der Gleichartigkeit
   • Zusammengehörige Informationen, z. B. Feldbezeichner, werden gleichartig dargestellt.
3. Gesetz der guten Gestalt
   • Der Mensch bevorzugt in seiner Wahrnehmung gute Gestalten (symmetrisch...)

Eigenschaften:

1. Einfachheit
2. Regelmäßigkeit
3. Symmetrie
4. Inners Gleichgewicht

Der Benutzer soll jederzeit sehen können:

1. Wo bin ich?
2. Wie kam ich hierhin?
3. Was kann ich hier tun?
4. Wohin und wie kann ich navigieren?

3.3.3 Interatkionsmodellierung

Benutzer:

Geld abheben

1. Der Benutzer Inseriert seine Bankkarte. ATM zeigt das Authentifizierungsmenü an.
2. Der Benutzer gibt sein Passwort ein um sich zu authentifizieren. ATM Zeigt das Menü zu weiteren Auswahl an.
3. Der Benutzer drückt „Abbruch“. ATM zeigt Menü zur weiteren Auswahl an.
4. Der Benutzer Wählt Betrag und Stückelung. ATM zahlt Betrag in gewünschter Stückelung aus, zeigt neuen Kontostand an und wirft Bankkarte aus.
5. Der Benutzer nimmt die Karte. ATM zeigt Willkommens Bildschirm

Geld einzahlen

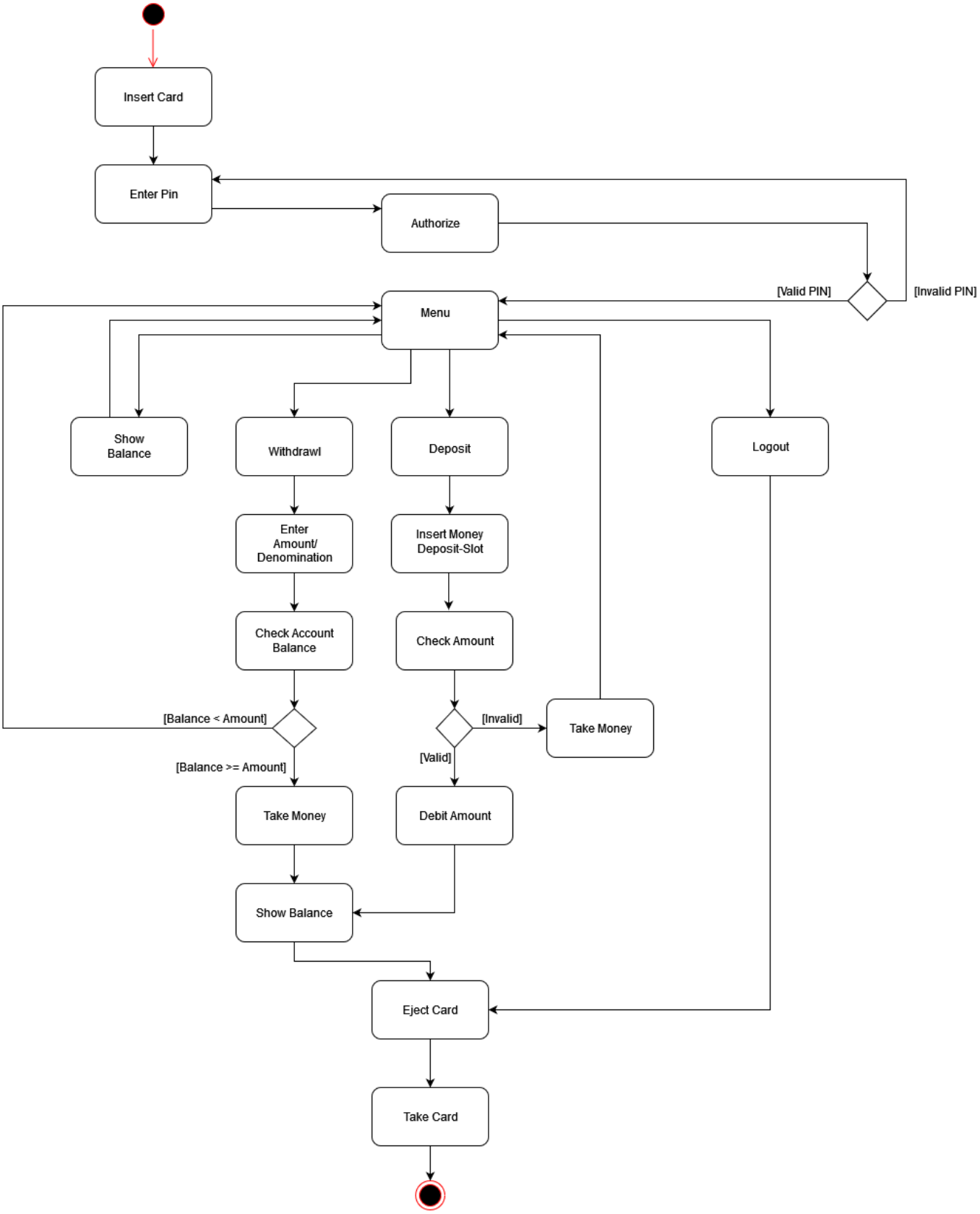
1. Der Benutzer Inseriert seine Bankkarte. ATM zeigt das Authentifizierungsmenü an.
2. Der Benutzer gibt sein Passwort ein um sich zu authentifizieren. ATM Zeigt das Menü zu weiteren Auswahl an.
3. Der Benutzer drückt „Geld einzahlen“ . ATM zeigt Informationsbildschirm und öffnet Deposit-Slot.
4. Benutzer drückt „Abbruch“. ATM zeigt Menü zur weiteren Auswahl an.
5. Benutzer legt Bargeld in den Deposit-Slot.
6. Benutzer drückt „Bestätigen“. ATM schließt den Deposit-Slot, validiert die Eingabe, bei erfolgreicher Prüfung wird der betrag dem Bankkonto gutgeschrieben und das Menü zu weiteren Auswahl angezeigt.
7. Benutzer drückt „Bestätigen“. ATM schließt den Deposit-Slot, validiert die Eingabe und bei nicht erfolgreicher Prüfung wird Deposit-Slot wieder geöffnet.
8. Benutzer entnimmt das Bargeld. ATM wirft Bankkarte aus und zeigt Willkommens Bildschirm an.
9. Der Benutzer nimmt die Karte. ATM zeigt Willkommens Bildschirm an.

Kontostand anzeigen

1. Der Benutzer Inseriert seine Bankkarte. ATM zeigt das Authentifizierungsmenü an.
2. Der Benutzer gibt sein Passwort ein um sich zu authentifizieren. ATM Zeigt das Menü zu weiteren Auswahl an.
3. Benutzer drückt „Kontostand anzeigen“. ATM zeigt Bildschirm mit Kontostand und Datum.
4. Benutzer drückt „Weitere Auswahl“. ATM zeigt Bildschirm zur weiteren Auswahl an.

Logout

1. Der Benutzer Inseriert seine Bankkarte. ATM zeigt das Authentifizierungsmenü an.
2. Der Benutzer gibt sein Passwort ein um sich zu authentifizieren. ATM Zeigt das Menü zu weiteren Auswahl an.
3. Der Benutzer drückt „Logout“. ATM wirft Bankkarte aus.
4. Der Benutzer nimmt die Karte. ATM zeigt Willkommens Bildschirm an.



4. Testdokumentation

coming soon

5. Abnahmedokumentation

coming soon

6. Benutzerdokumentation

coming soon

7. Projektdokumentation

coming soon

8. Codedokumentation

8.1 Code Ist-Dokumentation

8.1.1 Klassen

`ATMCaseStudy.java`

- Erstellt eine ATM Instanz und startet diese, wenn noch keine vorhanden

`ATM.java`

- Stellt die Hauptklasse des ATMs dar
- Initialisiert UI mit Keypad, CashDispenser, DepositSlot und Bankdatabase
- Es gibt viele unbenutzte konstante int Variablen
- Sobald Enter betätigt wird, wird die PIN überprüft (login)
- Wenn man eingeloggt ist, wird das Menü angezeigt, wenn man als Admin eingeloggt ist, wird das Admin-Menü angezeigt
- Im Menü kann man nun zwischen Funktionen wählen:
- `balance` : Eigenes Guthaben anzeigen
- `withdrawal` : Geld abheben, indem man die Scheine einzeln wählt
- `deposit` : Geld einzahlen. Geld ist erst verfügbar, wenn überprüft.
- `exit` : Führt Login erneut aus, öffnet allerdings neues Fenster
- Sollte man als Admin angemeldet sein, öffnet sich die Adminoberfläche mit diesen Funktionen:
- Kontostand jedes Nutzers einsehen
- Zwischen Accounts wecheln
- Accounts löschen
- Neue Accounts hinzufügen

`Transaction.java`

- Abstrakte Klasse, die mit einer AccountNummer, Dem Screen-Objekt und dem BankDatabase-Objekt initialisiert wird.

`BalanceInquiry.java`

- Erbt von Transactions und überschreibt die Execute-Funktion
- Die Execute-Funktion gibt den Kontostand auf dem Screen aus

`Withdrawal.java`

- Erbt von Transactions und überschreibt die Execute-Funktion
- Die Execute-Funktion zeigt die Buttons zur Scheinauswahl an
- Die Transaction-Funktion ermöglicht das abheben von Geld, wenn noch genügend auf dem Konto und im CashDispenser verfügbar ist.
- Man kann nur in 20er Scheinen abheben

`Deposit.java`

- Erbt von Transactions und überschreibt die Execute-Funktion
- Die Execute-Funktion zeigt UI zum Geldeinzahlen an
- Beim Geldeinzahlen wird geprüft, ob das Geld eingezahlt wurde

`DepositSlot.java`

- Klasse ist nicht vorhanden.
- Hier sollte überprüft werden, ob das Geld vorhanden ist

`CashDispenser.java`

- Startet mit 500 20$ Scheinen

`BankDatabase.java`

- Initialisiert alle Accounts
- Authentifiziert Nutzer anhand der PIN
- Funktionen um anhand der AccountNumber Daten über den Account abzurufen (verfügbares Guthaben, etc)
- Besitzt Funktionen um Guthaben von Accounts abzuziehen oder aufzuladen
- Fehler: `getaccpin` funktioniert nicht
- Funktion um temporär einen Account zu erstellen und dem Account-Array hinzuzufügen
- Funktion um temporär einen Account zu löschen

`Account.java`

- Besitzt Eigenschaften eines Benutzers
- Funktion um Pin mit aktuellem Account zu verifizieren
- Getter und Setter

`AccountFactory.java`

- Wird nicht verwendet
- Erbt von Account, initialisiert einen Account

`Iterator`

- Interface, das zwei Funktionen beinhaltet, die einen Wahrheitswert zurückgeben, ob von der aktuellen Position ein nächstes oder vorheriges Element existiert
- Funktion, die ein Objekt zurück gibt, anhand einer Position

`AccountIterator.java`

- Implementiert das Iterator Interface und überschreibt dessen Funktionen

`Screen.java`

- JFrame-Komponente, die Textfelder, Labels und Buttons besitzt
- Besitzt Funktionen um Nachrichten in der Konsole auszugeben
- Besitzt Funktionen um UI-Elemente anzuzeigen:
- Login
- Menü
- Kontostand
- Geldauszahlung
- Geldeinzahlung
- Admin-Ansicht

`Keypad.java`

- Besitzt unbenutze Scanner-Funktion
- Besitzt JButtons für ein Tastenfeld mit Löschen und Enter Funktionen
- Funktion um ein JPanel mit Buttons zu initialisieren und zurückgeben
- Fehler: Endlos-Schleife `userinput()`

8.2 Delta-Dokumentation

8.2.1 Verbesserungsvorschläge

- Pin-Authentifizierung beim Login
- Login mit Accountnummer?
- Länge des PIN`s?
- "Exit" schließt das Programm
- Verbesserung des Event-Handlings mit Actionlisteners
- Verbeserung der Klassenstruktur (weniger Klassen?)
- Verbesserung des UI-Handlings mit JFrame und JPanel
- Verbesserung der CashDispenser-Funktion
- Bankautomat in Deutsch oder Englisch
- Lizens nur 1x erwähnen