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

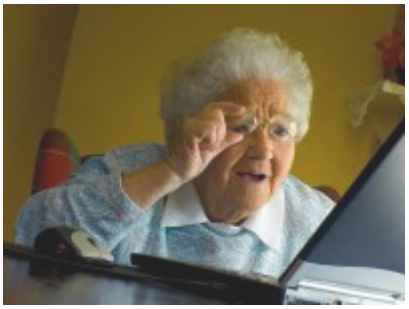
Hier werden die Rollen beschrieben, denen ein Benutzer angehören kann.

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

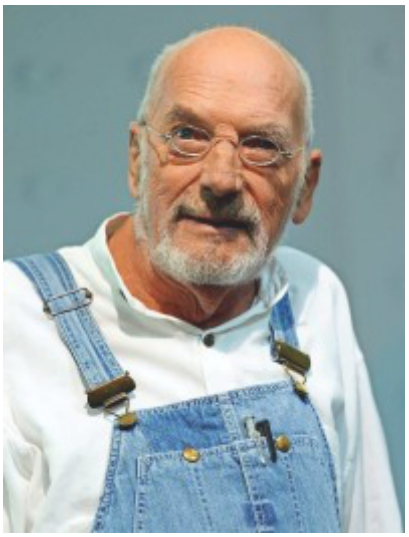
Personas veranschaulichen typische Vertreter Ihrer Zielgruppe.

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

User Stories sind Wünsche an eine Software, die aus Sicht des Endbenutzers verfasst wurden.

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

Auflistung aller Aufgaben dieses Projektes.

- 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

Hier werden alle fachspezifische Begriffe aufgelistet, die in diesem Projekt verwendet werden.

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	-

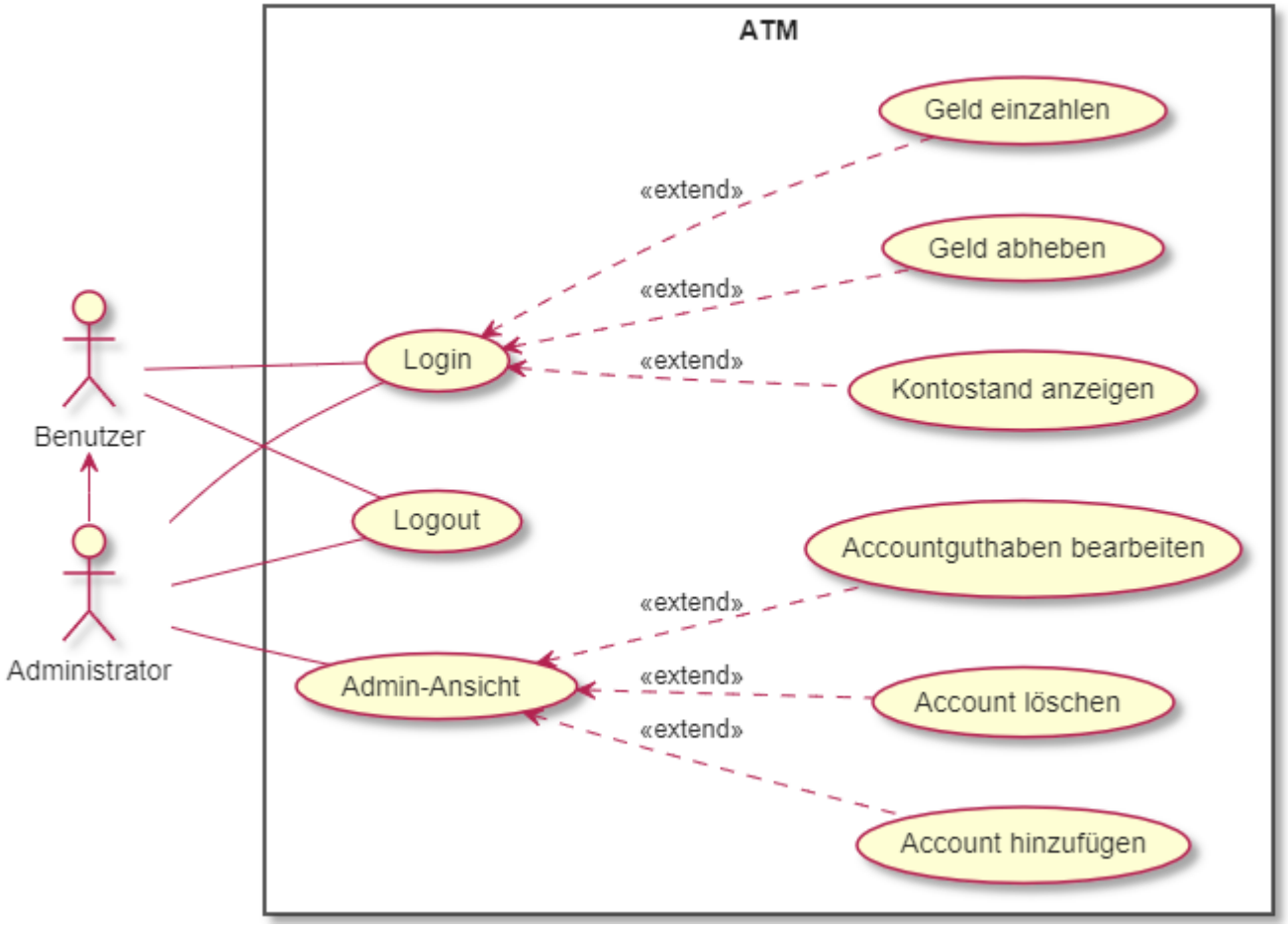
2.6 Mengengerüst

Das Mengengerüst beschreibt quantitativ die Komponenten eines Projektes.

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

In diesem Anwendungsfalldiagramm wird das nach außen sichtbare Verhalten des Systems aus Sicht der Nutzer beschrieben.



3. Architekturdokumentation

3.1 Beschreibung der Systemarchitektur

3.1.1 Priorisierung der nicht funktionalen Anforderungen

Qualitätsanforderungen

Änderbarkeit und **Wiederverwendbarkeit** waren uns besonders wichtig, da wir zu Beginn Schwierigkeiten hatten, uns einen Überblick über den bestehenden Code zu verschaffen. Aus diesem Grund entschieden wir uns, den Code noch einmal von Grund auf neu zu erstellen. Dadurch verbessert sich vor allem die **Brauchbarkeit** und **Wartbarkeit** des Codes.

Anforderungen an Lieferbestandteile

Eine vollständige Dokumentation in Form eines PDF Dokumentes und die Software bilden die Lieferbestandteile.

Anforderungen an die Benutzerschnittstelle

Eine weitere wichtige nicht funktionale Anforderung ist die **Bedienbarkeit** oder **Benutzerfreundlichkeit** des Programms. Da diese Anwendung für eine sehr große Menge an Benutzern ausgelegt ist, wurde die Bedienbarkeit und Benutzerfreundlichkeit des Programms auf eine höhere Priorität gesetzt. So wird gewährleistet, dass Benutzer aller Altersgruppen gut mit der Anwendung interagieren können.

3.1.2 Architekturprinzipien

Nach welchen Kriterien soll das System in Komponenten unterteilt werden? Wie sollen Komponenten strukturiert und verfeinert werden?

Das System wurde in verschiedene Komponenten unterteilt, die sich jeweils auf eine bestimmte Aufgabe beziehen, um eine enge Kopplung der Module untereinander zu reduzieren. Der verschachtelte Aufbau der UI Komponenten bildet eine Struktur, die leicht erweitert werden kann.

Welche Aspekte sollen in Komponenten zusammengefasst werden?

In der `ATM.java` Klasse werden die Änderungen von einem Modus in den Nächsten behandelt. Dem entsprechend wird die `Screen.java` Klasse angesteuert, um die UI Elemente zu aktualisieren.

Die Klasse `Screen.java` beinhaltet alle Funktionen, die zum Ändern der UI Elemente benötigt werden. In ihr werden die Klassen `Keypad.java` und `SidePanel.java` verwendet.

Welche Dienstleistungen sollen Komponenten nach außen an ihrer Schnittstelle anbieten? Wie sollen die Komponenten miteinander interagieren?

Die Komponente `Keypad.java` gibt über das `KeypadListener.java` Interface alle Events für Tastendrücke an die `Screen.java` Klasse weiter. Die Komponente `Screen.java` gibt über das Interface `ATMListener.java` Events wie z.B. einen Modus-Wechsel oder das Betätigen der Enter-Taste an die `ATM.java` Klasse weiter.

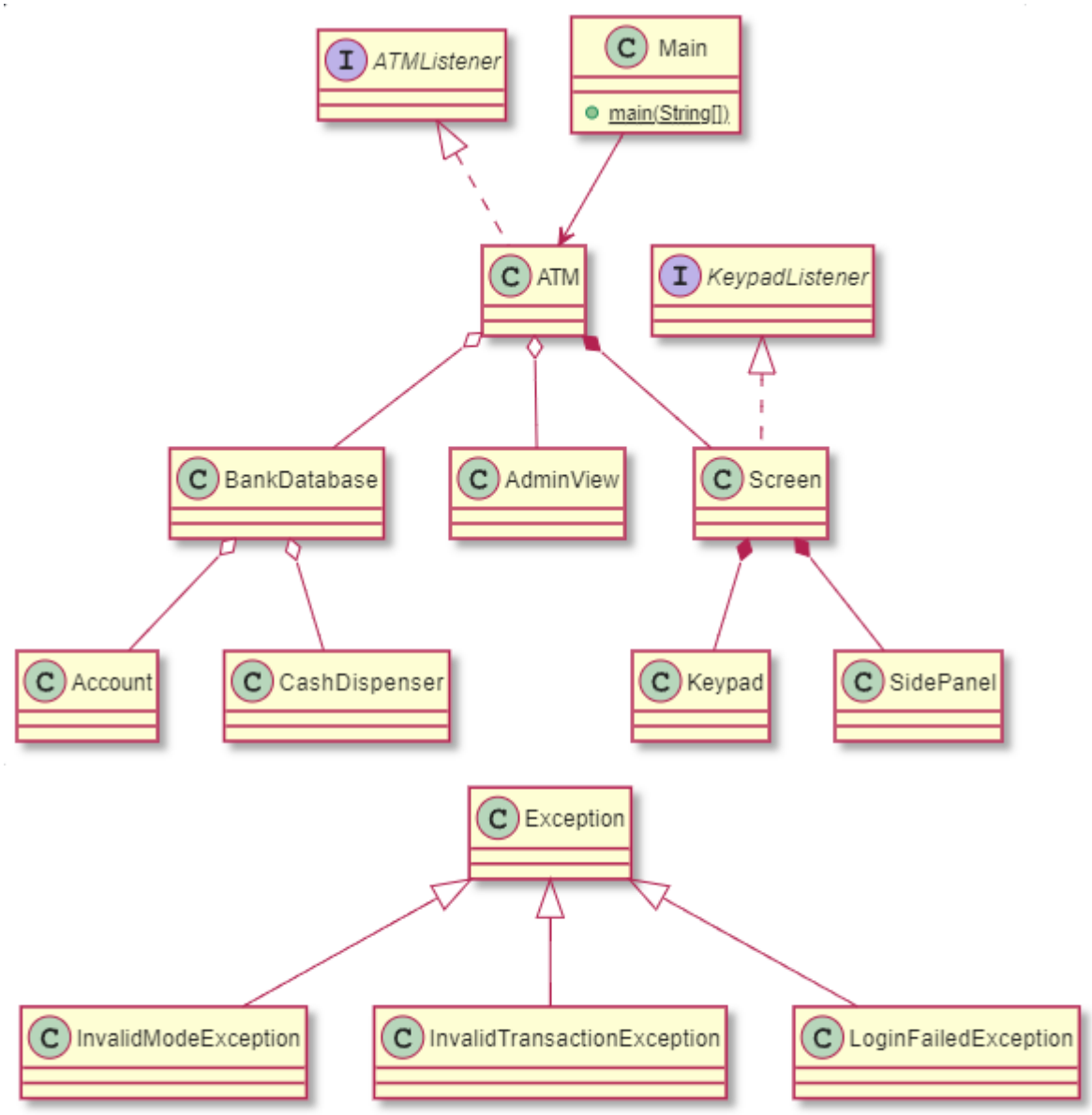
3.1.3 Schnittstellen

Hier werden alle Schnittstellen des Systems beschrieben.

- 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

Der Aufbau der Systemarchitektur ist weitestgehend modular gestaltet und ist hier in einem Klassendiagramm dargestellt.



3.2 Systementwurf

3.2.1 Systemdekomposition

Im folgenden Abschnitt werden die einzelnen Komponenten des Systems und ihre Funktionen beschrieben.

Das System lässt sich hauptsächlich durch die Bestandteile `Guthaben anzeigen`, `Geld abheben` und `Geld einzahlen` beschreiben. Zusätzlich gibt es ein `Menü` eine `Admin-Ansicht` und eine `Login`, sowie eine `Logout` Funktion.

Vom Menü aus, ist es einem Benutzer möglich alle relevanten Funktionalitäten durch das Drücken einer Zahl zu erreichen. Die Funktion `atmSwitchModeAction()` wechselt nun, je nach eingegebener Zahl, in den entsprechenden Modus. Eine weitere wichtige Komponente des Systems ist das `Keypad`, welches die verschiedenen Knöpfe darstellt. Dieses befindet sich immer in der linken Hälfte des Fensters und hilft dem Nutzer bei der Bedienung des Automaten. Es wird in dem Konstruktor der Klasse `Screen.java` zusammen mit dem SidePanel initialisiert.

Das `SidePanel` hat, wie das Keypad, eine eigene Klasse. Es befindet sich auf der rechten Hälfte des Fensters und beinhaltet unter anderem einen „Back-Button“. Mit diesem kann zurück in den „Menü-Modus“ gewechselt werden. In dem SidePanel befindet sich außerdem das Textfeld, in welchem die Benutzereingabe angezeigt wird, sowie ein JLabel. Dieses zeigt, je nach Modus, zum Beispiel das verfügbare Geld, oder die verschiedenen Optionen mit entsprechender Eingabe an.

Eine weitere Funktionalität ist die `Admin-Ansicht`. Loggt sich ein Admin ein, öffnet sich ein neues Fenster. In diesem können die Daten der Benutzer geändert und anschließend gespeichert werden.

3.2.2 Designalternativen und –Entscheidungen

Es wurde sich dazu entschieden die einzelnen Funktionalitäten mit Hilfe von verschiedenen Modi zu implementieren. Der Bankautomat befindet sich zu jedem Zeitpunkt in einem bestimmten Modus und reagiert, je nach Modus, unterschiedlich auf bestimmte Eingaben. Dieser Ansatz unterscheidet sich von der ursprünglichen Version des Automaten. Hier gab es keine Modi und die verschiedenen Funktionen, wie das Geldabheben, wurden von eigenen Klassen übernommen.

In der alten Version des Bankautomaten, konnte ein Admin mit Hilfe eines Iterators auf die einzelnen Benutzer zugreifen. In dem überarbeiteten Modell ist es möglich, aus einer Liste von Benutzern den gewünschten per Mausklick auszuwählen. Dies ermöglicht eine einfachere und schnellere Bearbeitung.

Zudem wird das Speichern der verschiedenen Benutzer nicht mehr innerhalb einer Java-Klasse übernommen, sondern außerhalb in einer JSON-Datei. Die Benutzerdaten werden mit Hilfe der Klasse `BankDatabase.java` in diese Datei übertragen.

3.2.3 Cross-Cutting-Concerns, NFRs

Nun werden kurz die Cross-Cutting-Concerns des Systems, sowie der Umgang mit diesen, vorgestellt.

Ein Benutzer soll in jedem Modus eine Eingabe tätigen können. Daher wurde das Keypad und ein entsprechendes Textfeld so implementiert, dass diese Komponenten stets sichtbar und verfügbar sind. Andere Komponenten werden teilweise unsichtbar gemacht, da diese nicht in jedem Modus gebraucht werden.

Ein weiterer Cross-Cutting-Concern ist das Geben von passendem Feedback an den Benutzer. Hier soll dem Benutzer, unabhängig von dem aktuellen Modus, stets mitgeteilt werden, wenn er eine ungültige Eingabe getätigt hat. Für diese Art von Fehlermeldungen wurde im untersten Bereich des Fensters ein Textfeld angelegt, welches die jeweilige Nachricht in roter Farbe anzeigt.

Außerdem ist die Validierung des Inputs bei einem Bankautomaten äußerst wichtig. Deshalb werden die Eingaben stets auf Richtigkeit überprüft. So wird beispielsweise sichergestellt, dass das eingezahlte Geld keinen Maximalwert überschreitet. Ebenso muss sichergestellt werden, dass ein Benutzer nicht mehr Geld abheben kann, als gerade für ihn verfügbar ist.

Bezüglich der Nicht-funktionalen-Anforderungen wurde auf eine hohe Performance und Bedienbarkeit geachtet. Dem Benutzer wird das Bedienen des Automaten durch ein intuitives Interface leichtgemacht. Die Wartezeiten sind kurz, da die Funktionen zur Berechnung von Überweisungen und Kontoständen eine geringe Laufzeit aufweisen.

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

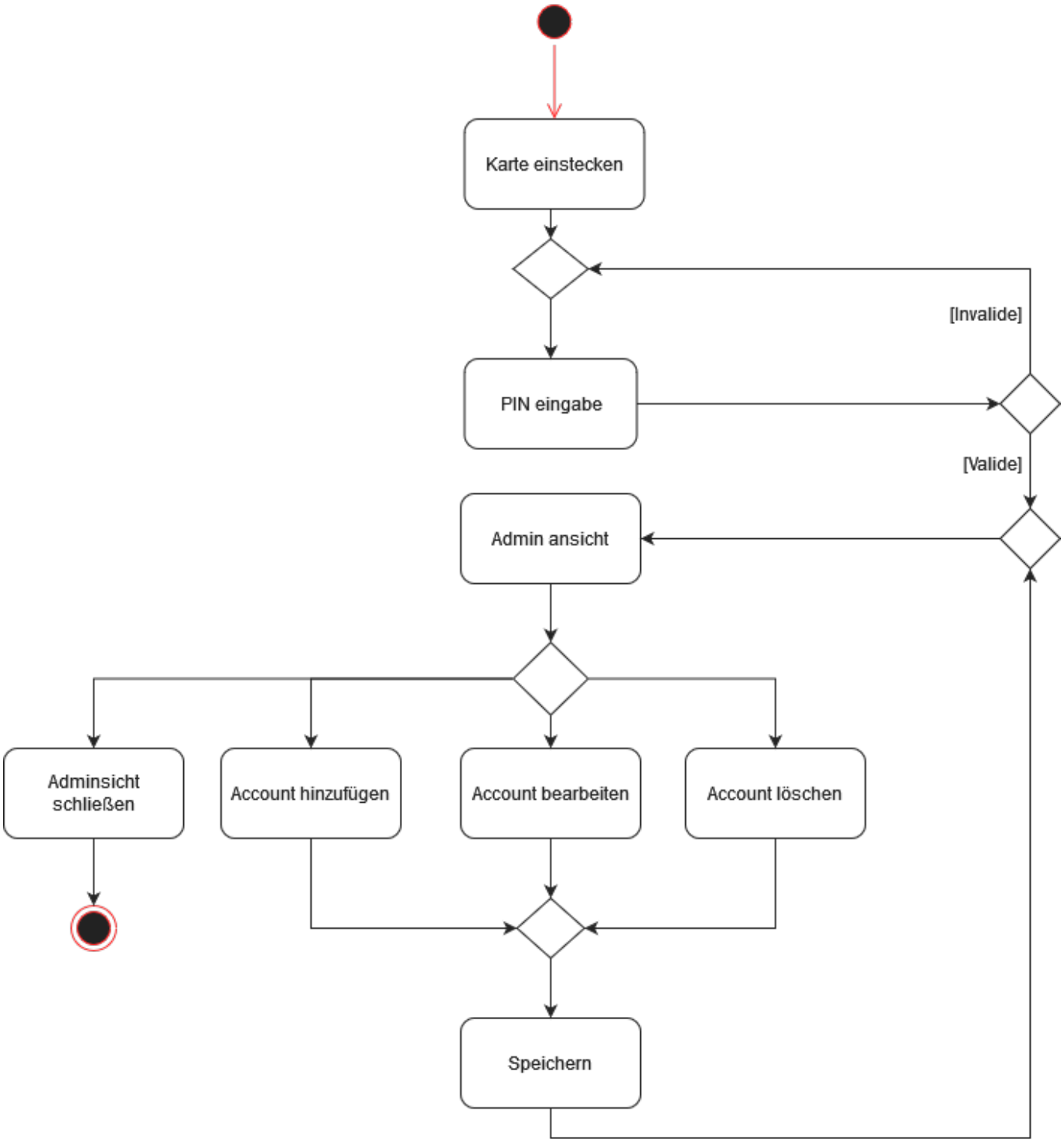
optische (Bildschirm-Ausgabe) und mechanische Ausgabe Information (Geldauszahlung) erzeugen. Die Rückgabe-Informationen werden vom Menschen visuell (Bildschirm-Information) und haptisch (Annahme des ausgezahlten Geldes) verarbeitet.

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

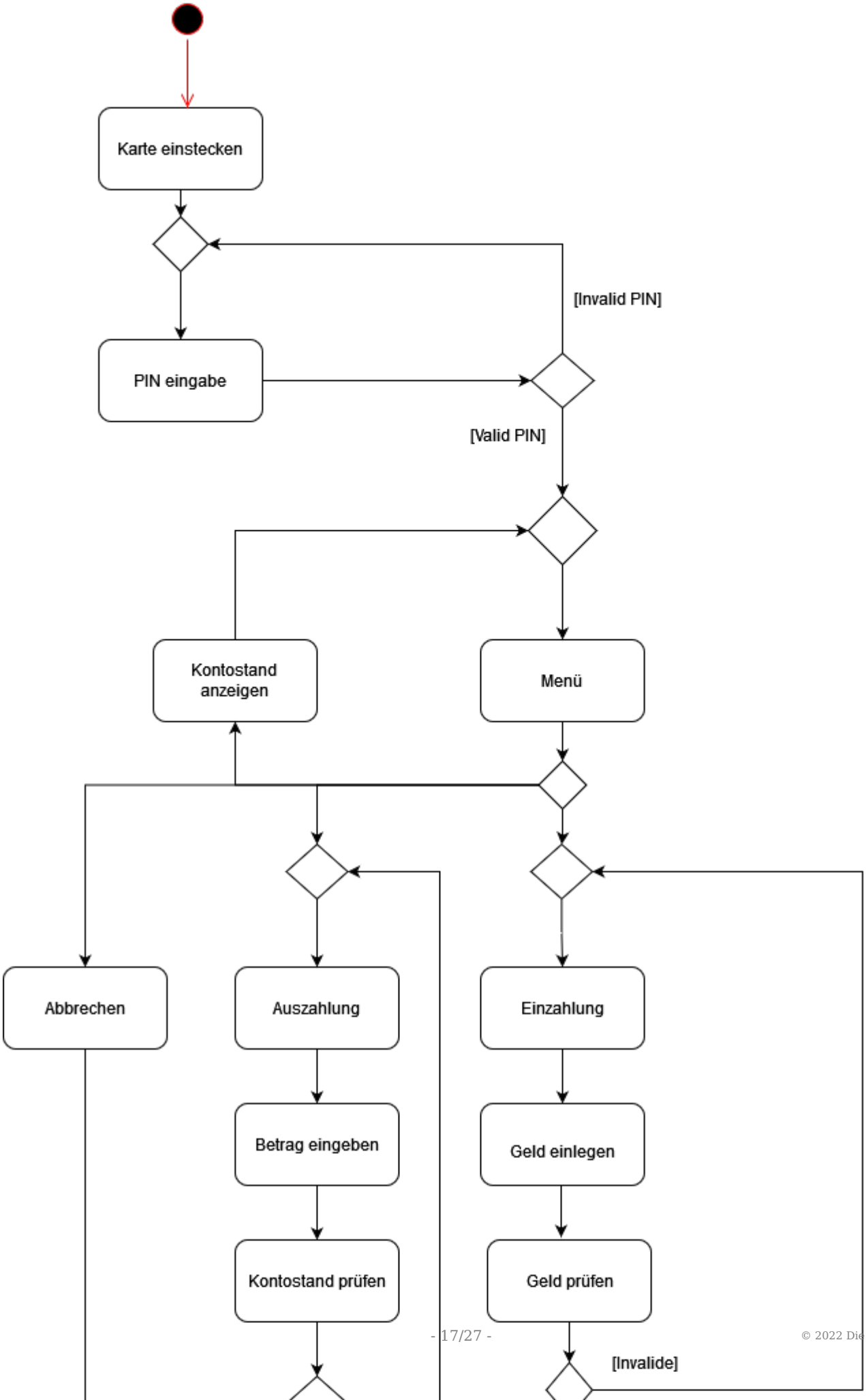
3.3.2 Gestaltungsprinzipien und Style-Guide

3.3.3 Interatkionsmodellierung

Adminansicht



Benutzeransicht



4. Testdokumentation

In der folgenden Dokumentation werden die für das Projekt durchgeführten Test beschrieben. Diese sind entweder manuell oder mit Hilfe von JUnit ausgeführt worden.

Name	**Sind Komponenten initialisiert**
Anforderung	Die ATM-Instanz soll einen screen und eine bankDatabase haben
Vorbedingung	ATM-Instanz wird erzeugt
Nachbedingung	Screen und bankDatabase des ATM sindinitialisiert
Testschritte	Stelle sicher, dass Komponenten nicht null sind

Name	**Wechsel in BALANCE Modus**
Anforderung	Mit dem Input "1" soll in den BALANCE Modus gewechselt werden
Vorbedingung	ATM-Instanz wird erzeugtUser loggt sich einMomentan im MENU Modus
Nachbedingung	Guthaben wird angezeigtATM im BALANCE Modus
Testschritte	Funktion atm.atmEnterAction() wird mit Input "1" aufgerufen

Name	**Falscher Input in Menü**
Anforderung	Bei falschem Input soll ATM im selben Modus bleiben
Vorbedingung	ATM-Instanz wird erzeugtUser loggt sich einMomentan im MENU Modus
Nachbedingung	ATM gibt Fehlermeldung, resettet das Textfeld und bleibt im selben Modus
Testschritte	Funktion atm.atmEnterAction() wird mit falschem Input aufgerufen

Name	**"Back" Button**
Anforderung	Der "Back" Button, soll den Modus zu MENU wechseln
Vorbedingung	ATM-Instanz wird erzeugtUser loggt sich einMomentan im MENU Modus
Nachbedingung	ATM befindet sich wieder im MENU Modus
Testschritte	Wechsel in BALANCE Modus, Drücken auf "Back" ButtonWechsel in WITHDRAWAL Modus, Drücken auf "Back" ButtonWechsel in DEPOSIT Modus, Drücken auf "Back" Button

Name	**"Clear" Button**
Anforderung	Bei Drücken auf den "Clear"-Button soll das Textfeld resettet werden
Vorbedingung	ATM-Instanz wird erzeugtUser loggt sich einMomentan im MENU Modus
Nachbedingung	Das Textfeld ist leer
Testschritte	Beliebiger Input wird in Textfeld eingegeben"Clear"-Button wird gedrückt

Name	**Ungültiger Pin Input**
Anforderung	Bei falscher Pin soll eine LoginFailedException geworfen werden
Vorbedingung	ATM-Instanz wird erzeugtEin neuer Account wird erstellt
Nachbedingung	ATM hat keinen Pin akzeptiert, da Pins aus 4 Ziffern bestehen müssenATM befindet sich noch im LOGIN Modus
Testschritte	Anmeldungsversuche mit verschieden ungültigen PinsZuerst ein Pin mit Buchstaben, dann ein Pin mit 5 Ziffern und zuletzt ein Pin mit 3 Ziffern

Name	**Neuen Account erstellen**
Anforderung	In der AdminView soll ein neuer Account erstellt werden können
Vorbedingung	ATM-Instanz wird erzeugt
Nachbedingung	Neuer Account wurde angelegtATM im ADMIN Modus
Testschritte	Neuer Admin-Account wird erstellt und der Datenbank hinzugefügtDer Admin loggt sich mit seiner Pin einÜberprüfen, ob die Länge der Account Liste sich um 1 erhöht hat

Name	**Credit und Debit Funktion**
Anforderung	Credit Funktion soll das Guthaben um mitgegebenen Betrag erhöhenDebit Funktion soll das Guthaben um mitgegebenen Wert verringern
Vorbedingung	ATM-Instanz wird erzeugtNeuer Account "a1" wird angelegt
Nachbedingung	Guthaben ist gleich hoch wie vor der Durchführung des Tests
Testschritte	a1.credit(5) wird aufgerufenÜberprüfen, ob sich Guthaben um 5 erhöht hata1.debit(5) wird aufgerufenÜberprüfen, ob sich Guthaben um 5 verringert hat

5. Abnahmedokumentation

coming soon

6. Benutzerdokumentation

coming soon

7. Projektdokumentation

coming soon

8. Codedokumentation

8.1 Code Ist-Dokumentation

Die Beschreibung in diesem Dokument ist zusätzlich zu dem kommentierten Code im Ordner `ATM-Machine-Old`.

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 Durchgeführte Veränderungen

- Änderung der PIN auf 4 Stellen
- Über das X kann das Programm beendet werden
- Über die Abbrechen-Funktion im Menu kann sich der Benutzer abmelden
- Die internen Klassen, die das Event-Handling übernahmen, wurden entfernt
- Event-Handling der UI Elemente werden mit zwei Interfaces umgesetzt
- `KeypadListener.java` kommuniziert die Tastendrücke
- `ATMListener.java` kommuniziert einen Modus-Wechsel und das Betätigen der Enter-Taste
- Auslagerung der Admin-Ansicht in ein neues Fenster `AdminView.java`
- Die Sprache des Programms wurde auf Deutsch umgesetzt
- Verbessertes Error-Handling
- Accounts werden mit einer .json Datei gelesen und gespeichert
- Geld wird in Scheinen aus dem Vorrat im Cash Dispenser ausgegeben
- Es werden die höchstmöglichen Scheine gewählt

8.2.2 Verbesserungsvorschläge

- Anbindung des Programms an eine Datenbank
- Besserer Algorithmus für die Ausgabe der Geldscheine aus dem Cash Dispenser