

# ATM Dokumentation

---

Die Panzerknacker

*Die Panzerknacker*

© 2022 Die Panzerknacker

# Inhaltsverzeichnis

---

1. ATM Dokumentation Startseite	3
1.1 Abstract	3
1.2 Das Team	3
2. Anforderungsdokumentation	4
2.1 Produktvision und Produktziele	4
2.2 Rollen und Personas	4
2.3 User Stories	8
2.4 Aufgaben	8
2.5 Begriffslexikon	8
2.6 Mengengerüst	9
2.7 Use Cases	9
3. Architekturdokumentation	10
4. Testdokumentation	11
5. Abnahmedokumentation	12
6. Benutzerdokumentation	13
7. Projektdokumentation	14
8. Codedokumentation	15
8.1 Code Ist-Dokumentation	15
8.2 Delta-Dokumentation	18

# 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

**Ernst Haft**

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

In progress!!

Ist-Dokumentation:

- Es sollte klar herauskommen, welche Funktionen die einzelnen Bausteine des Codes übernehmen, sodass Fachfremde ihn verstehen können (z.B. Konstruktor/Methode/..., übernimmt die Aufgabe des/erzeugt/implementiert/..., stellt die Funktion ... zur Verfügung).

Delta-Dokumentation:

- Erstellen Sie eine Optimierungsempfehlung des Codes: An welcher Stelle könnte dieser verbessert werden, hinsichtlich Bedienbarkeit, Wartbarkeit, Sicherheit usw. Implementieren sie die Verbesserungen und erstellen sie eine Delta-Dokumentation

## 2.5 Begriffslexikon

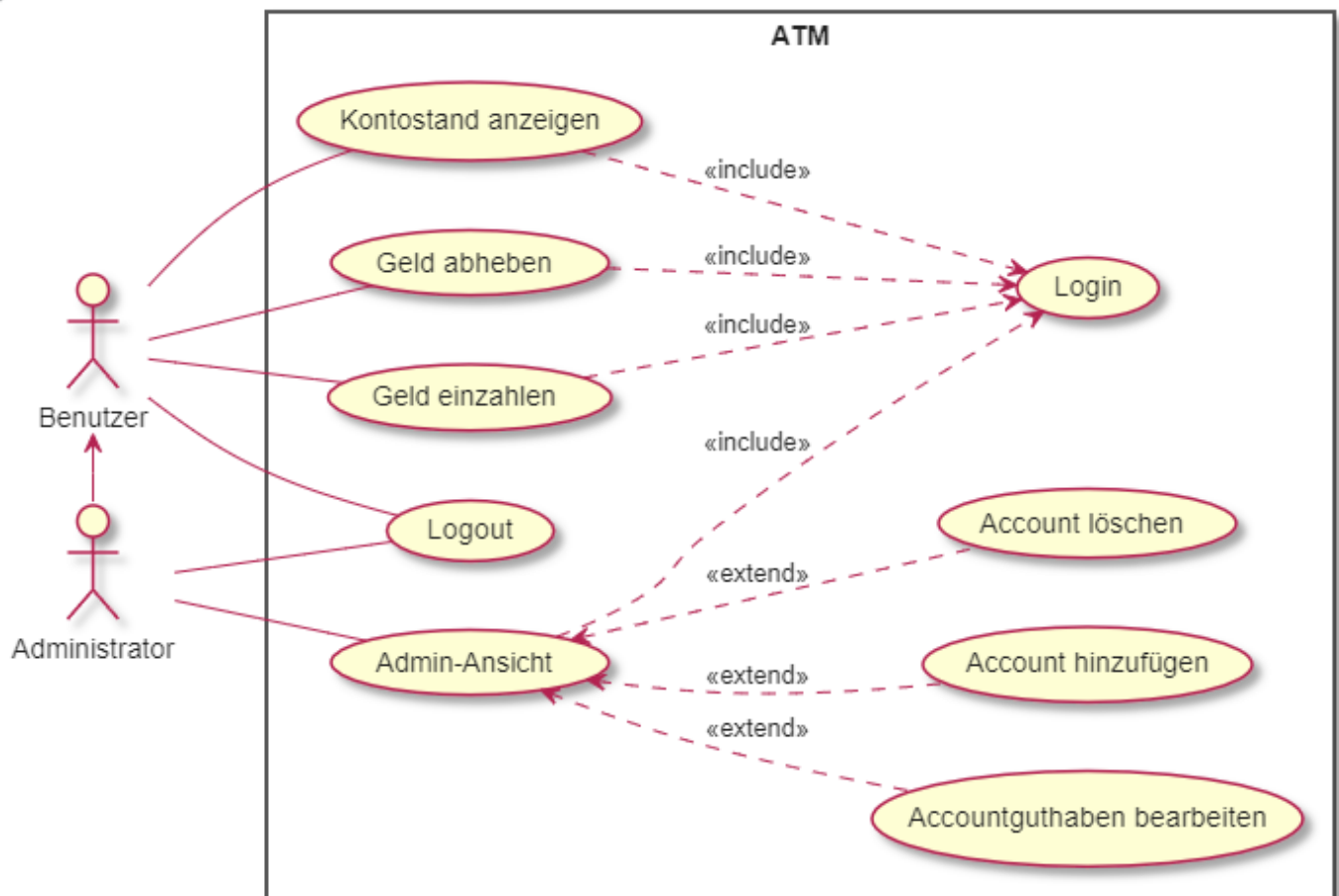
Begriff	Bedeutung	Beschreibung
<b>Cash Dispenser</b>	Bargeld im ATM-Dispenser	-
<b>Deposit Slot</b>	Geldfach zum Ein- und Auszahlen	-
<b>Balance</b>	Ist-Saldo auf einem Account	-
<b>Withdrawal</b>	Geld abheben	-
<b>Account Pin</b>	Geheimpin eines Accounts (unique)	-
<b>Account number</b>	Nummer eines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



## 2.6 Mengengerüst

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

## 2.7 Use Cases



## 3. Architekturdokumentation

---

coming soon

## 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 wechseln
- 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 unbenutzte 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
- Verbesserung der Klassenstruktur (weniger Klassen?)
- Verbesserung des UI-Handlings mit JFrame und JPanel
- Verbesserung der CashDispenser-Funktion
- Bankautomat in Deutsch oder Englisch
- Lizenz nur 1x erwähnen