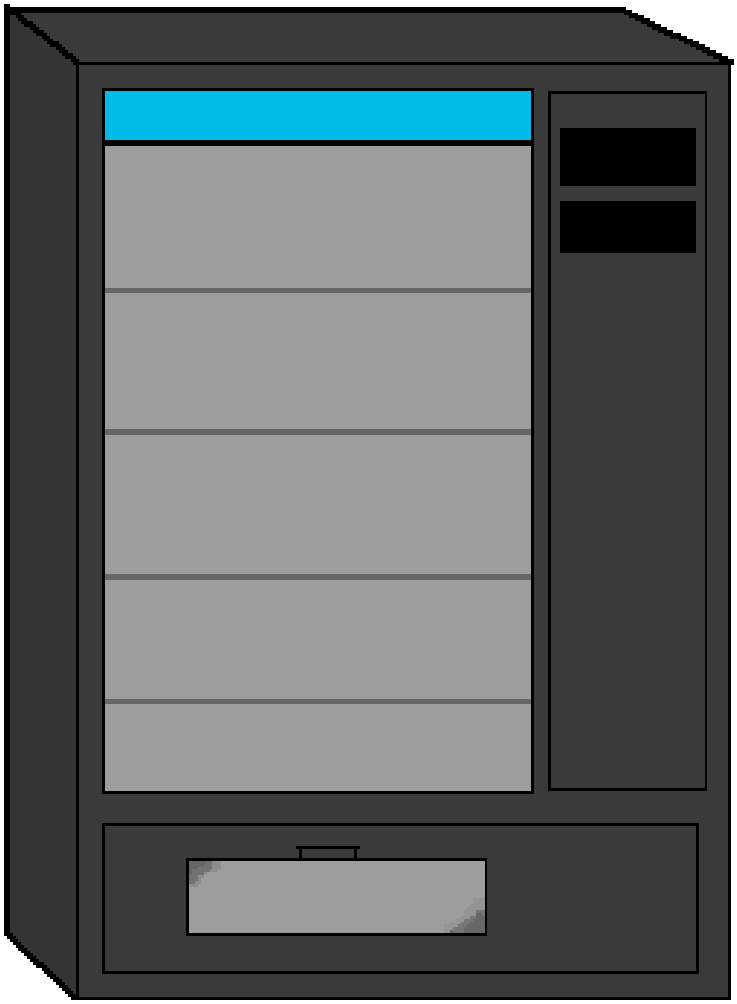

Projektauftrag «Snackautomat»

Thema: Java

Autoreninformationen

Autor: Nepomuk Crhonek & Michel Mahadeva & Denis Suciu

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Sinn und Zweck

2 Informieren

Der erste Schritt unseres Projektes ist das Informieren. Im Rahmen dieses Auftrags

2.1 Durchlesen des Auftrages

Anforderungen:

- Jedes Gruppenmitglied muss gleichmäßig zum Projekt beitragen (Nachweis durch Commit-Historie).
- VendingMachine (Snackautomat)
  - Verwaltet den Snackbestand.
  - Nimmt Geld entgegen und gibt Wechselgeld zurück.
  - Ermöglicht Wartung durch "Secret Key".
- Product (Produkt)
  - Attribute: Name, Preis, Anzahl im Bestand.
- Customer (Kunde)
  - Kann Geld einwerfen und Produkte auswählen.
  - Kann den Kauf abbrechen und Geld zurückfordern.
- Transaction (Kaufabwicklung)
  - Berechnet den Kaufprozess.
  - Gibt Wechselgeld zurück oder zeigt Fehler (z. B. zu wenig Geld).
- Admin (Wartungsperson)
  - Kann Produkte nachfüllen, Preise ändern und Produkte austauschen.
- Simulation (Steuerung der Automatenlogik)
  - Steuert den Ablauf der Simulation.
  - Verarbeitet Benutzereingaben.
- Starter (Main-Klasse)
  - Startet die Simulation.
- UML Activity Diagram
-

2.2 Offene Fragen klären

Wir haben vor dem Start des Projektes unsere offenen Fragen verschriftlicht. Diese haben wir dann mit dem Eisenhower-Prinzip priorisiert. Anbei findet man unsere Fragen und deren Priorisierung:

Frage	Priorität
Was für Branches sind nötig?	B
Von wo kriegen wir einen NFC reader?	D
Wie teilen wir die Arbeit auf?	A
Wie teilen wir unsere Zeit auf?	C

Tabelle 1 - Offene Fragen

2.3 Ziele

Die Ziele des Projektes haben wir in Muss- Soll- und Kann-Ziele definiert. Hauptziel dabei ist, die Muss-Ziele zu erreichen. Alles andere hat weniger Priorität und ist somit weniger dringend.

Muss-Ziele:

- Schrittweise Zahlen können
- Den Snackautomaten bearbeiten
- Produkte abziehen
- Eine Art von Authentifizierung der Mitarbeiter

Soll-Ziele

- User-CLI

Kann-Ziele

- GUI
- NFC-Authentification
- Mit Firebase verknüpfen

2.3 Checkliste Vorgehen

- ☐ Auftrag durchlesen
- ☐ Aufbau des Projektes besprechen
- ☐ Aufgaben definieren
- ☐ Meilensteine setzen
- ☐ Aufgaben aufteilen
- ☐ Übersicht zu den Klassen in Miro darstellen
- ☐ UML Diagramm zeichnen
- ☐ Grundfunktionen mit Attributen der Klassen implementieren
- ☐ Kontrollieren und Testen des Programmes

3 Planen

Im Schritt des Planens war uns wichtig, dass jedes Teammitglied wusste, was die Person zu dem Zeitpunkt machen muss. Dabei war uns zu Beginn wichtig, dass keine Person auf die andere warten musste, um die nächste Aufgabe zu machen.

3.1 Arbeitsaufteilung

Aufgabe	Erfüllt von…
Übersicht zu den Klassen in Miro darstellen	Denis
UML-Diagramm zeichnen	Nepomuk
Grundfunktionen mit Attributen der Klassen implementieren	Michel
Transaktion implementieren	Denis
Wartung der Maschine	Michel
Firebase	Nepomuk
UI erstellen und konfigurieren	Michel
Sicherheit	Nepomuk
Frontend Design erstellen und Produktbilder erstellen	Denis

Tabelle 2 - Arbeitsaufteilung

3.2 Arbeitsplanung

3.2.1 Nötige Arbeitsmittel

Zu diesem Projekt gibt es viele Mittel, die man braucht, um es so, wie wir es gemacht haben, zu vollenden. Anbei werden die wichtigsten aufgezählt.

3.3 Klassenaufteilung

Wir haben als erstes geschaut, wie viel Klassen wir für den Auftrag erstellen müssen. Dazu benutzten wir ein Miro Board als Notizpapier und sind auf den Entschluss gekommen, eine Class für den Automaten, sowie eine Class für die einzelnen Produkte mit einer ENUM für die einzelnen Produktsorten.

3.4 Übersicht der Branches

Branchname	Nutzen	Ersteller
User-Input	Alle Menüs in dem der User die Eingaben dem Programm geben kann	Michel
customer	Customer Class erstellen	Michel
vending-machine	Vending-Machine class mit Product ArrayList erstellen und buyProducts	Michel
Security	NFC Scanner scannt Karte von Admin	Nepomuk
payment		
User-interface	Eine Frontend mit visuelle Produkte	Denis

4 Entscheiden

Im Schritt des Entscheidens geht es darum, dass wir bei den offenen Fragen vom Beginn, welche mehrere Antworten für sich haben, uns entscheiden, welche wir dann umsetzen möchten.

Kriterium	Gewichtung (0.5-2)	(Variante A) Passwort	(Variante B) Badge(NFC)
Sicherheit	2.0	4	3
Benutzerfreundlichkeit	1.5	2	5
Implementierungsaufwand	1	3	2
Wartungsaufwand	0.5	2	4
Erweiterbarkeit	1	2	5
Gesamtscore	**6**	**19**	**21.5**
Rating (0-5)		**3.2**	**3.6**

Tabelle 3 - Entscheidungsmatrix

5 Realisieren

5.1 Einrichten zur Kollaboration

Im Rahmen dieses Auftrages mussten wir zum ersten Mal mit Branches im Git arbeiten. Also haben wir, sobald wir das Projekt erstellt und auf GitHub gepusht haben, die oben angegebenen Branches erstellt und die bereits aufgeteilten Aufgaben abgerabeitet.

5.2 Product Class

Die Product Class entspricht der einzelnen Produkte, wobei man herausfinden kann zu welcher Produktsorte sie gehören . Diese Produkte haben eine UUID und beim Kauf von Kunden werden solche Produkte von der Maschine entfernt.

5.3 Product Sort Class

In der Product Sort Class haben wir für jede Produktsorte eine Liste von Produkten erstellt. Jede Produktsorte kann man an man am Namen unterscheiden. Ausserdem hat jede Sorte einen anpassbaren Preis. In dieser Klasse werden die einzelnen Produkte aufgelistet in Form einer ArrayList.

5.4 Customer Class

Die Kunden unterscheiden wir vom Namen. Jeder Kunde hat bei unserem Programm einen aktuellen Kontostand.

5.5 Maintenance Class

Als Employee kann man beim Snackautomaten den Produktbestand etc. beeinflussen. Dafür braucht man einen Sicherheitsmechanismus. Jedoch sind in dieser Klasse nur die Funktionen, welche den Automaten beeinflussen.

5.6 Vending Machine Class

Die Vending Machine Class ist die zentrale Komponente unseres Programms. Sie verwaltet die verfügbaren Produktsorten in einer ArrayList und wickelt alle Transaktionen ab. Diese Klasse enthält Methoden zum Befüllen des Automaten mit Produkten, zum Kauf von Produkten und zur Abwicklung von Zahlungsvorgängen.

Der Snackautomat hat eine maximale Kapazität für jede Produktsorte. Wenn der Produktbestand niedrig ist (bei uns unter 30% der Kapazität), füllt der Automat die Produkte automatisch auf, um einen kontinuierlichen Service zu gewährleisten. Darüber hinaus verwaltet der Automat den Zahlungsprozess. Hierbei wird von der PayCoin-Klasse jeweils ein JFrame Fenster gestartet um die nächste Münze des Kunden zu erhalten. Sobald eine Münze ausgewählt wird, schliesst das aktuelle Fenster und bei Bedarf wird ein neues Fenster geöffnet. Bei diesem Fenster hatten wir zuerst Probleme, diese konnten wir jedoch mit folgendem while beheben.

```
while(coin.get() == null);
setVisible(false);
dispose();
return coin.get();
```

Hierbei wird erst dann ein Wert returned, wenn eine Münze ausgewählt wird.

5.7 Menu Class

Die Menu Class stellt die Schnittstelle zwischen dem Kunden/Mitarbeiter und dem Snackautomaten dar. Sie zeigt verschiedene Menüs an, je nachdem, ob der Benutzer ein Kunde oder ein Mitarbeiter ist. Für Mitarbeiter enthält die Menu Class einen

Authentifizierungsprozess mittels NFC-Technologie, um Administratorrechte zu überprüfen, bevor der Zugriff auf Wartungsfunktionen gewährt wird.

Abbildung 1 - unser NFC Reader

5.8 Main Class

Die Main Class dient als Einstiegspunkt in unsere Anwendung. Sie bietet Benutzern die Wahl zwischen einer grafischen Benutzeroberfläche (GUI) und einer Kommandozeilenschnittstelle (CLI). Je nach Auswahl des Benutzers initialisiert sie die entsprechende Schnittstelle und startet die Anwendung.

5.9 UI Class

Die UI Class implementiert die grafische Benutzeroberfläche für unseren Snackautomaten. Sie erweitert JPanel und implementiert Runnable, um eine reaktionsschnelle Oberfläche zu erstellen, die mit 60 Bildern pro Sekunde aktualisiert wird. Die UI zeigt die Produkte des Automaten grafisch an und ermöglicht es Benutzern, mit dem Automaten über eine Maus zu interagieren.

Abbildung 2 - Seitenanzeige des GUI's

5.9.1 Produktwahl

Mithilfe von MouseEvents konnten wir mit der Position der Maus, das ausgewählte Produkt herausfinden. Danach kann man mit Interaktionen mit den Benutzern Angaben wie die Anzahl des Produktes herausfinden. Die Transaktion erfolgt dann wie beim CLI.

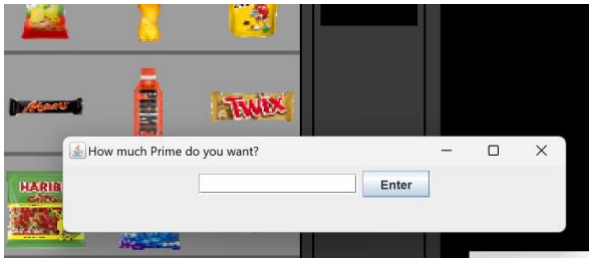

Abbildung 3 - Interaktionsbeispiel mit dem Nutzer

5.10 MovementMouse Class

Die MovementMouse Class erweitert MouseAdapter, um Mausbewegungen und -klicks für die GUI zu verfolgen. Sie zeichnet die x- und y-Koordinaten des Mauszeigers auf und verfolgt, wann die Maustaste gedrückt oder losgelassen wird.

6 Kontrollieren

6.1 Testing

Testing ist ein wichtiger Schritt unseres Projektes. Wir haben Wert darauf gelegt, dass nicht diejenigen eine Funktion testen, welche sie selbst gemacht haben. Dies dient dazu, dass man durch einer anderen Perspektive das Programm testen kann und einem eventuell etwas Neues auffällt.

Bevor wir mit dem GUI fertig wurden, haben wir bereits die meisten Funktionen mithilfe unseres CLI's testen können.

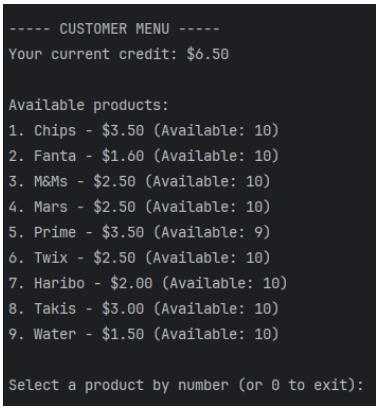

Abbildung 4 - Ansicht CLI für Kunden

6.1.1 White-Box Testing

Während des White-Box-Testings haben wir die internen Strukturen unserer Anwendung untersucht, um sicherzustellen, dass jede Komponente korrekt funktioniert. Wir haben getestet:

- Korrekte Initialisierung der Produktbestände beim Befüllen des leeren Automaten
- Richtige Kreditberechnung während des Produktkaufs
- Genaue Zahlungsabwicklung mit verschiedenen Münzeinheiten
- Authentifizierungsmechanismus für den Administratorzugang
- Produktauffüllungslogik, wenn der Bestand unter den Schwellenwert fällt

6.1.2 Black-Box Testing

Beim Black-Box-Testing haben wir die Anwendung aus der Endbenutzer-Perspektive getestet, ohne den internen Code zu berücksichtigen. Wir haben überprüft:

- Kundenablauf beim Kauf von Produkten
- Admin-Authentifizierung und Wartungsoperationen
- Reaktionsfähigkeit und Benutzerfreundlichkeit der Schnittstelle
- Fehlerbehandlung bei unzureichendem Guthaben
- Produktauswahl und Transaktionsabschluss

6.2 Kontrollraster Anforderungen

Nr.	Test	Eingabe	Kontollpunkt	Ergebnis	Datum	Tester*in	Status
1	Anwendung starten	Main-Klasse ausführen	Anwendung startet und zeigt Schnittstellenauswahl	Erfolgreich	27.02.2025	Denis	✅
2	Produktkauf durch Kunden	Produkt auswählen und bestätigen	Guthaben reduziert und Produkt ausgegeben	Erfolgreich	27.02.2025	Nepomuk	✅
3	Behandlung unzureichender Mittel	Versuch, mit unzureichendem Guthaben zu kaufen	System fordert zur zusätzlichen Zahlung auf	Erfolgreich	27.02.2025	Nepomuk	✅
4	Admin-Authentifizierung	Autorisierte NFC-Karte scannen	Admin-Menü erscheint	Erfolgreich	27.02.2025	Michel	✅
5	Produktpreisänderung	Preis über Admin-Menü ändern	Neuer Preis im Kundenmenü angezeigt	Erfolgreich	27.02.2025	Denis	✅
6	Neues Produkt hinzufügen	Produkt über Admin-Menü hinzufügen	Neues Produkt erscheint im Kundenmenü	Erfolgreich	27.02.2025	Michel	✅
7	Automatisches Auffüllen bei niedrigem Bestand	Produkte kaufen bis unter 30%	System füllt Bestand automatisch auf	Erfolgreich	27.02.2025	Nepomuk	✅
8	GUI-Reaktionsfähigkeit	Mit GUI-Schnittstelle interagieren	UI reagiert korrekt auf Mausereignisse	Erfolgreich	27.02.2025	Denis	✅

Tabelle 4 – Kontrollraster

7 Auswerten

7.1 Reflektionsraster

Nepomuk	Was lief gut?	Was lief nicht so gut?	Was würden wir nächstes Mal anders machen?
Umfang und Qualität vom Ergebnis	Ich finde unser Ergebnis ist sehr beeindruckend	Admin GUI	Zeit besser einplanen
Planung und Vorgehen vom Projekt	Arbeiten gut eingeteilt	Firebase	Für Firebase genügend Zeit einplanen
Aufteilung und Zusammenarbeit im Team	Ausgeglichene Aufteilung und jeder konnte für sich gut arbeiten	Zu Beginn war noch nicht jedem klar, was man machen muss	Alle lesen den Auftrag durch statt eine Person, welche dann den anderen Leuten erklärt

Tabelle 5 – Reflexionsraster Nepomuk

Denis	Was lief gut?	Was lief nicht so gut?	Was würden wir nächstes Mal anders machen?
Umfang und Qualität vom Ergebnis	Ich finde unser Ergebnis ist sehr beeindruckend	ADMIN GUI und Bezahlungsart	Andere Lösung für Bezahlungsart finden
Planung und Vorgehen vom Projekt	Ziele definieren	ADMIN GUI, und bisschen bessere Einteilung	Eine bisschen bessere Zeiteinteilung machen
Aufteilung und Zusammenarbeit im Team	-	Der Anfang war verwirrend wegen der Einteilung	Uns alle zusammen hinsetzen und die genaue Aufgaben Einteilung machen

Tabelle 6 - Reflexionsraster Denis

Michel	Was lief gut?	Was lief nicht so gut?	Was würden wir nächstes Mal anders machen?
Umfang und Qualität vom Ergebnis	GUI für Kunden und CLI	Bezahlungsart	Andere Lösung für Bezahlungsart finden
Planung und Vorgehen vom Projekt	Ziele definieren	Vorgehen hätte zu Beginn strukturierter sein können	Allen genau mitteilem, was sie machen sollten
Aufteilung und Zusammenarbeit im Team	Ausgeglichene Aufteilung	Anfang war verwirrend wegen der Einteilung	Alle sollten Auftrag lesen

Tabelle 7 - Reflexionsraster Michel

8 Quellen

8.1 Referenzdokumente

[1] 2024-2025_snackautomat-simulator-final.pdf

[2] Java API Dokumentation 17.0

8.2 Webquellen

NFC Scanner Java

Schach Tutorial für GUI

Erstellen von Pixelart