Snackautomat Challenge

Thema: Gruppenprojekt mit Java

Dokumentinformationen

Dateiname: BLJ2024-Snackautomat-leon-jan-thomas
Speicherdatum: 28.02.2025

Autoreninformationen

Autoren: Leon Probst, Jan Ludwig, Thomas Stern

E-Mail: leon.probst@noseryoung.com
jan.ludwig@noseryoung.com
thomas.stern@noseryoung.com

GitHub: Repository

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Einleitung

1.1 Sinn und Zweck

Diese Dokumentation beschreibt die Entwicklung eines Snackautomaten als Gruppenprojekt in Java. Ziel ist es, ein Programm zu erstellen, das den Verkaufsprozess eines realen Automaten simuliert und dabei Benutzern die Auswahl zwischen Snacks, Getränken und Zusatzartikeln ermöglicht. Neben dem Kaufprozess sollen auch Wartungsfunktionen wie das Auffüllen des Lagerbestands und die Verwaltung des Geldbestands implementiert werden. Das Projekt orientiert sich nach IPERKA, wobei die Phasen «Informieren» und «Planen» wichtig sind, um Anforderungen und potenzielle Fehler frühzeitig zu identifizieren, und «Kontrollieren» sicherstellt, dass die Implementierung den geplanten Standards entspricht.

2 Projektbeschreibung und Anforderungen

2.1 Zielsetzung

Das Ziel des Projekts ist die Entwicklung eines funktionalen Snackautomaten, der folgende Hauptanforderungen erfüllt:

- Simulation eines Verkaufsprozesses mit Produktauswahl, Zahlung und Ausgabe.
- Verwaltung eines Produktlagers und den Geldstand.
- Behandlung von Fehlerszenarien (zum Beispiel zu wenig Geld, ausverkaufte Produkte).
- Bereitstellung von Wartungsfunktionen für den Betreiber.

2.2 Funktionale Anforderungen

- Der Benutzer kann zwischen verschiedenen Produktkategorien (Snacks, Getränke, Zusatzartikel) wählen.
- Das Programm akzeptiert das Guthaben (CHF) des Benutzers als Eingabe.
- Fehlermeldungen werden bei ungültigen Eingaben oder nicht verfügbaren Produkten angezeigt.
- Wartungsmodus: Lagerbestände können aufgefüllt und Produkte können überprüft werden.

2.3 Nicht-funktionale Anforderungen

- Die Anwendung nach dem EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe) modelliert werden, um Benutzerinteraktion und Verarbeitung zu trennen.
- Der Code soll gut dokumentiert und einfach zu ändern sein.
- Die Benutzeroberfläche bleibt auf eine einfache Konsolenanwendung beschränkt.

3 Fehlerszenarien

Folgende typische Fehler wurden identifiziert und werden behandelt:

- Zu wenig Geld eingeworfen für das gewählte Produkt.
- Lagerbestand eines Produkts beträgt 0.
- Ungültige Eingaben (wie negative Geldbeträge).

4 Vorgehensweise nach IPERKA

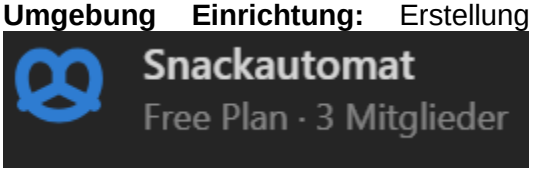
Wir nutzen die IPERKA-Methodik (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten), um das Projekt strukturiert umzusetzen.

4.1 Informieren

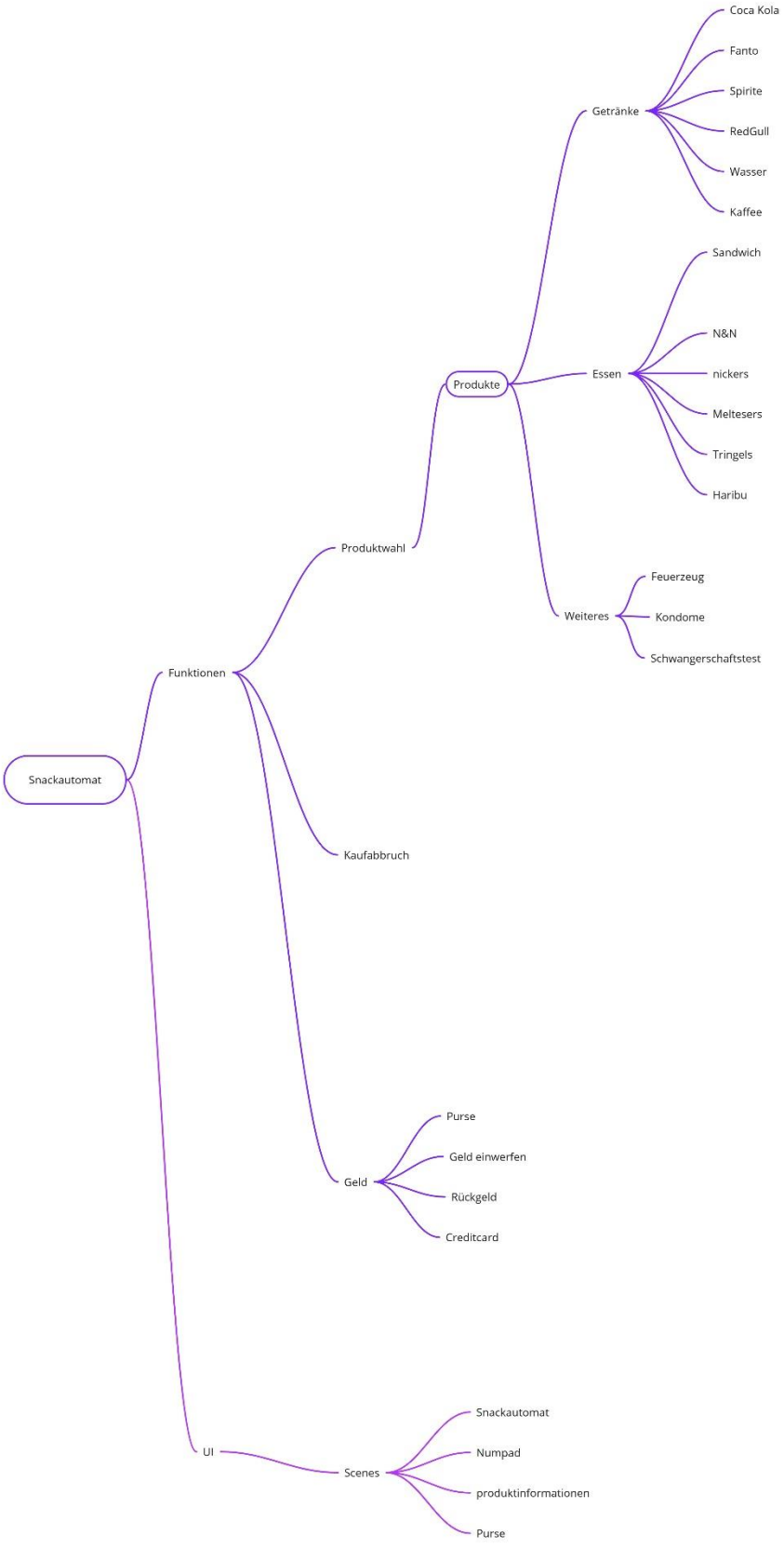
In dieser Phase haben wir Informationen gesammelt, um ein gutes Verständnis für den Snackautomaten zu entwickeln. Dazu gehörte die Analyse eines echten Automaten sowie die Identifikation von Anforderungen.

Aktivitäten

- **Recherche:** Untersuchung des typischen Ablaufs eines Snackautomaten: Produktauswahl -> Bezahlung -> Ausgabe
- **Fehleranalyse:** Identifikation von Szenarien wie «zu wenig Geld» oder «Produkt nicht verfügbar».
- **Umgebung Einrichtung:** Erstellung von Workspace in Notion und Miro.



- **Mindmap:** Erstellung eines Mindmaps auf Miro, um Ideen zu sammeln und erste Konzepte zu visualisieren:

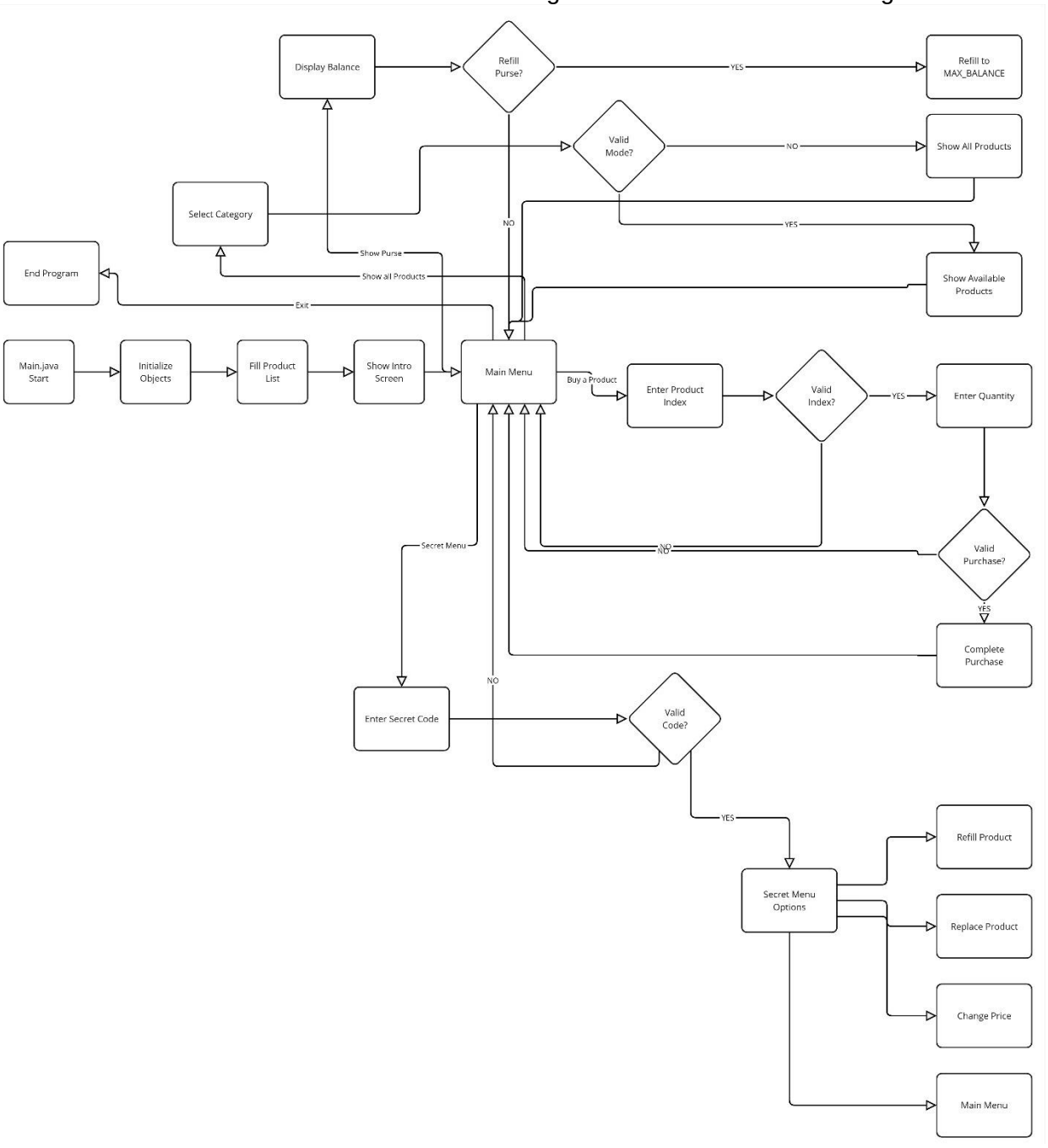


4.2 Planen

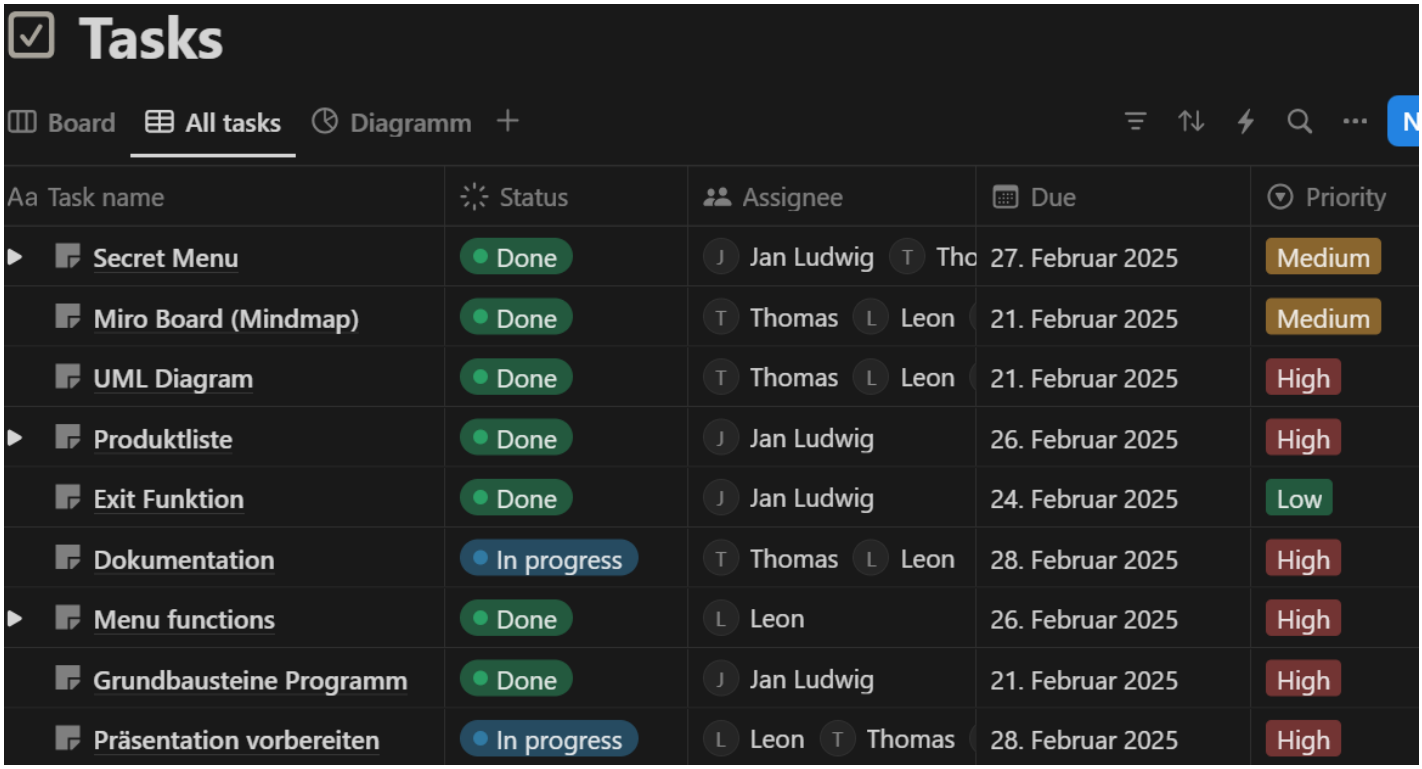
Basierend auf den gesammelten Informationen haben wir die Projektstruktur geplant, Klassen und Git-Branches entworfen und Aufgaben verteilt. Ziel war es, eine klare Roadmap zu schaffen und Missverständnisse zu vermeiden.

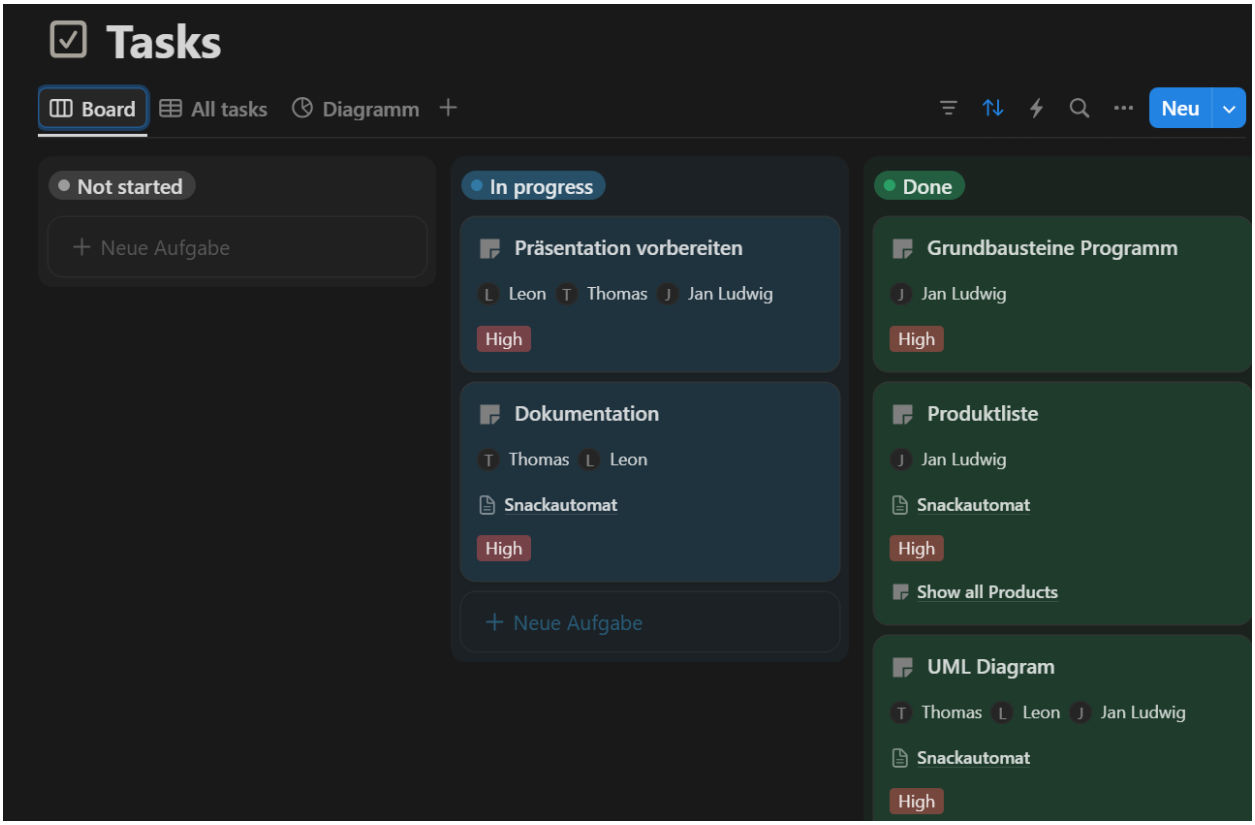
Aktivitäten

- **Klassenentwurf:** Definition von Klassen dargestellt in einem UML-Diagramm.



- **Aufgabenplanung:** Erstellung von Tasks und einem Kanban-Boards auf Notion mit Tasks wie, «Klassen implementieren» und «Dokumentation schreiben».





- **EVA-Prinzip:** Trennung von Eingabe/Ausgabe (Benutzeroberfläche) und Verarbeitung (Snackautomat, Zahlung).

Ergebnisse

- Mindmap und UML-Diagramm als Entwurf erstellt.
- Kanban-Board als Arbeitsplan mit klaren Verantwortlichkeiten.

4.3 Entscheiden

In dieser Phase wurden endgültige Entscheidungen über Aufteilung, Vorgehen und Teamorganisation getroffen, um die Umsetzung vorzubereiten.

Aktivitäten

- **Aufbau:** Konsolenanwendung als UI.
- **Toolwahl:** IntelliJ (IDE), Miro (UML, Mindmap), Notion (Tasks, Kanban-Board).
- **Teamentscheidungen:** Flexible Rollenverteilung, häufige Meetings, Nutzung visueller Hilfsmittel (Mindmap, UML) zur Klärung von Vorstellungen.

Ergebnisse

- Jedes Teammitglied kennt die geplanten Schritte und weiss, wie die Beiträge zum Gesamtprojekt beitragen.
- Vorstellung vom Snackautomaten wurde geschaffen
- Effektive Zusammenarbeit

4.4 Realisieren

In der Realisierungsphase wird der geplante Snackautomat in Java umgesetzt. Hier setzen wir die im UML-Diagramm definierten Klassen in Code um, integrieren die Benutzeroberfläche mit der Logik und beginnen mit ersten Tests. Ziel ist es, eine lauffähige Version zu erstellen, die die Pflichtfunktionen (Produktauswahl, Zahlung und Ausgabe) abdeckt, während wir gleichzeitig flexibel bleiben, um Anpassungen vorzunehmen.

Implementierung der Klassenstruktur:

- Entwicklung der Klasse Products mit Attributen wie Name, Preis und Getters.
- Umsetzung der Zahlungsfunktion zur Abrechnung des Geldes und Prüfung ob man genug Geld hat für einen Artikel

- Implementierung der Benutzeroberfläche, die die Konsoleninteraktion übernimmt zum Beispiel Menü anzeigen, Eingaben lesen etc…

Technische Umsetzung:

Wir haben unser Git-Repository mit fünf Branches eingerichtet: drei Feature-Branches (feature-leon, feature-jan, feature-thomas), ein develop-Branch zum Testen der integrierten Änderungen und der main-Branch für die stabile Version. Die ersten Merges von Feature-Branches in develop liefen erfolgreich, jedoch kamen zum Ende einige Conflicts, die wir aber wiederum beheben konnten.

Ergebnisse:

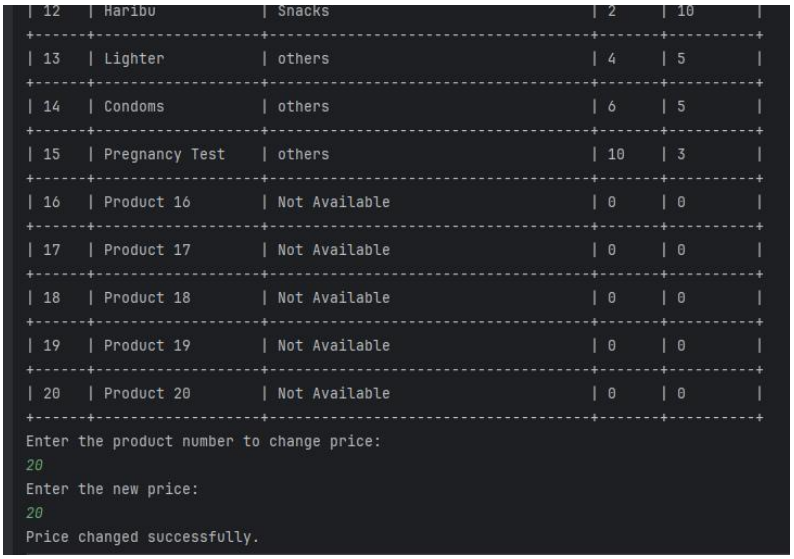
- **Teilweise funktionale Anwendung:** Eine erste Version des Snackautomaten ist lauffähig, die es ermöglicht, ein Produkt auszuwählen, eine Zahlung einzugeben und eine Rückmeldung, zum Beispiel: «Error, Zu wenig Gel», zu erhalten. Die Konsolenoberfläche zeigt eine ASCII-Art und ein einfaches Menü mit verschiedenen Optionen an.

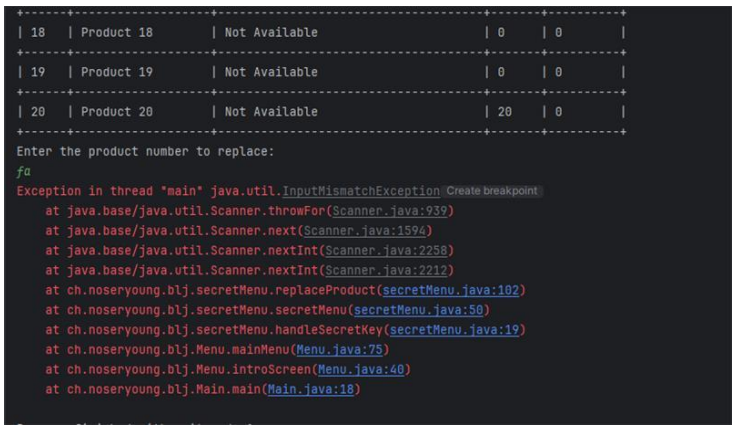
- **Fehlerbehandlung:** Szenarien wie «zu wenig Geld» oder «Produkt nicht mehr verfügbar» werden korrekt erkannt und führen zu entsprechenden Errors in der Konsole.
- **Teamfortschritt:** Das Kanban-Board auf Notion zeigt abgeschlossene Tasks (zum Beispiel und offene Aufgaben, was eine klare Übersicht über den aktuellen Stand bietet.

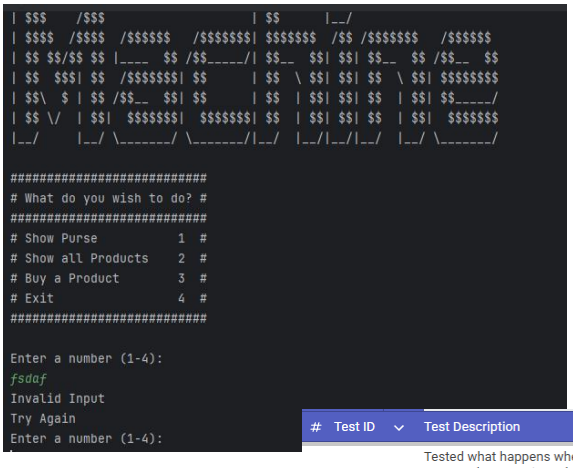
4.5 Kontrollieren

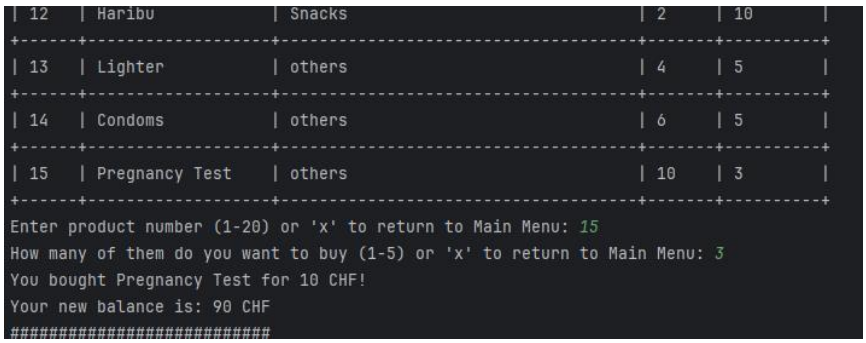
In der Kontrollphase überprüfen wir die Implementierung des Snackautomaten auf Funktionalität, Error frei und Übereinstimmung mit den geplanten Anforderungen. Ziel ist es, Fehler frühzeitig zu erkennen, die Qualität der Anwendung zu sichern und sicherzustellen, dass alle Kernfunktionen wie Produktauswahl, Zahlungsabwicklung und Fehlerbehandlung korrekt arbeiten.

Hier sind die entsprechenden Tests:









#	Test ID	Test Description	Expected Result	Actual Result	Pass/Fail	Status
1		Tested what happens when in Main Menu , when entering other numbers then 1-4	Show the Menu again with the Option to choose	Show the Menu again with the Option to choose	Pass	-
2		Tested all Menus and see what happens when entering random letters	Show the Menu again with the Option to choose	Show the Menu again with the Option to choose	Pass	-
3		Tested what happens when buying multiple things	Should buy requested amount	Only bought 1, when entered anything above 1	Fail	Resolved
5		Chaning the Price of a Product which doesnt exist	Shouldn't change the price and leave it at 0	Changed the price and when entered a string crashed	Fail	Resloved
6		What happens when I run restock all in secretMenu	It should restock all the items to 20 items	It restocked all the items to 20 items	Pass	
7						
8						

Es lief nicht alles wie erwartet, aber wir konnten alle Bugs beheben und sind mit dem Resultat zufrieden.

4.6 Auswerten

Die letzte Phase, die Auswertungsphase dient dazu, den gesamten Projektverlauf kritisch zu reflektieren, die erreichten Ziele zu bewerten und Erkenntnisse für zukünftige Projekte abzuleiten. Diese Phase schliesst das Projekt ab und bereitet uns auf die Präsentation vor, indem wir unsere Ergebnisse zusammenfassen und präsentierbare Highlights herausstellen.

Technische Bewertung:

- Überprüfung der Funktionalität: Werden alle Anforderungen erfüllt (wie die Produktauswahl, Zahlungsabwicklung, Fehlerbehandlung)? Gibt es noch Bugs oder Einschränkungen?
- Analyse der Codequalität: Ist der Code komplett, und gut kommentiert? Entspricht er dem EVA-Prinzip und den geplanten Klassen im UML-Diagramm?

Teamreflexion:

- Gemeinsames Meeting zur Besprechung: Welche Aspekte der Zusammenarbeit haben gut funktioniert. Wo gab es Herausforderungen?
- Tool-Reflektion: Waren Notion und Miro hilfreich? Gab es Einschränkungen oder Alternativen, die besser gepasst hätten?

Ergebnisse:

- Produktauswahl und Zahlungsabwicklung funktionieren, inklusive wie viel Geld man übrig hat. Fehlerbehandlungen sind stabil und wichtige Kommentare sind vorhanden.
- Flexible Rollen förderten unser Lernen und die Meetings verbesserten Absprache.
- Herausforderungen: Merge-Konflikte und Zeitdruck durch verbliebende Tests.

Lessons Learned:

- Welche Entscheidungen haben sich bewiesen? Welche Fehler würden wir vermeiden?
- Diskussion über persönliche Lernziele: Welche Fähigkeiten haben wir gestärkt?

Ergebnisse:

Workspaces auf Miro und Notion einzurichten war eine gute Idee, die uns geholfen hat, immer auf dem neusten Stand der Dinge zu bleiben und eine gute Übersicht zu haben. Bei neu implementierten Funktionen sollte man immer überprüfen ob die bereits funktionierenden

Funktionen, immer noch gehen. Wir alle haben unser Verständnis und die Zusammenarbeit mit Git verbessert.

5 Herausforderungen und Lösungen

- **Git:** Merging lief ohne Probleme, nur am Schluss sind einige Konflikte aufgetreten. Diese konnten wir jedoch wieder beheben mit einem neuen gitignore und manuelles Merging über IntelliJ.

- **Lernen:** Durch die Rotation der Aufgaben konnten wir alle neue Erfahrungen in der Zusammenarbeit mit Git sammeln.

6 Implementierungsplan

Tag 1:

Fertigstellung des UML-Diagramms, Notion, Miro und Git-Einrichtung.

Tag 2:

Klassen nach UML-Diagramm erstellt, danach gearbeitet und auf Notion alles festgehalten.

Tag 3:

Entwicklung der Zahlungslogik und Benutzeroberfläche mit ASCII-Arts. Beides abgeschlossen.

Tag 4:

Tests durchgeführt und letzte Fehler behoben.

Tag 5:

Dokumentation feingeschliffen und Präsentation vorbereitet.

7 Fazit und Ausblick

Reflexion

- **Erfolge:** Die erste Version des Snackautomaten ist funktional, mit einer ansprechenden ASCII-Art und stabiler Fehlerbehandlung. Die Nutzung von Git mit fünf Branches (drei Feature-Branches, ein develop und main Branch) hat die Zusammenarbeit erleichtert.
- **Herausforderungen:** Zeitdruck am Ende durch unterschätzte Aufgaben wie testing.

In zukünftigen Projekten werden wir frühere Tests und unsere Zeitplanung genauer einhalten.