Projekt Snackautomat
(Dokumentation)

I:

Informieren

Zu Beginn unseres Projekts haben wir uns mit der Aufgabenstellung auseinandergesetzt und analysiert, welche Anforderungen und Ziele zu erfüllen sind. Dabei haben wir das Dokument genau durchgelesen und die wichtigsten Punkte zusammengefasst.

Um ein besseres Verständnis für die Programmierung eines Snackautomaten zu erhalten, haben wir nach Programmierlösungen gesucht. Dabei haben wir YouTube-Videos und Online-Tutorials angeschaut, um uns Inspiration zu holen.

2024-2025_snackautomat-simulator-final (1).pdf

P:

Planen

Nachdem wir uns über die Aufgabenstellung informiert hatten, haben wir uns als Team zusammengesetzt, um die Planung des Projekts strukturiert anzugehen. Zunächst haben wir alle notwendigen Schritte in einem Word-Dokument festgehalten. Wichtig in unserer Planung war die faire Aufteilung der Aufgaben. Jeder hat klare Aufgaben bekommen, sodass alle aktiv zum Projekt beitragen konnten. Zudem haben wir uns darauf geeinigt, regelmässig zu treffen, um uns zu besprechen und zu unterstützen.

Snackautomat .docx

E:

Entscheiden

In der Entscheidungsphase haben wir verschiedene Möglichkeiten für die Umsetzung unseres Snackautomaten gehabt. Dabei standen mehrere Optionen zur Auswahl, beispielsweise:

- Die Bezahlung mit einer eigenen Datenbank zu speichern, damit sie realistischer funktioniert.
- Statt das Programm nur in der Konsole auszugeben, eine Oberfläche zu erstellen, die wie ein Getränkeautomat aussieht.

Schlussendlich haben wir und dafür entschieden, erstmal die Mindestanforderungen des Projekts umzusetzen und falls wir noch Zeit haben die Zusatzaufgaben zu implementieren.

R:

Realisieren

Beim Realisieren des Snackautomaten haben wir zuerst das ganze mit GitHub verbunden, sodass wir beide darauf zugreifen und gemeinsam arbeiten konnten. Es gab zwar ein paar Probleme mit Ramadans GitHub, da irgendwie zwei Konten miteinander verbunden waren und er keine Änderungen pushen konnte, aber das konnten wir mit Julias Hilfe beheben, indem er das Ganze über die Konsole gepusht und gepullt hat.

Im Projekt haben wir zuerst unsere Grundstruktur erstellt, nämlich die bereits vorhandene Main-Datei, Snack, in der wir unsere Snacks initialisieren, Payment, in der das Bezahlen der Produkte geregelt wird, und das wichtigste Element: SnackAutomatGUI, in dem das Ganze mit Knöpfen und Bildern dargestellt wird.

Zuerst haben wir die Basics des Auftrags umgesetzt, sodass die Interaktion mit Scannern in der Konsole funktionierte. Doch schnell merkten wir, dass wir noch genügend Zeit hatten und uns an etwas Schwierigeres wagen konnten nämlich

das Arbeiten mit GUI. Da wir wenig Erfahrung damit hatten, mussten wir viel im Internet recherchieren, aber am Ende hat es funktioniert.

Als Erstes haben wir ein JFrame erstellt, um das Hauptfenster darzustellen. Danach haben wir Knöpfe und ein Eingabefeld hinzugefügt, um das Verhalten einer echten Snackmaschine zu simulieren. Zudem haben wir einen zusätzlichen Knopf (S) eingebaut, der die Admin-Funktion beinhalten sollte.

Da wir am Anfang alles mit Scannern umgesetzt hatten, mussten wir sie durch GUI-Elemente ersetzen, sodass eine Verbindung zwischen der Eingabe und der Verarbeitung der Daten bestand.

Beim Bezahlen der Produkte gibt es die Möglichkeit, zwischen Bargeld und Kreditkarte zu wählen.

- Bei Bargeld kann der Benutzer entscheiden, wie viel er einzahlt, und erhält dementsprechend Rückgeld.
- Bei der Kreditkarte ist es wie im echten Leben: Die Karte wird gescannt, der Benutzer gibt einen vierstelligen Code ein, und die Zahlung wird abgeschlossen.

Zu Beginn hat man 100 Fr. Guthaben, das sich bei jeder Transaktion automatisch aktualisiert. Eigentlich war der Plan, das Ganze mit einer Datenbank zu verbinden, aber wir merkten schnell, dass die gegebene Zeit dafür nicht ausreichen würde, also haben wir die Idee verworfen.

Beim Admin-Knopf haben wir es laut Auftrag so umgesetzt, dass nach einem Code gefragt wird. Falls dieser korrekt eingegeben wird, erhält der Benutzer die Berechtigung, die Produkte wieder aufzufüllen.

Das gesamte Programm wird mit der Zahl Null beendet.

K:

Kontrollieren

Nachdem wir fertig waren mit dem Programmieren, haben wir das gesamte Programm nach Fehlern durchsucht.

Als Erstes haben wir überprüft, ob die Knöpfe richtig funktionieren. Dabei gab es keine allzu grossen Fehler, nur dass einmal der Bestätigungs-Button

und der Admin-Button geschrumpft wurden. Wir mussten sie wieder auf die normale Grösse setzen, was wir mit getScaledInstance() gelöst haben. Ein weiteres Problem war, dass die Druckfläche des Knopfes zu gross war, also grösser als der eigentliche Knopf. Dadurch waren die Abstände zwischen den Knöpfen zu gross. Dies liess sich beheben, indem wir im Code die Grösse der Druckfläche explizit definiert haben.

Als Nächstes haben wir die Eingaben überprüft, um sicherzustellen, dass alle wirklich erkannt und verarbeitet werden. Hier gab es keine Fehler.

Zuletzt haben wir noch geprüft, ob die Passwörter richtig funktionieren – also dass der Zutritt verweigert wird, wenn das falsche Passwort eingegeben wird, und gewährt wird, wenn das richtige eingegeben wird.

Bei der Kreditkarte ist es ein wenig speziell, da der Benutzer beliebige vier Zahlen eingeben muss, um den Kauf fortzusetzen. Wir haben uns bewusst dagegen entschieden, ein festes Passwort zu setzen, da der Benutzer es ohnehin nicht kennen würde. Die einzige Bedingung ist, dass es vier Zahlen sein müssen.

A:

Auswerten

Im Allgemeinen sind wir zufrieden mit dem Resultat unserer Snackmaschine. Es erfüllt alle Anforderungen: Snack Auswahl, Bezahlung Guthaben und Admin Funktion. Die Benutzerfläche ist verständlich gestaltet und das Rückgeld System funktioniert. Auch wird das Guthaben korrekt aktualisiert. Die Herausforderungen waren klar und deutlich im GUI nämlich als erstes die Anpassung der Button Grössen und Bildskalierung auch das weitergeben der GUI Referenz an Payment war anfangs problematisch aber wir konnten es mithilfe des Internets ein Lösung finden. Was wir noch optimieren könnten wäre das wir das Layout ein bisschen verschönern könnten da es jetzt alles sehr kompakt aussieht da wir keine Zeitfanden das noch zu optimieren.