ZhanLxMaherD

Software-Projekt: Getränkeautomat

Inhaltsverzeichnis

[Einleitung 1](#_Toc170503339)

[Systemvoraussetzungen 2](#_Toc170503340)

[Softwarevoraussetzungen 2](#_Toc170503341)

[Hardwarevoraussetzungen 2](#_Toc170503342)

[Aufbau 2](#_Toc170503343)

[Klasse Automat 2](#_Toc170503344)

[Hashmap? 2](#_Toc170503345)

[Wichtige Methoden 3](#_Toc170503346)

[Klasse GUI 4](#_Toc170503347)

[Wichtige Komponente 4](#_Toc170503348)

[ActionListener 4](#_Toc170503349)

[Klasse Produkt 5](#_Toc170503350)

[Wichtige Komponente 5](#_Toc170503351)

[Ziele und Reflexion 6](#_Toc170503352)

[Äußere Ziele 6](#_Toc170503353)

[Innere Ziele 6](#_Toc170503354)

[Selbstreflexion 6](#_Toc170503355)

[Hinweis 6](#_Toc170503356)

[Probleme 7](#_Toc170503357)

# Einleitung

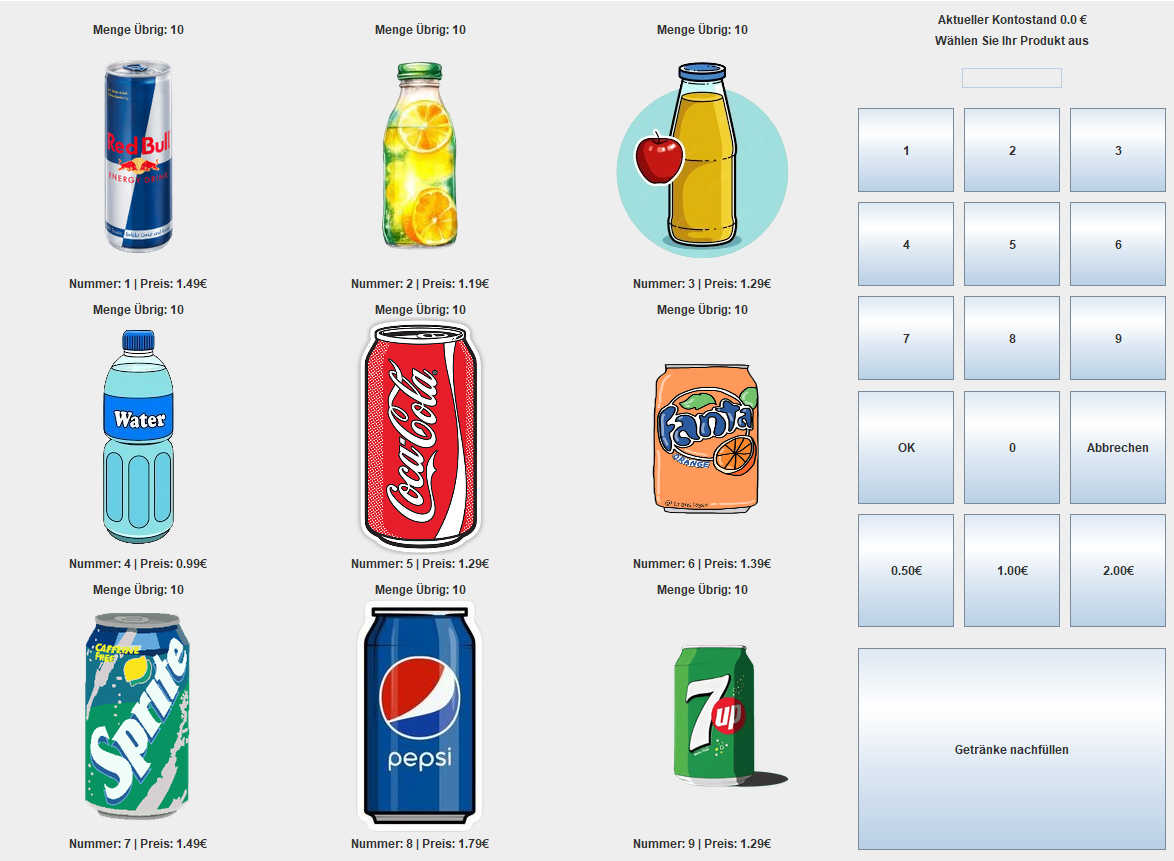
Unsere Dokumentation beschreibt detailliert die Entwicklung, Funktion und die Implementierung eines gängigen Getränkeautomaten (Simulation). Das Projekt, welches mit der Programmiersprache Java realisiert wurde, bietet eine digitale Nachbildung eines modernen Getränkeautomaten.

Unsere Simulation besteht aus einer intuitiven und benutzerfreundlichen grafischen Oberfläche, die es Anwendern ermöglicht, auf eine einfache Art mit dem virtuellen Getränkeautomaten zu interagieren. Es wurden dabei die wesentlichen Funktionen und Prozesse eines echten Getränkeautomaten sorgfältig nachgebildet, um einen authentisches Nutzererlebnis zu gewährleisten.

Im Laufe unserer Dokumentation werden wir die verschiedenen Aspekte des Projekts beleuchten, wie zum Beispiel die grundlegende Konzeption, technische Umsetzung, die implementierten Funktionalitäten aber auch die Herausforderungen, die wir während unserer Umsetzung festgestellt haben und diese Herausforderungen gemeistert haben.

Wir haben besonders auf die Softwarearchitektur, die verwendeten Designmuster und die Integration der Benutzeroberfläche unseren Fokus gelegt.

Zusätzlich werden wir die Logik hinter den verwendeten Kernfunktionen wie der Getränkeauswahl, dem Bezahlvorgang und dem Ausgabemechanismus erläutern.



Systemvoraussetzungen

Für die Ausführung des Programms sind bestimmte Systemvoraussetzungen erforderlich, um eine reibungslose und effiziente Nutzung des Programms zu gewährleisten. Die Voraussetzungen bestehen sowohl aus einem Software- als auch einem Hardware- Teil.

## Softwarevoraussetzungen

Zunächst benötigen wir Java Runtime Environment(JRE) der Version 8 oder höher. Die JRE Version 8 oder höher stellt sicher, dass die verwendeten Java-Bibliotheken und Funktionen unterstützt werden. Die JRE kann von der Oracle-Webseite heruntergeladen werden.

## Hardwarevoraussetzungen

Der Speicherbedarf des Programms ist gering, da es eine einfache Simulation ist. 256 MB RAM sind ausreichend, um das Programm ausführen zu können und eine gute Benutzererfahrung zu gewährleisten.

Man braucht auch sehr wenig Speicherplatz (Nur 50 MB). Man braucht den Speicher für die Programmdateien, die Konfigurationsdateien und eventuell benötigte temporäre Dateien.

Der Gebrauch eines Bildschirms mit mindestens 1024x768 Pixeln Auflösung ist außerdem auch sehr wichtig. Diese Auflösung stellt sicher, dass alle Bedienelemente und Anzeigen korrekt dargestellt werden und der Benutzer eine angenehme Interaktion mit dem Programm hat.

# Aufbau

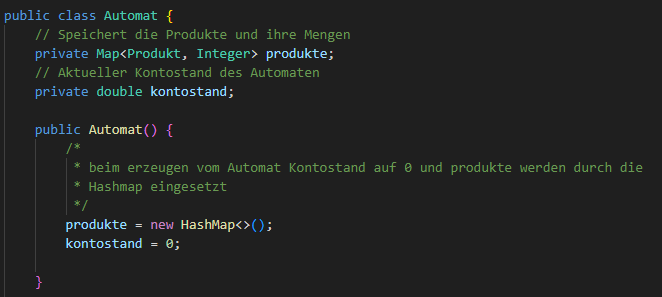
Das Programm besteht aus drei Klassen. Die Klasse Automat, GUI und Produkt. Die Klassen arbeiten zusammen, um einen Getränkeautomat zu simulieren und eine grafische Benutzeroberfläche (GUI) bereitzustellen.

## Klasse Automat

Diese Klasse verwaltet den Zustand des Getränkeautomaten. Diese Klasse speichert die Produkte, die zur Verfügung stehen und deren Mengen in einer HashMap. Außerdem verwaltet diese Klasse den Kontostand des Automaten.

### Hashmap?

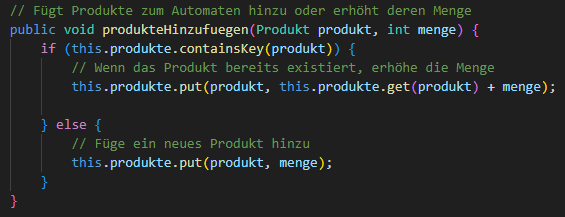
Hashmap ist eine Datenstruktur. Sie ermöglicht die Speicherung von Schlüssel-Wert-Paaren und bietet einen schnellen Zugriff auf die Werte basierend auf den Schlüsseln. In der Klasse “Automat” wird eine Hashmap verwendet, um die Produkte und die dazugehörigen Mengen zu speichern.



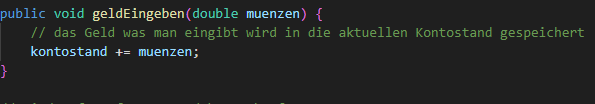
### Wichtige Methoden

Der Konstruktor Initiallisiert den Automaten mit einem leeren Produktbestand und einem Kontostand von 0 Euro.

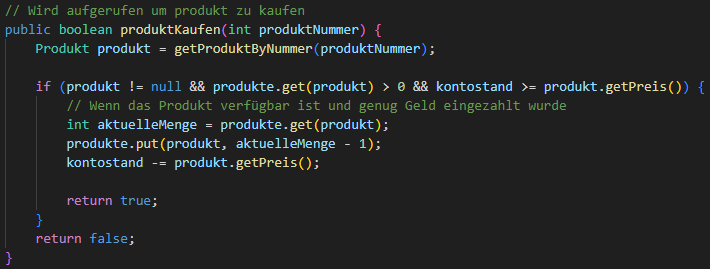
Die Methode produkteHinzufuegen(Produkt product, int menge) fügt ein Produkt zum Automaten hinzu. Die Methode kann aber auch stattdessen die Menge eines bereits vorhandenen Produkts erhöhen.



Die Methode geldEingeben(double muenzen) erhöht den Kontostand.



Die Methode produktKaufen(int produktNummer) “versucht” ein Produkt zu kaufen, falls dieses zur Verfügung steht und der Benutzer genügend Geld hat. Falls der Benutzer ein Produkt kauft, wird die Menge des ausgewählten Produkts reduziert.

Die Methode rueckgabe() gibt den aktuellen Kontostand zurück und setzt ihn auf 0. Zum Finden eines Produkts anhand der Produktnummer brauchen wir die Hilfsmethode getProduktByNummer(int produktNummer) ist eine Hilfsmethode, die zum Finden eines Produkts anhand der Produktnummer dient.

Schließlich ermöglicht uns die Methode getProduktListe() den Zugriff auf die Produkte und deren Mengen. Während die Methode getKontostand() den aktuellen Kontostand zurückgibt.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

## Klasse GUI

Diese Klasse erstellt die grafische Benutzeroberfläche. Sie besteht aus mehreren Panels und vielen Steuerelementen, die dem Benutzer die Möglichkeit geben, Produkte auszuwählen, das Geld einzuwerfen und die entsprechenden Produkte zu erwerben.

### Wichtige Komponente

Das “mainPanel” ist das Hauptpanel, das alle weiteren Panels enthält. Das “GetränkePanel” enthält drei Steuerelemente: das Numpad, die Geldwurfbuttons und den Nachfüll-Button. Das “textLabel” zeigt die Statusmeldungen und Anweisungen an und das “produktNummerFeld” ist ein Textfeld zur Eingabe der. Zum Schluss gibt es das “kontoStandLabel”. Es zeigt den aktuellen Kontostand an. (Siehe Code, da die Screenshots sonst zu viel Platz verbrauchen).

### ActionListener

Die Methode “kontostandAktualisieren()” aktualisiert die Anzeige des Kontostands, während die Methode “aktuallisiereGetränkePanel()” die Getränke-Anzeige aktualisiert.

Die Methode “aktionPerformed(ActionEvent e)” behandelt die Aktionen der Buttons und die Methode “nachfuelModusAktivieren()” aktiviert den Nachfüllmodus, wenn man das Passwort des Betreibers eingegeben hat und somit als der Betreiber agiert. Im Gegensatz dazu sorgt die Methode “nachfuelModusDeaktivieren()” für die Deaktivierung des Nachfüllmodus, wie man schon am Namen erkennen kann.

(Hier ist ein Beispiel wie der Action Listener funktioniert)

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

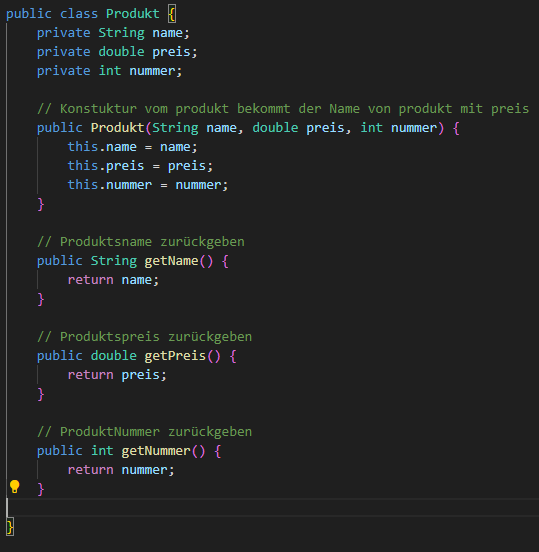
## Klasse Produkt

Diese Klasse repräsentiert jedes einzelne Produkt im Getränkeautomaten. Die Klasse enthält die grundlegenden Methoden, die für die Verwaltung und den Zugriff auf die Produktinformationen erforderlich sind.

### Wichtige Komponente

Es gibt Attribute, die die einzelnen Informationen des Produkts speichern. Es gibt auch einen Konstruktor, der drei Parameter (name, preis, nummer) entgegennimmt.

Ein wichtiger Teil sind die Getter-Methoden, die entweder den Namen, Preis oder die Nummer zurückgeben. Die Getter-Methoden sind wichtig, da die Attribute “private” sind und der Zugriff nur über diese öffentlichen Methoden stattfinden kann.



# Ziele und Reflexion

## Äußere Ziele

Das äußere Ziel unseres Projekts war es, unseren Horizont in der Sprache Java zu erweitern und praktische Erfahrungen in der Entwicklung einer grafischen Benutzeroberfläche (GUI) zu sammeln.

Bei uns standen das Verständnis für die komplexe Struktur und der Aufbau der Anwendung im Vordergrund. Die Umsetzung diente uns dazu, die theoretischen Konzepte in die Praxis umzusetzen und dabei dementsprechend die Problemlösungskompetenzen weiter zu entwickeln.

## Innere Ziele

Das Hauptziel des Projekts war es, die Entwicklung eines voll funktionsfähigen virtuellen Getränkeautomatens mit einer grafischen Oberfläche, die benutzerfreundlich gestaltet ist. Der Getränkeautomat sollte verschiedene Getränke anbieten, Geldeinwurf simulieren, Wechselgeld ausgeben und Produkte verkaufen können.

Noch ein Ziel war es die Implementierung eines Nachfüllmodus für den Betreiber, um den Automaten richtig zu verwalten. Unsere Anwendung sollte robust sein, Fehleingaben abfangen und eine intuitive Bedienung ermöglichen.

## Selbstreflexion

Bei der Umsetzung des Projekts wurden wir mit verschiedenen Herausforderungen konfrontiert, wie der Implementierung der Produktverwaltung mithilfe einer HashMap, der Gestaltung einer ansprechenden und funktionalen GUI und der Behandlung verschiedener Benutzerinteraktionen.

Der Nachfüllmodus erforderte zusätzliche Überlegungen zur Sicherheit und Benutzerfreundlichkeit.

Durch die Arbeit an diesem Projekt wurde nicht nur unsere Programmierfähigkeit gefördert, sondern auch das Verständnis für Softwaredesign, und natürlich wurde unser Wissen über die Benutzerinteraktion vertieft. Die erfolgreiche Umsetzung des Projekts demonstriert die Fähigkeit, ein komplexes System zu planen, umzusetzen und zu dokumentieren.

## Hinweis

Das Getränkeautomaten-Projekt war eine sehr tolle Teamarbeit. Wir haben von Anfang bis zum Ende alles zusammen gemacht. Jeden noch so kleinen Teil des Programms - die Produkte, wie der Automat funktioniert und wie er aussieht - haben wir gemeinsam erstellt.

Wir haben oft miteinander über unsere Ideen geredet. So konnten wir die besten Lösungen finden, besonders bei schwierigen Aufgaben wie dem Nachfüllen der Getränke oder dem Gestalten der Benutzeroberfläche.

Nach dem Programmieren haben wir immer den Code des anderen angeschaut und verbessert. Dies hat uns geholfen, Fehler zu finden und voneinander zu lernen. Somit wurde unser Programmierstil besser.

Durch das alles haben wir ein Projekt geschaffen, das gut aufgebaut ist. Jeder von uns versteht, wie alles funktioniert - von den Produkten bis zu dem, was der Benutzer sieht und macht.

Auch die Dokumentation unseres Projekts haben wir gemeinsam geschrieben. Wir haben uns immer besprochen, was wichtig zu erklären ist, und haben uns gegenseitig beim Schreiben unterstützt. Hiermit konnten wir sicherstellen, dass unsere Dokumentation vollständig und verständlich ist.

Diese Art zu arbeiten hat nicht nur Spaß gemacht, sondern auch zu einem guten Ergebnis geführt.

## Probleme

Bei unserer Entwicklung stießen wir auf verschiedene Herausforderungen, die wir schrittweise meistern mussten. Zunächst war die Verwendung einer HashMap zur Speicherung der Produkte und deren Mengen im Automaten eine neue Erfahrung. Wir mussten uns anfangs mit diesem neuen Konzept vertraut machen und lernen, wie man es effektiv einsetzt, um den Produktbestand zu verwalten. Die Aufgabe erforderte viel Zeit und Übung, bis wir damit umgehen konnten. Ein weiterer komplizierter Aspekt war das Hinzufügen von Bildern für die jeweiligen Produkte in die Benutzeroberfläche. Wir mussten erst herausfinden, wie man Bilder in eine Java-Swing-Anwendung einbinden und den richtigen Produkten zuordnen kann. Dies war nicht so einfach, wie wir dachten. Es erforderte viel Recherche.

Nachdem wir die Bilder erfolgreich hinzugefügt hatten, bemerkten wir, dass die Größen der Bilder unterschiedlich waren. Es sah dementsprechend uneinheitlich in der Benutzeroberfläche aus. Wir mussten die Bildgrößen anpassen. Da wir anfangs nicht wussten, wie man das in Java macht, haben wir verschiedene Webseiten durchsucht. Wir fanden schließlich die Webseiten "https://imgupscaler.com" und "https://picresize.com". Diese Webseiten halfen uns, die Darstellung in der GUI zu verbessern. Eine Sache, die uns viele Nerven gekostet hat, war der Nachfüll-Modus für den Automaten. Dieser Modus sollte es, wie oben schon gesagt, ermöglichen, die Mengen vorhandener Produkte zu erhöhen. Um für die Sicherheit zu sorgen, mussten wir einen Passwortschutz einbauen. Zusätzlich war es wichtig, die Eingaben der Benutzer für Produktnummern und die Mengen sorgfältig zu überprüfen, um Fehler zu vermeiden.

Wir wollten die Benutzeroberfläche flüssig und reaktionsschnell gestalten. Dafür mussten wir uns mit Multithreading auseinandersetzen. Es war besonders wichtig für Aktionen wie den Kauf von Produkten oder die Rückgabe von Geld, bei denen wir Verzögerungen einbauen und die Anzeige aktualisieren mussten, ohne dass die gesamte Anwendung "einfriert". Der Einbau von Threads war nicht so einfach zu verstehen, aber mit der Zeit lernten wir, wie man sie effektiv einsetzt.

Ein weiterer Aspekt war die Fehlerbehandlung und die entsprechende Überprüfung von Benutzereingaben. Wir mussten sicherstellen, dass der Automat korrekt auf ungültige Eingaben reagiert, egal ob bei der Auswahl von Produkten, der Eingabe von Geldbeträgen oder beim Nachfüllen von Produkten. Wir lernten auch neue Konzepte kennen und wendeten sie praktisch an. Obwohl es manchmal frustrierend war, wenn etwas nicht sofort funktionierte, war es auch sehr lehrreich. Mit jedem gelösten Problem wuchs unser Verständnis und Selbstvertrauen im Umgang mit komplexen Aufgaben. Am Ende all dieser Bemühungen stand ein funktionierender Getränkeautomat, auf den wir (Maher und Zhan) sehr stolz sein können. Die Erfahrung, die wir bei diesem Projekt gesammelt haben, wird uns bestimmt in der Zukunft weiterhelfen.