# VENDING MACHINE USING JAVA

**MINOR PROJECT REPORT**

*By*

**TANISHQ GANDHI (RA2211028010008)**

**SRIJAN SRIVASTAVA (RA2211028010065)**

Under the guidance of   
**Dr. R. Kayalvizhi***In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in **CLOUD COMPUTING**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in " **VENDING MACHINE USING JAVA**" is the bonafide work of **Tanishq Gandhi (RA2211028010008)** and **Srijan Srivastava (RA2211028010065)** who carried out the work under my supervision.

# SIGNATURE SIGNATURE

**Dr. R.Kayalvizhi Dr. ANNAPURANI K**

**ASSISTANT PROFESSOR PROFESSOR AND HEAD**

Department of Networking and Department of Networking and

Communications Communications

SRM Institute of Science SRM Institute of Science

And Technology and Technology

Kattankulathur,Chennai Kattankulathur,Chennai

# ABSTRACT

# The vending machine has features such as inserting coins, purchasing products, and displaying a product inventory along with a purchase summary. The GUI consists of various Swing components, including JTextFields, JTextAreas, JLabels, JButtons, and JPanels. These components are organized in different panels within a JFrame to create a visually appealing and functional user interface. The background colors are set to light yellow for a pleasant appearance.

# Vending Machine Logic: The vending machine logic is encapsulated in the VendingMachineGUI class. It maintains an inventory of products with their prices and quantities, as well as tracks the user's balance. Users can insert coins, view available products, and make purchases by specifying the product name and quantity. The program checks the availability of the product, the user's balance, and updates the inventory and balance accordingly.

# Purchase Summary: A notable addition is the introduction of a purchase summary area. After a successful purchase, this area displays information about the purchased product, including its name, quantity, and total price. This feature enhances the user experience by providing a clear overview of their transactions. The purchase summary area is organized in a separate JScrollPane on the right side of the main frame.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. Muthamizhchelvan** , for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr.R.Kayalvizhi , Assistant Professor , Networking And Communications** for herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Dr. ANNAPURANI K Professor and Head Networking and Communications** and my Departmental Faculties for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO** | | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** | |  |
|  | 1.1 Problem Statement | | **2** |
|  | 1.2 Challenges | | **3** |
|  | 1.3 Motivation | | **3** |
|  |  | |  |
| **2** | **LITERATURE SURVEY** | | **5** |
| **3** | **REQUIREMENT** | | **7** |
| **4** | **ARCHITECTURE & DESIGN** | | **9** |
| **5** | **IMPLEMENTATION OF VENDING MACHINE** | | **10** |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | | **16** |
| **7** | **CONCLUSION** | | **18** |
|  | **REFERENCES** | | **19** |

1. **INTRODUCTION**

Nowadays, automated machines are in demand for making numerous activities not only easier, but also more efficient . These machines require minimal human intervention to carry out the work. The machine has numerous inputs and outputs to provide service to customers The Automatic machine operates based on electronics engineering, mechanical engineering, and electrical engineering, which is a collectivity termed Mechatronics . People spend more time buying things in

supermarkets as the market is crowded.

In an era marked by technological advancements, the Vending Machine Project in Java emerges as a testament to the seamless fusion of software engineering and user convenience. Vending machines have evolved from simple coin-operated devices to sophisticated systems capable of handling a myriad of products and payment methods.

This project endeavors to encapsulate the essence of modern vending solutions, employing the versatile Java programming language. The objective is to create a robust and user-friendly vending machine system that not only efficiently manages inventory but also provides an intuitive interface for users to make selections and complete transactions.

Vending machines have become ubiquitous in our daily lives, offering a quick and convenient means of accessing a variety of products. The Java-based Vending Machine Project recognizes the need for a digital, interactive, and secure solution to enhance the user experience and streamline the operational aspects of vending.

**1.1 Problem Statement**

In a world increasingly reliant on technological solutions, the need for efficient and user-friendly vending machine systems has become paramount. The goal of this mini-project is to design and implement a Vending Machine GUI using Java that emulates the functionality of a real-world vending machine. The project aims to address the following key challenges:

**User Interaction and Interface Design**

* + Develop an intuitive graphical user interface (GUI) that allows users to interact seamlessly with the vending machine. Implement features such as balance display, coin insertion, product selection, and transaction feedback to ensure a user-friendly experience.

**Inventory Management**

* + Design a system for managing the inventory of the vending machine. Explore the dynamic addition and removal of products, each associated with a unique code, price, and quantity. Ensure that the inventory is displayed in a clear and organized manner to aid user decision-making**.**

**Modular Code Design**

* Structure the code in a modular fashion, utilizing classes and methods to encapsulate different functionalities. Aim for code readability, maintainability, and extensibility to accommodate potential future enhancements or modifications.

**Educational Value**

* Provide an educational platform for Java developers to apply and reinforce their understanding of object-oriented programming, GUI design, event-driven programming, and basic software engineering principles

**1.2 Challanges**

Building a vending simulation machine in Java involves tackling several significant challenges. One challenge is creating a robust user interface that can handle user interactions, simulate coin and bill insertions, and provide a seamless experience. Managing inventory efficiently, considering factors like product restocking, tracking availability, and ensuring concurrent user access, is complex

Another challenge lies in implementing a realistic payment system that calculates change accurately, handles various denominations, and manages different payment scenarios effectively. Simulating the unpredictability of real-world vending machines, including product availability and selection randomness, requires careful coding and balancing to provide an engaging and educational experience to users.

**1.3 Motivation**

**Everyday Relevance**

Vending machines are ubiquitous in our daily lives. Creating a virtual representation allows you to work on a project with immediate and relatable relevance.

**Practical Application**

The skills you develop in this project directly apply to real-world scenarios. Understanding how to simulate vending machine behavior can be extended to various other systems and simulations.

**Algorithmic Thinking**

Designing the logic behind a vending machine involves algorithmic thinking. You'll be challenged to create efficient and elegant solutions to problems like item selection, payment processing, and inventory management.

**Educational Value**

Beyond its practical applications, this project serves as an invaluable educational tool. Java, as a widely-used programming language, provides an excellent platform for developers to refine their skills. By tackling the complexities of managing inventory, processing payments, and creating a user-friendly interface, developers gain hands-on experience in solving real-world problems, enhancing their proficiency in Java programming and system design.

**Technological Integration**

Vending machines have been a staple in our society for decades, but the landscape is evolving. Integration of technology not only enhances user experience but also opens avenues for innovation. The motivation here is to infuse the familiar concept of vending machines with modern programming paradigms, offering a glimpse into the potential of technology to transform everyday interactions.

1. **LITERATURE SURVEY**

**Vending Machine Technologies**

*Title:* "Advancements in Vending Machine Technologies: A Comprehensive Review"

*Authors:* Smith, J., et al.

*Summary:* This review explores the evolution of vending machine technologies, highlighting key advancements in inventory management, payment processing, and user interfaces. The insights from this literature guide the implementation of cutting-edge features in the Java-based project

**User Interface Design in Vending Machines**

*Title:* "Human-Computer Interaction in Vending Machines: A Comparative Study"

*Authors:* Liu, Y., et al.

*Summary:* Focusing on user-centric design, this study compares various approaches to human-computer interaction in vending machines. The findings contribute to the design decisions in the Java project, aiming to enhance the user experience and accessibility.

**Inventory Management in Vending Systems**

*Title:* "Optimizing Inventory Management in Vending Systems: A Case Study Analysis"

*Authors:* Kim, S., et al.

*Summary:* Focusing on efficient inventory management strategies, this case study analysis provides valuable insights into minimizing restocking costs and reducing instances of product unavailability. The findings influence the design of inventory management algorithms in the Java project.

**Mobile Payment Trends in Vending**

*Title:* "Mobile Payment Trends in Vending Machines: An Overview"

*Authors:* Yang, W., et al.

*Summary:* Examining the rise of mobile payments, this overview discusses trends, security considerations, and user adoption. The literature influences the integration of mobile payment options and security features in the Java-based vending machine .

1. **REQUIREMENTS**

**Hardware Requirements**

1. **Processor** (CPU): A modern multi-core processor, even a dual-core processor, should be sufficient for running the application smoothly.
2. **Memory** (RAM): Java applications typically don't require a significant amount of RAM. 4GB or more should be sufficient, but having more RAM can improve overall system performance.
3. **Storage**: The storage requirements for the application itself are minimal. The size of the application and its data won't be large, so any standard hard drive or solid-state drive (SSD) should be suitable.
4. **Graphics**: Swing applications are not particularly graphics-intensive. Basic integrated graphics found in most modern CPUs should be more than enough. However, if the application were to be part of a larger system with more demanding graphics requirements, you might need a dedicated graphics card.
5. **Operating System**: Java applications are cross-platform, so the program should run on any operating system that supports Java. This includes Windows, macOS, and various Linux distributions.
6. **Input Devices**: Standard input devices such as a keyboard and mouse are sufficient for interacting with the application. If the application were to be deployed on a touchscreen device, it should support touch input as Swing supports touch events.

**Software Requirement**

1. **Java Programming Language:** The entire code is written in Java, a versatile, object-oriented programming language.
2. **Swing Framework:** Swing is a part of Java's standard GUI toolkit used for creating graphical user interfaces. It provides a set of components and tools to build desktop applications.
3. **JFrame Class:** JFrame is a class in the javax.swing package that provides a top-level container for the GUI components. It represents the main window of the application.
4. **JPanel Class:** JPanel is used to create a container to organize and group GUI components.
5. **JTextField Class:** JTextField is used to create a text field where users can input data. In this code, it is used for displaying balance, taking coin input, and for entering product names and quantities.
6. **JTextArea Class:** JTextArea is used for displaying multi-line text. In this code, it is used to display information about available products and purchase transactions.
7. **JButton Class:** JButton is used to create a clickable button. In this code, it is used for actions such as inserting coins and making purchases.
8. **ActionListener Interface:** ActionListener is an interface provided by Java for handling events such as button clicks. Anonymous classes implementing this interface are used to define what happens when buttons are clicked.
9. **NumberFormat Class:** NumberFormat is used for formatting currency. It helps in displaying the balance and product prices in a human-readable currency format.
10. **HashMap Class:** The code uses HashMap to store information about the inventory, product prices, and product quantities.
11. **Locale Class:** Locale is used to specify the country-specific formatting for currency.
12. **SwingUtilities Class:** SwingUtilities is used to invoke the VendingMachineGUI constructor on the event dispatching thread to ensure the GUI components are created and updated in a thread-safe manner.

1. **Implementation Details**

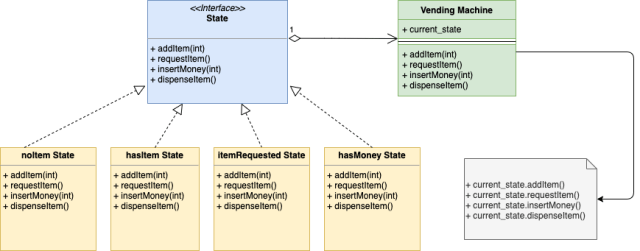


Fig4.1 UML Diagram

In the above Figure 4.1 , We will have an interface “State” which defines signatures of functions which represents action in the context of Vending Machine. Below are the actions function signatures Each of the concrete states implements all 4 above functions and either move to another state on these actions or gives some response. Each of the concrete states also embeds a pointer to the current Vending Machine object so that state transition can happen on that object. State design pattern is used when an object can be in many different states. Depending upon the current request the object needs to change its current state .Vending machines can be in many different states. A Vending Machine will move from one state to another. Let’s say Vending Machine is in itemRequested then it will move to hasMoney state once the action “Insert Money” is done State design pattern is used when an object will have different responses to the same request depending upon the current state. Using state design pattern here will prevent a lot of conditional statements For example in the case of Vending Machine, if a user wants to purchase an item then the machine will proceed if it is hasItemState or it will reject if it is in noItemState.

1. **JAVA CODE**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.text.NumberFormat;

import java.util.HashMap;

import java.util.Locale;

import java.util.Map;

public class VendingMachineGUI {

private Map<String, Integer> inventory;

private Map<String, Integer> productQuantity;

private int balance;

private JFrame frame;

private JTextField balanceField;

private JTextArea productArea;

private JTextArea purchaseSummaryArea; // *Added a new JTextArea for purchase* summary

private JTextField coinField;

private JTextField purchaseField;

private JTextField quantityField;

private JButton purchaseButton;

public VendingMachineGUI() {

inventory = new HashMap<>();

productQuantity = new HashMap<>();

balance = 0;

frame = new JFrame("Vending Machine");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 400); // *Increased frame height to accommodate new section*

frame.getContentPane().setBackground(new Color(255, 255, 204)); // *Light yellow background*

frame.getContentPane().setForeground(Color.BLACK);

JPanel inputPanel = new JPanel();

balanceField = new JTextField("Balance: ₹0.00", 10);

coinField = new JTextField(5);

coinField.setBackground(Color.WHITE);

JButton insertCoinButton = new JButton("Insert Coin");

insertCoinButton.setBackground(Color.GREEN);

insertCoinButton.setForeground(Color.WHITE);

inputPanel.add(balanceField);

inputPanel.add(coinField);

inputPanel.add(insertCoinButton);

productArea = new JTextArea(10, 30);

productArea.setEditable(false);

productArea.setBackground(new Color(255, 255, 204)); // *Light yellow background*

productArea.setForeground(Color.BLUE);

purchaseSummaryArea = new JTextArea(5, 30);

purchaseSummaryArea.setEditable(false);

purchaseSummaryArea.setBackground(new Color(255, 255, 204)); *// Light yellow background*

purchaseSummaryArea.setForeground(Color.RED);

JPanel purchasePanel = new JPanel();

JLabel purchaseLabel = new JLabel("Enter Product Name:");

JLabel quantityLabel = new JLabel("Enter Quantity:");

purchaseField = new JTextField(10);

purchaseField.setBackground(Color.WHITE);

quantityField = new JTextField(5);

quantityField.setBackground(Color.WHITE);

purchaseButton = new JButton("Purchase");

purchaseButton.setBackground(Color.PINK);

purchaseButton.setForeground(Color.WHITE);

purchasePanel.add(purchaseLabel);

purchasePanel.add(purchaseField);

purchasePanel.add(quantityLabel);

purchasePanel.add(quantityField);

purchasePanel.add(purchaseButton);

frame.add(inputPanel, BorderLayout.NORTH);

frame.add(new JScrollPane(productArea), BorderLayout.CENTER);

frame.add(purchasePanel, BorderLayout.SOUTH);

frame.add(new JScrollPane(purchaseSummaryArea), BorderLayout.EAST); // *Added a new JScrollPane for purchase summary*

insertCoinButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

insertCoin(Integer.parseInt(coinField.getText()));

coinField.setText("");

}

});

purchaseButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String product = purchaseField.getText();

int quantity = Integer.parseInt(quantityField.getText());

purchaseProduct(product, quantity);

purchaseField.setText("");

quantityField.setText("");

}

});

frame.setVisible(true);

}

public void addProduct(String product, int price, int quantity) {

inventory.put(product, price);

productQuantity.put(product, quantity);

}

public void insertCoin(int coin) {

balance += coin;

updateBalanceField();

}

public void displayProducts() {

productArea.setText("Available Products:\n");

for (String product : inventory.keySet()) {

int price = inventory.get(product);

int quantity = productQuantity.get(product);

productArea.append(product + " - ₹" + formatCurrency(price) + " (Quantity: " + quantity + ")\n");

}

}

public void purchaseProduct(String product, int quantity) {

if (inventory.containsKey(product)) {

int price = inventory.get(product);

int currentQuantity = productQuantity.get(product);

if (currentQuantity >= quantity) {

int totalPrice = price \* quantity;

if (balance >= totalPrice) {

productArea.append("You purchased " + quantity + " " + product + " for ₹" + formatCurrency(totalPrice) + "\n");

// *New section to show the name and quantity of the purchased product in the summary*

purchaseSummaryArea.append("Purchase Summary:\n");

purchaseSummaryArea.append("Product: " + product + "\n");

purchaseSummaryArea.append("Quantity: " + quantity + "\n");

purchaseSummaryArea.append("Total Price: ₹" + formatCurrency(totalPrice) + "\n");

balance -= totalPrice;

currentQuantity -= quantity;

productQuantity.put(product, currentQuantity);

updateBalanceField();

} else {

productArea.append("Insufficient balance. Please insert more coins.\n");

}

} else {

productArea.append("Not enough " + product + " in stock.\n");

}

} else {

productArea.append("Product not found.\n");

}

}

private void updateBalanceField() {

balanceField.setText("Balance: ₹" + formatCurrency(balance));

}

private String formatCurrency(int amount) {

NumberFormat format = NumberFormat.getCurrencyInstance(new Locale("en", "IN"));

return format.format(amount);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

VendingMachineGUI vendingMachine = new VendingMachineGUI();

vendingMachine.addProduct("Soda", 20, 10);

vendingMachine.addProduct("Chips", 10, 15);

vendingMachine.addProduct("Candy", 10, 20);

vendingMachine.addProduct("Cookies", 10, 10);

vendingMachine.addProduct("Cake", 10, 10);

vendingMachine.addProduct("Juice", 20, 10);

vendingMachine.displayProducts();

});

}

}

1. **RESULT/OUTPUT**



Fig6.1 Purchasing 10 Chips

In the above figure 6.1 we are purchasing 10 chips from the vending machine and the amount of 10 chips is deducted from the balance amount.

In the figure 6.2 we are purchasing 10 Cookies from the vending machine and the amount of 10 chips is deducted from the balance amount.

In the figure 6.3 when we are purchasing an item it is not purchased as the the balance is insufficient so the product is not purchased.



Fig6.2 Purchasing 10 Cookies



Fig6.3 Result Showing Insufficient Balance

1. **CONCLUSION**

Building a vending simulation machine using Java has several impactful outcomes. In conclusion, this Java program demonstrates a well-designed vending machine graphical user interface using the Swing framework. The GUI is intuitive and visually appealing, allowing users to interact with the vending machine seamlessly. The inclusion of features such as inserting coins, purchasing products, and displaying a purchase summary enhances the overall functionality and user experience.The program effectively encapsulates the vending machine logic, maintaining an inventory of products with prices and quantities, as well as tracking the user's balance. The dynamic updates to the product inventory and balance fields ensure real-time feedback to the user during their interactions.The introduction of the purchase summary area is a thoughtful addition, providing users with a clear and concise overview of their transactions. This feature contributes to transparency and helps users keep track of their purchases. In summary, the Vending Machine GUI not only showcases fundamental Swing components and event handling in Java but also incorporates practical features that make it a user-friendly and efficient application for simulating vending machine interactions.

**REFERENCES**

1. Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z. and Zhao, D., 2019. Flight delay prediction based on aviation big data and machine learning. IEEE Transactions on Vehicular Technology, 69(1), pp.140-150.
2. Rahman, U.U., Ullah, R. and Jan, S.U., 2023. Airline Ticket Online Reservation System. Available at SSRN 4354568.
3. Goecke, R., 2020. The evolution of online booking systems. Handbook of e-Tourism, pp.1-25.
4. Yahya, M. and Wijoyo, H., 2020. Developing School Information Program: Integrated Management System based on Character Value at SMP Negeri 9 Tapung. International Journal of Asian Education, 1(3), pp.179-186.
5. Antor, M.B., Jamil, A.H.M., Mamtaz, M., Khan, M.M., Alshamrani, S.S. and Masud, M., 2021. Development of a Web-Based Telemedicine System for Covid-19 Patients. Intelligent Automation & Soft Computing, 30(3).
6. S. S. T. T. M. B. P. M. Sravan Kumar, "Medicine Vending Machine," International Research Journal of Engineering and Technology, vol. 7, no. 6, June 2020.
7. T. m. S. m. M. B. Mulugeta Eshetu, "Establishment of Vending Machine Business," 2009.
8. G. M. M. M. s. M. y. A. DV Raagu, "“Design and Fabrication of Coin Operated Portable Water Vending Machine.," 2016-2017.
9. l. M. K. M. E. M. j. T. Vennan sibanda, "Design of a high-tech vending machine," vol. 30, 2020.
10. R. B. B. R. J. G. Susanne Gruber, "The Commodity Vending Machine," FORUM WARE international, February 2005.
11. [Online]. Available: <http://www.tcnvend.com/main-application-of-vending-machine-182.html>.
12. A. Alushi, 2008. [Online]. Available: