# Getting Ready: Vending Machine

Understand the vending machine problem and learn the questions to further simplify this problem.

**We'll cover the following\\**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [States of the vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States-of-the-vending-machine)
  + [Money handling](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Money-handling)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design patterns](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-patterns)

## Problem definition

A **vending machine** is an automated machine. The machine provides items, including snacks, chocolates, and beverages to consumers. There are multiple racks inside the vending machine with products on each rack. A user inserts money into the machine, selects the rack number from which they want to buy a product, and presses the button. A vending machine dispenses the product to the users based on the amount of money inserted and the selection of the product.

Modern vending machines were first developed in England in the early 1880s and dispensed postcards. Today, there are specialized vending machines that provide specific products. The vending machine is a 24x7 standalone unit that requires a standard power supply connection to function. It consists of electromechanical systems that help automate the entire vending process. Therefore, its basic function is to flawlessly issue users with a diverse range of products.

The diagram below elaborates on the process of product purchasing using the vending machine:

Step1: Insert cash according to the price of the product you want to buy

**1** of 5

## Expectations from the interviewee

Although the vending machine problem is a simpler design problem asked in interviews, the interviewer still has some expectations. The following provides an overview of what the interviewer wants to hear you discuss in more detail during the interview.

### States of the vending machine

An interviewer would also expect you to discuss the different states of the vending machine. You may ask the following set of questions:

* What function do the vending machines perform? Alternatively, how many different states can the vending machines have?
* After inserting money into the machine, what does the system do?
* Who presses the vending machine button, and what happens after pressing it?
* What does the dispense function do?
* If the vending machine is in a dispense state, is it possible to insert money?
* If you are in NoMoneyInsertedState and try to select a product without paying money, would you be able to select a product?

### Money handling

One of the most significant attributes of the vending machine system is how it receives, calculates, and returns money. You may ask the interviewer the questions listed below:

* What should the system do if we pay less money than the product price?
* What should the system do if we pay more money than the product price?
* Can the credit card be used to input money or can only cash be used?

## Design approach

We’ll design this vending machine problem using the bottom-up design approach. For this purpose, we’ll follow the steps below:

* Identify and design the smallest components first, like, a product in the machine.
* Use these small components to design bigger components, for example, the inventory.
* Repeat the steps above until we design the whole system, which is the vending machine.

## Design patterns

It is always a good practice to discuss the design patterns that the vending machine falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design pattern is used to design the vending machine:

* State design pattern
* Singleton design pattern

Let’s explore the requirements of the vending machine problem in the next lesson.

Back

**Requirements for the Vending Machine**

Let's look at the requirements of the vending machine.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)

In this lesson, we’ll list down the requirements of our vending machine problem. This is a very crucial step as requirements define the scope of a problem, so getting them right from the interviewer and understanding them well will make the design of the rest of the problem smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label "Rn", where "R" is short for Requirement and "n" is a natural number.

**Requirement collection**

The requirements for the vending machine problem are defined below:

**R1:** There are different products placed at different positions in the vending machine.

**R2:** The vending machine can be in one of these three states:

* NoMoneyInsertedState: There is no money inserted into the machine.
* MoneyInsertedState: Money is inserted into the machine.
* DispenseState: The machine gives out the product.

**R3:** There can be two actors in the system. One is the user and the other is the admin.

**R4:** The admin can add a product to the machine or remove a product from the machine.

**R5:** The system should allow the users to select a product they want to purchase from the machine by specifying the rack number.

**R6:** The user can insert money into the machine in the form of cash.

**R7:** The system should be able to calculate the money inserted into the machine.

**R8:** The system should check whether the user inserted the exact amount required for the specific product into the machine.

**R9:** If the amount is greater than the product price, the system should change back the user and dispense the product.

**R10:** If the amount is less than the product price, the system should display an error message and return the money.

We've identified our requirements for the problem, and in the next lesson, we will define the class diagram of our vending machine system.

# Use Case Diagram for the Vending Machine

Learn how to define use cases and create the corresponding use case diagram for the vending machine.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-cases)
  + [Customer](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Customer)
  + [Operator](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Operator)
  + [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
  + [Extend](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Extend)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram for the vending machine and understand the relationship between its different components.

First, we’ll define the different elements of our vending machine, followed by the complete use case diagram of the system.

## System

Our system is a "Vending machine."

## Actors[#](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)

Now, we’ll define the main actors of our vending machine.

### Primary actors

* **Customer:**This actor can view, select and take products, insert money, and take out change from the machine.
* **Operator:** This actor can do everything a "Customer" can do. It can also add or remove products and remove cash from the machine.

### Secondary actors

* **System:** This actor can search for the selected product and dispatch it after validating money.

## Use cases

In this section, we will define the use cases for the vending machine. We have listed down the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times because they are shared among different actors in the system.

### Customer

* **View products:**To view all available products in the vending machine
* **Select products:**To select a product to buy from the vending machine
* **Insert money:**To insert money to buy products from the vending machine
* **Take product:**To take out products from the vending machine
* **Take change:** To take out change from the vending machine

### Operator

* **Add product:** To add new products inside the vending machine
* **Remove product:** To remove products from the vending machine
* **Cash remove:** To remove collected cash from the vending machine

### System

* **Search product:**To search for the selected product in the machine to dispatch it
* **Validate money:**To validate that the money is legal
* **Dispense product:** To dispense selected products so customers can take them
* **Return change:**To return the change to the customer if the inserted amount is less than the purchased product price

## Relationships

This section describes the relationships between and among actors and their use cases.

### Generalization

The customer and operator are two actors who interact with the vending machine. The consumer can only engage with the system to purchase a product. An operator can execute all the duties that a customer can, along with certain administrative responsibilities. Therefore, the “Operator” actor has a generalization relationship with the “Customer” actor.

### Associations

The below table shows the association relationship between actors and their use cases.

|  |  |  |
| --- | --- | --- |
| **Customer** | **Operator** | **System** |
| View products | Add product | Search product |
| Select products | Remove product | Dispense product |
| Insert money | Cash remove | Validate money |
| Take product | View products | Return change |
| Take change | Select products |  |
|  | Insert money |
| Take product |
| Take change |

### Include

* When a customer selects a product to buy, the system then searches for the product’s location and dispatches it. Therefore, the “Select products” use case has an include relationship with the “Search product” use case.
* When a customer selects a product to buy, the system then validates the money that the customer inserted and then dispenses the product. Therefore, the “Validate money” use case has an include relationship with the “Dispense product” use case.

### Extend

* When a customer selects a product to buy, the system then validates the money that the customer inserted and then returns the change if the amount is greater than the price of the purchased product. Therefore, the “Return change” use case has an extend relationship with the “Validate money” use case.

## Use case diagram

# Class Diagram for the Vending Machine

Learn to create a class diagram for the vending machine using the bottom-up approach.

**We'll cover the following**

* [Components of a vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-a-vending-machine)
  + [State](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#State)
  + [Product](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Product)
  + [Rack](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Rack)
  + [Inventory](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inventory)
  + [Vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Vending-machine)
  + [Enumeration](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumeration)
    - [ProductType](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ProductType)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Aggregation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Aggregation)
  + [Association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Association)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram for the vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-for-the-vending-machine)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)
* [Additional requirements](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Additional-requirements)

In this lesson, we’ll identify and design the classes and interfaces based on the requirements that we have previously gathered from the interviewer for our vending machine system.

## Components of a vending machine

As mentioned earlier, we should design the vending machine using a bottom-up approach.

### State

State is an interface that represents the current state of the vending machine. There can be three possible states of a vending machine, i.e., no money inserted state, money inserted state, and the dispense state. The NoMoneyInsertedState class represents the state when there is no money inserted in the machine. When the user inserts the money into the machine, the state changes to MoneyInsertedState. Furthermore, when the machine dispenses the required product to the user, it transitions to the DispenseState.

This problem follows the State design pattern since the vending machine changes its behavior based on its state. Here, the State class defines an interface for declaring what the subclasses (NoMoneyInsertedState, MoneyInsertedState, DispenseState) should do. The subclasses provide the implementation for methods defined in the State, and the implementation of each method changes with the change of the state. Every state implements some functions, as shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Inheritance

The following classes show an inheritance relationship:

* The NoMoneyInsertedState, MoneyInsertedState, and DispenseState classes implement the Stateinterface.

**Note:**We have already discussed this inheritance relationship between classes in the component section above.

## Class diagram for the vending machine

Here is the complete class diagram for our vending machine:

A diagram of a machine

Description automatically generated

* No money inserted state
* Money inserted state
* Dispense state

All these states have the same methods but the implementation of each method in each state changes with the change of the state.

## Additional requirements

The interviewer can introduce some additional requirements in the vending machine, or they can ask some follow-up questions. Let's see some examples of additional requirements:

**Refund/Cancel**: The vending machine should have the option to cancel the operation. In that case, the customers will get a refund. The class diagram provided below shows the refund functionality in all states:

A screenshot of a diagram

Description automatically generated

* customer uses the refund option, then the refundFullMoney() function will call and return the full amount back in the change tray.
* DispenseState: In this state, the refundFullMoney() function is blocked. We can say that if the customer uses the refund option, then the system does not do anything. This is because, in this state, the customer has already selected a product, and the vending machine is in the dispense state.

We have completed the class diagram of a vending machine according to the requirements. Now let's go ahead and design the activity diagram of the vending machine in the next lesson.

# Activity Diagram for the Vending Machine

Create an activity diagram for the vending machine.

**We'll cover the following**

* [Product purchase from the vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Product-purchase-from-the-vending-machine)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)

Activity diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our vending machine. In this lesson, we will create an activity diagram for the following activity.

## Product purchase from the vending machine

The following are the states and actions involved in this activity diagram.

### States

**Initial state:**The customer inserts money in the vending machine.

**Final state:**The customer receives the desired product.

### Actions

The customer interacts with the vending machine and inserts money. The customer selects their desired product. If that product is available, they get the product and the change. Otherwise, the system shows a message of product unavailability and asks the customer to select some other product.

Based on the order above, the activity diagram of a product purchase from the vending machine is shown below:

A screenshot of a diagram

Description automatically generatedA screenshot of a diagram

Description automatically generated

# Code for the Vending Machine

Write the object-oriented code to implement the design of the vending machine problem.

**We'll cover the following**

* [Vending machine classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Vending-machine-classes)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [State](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#State)
  + [Product, rack and inventory](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Product,-rack-and-inventory)
  + [Vending machine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Vending-machine)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

We’ve reviewed different aspects of the vending machine problem and observed the attributes attached to the problem using various UML diagrams. Let us now explore the more practical side of things, where we will work on implementing the vending machine using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the vending machine:

* Java
* C#
* Python
* C++
* JavaScript

## Vending machine classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Enumerations

The following code provides the definition of the enumeration used in the vending machine system:

**Note:**JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

Java

// Enumerations

enum ProductType {

CHOCOLATE,

SNACK,

BEVERAGE,

OTHER

}

### State

State is an interface and the NoMoneyInsertedState, MoneyInsertedState, and DispenseState classes implement the State interface. The definition of these classes is provided below:

Java

C#

Python

// State is an interface

public interface State {

// Interface method (does not have a body)

public void insertMoney(VendingMachine machine, double amount);

public void pressButton(VendingMachine machine, int rackNumber);

public void returnChange(double amount);

public void updateInventory(VendingMachine machine, int rackNumber);

public void dispenseProduct(VendingMachine machine, int rackNumber);

}

public class NoMoneyInsertedState implements State {

@override

public void insertMoney(VendingMachine machine, double amount) {

// changes state to MonenInsertedState

}

public void pressButton(VendingMachine machine, int rackNumber) {}

public void returnChange(double amount) {}

public void updateInventory(VendingMachine machine, int rackNumber) {}

public void dispenseProduct(VendingMachine machine, int rackNumber) {}

}

public class MoneyInsertedState implements State {

@override

public void insertMoney(VendingMachine machine, double amount) {}

public void pressButton(VendingMachine machine, int rackNumber) {

// check if product item is available

// validate money

// change state to DispenseState

}

public void returnChange(double amount) {}

public void updateInventory(VendingMachine machine, int rackNumber) {}

public void dispenseProduct(VendingMachine machine, int rackNumber) {}

}

public class DispenseState implements State {

@override

public void insertMoney(VendingMachine machine, double amount) {}

public void pressButton(VendingMachine machine, int rackNumber) {}

public void returnChange(double amount){}

public void updateInventory(VendingMachine machine, int rackNumber) {}

public void dispenseProduct(VendingMachine machine, int rackNumber) {

// dispense product

// change state to NoMoneyInsertedState

}

}

### Product, rack and inventory

We will explore the Product, Rack, and Inventory classes that provide the details of the items available as well as their positions inside the vending machine. The definition of these classes is provided below:

public class Product {

private String name;

private int id;

private double price;

private ProductType type;

}

public class Rack {

private int productId;

private int rackNumber;

public boolean isEmpty();

}

public class Inventory {

private int noOfProducts;

private List<Product> products;

public void addProduct(int productId, int rackId);

public void removeProduct(int productId, int rackId);

}

### Vending machine

The VendingMachine class is the final class of the system, and it will also be a Singleton class so that there will only be a single instance of this class in the whole system. The definition of this class is given below:

public class VendingMachine {

private State currentState;

private double amount;

private int noOfRacks;

private List<Rack> racks;

private List<int> availableRacks;

// The VendingMachine is a Singleton class that ensures it will have only one active instance at a time

private static VendingMachine vendingMachine = null;

// Created a private constructor to add a restriction (due to Singleton)

private VendingMachine();

// Created a static method to access the singleton instance of VendingMachine

public static VendingMachine getInstance() {

if (vendingMachine == null) {

vendingMachine = new VendingMachine();

}

return vendingMachine;

}

public void insertMoney(double amount) {}

public void pressButton(int rackNumber) {}

public void returnChange(double amount) {}

public void updateInventory(int rackNumber) {}

public void dispenseProduct(int rackNumber) {}

public int getProductIdAtRack(int rackNumber) {}

}

## Wrapping up

We've explored the complete design of a vending machine in this chapter. We've looked at how a basic vending machine can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.