Vending Machine v1.0

Technical Design and Implementation

Han Truong (han.truong@cmegroup.com) 408-806-2938

Source code: https://github.com/hantruongth/VendingMachine
Profile: https://www.hantruong.us

Contents

Problem Statement

A vending machine sells items for various prices and can give changes. At the start of the day, it is loaded with a certain number of coins of various denominations e.g. 100 x 1c, 50 x 5c, 50 x 10c, 50 x 25c etc. When an item is requested, a certain number of coins are provided. Write code that models the vending machine and calculates the change to be given when an item is purchased (e.g. 2 x25c used to purchase an item costing 35c might return 1 x 10c and 1 x 5c).

Provide a solution that includes an interactive interface (a console interface, for e.g.) that allows a user to select items, pay for them in coins and get any change.

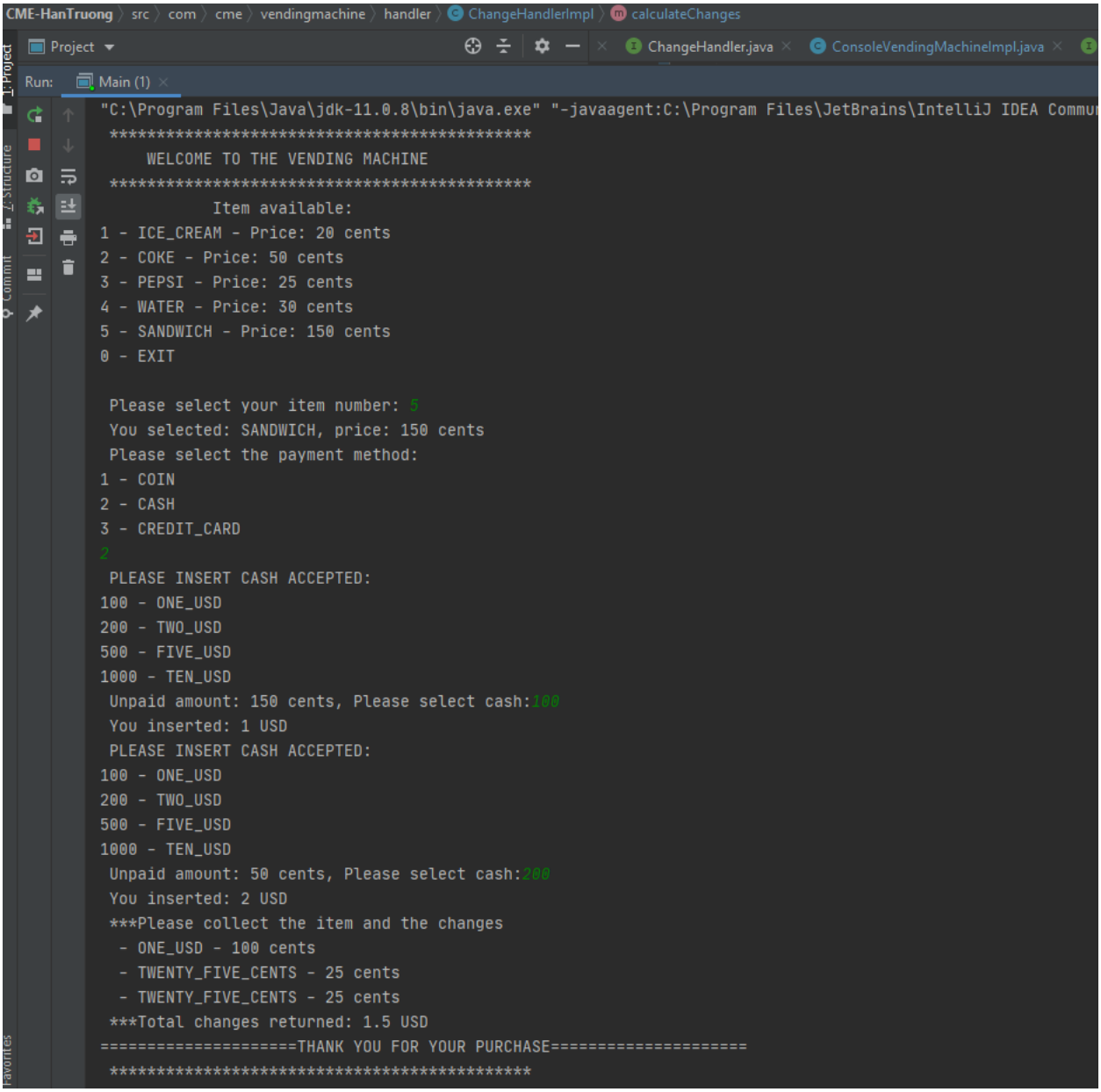
Solution Overview

The solution allows Vending machine to sell items and give the changes. The machine can load the selling items, coin and cash into inventory, allows buyer to make payment by different payment methods such as Coin, Cash or Credit Card payment.

In order to conduct the purchase, the machine will check whether the item is still available in inventory (In stock) and check if there is enough coin/cash to return the changes to the buyer.

1. Accepts the payments:
   - Coin: 1, 5, 10 and 25 cents.
   - Cash: 1, 2, 5 and 10 dollars bills.
   - Credit Card.
2. Allow user to select item to buy: Ice Cream, Coke, Pepsi, Water and Sandwich.
3. Cancel the operation and refund.
4. Return the changes.
5. Reset the operation.

The problem analysis and designs follow the Object-oriented analysis and design concept from Object-oriented analysis, Object-oriented design, Object-oriented modeling by applying the Object-Oriented programming (OOP). I apply five SOLID design principles that intend to make designs more understandable, reusable, flexible, and maintainable. It is reusable object-oriented design and design patterns of Gang of Four (GOF) principals such as Factory Pattern, Factory Method, Strategy, etc. The solution firstly implemented with Console interface as input and output as the following figure. However, we can easily implement a new interface.



Software Requirement

- Programming language: Java 8 or later.
- Testing: JUnit-4.13.2.jar, hamcrest-core-1.3.jar
- Tool/Plugins: IntelliJ, PlantUML plugin.

Architecture

Use Cases

1. The machine displays a welcome message, list of items and price to sell.
2. The machine asks user to select an item to buy.
3. The machine asks user to select the payment method options (pay by Coin, Cash or Credit Card)
4. User enters enough amount for the purchase (the machine keeps asking user to input enough amount to pay the item to continue or cancel/refund).
5. The machine calculates the change to return to user and return the item to user.
6. The machine updates the coin/cash/item inventory.
7. The machine displays the message with the change and the item when the purchase is successful.

UML Use Case Design

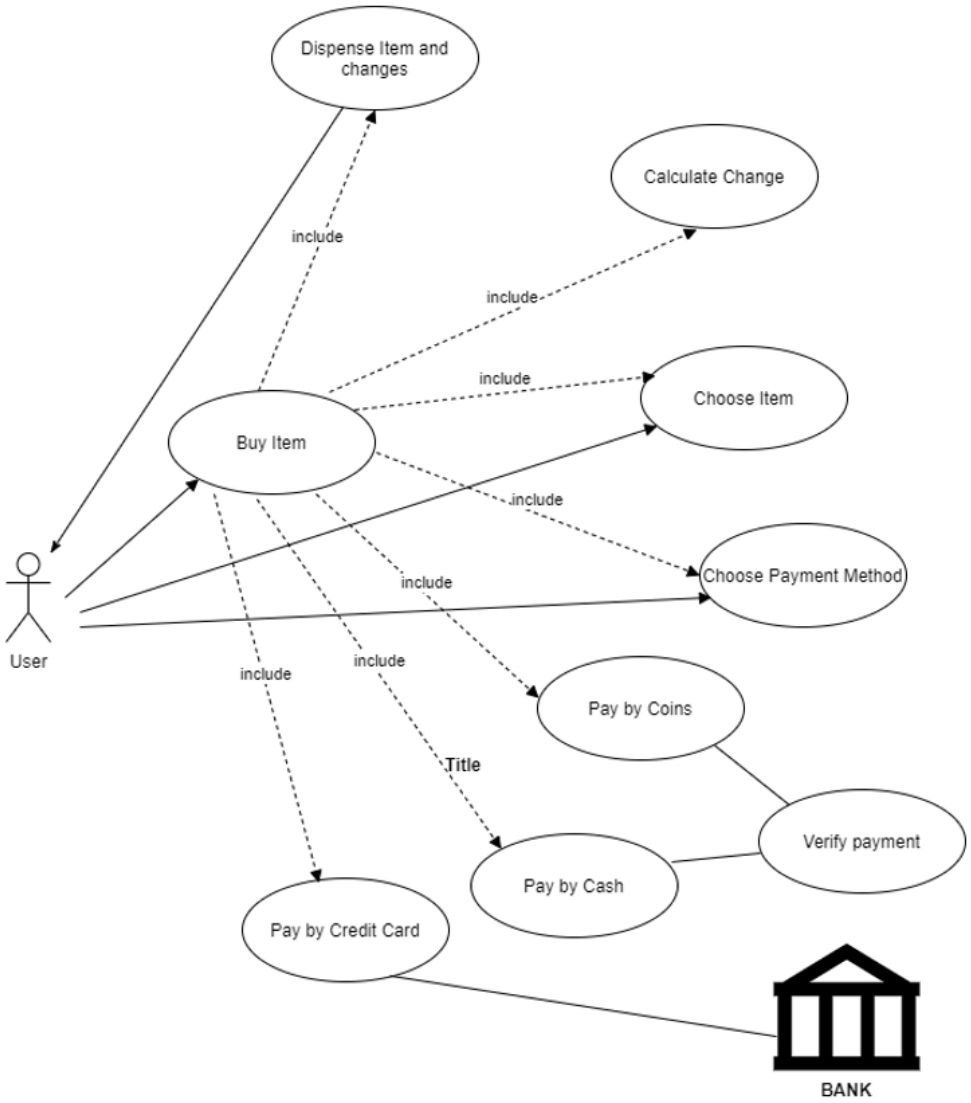
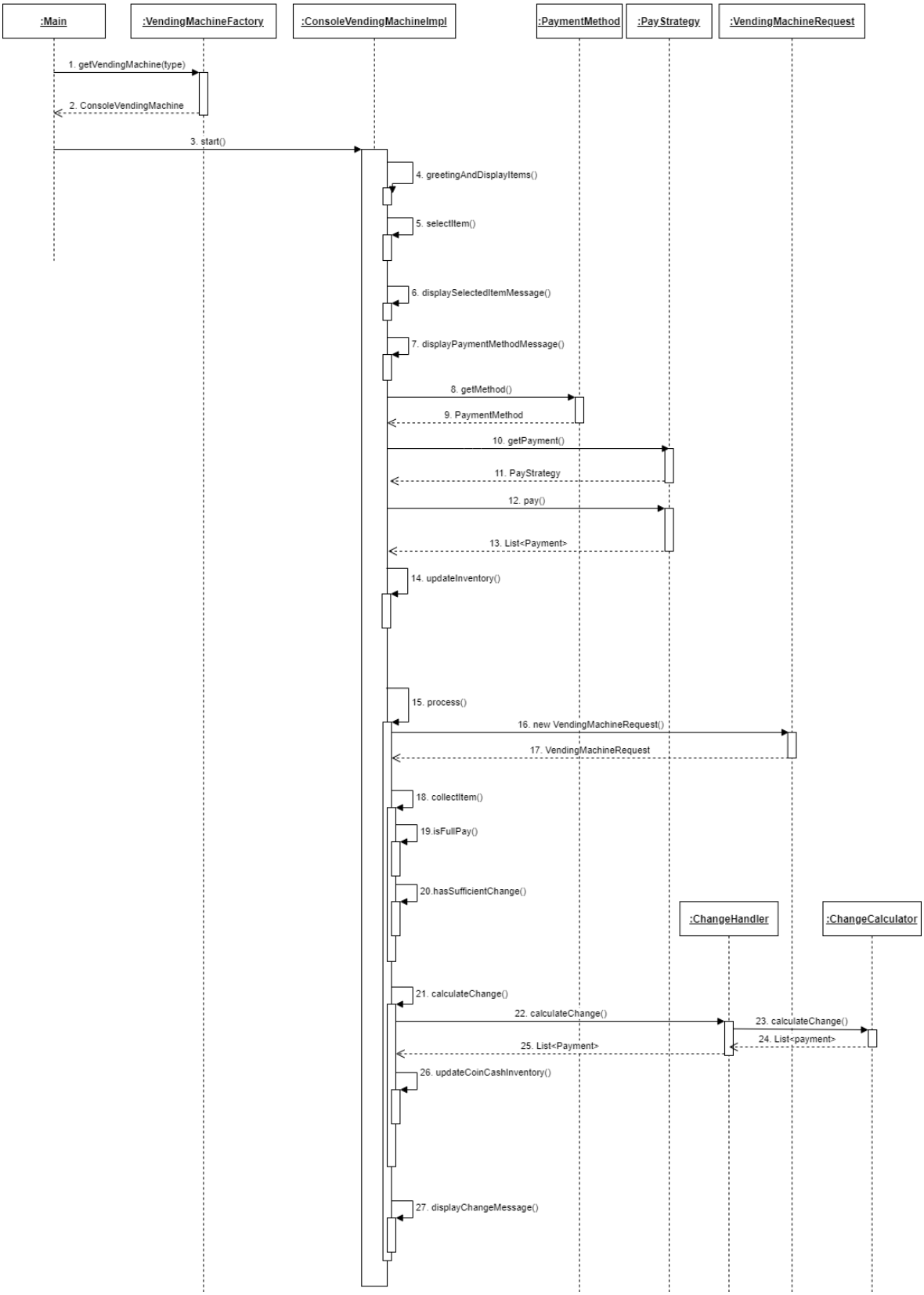


Figure: Use Case Diagram

Sequence Diagram Design



Class Diagram Design

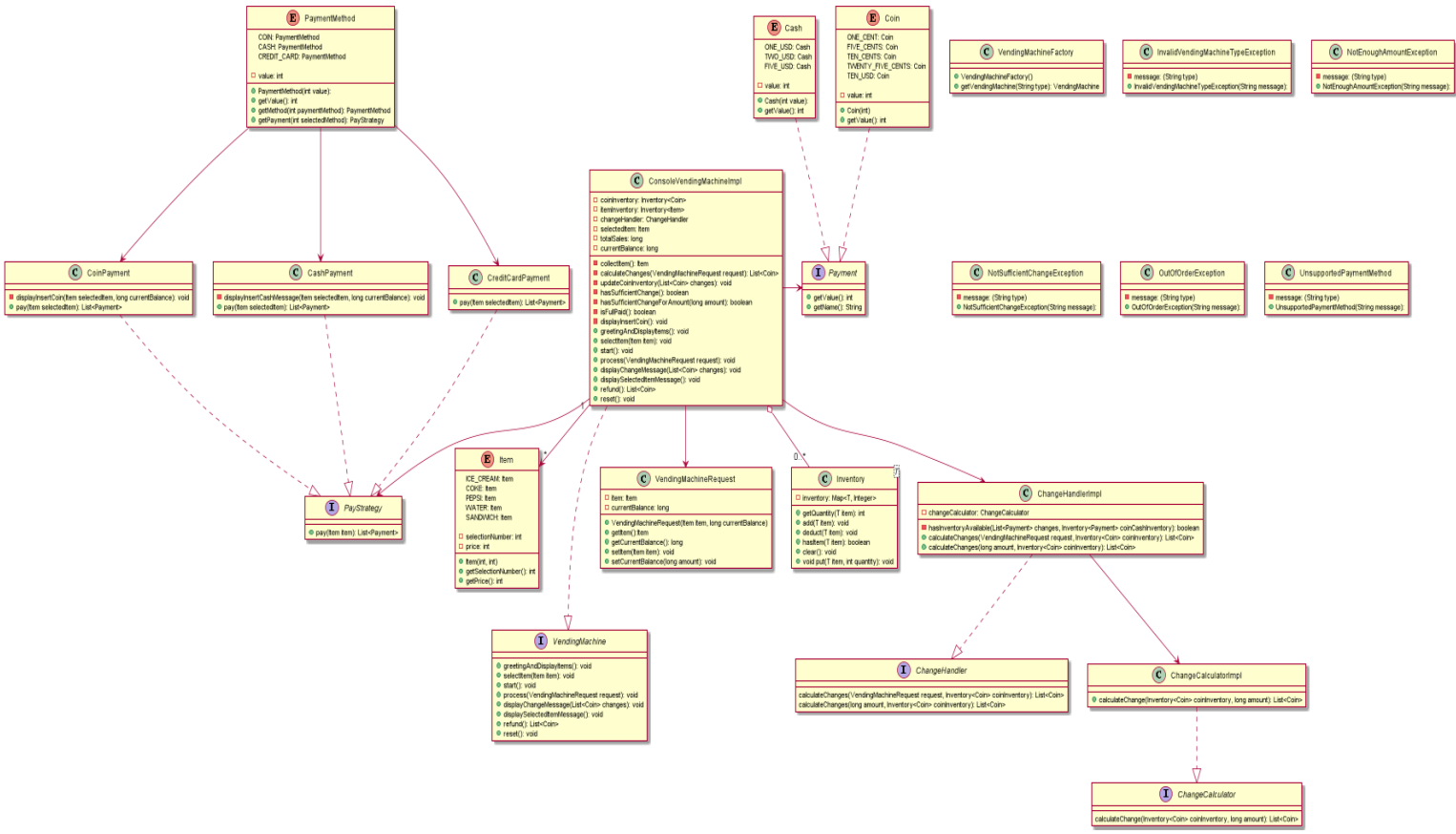


Figure: Class Diagram

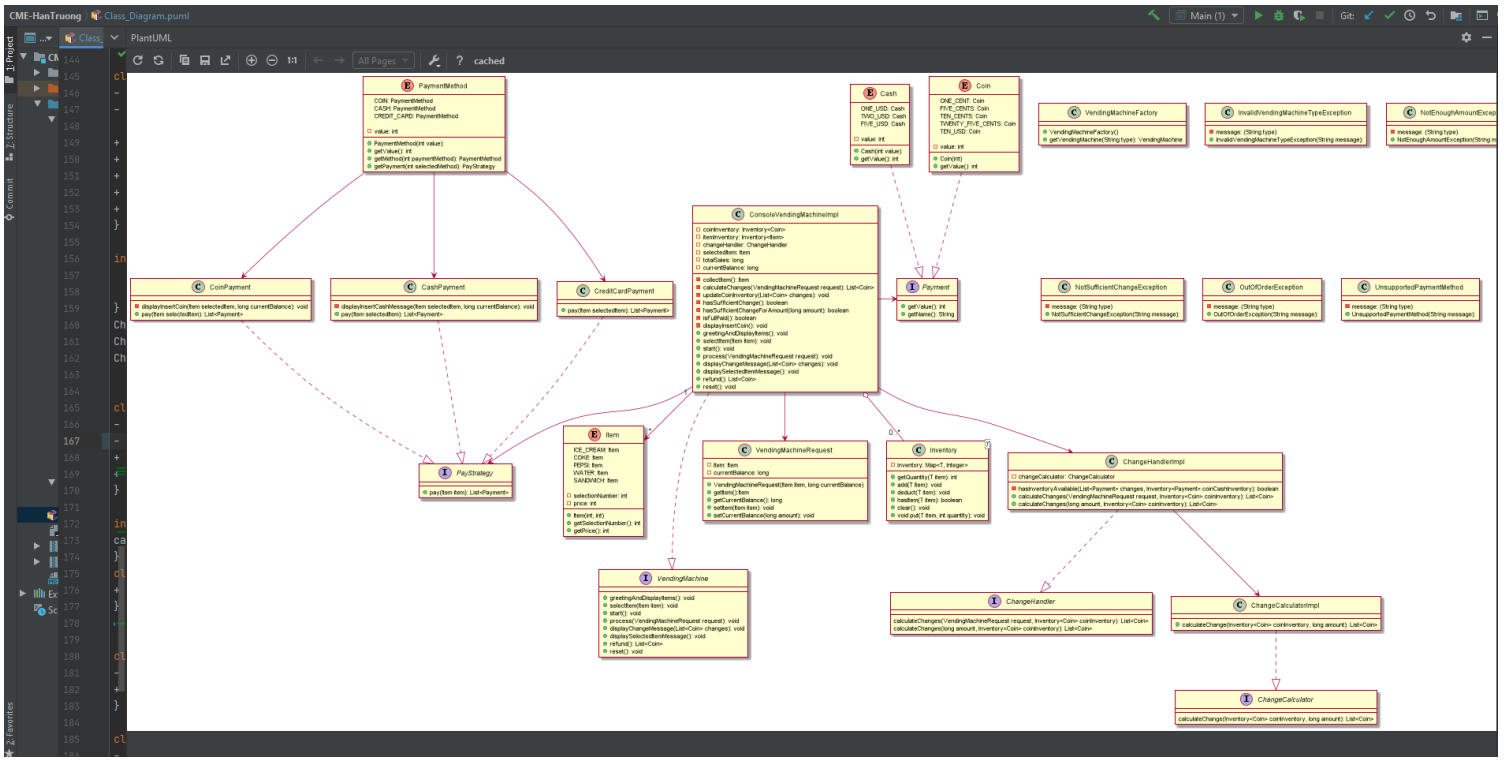


Figure: Class Diagram in PlantUML Plugin

Code Implementation

The problem develops in Java programming language with Object-Oriented programming (OOP) concept.

Vending Machine has the following classes and interfaces:

VendingMachine	The public API of a vending machine, usually, all high-level functionalities.
ConsoleVendingMachineImpl	A console implementation of VendingMachine.
VendingMachineFactory	A Factory class to create different kinds of Vending Machine.
Item	Java Enum to represent Item served by Vending Machine.
Inventory	Java class to represent an Inventory, used for creating the case and item inventory inside Vending Machine.
Payment	An interface for the payment like Coin, Cash or credit card.
Coin	A specific payment type of Payment interface.
Cash	A specific payment type of Payment interface.
PaymentMethod	A Java Enum to represent the supported payment method like Coin, Cash or Credit Card.
PayStrategy	An interface for the payment options Cash, Coin, Credit Card.
CashPayment	A specific payment method of interface PayStrategy.
CoinPayment	A specific payment method of interface PayStrategy.
CreditCardPayment	A specific payment method of interface PayStrategy.
VendingMachineRequest	An object represents a request sent to Vending Machine.
ChangeHandler	An interface to handle the calculation of changes.

ChangeHandlerImpl	An implementation of the ChangeHandler.
ChangeCalculator	An interface to calculate the changes.
ChangeCalculatorImpl	An implementation of ChangeCalculatorImpl.
InvalidVendingMachineTypeException	Invalid vending machine type exception.
NotEnoughAmountException	Not enough amount exception.
NotSufficientChangeException	Not sufficient change exception.
OutOfOrderException	Out of order exception.
UnsupportedPaymentMethod	Unsupported payment method exception.
Main	The main class to execute the program.

Java packages

- Exception
- Factory
- Handler
- Impl
- Model
- Payment
    - Impl
- Test

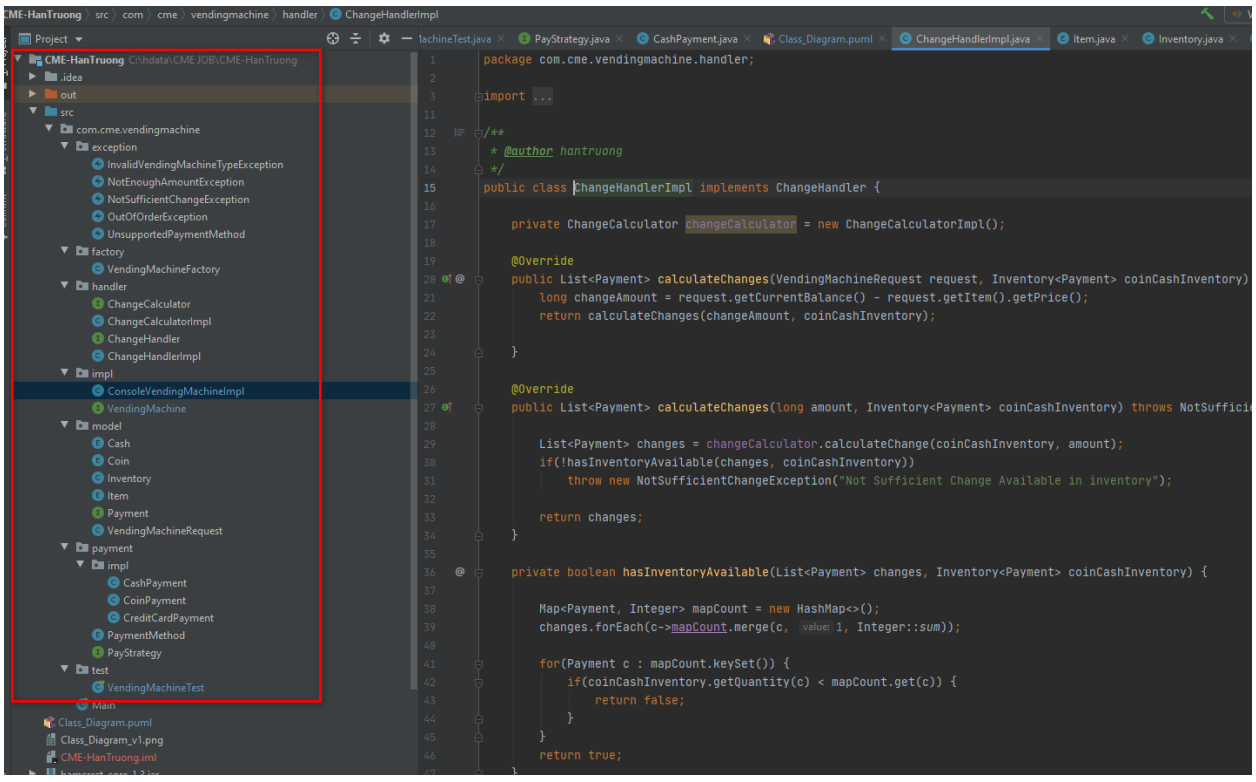


Figure: All Java packages

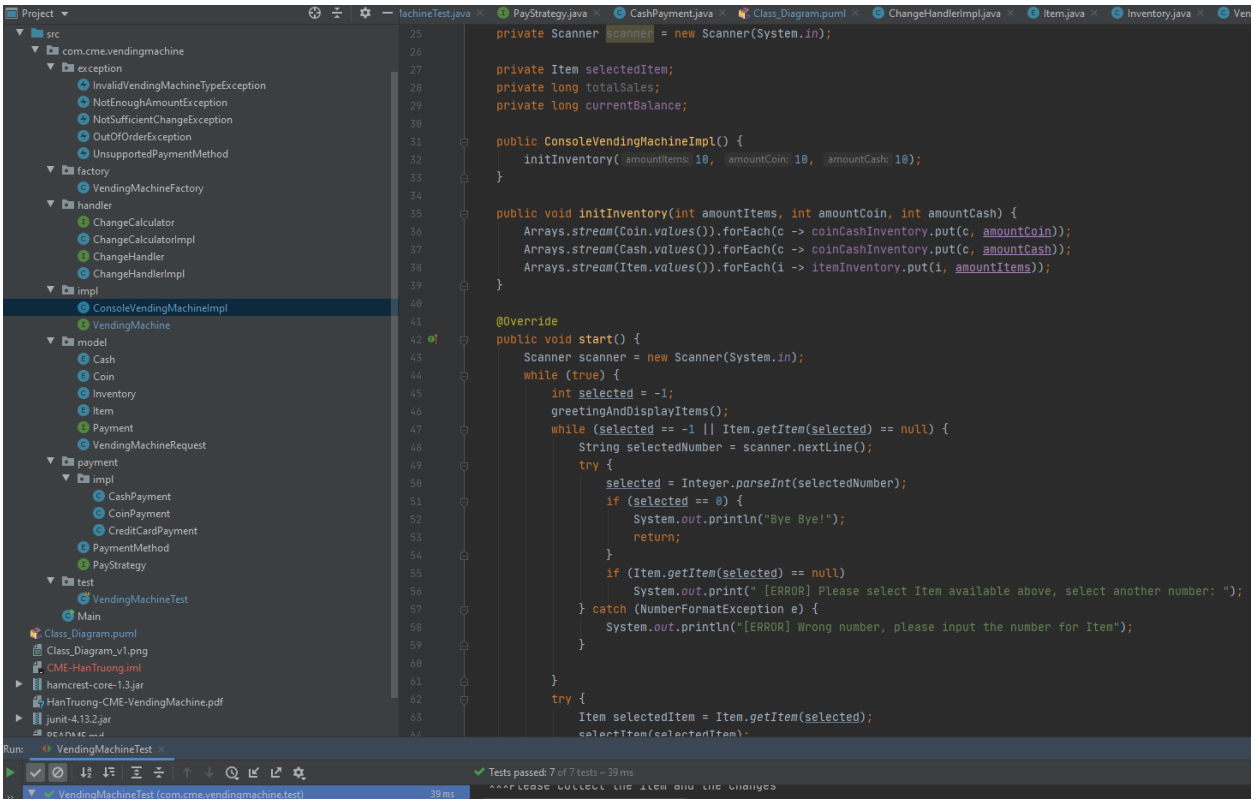


Figure: ConsoleVendingMachineImpl class

How To Run

1. Import project into IntelliJ
2. Add 2 jars junit-4.13.2.jar andhamcrest-core-1.3.jar as dependencies for Junit testing. Select **Files > Project Structure > Modules.**

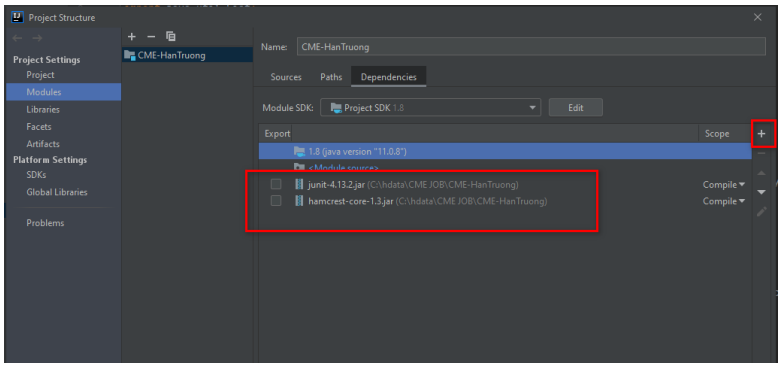


Figure: Add Jars to build path

3. Build and run the **Main** method in **Main** class.

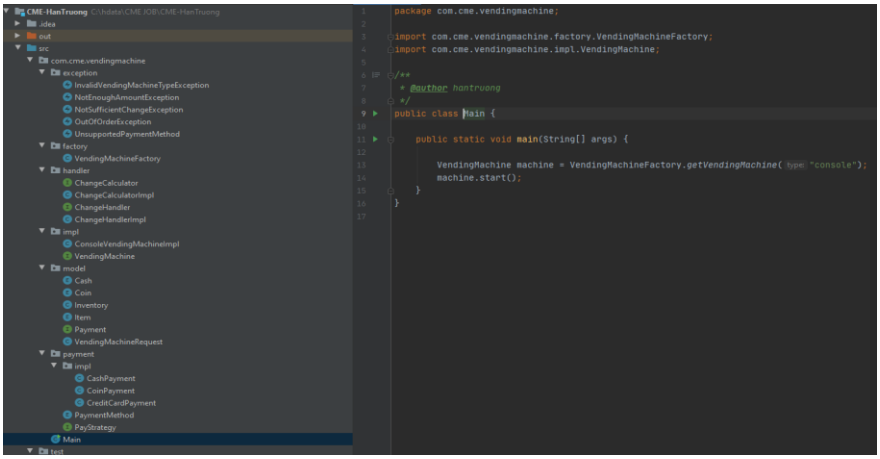


Figure: Main class

4. How the machine works?
    a. User selects the Item to buy, by entering number, e.g.: 5
    b. The machine asks user to select the Payment method. The user selects the payment method by entering the number, e.g.: 1 for Coin, 2 for Cash or 3 for Credit Card.
    c. User inserts the payment until it has enough amount to pay for the item.
    d. The machine will dispense the Item and return the changes.

5. The Machine Run in Console.

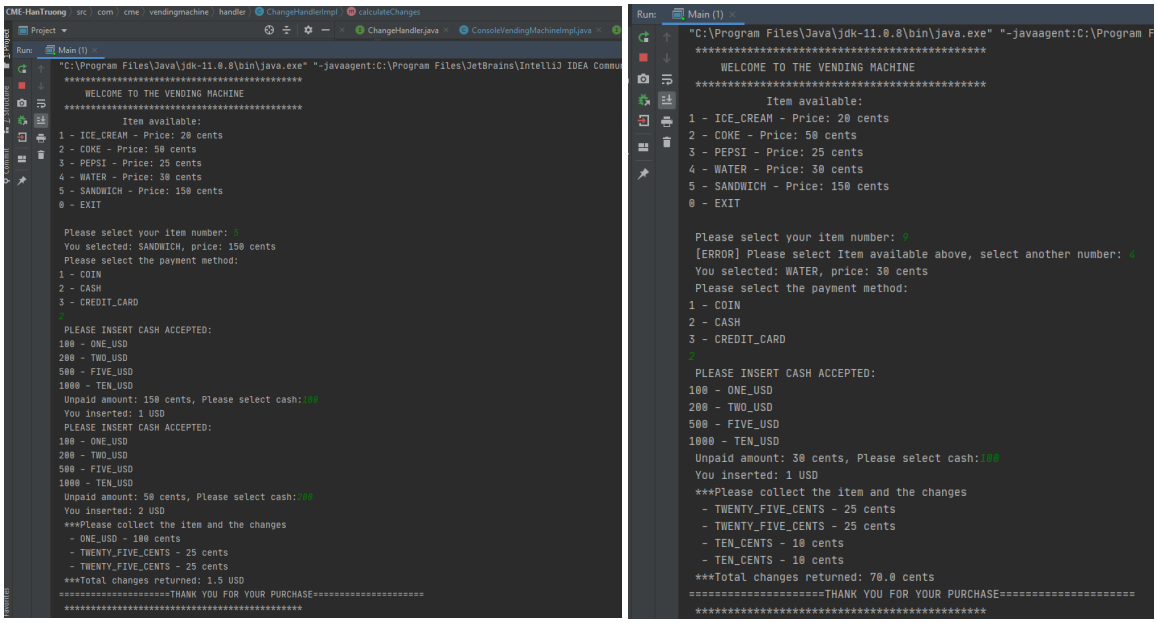


Figure: The machine running in Console

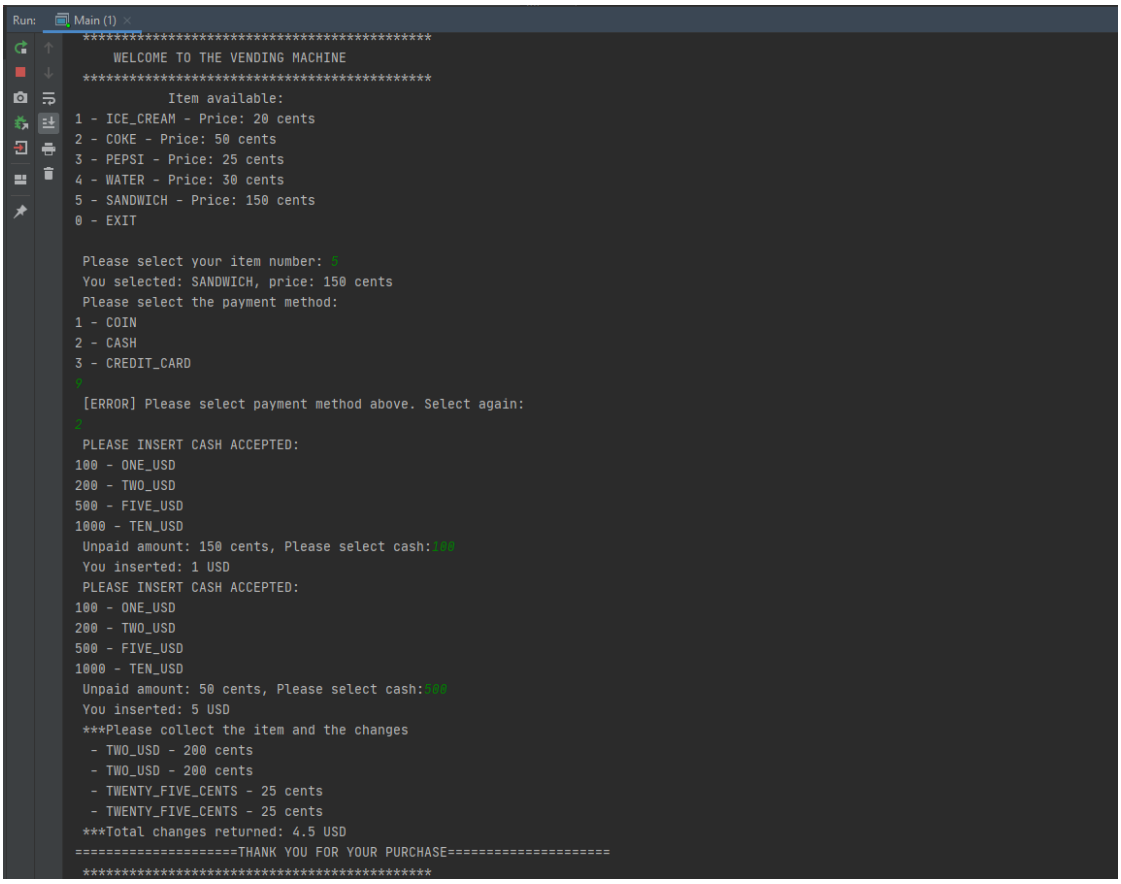


Figure: The machine running in Console

Integration

Execute the test **VendingMachineTest** class.

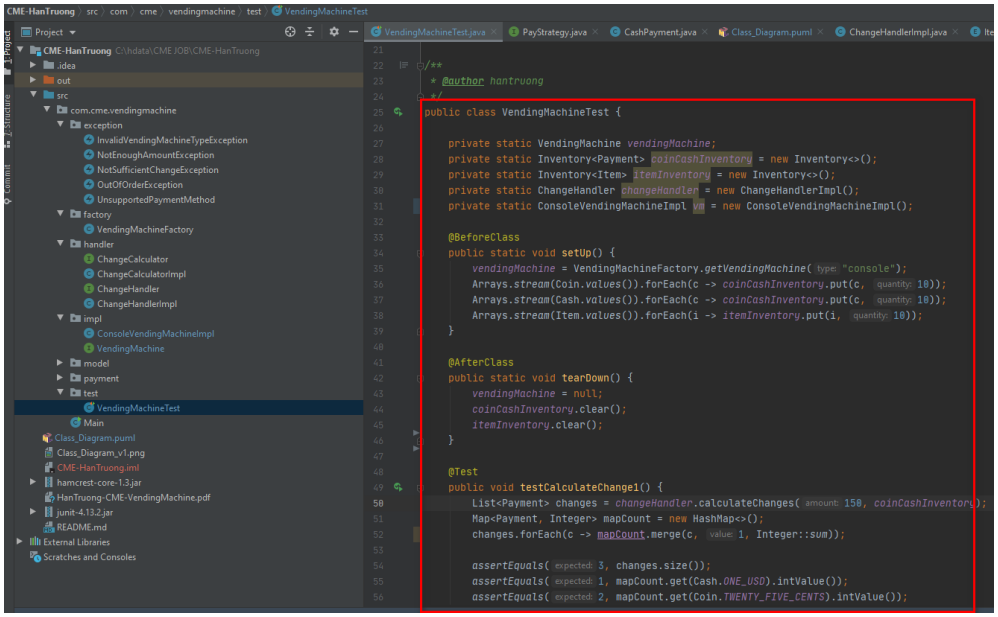


Figure: VendingMachineTest class

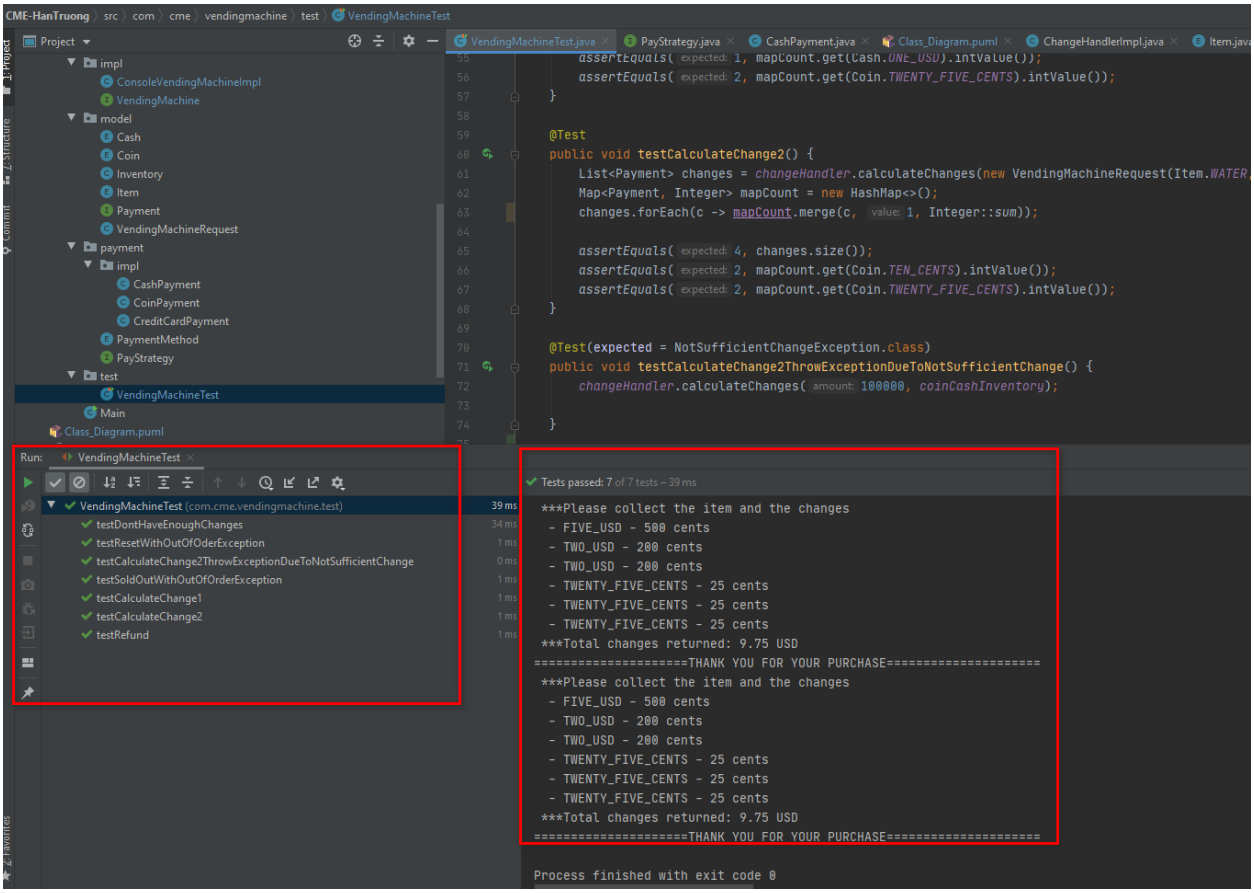


Figure: Tests Run Passed

----------Thank you so much ----------