

B202 Advanced Programming Project

Student: Sami Chamali

Student-ID: GH1033269

Date of Submission: 19/12/2025

Here are the required links to related to this project:

Explanation Video: <https://youtu.be/tz6CXXh1tDc>

Github Link: <https://github.com/samichamali/JavaProject>

Market Trading Simulator

This project simulates the fluctuation in the crypto market at a much quicker rate than real life.



It tries to imitate the constant radical fluctuation in the crypto market by utilising random functions adjusted by set chances.

Introduction:

It's undeniable that the jump onto the crypto trade trend has caused a lot of damage to wallets of those clueless in the field, from ignorance in fluctuation analysis to tying market values to real-life events. People don't understand the numbers nor why they change! Which is why this project has been created to provide a "curious trader" the opportunity to play around with a simulated crypto market and control the chances of increase/decrease and if the coin gets to float or not, basically check the effects of both fruition and failure.

With this project, people can finally decide whether they want to partake in this field or stay away from...

"Are you able to carry the chainsaw without hurting yourself?"

System Architecture:

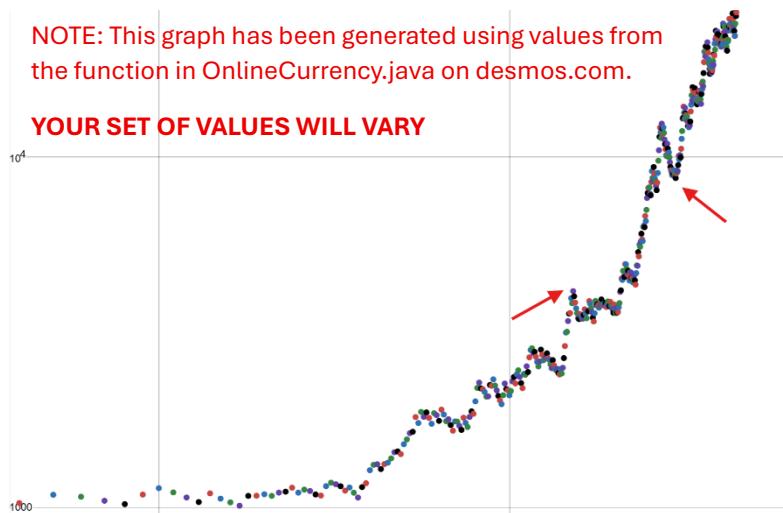
This is a console-based project developed using the Java Programming Language. It initiates a class-thread that runs a sophisticated random value generator that is used by other classes to read an always updating variable. All the other classes are very well-connected and access each others' important variables. And all the transactions are logged into a TXT File.

Let's discuss the most important phases of this project's development life cycle.

Unlike a computer, the crypto market might look random, but that's not entirely true. The value of a coin can fluctuate only through a specific range under certain market conditions.

In short: The coin won't drop or rise to an extreme point all by itself; that is only possible if real-world events are able to dictate its value.

AI was Utilised to generate mathematical equations to use in a function that runs in a separate thread and tries to generate coin-values that look like that of an actual crypto-coin, example: Bitcoin.



```
// Initiating the Crypto Currency Market Simulator Function in a separate thread
OnlineCurrency.Market Crypto_Market = new OnlineCurrency().new Market();
Crypto_Market.start();
```

What are the chances of fluctuation in this simulated market?

The settings/numbers chosen to generate random market values have been set to have the coin flourish after a certain period, which means that it's essentially a **logarithmic** function.

Chances

The loop generates a random number between 0 and 999; 1000 Total Chances.

These Ranges have been set **purely by AI to stabilise the value of the coin as time progresses.**

Chance Range	Probability	Event	Effect on Price
0 to 249	25%	Slight Increase	The price moves up slightly, and volatility begins to dampen (decrease).
250 to 599	35%	Slight Decrease	The price moves down slightly, countering the upward bias, and volatility continues to dampen.
600 to 998	40%	Significant Jump (With Bias on Increase)	The price jumps upward quickly. This case also includes a small resistance factor to prevent infinite growth. Volatility increases.
999	0.1% (1 in 1000)	Significant Drop (Crash)	The price instantly drops by 40% to 60%. This event immediately causes volatility to spike dramatically.

Let's talk about the most technical aspects of the project and how it works:

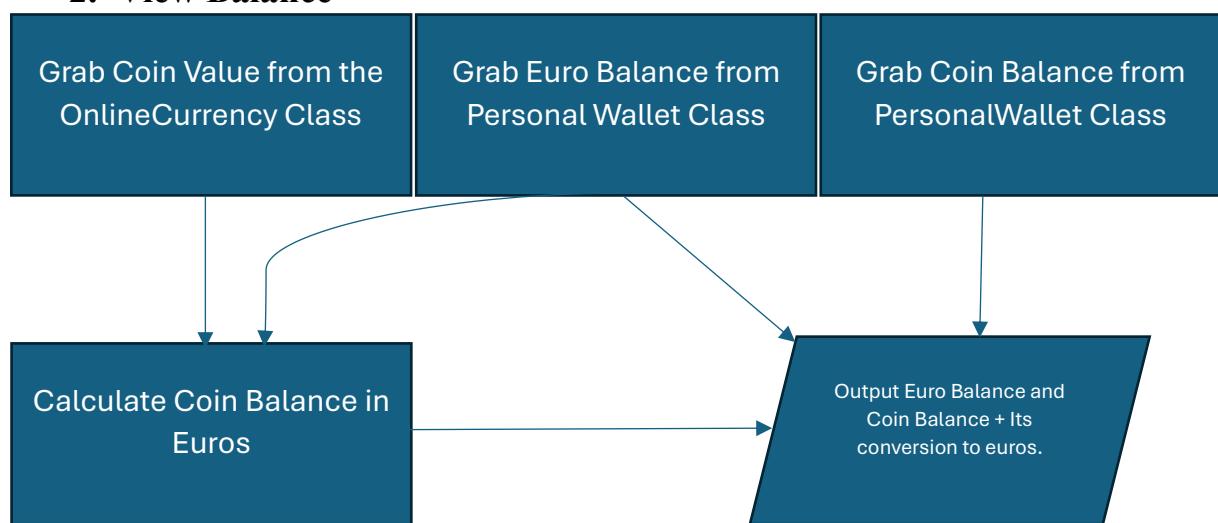
The user is greeted in the console with 3 options:

1. View Coin Value
2. View Balance
3. Trade

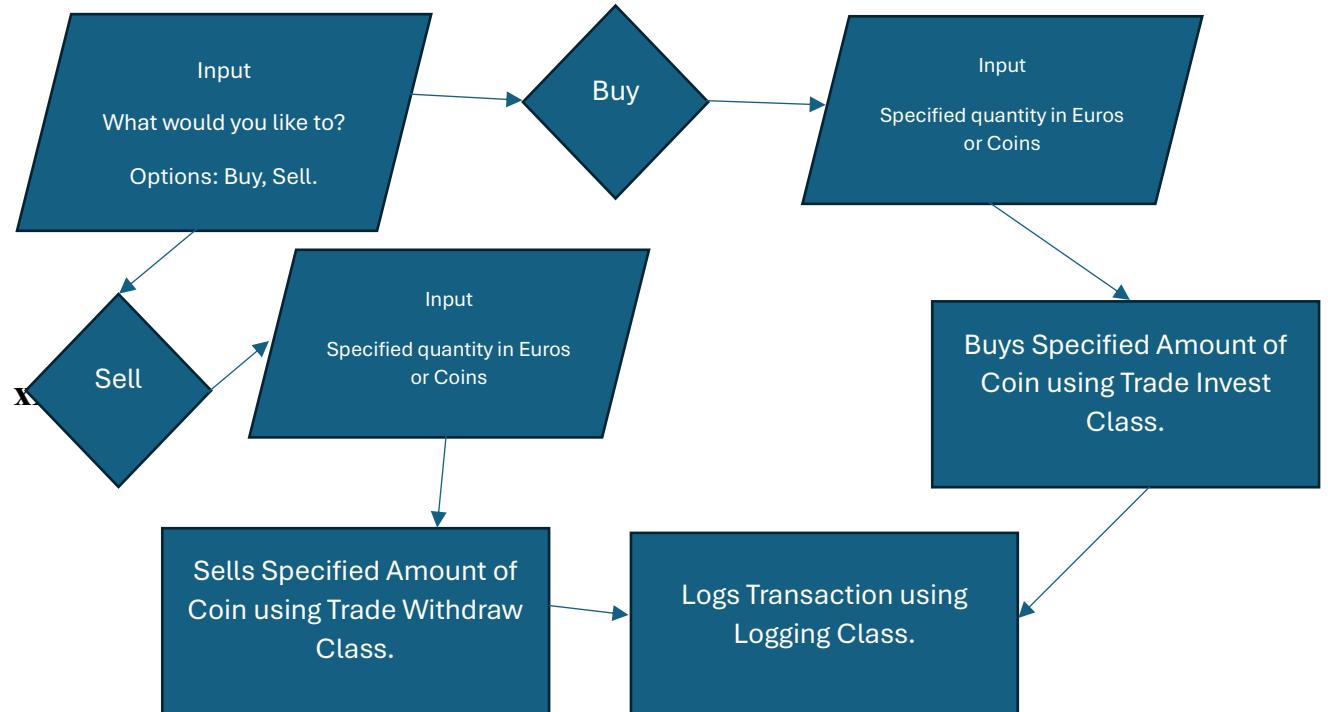
1. View Coin Value



2. View Balance



3. Trade



Results – Let's take a look at the interface and how to use it!

```
Welcome to the Commerce Simulator!
=====
Choose one of the following options:
1. View Coin Value
2. View Balance
3. Trade
-----
Choice: 1
Coin Value: 10671.711€
-----
Click Enter to continue
=====
Choose one of the following options:
1. View Coin Value
2. View Balance
3. Trade
-----
Choice: 2
Euro Balance: 10000.0€
Coin balance: 0.0 / 0.0€
-----
Click Enter to continue
=====
Choose one of the following options:
1. View Coin Value
2. View Balance
3. Trade
-----
Choice: 3
What would you like to do?
1- Buy Coin
2- Sell Coin
Choice: 1
-----
How do you want to buy?
1- In Coins
2- In Euros
Choice: 2
-----
Please Enter the amount you would like to invest.
Euros: 10000
Successfully invested!
-----
Click Enter to continue
=====
Choose one of the following options:
1. View Coin Value
2. View Balance
3. Trade
-----
Choice: 1
Coin Value: 12063.035€
-----
Click Enter to continue
=====
Choose one of the following options:
1. View Coin Value
2. View Balance
3. Trade
-----
Choice: 2
Euro Balance: 0.0€
Coin balance: 0.82679754 / 12045.942€
-----
Click Enter to continue
=====
```

Option 1 has been chosen, to view the value of the coin.



Option 2 has been chosen, to view my current balance.



Option 3 has been chosen, to start the trade process. Where we're asked if we to Buy or Sell coins. Let's select Buy.



The program then asks us if we the amount we would like to input is in Coins or Euros. Since I only have 10,000 euros, and haven't calculated how much that is in coins, I want to buy the maximum amount I can with my current balance of 10,000€. Let's go all in.



Let's check if the coin had increased in value, and it did!



After a bit of patience, the coin had increased in value, which then gave us a profit of 2045.942€. A good Investment!



Challenges and Solutions:

The challenges I have faced while developing this project were mostly centered around keeping the coin stable and avoiding any sudden rises or drops, this was very difficult given the fact that computers aren't very good at generating random numbers to begin with... but we've got to work with what we have.

Using the **help of AI and Desmos**, I was able to generate an almost logarithmic function that starts off with small fluctuations in the coin's value, and then after some time, the coin starts to flourish, meaning that its value will sky-rocket. This has been intentionally designed to see if the user is knowledgeable enough to buy a significant portion of the coin before that increase and then sell at the top, which would be extremely profitable, or miss out because of late investment.

The rest of the project was created entirely by me.

Conclusion

As we reach our final words, this project has been created with the goal of educating a person who isn't knowledgeable in the field of trading about the dangers and reasons behind its results. It could be extremely beneficial to those who want to test the market under certain conditions, an example could be where failure is more prone than success (editing chances to see the result of a coin's value on a graph).

Where could this project improve, and what are some future considerations I should take to enhance this project?

Adding a **Graphical User Interface** would have given the program a much better feel compared to its current console-based state, I was thinking of adding a GUI but faced a lot of trouble trying to create visual updating graphs, I deemed it too time consuming and had to put it in my future considerations list.

And here is my other more general list of points of improvement:

1. Adding Multiple Coins that differ in Value (Multiple Threads)
2. Adding Multiple Currencies (Conversion Class)
3. Using an API Framework and having Multiple Traders (Play Framework)
4. Providing the User with Graphs and Visual Indicators of a Coin's Value.

It has certainly been a blast developing this project, and thanks for reading!