

Development and Implementation of a Capital Gains Tax Calculation Program in Java

*A Report on SENG1110 Assignment 2*

**The University of Newcastle**

*SCHOOL OF INFORMATION AND PHYSICAL SCIENCES*

SENG1110 - Object-Oriented Programming

**Authors**: Min Thu Khaing, Thet Paing Hmu

**Instructor**: Dr. Poh Kok Loo

**Submission** **Date**: 5/04/2024

Table of Contents

[Abstract 2](#_Toc163223323)

[Introduction 3](#_Toc163223324)

[Design and Implementation 3](#_Toc163223325)

[Functionality 4](#_Toc163223326)

[Error Handling 4](#_Toc163223327)

[Results and Analysis 5](#_Toc163223328)

[I. Testing the Program with Valid Input 5](#_Toc163223329)

[II. Testing the Program with Invalid Inputs 21](#_Toc163223330)

[Conclusion 26](#_Toc163223331)

[Appendix 27](#_Toc163223332)

[Appendix A : Sample Code Snippets 27](#_Toc163223333)

## Abstract

This report details the implementation and extension of Assignment 1 which calculates Capital Gains Tax (CGT) and profits from cryptocurrency transactions at the University of Newcastle, Trimester 1, 2024. The objective of Assignment 2 is to create a program that manages user information and investments, with specific functionalities such as adding users, deleting users, displaying user information, managing investments, and saving data to files.

The program allows for up to 5 users, each capable of having 2 investments. Users input their information, including name, annual salary, residency status, and investment details such as buying price, selling price, and holding duration for cryptocurrencies. The program provides options to add, display, and delete users and investments, as well as to save data to files.

Three classes are defined for the implementation: Investment, User, and CgtInterface. The Investment class handles investment details, including deposit amounts and cryptocurrency selection. The User class manages user data and investment accounts. The CgtInterface class serves as the interface for user interaction, facilitating input validation and output display.

Overall, this report describes the design and functionalities of the program, providing guidance on usage and implementation to fulfill the assignment requirements effectively.

## Introduction

The objective of this assignment is to enhance the functionality of a previous program by incorporating arrays and external file handling. Building upon the concepts introduced in Assignment 1 and utilizing the principles of object-oriented programming, the aim is to create a robust system capable of managing user information and investments efficiently.

In this report, we explore the detailed specifications and requirements of the program, clarifying its functionalities and the essential steps for the implementation. The program is designed to manage a maximum of 5 users, each endowed with the capability to maintain 2 distinct investments. Users are prompted to input their personal information and investment criteria, with provisions for seamlessly adding, deleting, and displaying both user profiles and investment portfolios. Furthermore, the program offers the convenience of data preservation by facilitating the storage of user and investment details in external files.

## Design and Implementation

The program is designed using object-oriented principles, with a focus on modularity, and scalability, and is structured with three key classes:

1. **CgtInterface** - The CgtInterface class serves as the main interface for user interaction, facilitating input/output operations and validation. It arranges the functionalities of the program, including adding, deleting, and displaying users and investments, as well as saving data to external files. This class communicates with users, validates inputs, and invokes appropriate methods from the Investment and User classes to execute requested actions.
2. **Investment** - The Investment class encapsulates investment details, including deposit amounts for three consecutive years (year1Deposit, year2Deposit, year3Deposit) and the selection of the cryptocurrency (coinSelection). It features constructors to initialize these variables and methods to modify and access them as necessary.
3. **User** - The User class manages user-specific data such as name, annual salary, residency status, and investment accounts. It incorporates an array of Investment objects (investAccount) to keep track of two investments per user. This class also provides methods for accessing and updating user information. This class includes information such as the user's name, annual salary, residency status, investment details, tax rate, CGT, and actual profit after deducting CGT.

## Functionality

The functionality includes:

1. **Adding Users**: Users can be added to the system, providing their personal details such as name, annual salary, residency status, and investment parameters. The program ensures that the number of users does not exceed the limit of 5.
2. **Deleting Users**: Users can be removed from the system by specifying their name. The program verifies the existence of the user and displays appropriate messages based on the outcome.
3. **Displaying User Information**: Users' information, including their name, residency status, investment details, and remaining investment capacity, can be displayed. The program validates the user's existence and provides comprehensive information accordingly.
4. **Adding Investments**: Users can add investments to their account, specifying details such as initial deposit, yearly investment amounts, and cryptocurrency choice. The program ensures that users do not exceed the maximum limit of two investments and validates input feasibility.
5. **Displaying Investment Information:** Users' investment details, including yearly profits for the next three years, can be displayed. The program validates the existence of the investment and provides detailed information accordingly.
6. **Deleting Investments**: Users can remove investments from their account, specifying the investment number. The program validates the existence of the investment and displays appropriate messages based on the outcome.
7. **Saving Data to External Files**: User and investment information can be saved to external files in text format. The program ensures proper formatting and handles cases where there are no users or investments. The program prompts users to input relevant data such as name, annual salary, cryptocurrency transaction details, and investment preferences.

Input validation ensures that only appropriate values are accepted, enhancing the reliability of calculations.

## Error Handling

Error checking and input validation are performed in the class CgtInterface, within the methods getValidatedNumInput() for double data type, and getValidatedInput() for Strings. This is achieved by comparing the input value against the specified regular expression.

**getValidatedNumInput()** - Retrieves and validates numerical input from the user and returns the validated numerical input from the user. And it takes the following parameters.

1. minimum - The minimum allowed value for the input.
2. maximum - The maximum allowed value for the input which can be canceled out by passing the value “-1”.
3. acceptEqualToMinimum - Boolean value whether going to accept equal amount to minimum amount.
4. prompt - The message prompting the user for input.
5. console - The Scanner object used for input.
6. regex - The regular expression used to validate the input.
7. invalidMessage - The message displayed when the input is invalid.

**getValidatedInput()** - Retrieves the String input from console and validates it using given regular expression and return the value if successful. And it takes the following parameters.

1. prompt - The message prompting the user for input.
2. console - The Scanner object used for input.
3. isName - Boolean value whether the function should expect a name or not.
4. regex - The regular expression used to validate the input.
5. invalidMessage - The message displayed when the input does not match the regular expression.

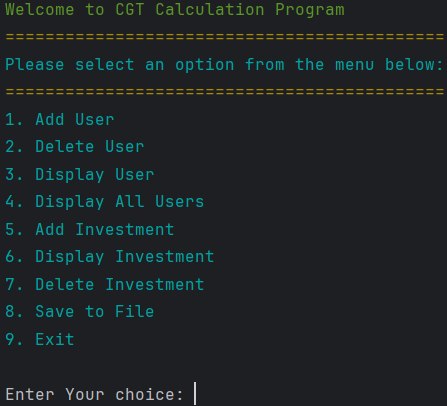
(Refer to this [section](#_Appendix_A_:) to view the code implementation.)

## Results and Analysis

We conducted extensive testing to ensure the accuracy and reliability of the program. Test cases covered various scenarios including valid inputs, invalid inputs, boundary cases, and error handling which we use the same functions to check the validation of the input from Assignment 1. The program consistently produced correct results, demonstrating its effectiveness in calculating CGT (Capital Gains Tax) and profit for cryptocurrency transactions.

### I. Testing the Program with Valid Input

The program will allow up to 5 users and each user will be able to have 2 investments. When the user starts executing the program, the program will ask the user to choose from the options of Add User, Delete User, Display User, Display All Users, Add Investment, Display Investment, Delete Investment, Save to File and Exit. These options will be the main window which the user will be asked to choose throughout the program.



A screenshot of a computer screen

Description automatically generated

If the user chooses to add User which is option 1, the program will ask the name of the user, annual salary, residential status, buying price, selling price and number of years held of the cryptocurrency.

A black screen with colorful text

Description automatically generated

The user is successfully added to the program.

A screenshot of a computer program

Description automatically generated

If the user chooses option 2 ,delete user, then the program will ask the name of the user to be deleted. The program will search for the user in the program afterward.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

The user is successfully deleted from the program.

A screenshot of a computer program

Description automatically generated

If the user chooses option 3 which is to display a user within the program, the program will ask the name the user wants to display. The program will then search for the user in the program.

A screenshot of a computer

Description automatically generated

If the user has already been created, the program will display the user’s information, capital Gains Tax, Available Balance, Investment Account, and the Profit from the coin invested.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Choosing option 4 will display all the users that have been created into the program.

The following is the output of all users in the program.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

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 screen

Description automatically generated

After showing all the users saved inside the program, the program will return to the option menu.

A screenshot of a computer program

Description automatically generated

If the user chooses option 5 that adds investment to the user, the program will ask the Name of the user to add the investment. Since the username “Dave” already has 1 investment, it is possible to add another investment. The program searches for the username “Dave”, asking the necessary questions concerning about investment coin.

A screen shot of a computer

Description automatically generated

The user chooses one investment coin, and the investment is successfully added to the user name “Dave”.

A screenshot of a computer program

Description automatically generated  
Option 6 will display the investment of a single user within the system. The image above displays the investment for the user “Dave”.

A screenshot of a computer

Description automatically generated

Since the user “Dave” has 2 investment accounts, the user can choose which investment account to display.

A screenshot of a computer

Description automatically generated

If the user wants to delete investment accounts, the user can choose option 7 to delete any account the program has saved to that name. The image below shows that the deleted account cannot be found in the system under the username “Dave”.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

To save all the data into the file, the user can choose option 8. For a

A screen shot of a computer code

Description automatically generated

The images below are the data saved into the “usersdata.txt” file.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

If the user enters option 9, the program will end there. If one of the users does not have an investment account, there will be a message showing no investment account for that user.

A screenshot of a computer program

Description automatically generated

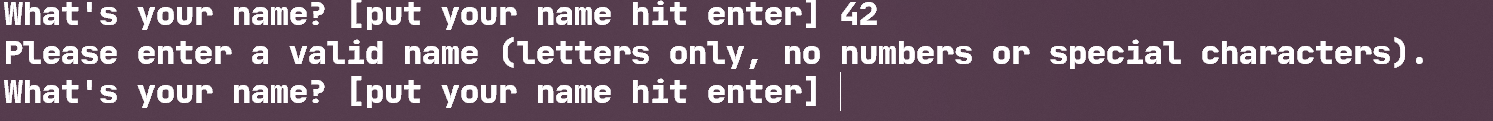
If the user enters option 9, the program will end and exit there.

### II. Testing the Program with Invalid Inputs

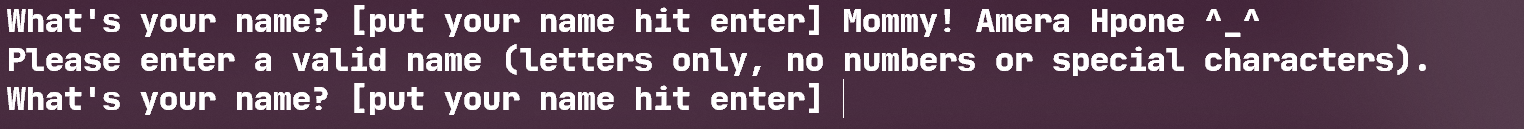
Extensive testing was conducted to evaluate the program's error-handling abilities, covering scenarios like using special characters or numbers in names, inputting negative values, leaving fields empty, and exceeding specified ranges.

#### 1. Name

First: Using numbers.



Second: Using special characters.

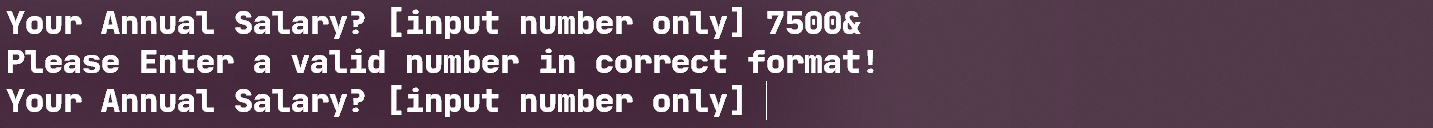


Third: Using an empty input.

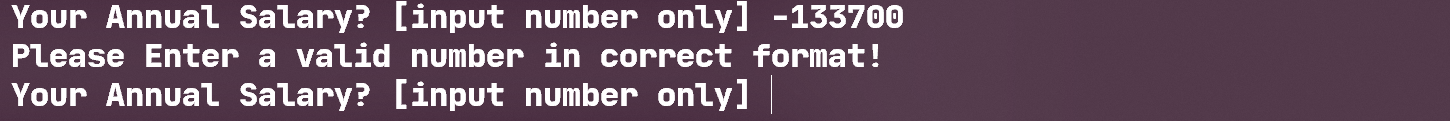
#### 

#### 2. Salary

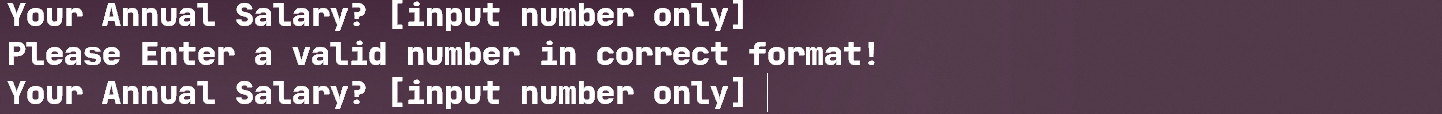
First: Using special characters.



Second: Using numbers other than a positive value.

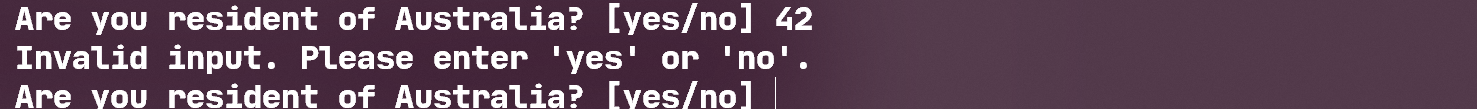


Third: Using an empty input.

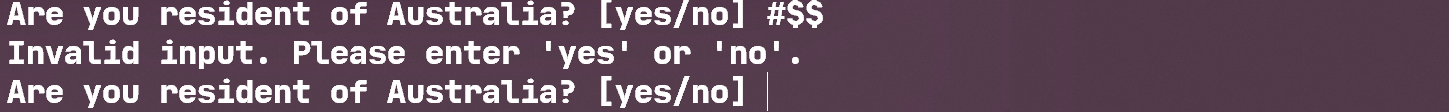


#### 3. Residential Status

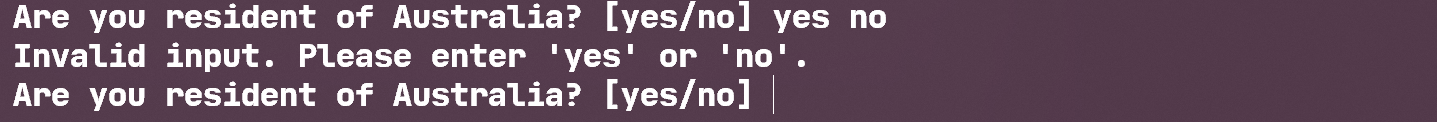
First: Using numbers.



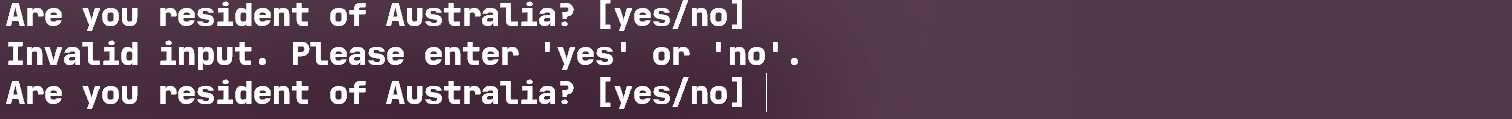
Second: Using special characters.



Third: Providing both yes and no at the same time.

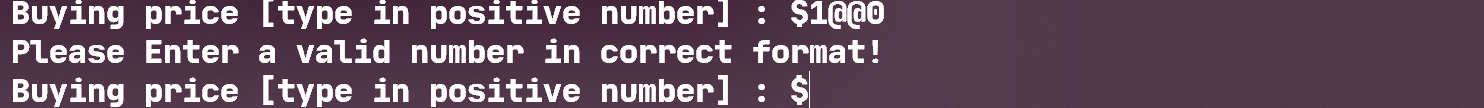


Fourth: Using an empty input.



#### 4. Buying Price and Selling Price

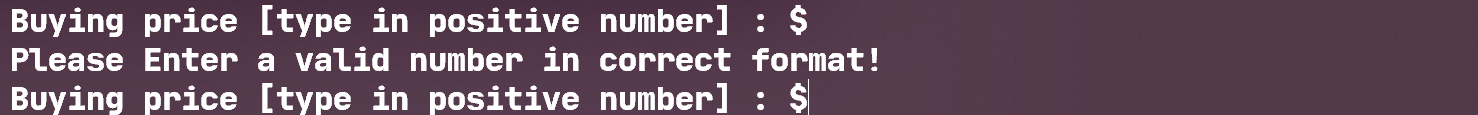
First: Using special characters.



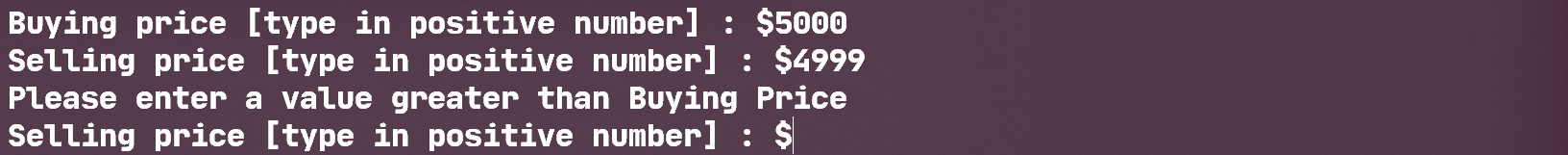
Second: Using numbers other than a positive value.



Third: Using an empty input.

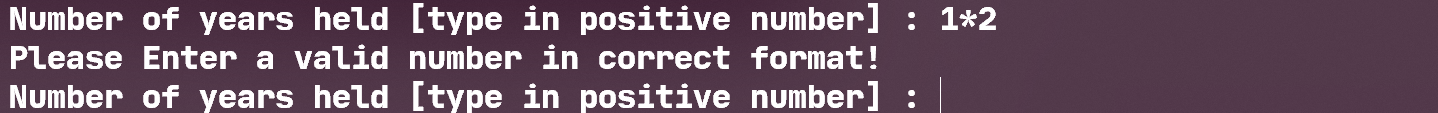


Fourth: Using a value less than or equal to the Buying Price.

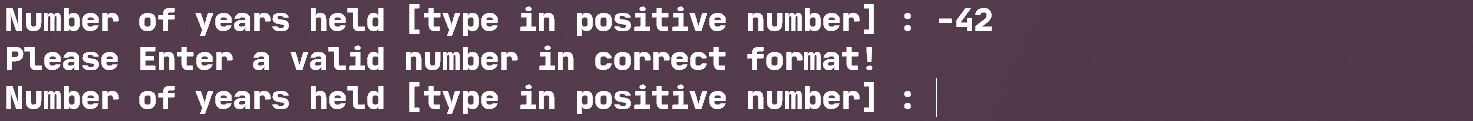


#### 5. The number of years held.

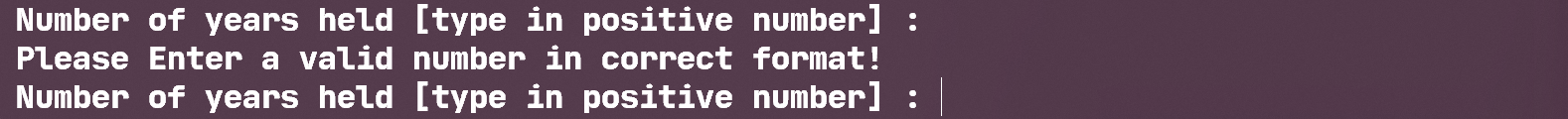
First: Using special characters.



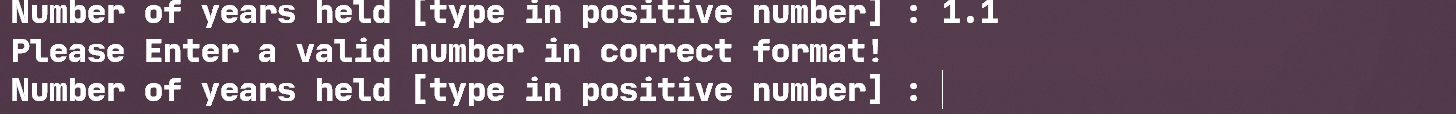
Second: Using numbers other than a positive value.



Third: Using an empty input.

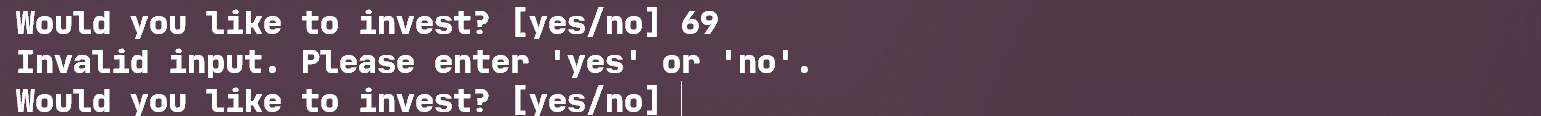


Fourth: Using decimal values.

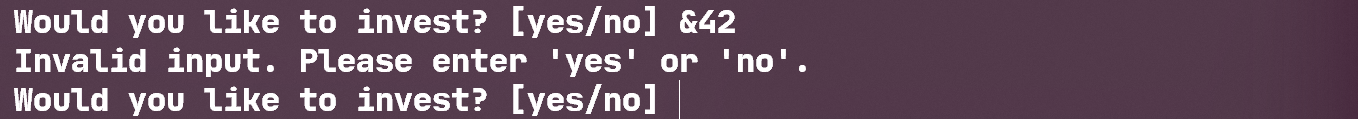


#### 6. Prompt to Invest.

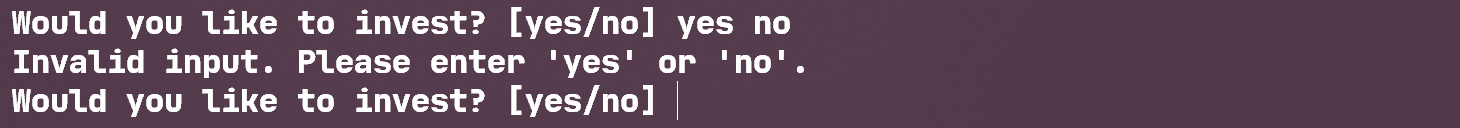
First: Using numbers



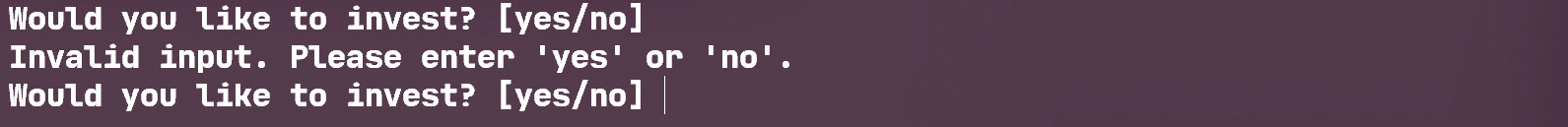
Second: Using special characters.



Third: Providing both yes and no.

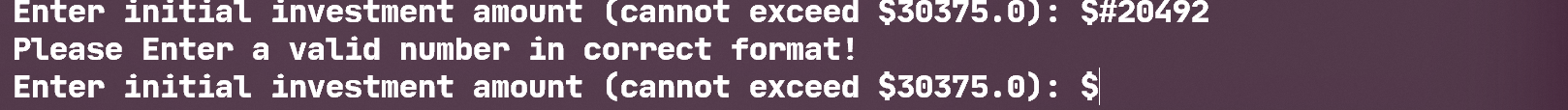


Fourth: Using an empty input.

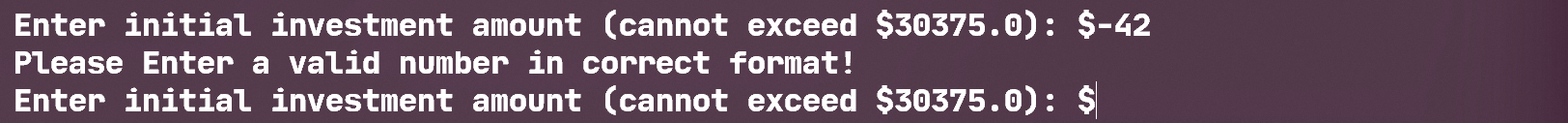


#### 7. Input for Initial Investment Amount

First: Using special characters.



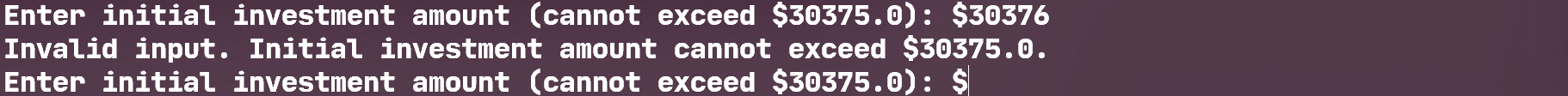
Second: Using numbers other than a positive value.



Third: Using an empty input.

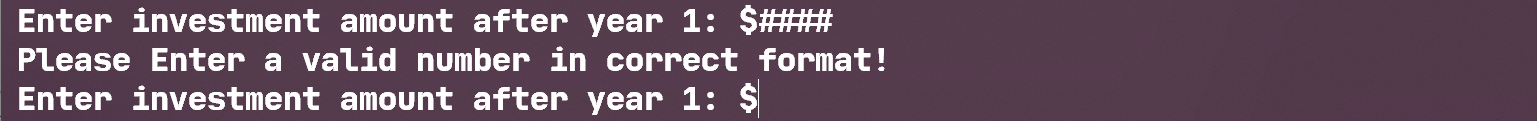


Fourth: Using a value greater than the profit the user made after taxation.



#### 8. Input for Second Year

First: Using special characters.



Second: Using numbers less than zero.



Third: Using an empty input.

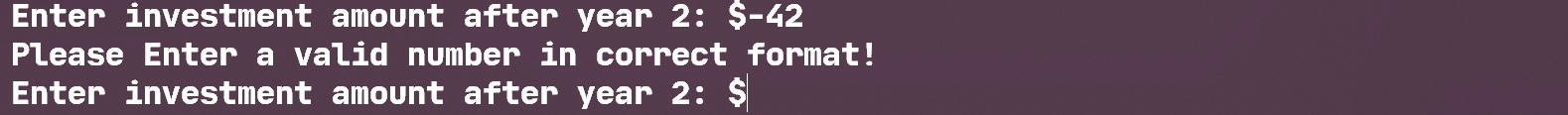


#### 9. Input for Third Year

First: Using special characters.



Second: Using numbers other than a positive value.



Third: Using an empty input.

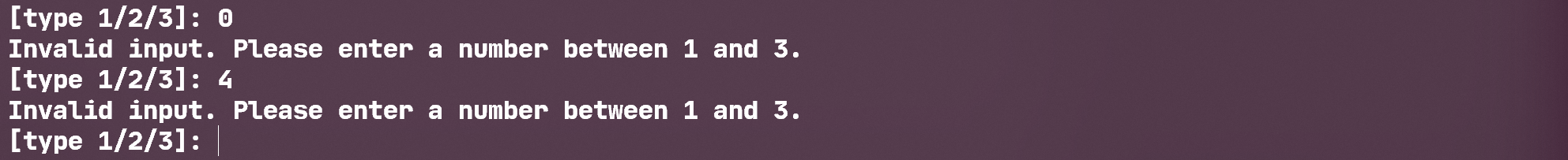


#### 10. Input for Coin Selection

First: Using special characters.



Second: Using numbers outside the range of 1 to 3.



Third: Using an empty input.



## Conclusion (DONE TILL HERE)

In conclusion, the program developed for the SENG 1110 Programming Assignment 2 fulfills the objective of managing user information and investments effectively. By implementing a structured design and functionalities, the system provides users with a comprehensive platform for investment management.

The integration of error handling mechanisms and input validation routines enhances the reliability and integrity of the system, minimizing the occurrence of errors and ensuring smooth program execution. The program offers a user-friendly interface, allowing users to seamlessly interact with the system and perform various actions such as adding, deleting, and displaying users and investments. Additionally, the capability to save data to external files enables data persistence and facilitates seamless data management across program executions.

Testing the program with valid inputs demonstrated its capability to process user data accurately. For instance, when provided with realistic inputs such as purchase price, selling price, and the duration of investment, the program consistently computed CGT percentages and actual profits after taxation in alignment with real-world calculations. This verification against known CGT formulas bolstered confidence in the program's accuracy and reliability.

Overall, the design and implementation of the program demonstrates a systematic approach to software development, emphasizing functionality, reliability, and user experience. By meeting the specified requirements and addressing the needs of users effectively, the program serves as a valuable tool for investment management, empowering users to make informed decisions and manage their investments efficiently.

## Appendix

### Appendix A : Sample Code Snippets

Algorithm for Retrieving and Validating Input String.

1. /\*\*
2. \* Retrieves and validates input from the user.
3. \*
4. \* @param prompt         The message prompting the user for input.
5. \* @param console        The Scanner object used for input.
6. \* @param isName         Boolean value whether it's name or not.
7. \* @param regex          The regular expression used to validate the input.
8. \* @param invalidMessage The message displayed when the input does not match the
9. \*                       Regex.
10. \* @return The validated input from the user.
11. \*/
12. **private** String getValidatedInput(String prompt, Scanner console, **boolean** isName, String regex,
13. String invalidMessage) {
14. String input;
15. **do** {
16. System.out.print(prompt);
17. **if** (isName)
18. input = console.nextLine();
19. **else**
20. input = console.nextLine().toLowerCase();
21. **if** (!input.matches(regex)) {
22. System.out.println(invalidMessage);
23. }
24. } **while** (!input.matches(regex));
25. **return** input;
26. }

Algorithm for Retrieving and Validating Number.

1. /\*\*
2. \* Retrieves and validates numerical input from the user.
3. \*
4. \* @param minimum              The minimum allowed value for the input.
5. \* @param maximum              The maximum allowed value for the input. (put -1
6. \*                             to
7. \*                             ignore maximum value check)
8. \* @param acceptEqualToMinimum Boolean value whether going to accept equal
9. \*                             amount to minimum amount.
10. \* @param prompt               The message prompting the user for input.
11. \* @param console              The Scanner object used for input.
12. \* @param regex                The regular expression used to validate the
13. \*                             input.
14. \* @param invalidMessage       The message displayed when the input is invalid.
15. \* @return The validated numerical input from the user.
16. \*/
17. **private** **double** getValidatedNumInput(**double** minimum, **double** maximum, **boolean** acceptEqualToMinimum, String prompt,
18. Scanner console, String regex,
19. String invalidMessage) {
20. **double** value = -1; //Setting value to -1 to pass the minimum value check, also compiler check, never set it to 0,
21. String input;
22. **do** {
23. System.out.print(prompt);
24. input = console.nextLine();
25. **if** (!input.matches(regex)) {
26. System.out.println("Please Enter a valid number in correct format!");
27. **continue**;
28. }
29. value = Double.parseDouble(input);
30. **if** (value <= minimum && !acceptEqualToMinimum || value < minimum && acceptEqualToMinimum
31. || (maximum != -1 && value > maximum)) {
32. System.out.println(invalidMessage);
33. }
34. } **while** (value <= minimum && !acceptEqualToMinimum || value < minimum && acceptEqualToMinimum
35. || (maximum != -1 && value > maximum));
36. **return** value;
37. }

The yearly Profit and Total Profit calculation is hard coded since we only need three subsequent years.

1. /\*\*
2. \* Calculates the predicted profits for the investment based on the selected
3. \* coin and deposits.
4. \*/
5. **public** **void** calculateInvestment() {
6. /\*
7. \* Years Yearly profit Total Profit
8. \* 1 | $500\* 0.15 = $75 | $75
9. \* 2 | ($500 + $1000) \* 0.15 = $225 | $75 + $225 = $300
10. \* 3 | ($500 + $1000 + $500) \* 0.15 = $300 | $75 + $225 + $300 = $600
11. \*/
13. /\*
14. \* Predicted Profit for Investment in Fast Coin
15. \* Years | YearlyProfit | TotalProfit
16. \* \_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
17. \* 1 | $75 | $75
18. \* 2 | $225 | $300
19. \* 3 | $300 | $600
20. \*/
22. **double** predictedProfitRate;
24. predictedProfitRate = **switch** (coinSelection) {
25. **case** 1 -> 0.18;
26. **case** 2 -> 0.12;
27. **case** 3 -> 0.15;
28. **default** -> **throw** **new** IllegalStateException("Unexpected value: " + coinSelection);
29. };
31. yearOneProfit = (year1Deposit) \* predictedProfitRate;
32. yearTwoProfit = (year1Deposit + year2Deposit) \* predictedProfitRate;
33. yearThreeProfit = (year1Deposit + year2Deposit + year3Deposit) \* predictedProfitRate;
35. yearOneTotalProfit = yearOneProfit;
36. yearTwoTotalProfit = yearOneProfit + yearTwoProfit;
37. yearThreeTotalProfit = yearOneProfit + yearTwoProfit + yearThreeProfit;
38. }

Algorithm for calculating Capital Gains Tax

1. /\*\*
2. \* Calculates the capital gains tax (CGT) based on user's financial information.
3. \*/
4. **public** **void** calculateCgt() {
6. **double** profit;
7. **double** profitForCGT;
8. **double** totalAnnualIncome;
9. taxRate = 0.0;
11. // CALCULATION
12. /\*
13. \* Profit = Selling price – Buying price
14. \* Profit for CGT = Profit / Number of years cryptocurrency is held
15. \* Total Annual income = Annual salary + Profit for CGT
16. \*/
18. profit = sellingPrice - buyingPrice;
19. profitForCGT = profit / years;
20. totalAnnualIncome = annualSalary + profitForCGT;
22. /\*
23. \* Find the tax rate for Total Annual income as Tax rate.
24. \*
25. \* Tax rates – residents
26. \* $0 – $18,200 0%
27. \* $18,201 – $45,000 19%
28. \* $45,001 – $120,000 32.5%
29. \* $120,001 – $180,000 37%
30. \* Over $180,001 45%
31. \*
32. \* Tax rates – non-residents
33. \* $0 – $120,000 32.5%
34. \* $120,001 – $180,000 37%
35. \* Over $180,001 45%
36. \*/
37. **if** (resident) {
38. **if** (totalAnnualIncome >= 0 && totalAnnualIncome <= 18200) {
39. taxRate = 0;
40. } **else** **if** (totalAnnualIncome >= 18201 && totalAnnualIncome <= 45000) {
41. taxRate = 0.19;
42. } **else** **if** (totalAnnualIncome >= 45001 && totalAnnualIncome <= 120000) {
43. taxRate = 0.325;
44. } **else** **if** (totalAnnualIncome >= 120001 && totalAnnualIncome <= 180000) {
45. taxRate = 0.37;
46. } **else** **if** (totalAnnualIncome >= 180001) {
47. taxRate = 0.45;
48. }
49. } **else** {
50. **if** (totalAnnualIncome >= 0 && totalAnnualIncome <= 120000) {
51. taxRate = 0.325;
52. } **else** **if** (totalAnnualIncome >= 120001 && totalAnnualIncome <= 180000) {
53. taxRate = 0.37;
54. } **else** **if** (totalAnnualIncome >= 180001) {
55. taxRate = 0.45;
56. }
58. }
59. /\*
60. \* Calculation
61. \* CGT = Tax rate \* Profit for CGT
62. \* Actual Profit = Profit for CGT - CGT
63. \*/
64. cgt = taxRate \* profitForCGT;
65. actualProfit = profitForCGT - cgt;
66. }