20.11 Program 6: Stock Application

Objectives

* Abstraction using Interfaces and Abstract Classes
* Polymorphism with Overloading
* Inheritance by extending classes and implementing interfaces
* Code in an IDE and upload to grading program
* Use documentation and code structure according to class guidelines

Program Description

In this programming assignment, you will be programming an application that is representative of the basic model of a Stock Broker application. This would help you understand the major OOP concepts like Abstraction, Inheritance and Polymorphism. Assume that all accounts are in USD$.

StockAccount

StockAccount.java will be an interface that will define the following methods:

* setName(String): returns void
* getName( ): returns String
* setCategory(String): returns void
* getCategory( ): returns String
* getID( ): returns int
* getStocksOwned( ): returns ArrayList< Stock >
* addStock(Stock): returns boolean
* removeStock(Stock): returns boolean

Institutional

Institutional.java will be a concrete class implementing the StockAccount interface. The following variables will be declared:

* ID of type int
* manager of type String
* name of type String (i.e. name of institution)
* category of type String
* stocksOwned of type ArrayList

The following methods will be implemented in the class:

* Institutional( ) constructor
  + Initialize ID to StockApp.totalAccounts variable (you must also increment the StockApp.totalAccounts variable by 1)
  + Initialize manager to {"Manager" + ID}
  + Initialize name to {"Institutional" + ID}
  + Initialize category to "Institutional"
  + Initialize the ArrayList< Stock > stocksOwned
* Institutional(String manager, String name) constructor
  + Initialize ID to StockApp.totalAccounts variable
  + Initialize manager to parameter manager
  + Initialize name to parameter name
  + Initialize category to "Institutional"
  + Initialize the ArrayList< Stock > stocksOwned
* addStock(Stock): returns boolean
  + If the Stock already exists, return false.
  + Adds the given Stock to the ArrayList and returns true, if successful.
* removeStock(Stock): returns boolean
  + Removes the given Stock only if the numShares are 0 and returns true.
  + If numShares > 0 or Stock does not exist, return false

Retail

Retail.java will be a concrete class implementing the StockAccount interface. The following variables will be declared:

* ID of type int
* name of type String
* category of type String
* stocksOwned of type ArrayList

The following methods will be implemented in the class:

* Retail( ) constructor
  + Initialize ID to StockApp.totalAccounts variable (you must also increment the StockApp.totalAccounts variable by 1)
  + Initialize name to {"Retail" + ID}
  + Initialize category to "Retail"
  + Initialize the ArrayList stocksOwned
* Retail(String name) constructor
  + Initialize ID to StockApp.totalAccounts variable
  + Initialize name to parameter name
  + Initialize category to "Retail"
  + Initialize the ArrayList stocksOwned
* addStock(Stock): returns boolean
  + If total number of Stock (not numShares) are more than 10 or Stock already exists, return false
  + Adds the given Stock to the ArrayList and returns true, if successful.
* removeStock(Stock): returns boolean
  + Removes the given Stock only if the numShares are 0 and returns true.
  + If numShares > 0 or Stock does not exist, return false

Stock

You have already worked on Stock.java in Program 3a. I have provided the solution as a template. I have added one more method, which was not in the prompt for Program 3a.

Financials

You have already worked on Financials.java in Program 3a. I have provided the solution as a template.

StockApp

The StockApp.java will be the main driver for this programming assignment. The following variables will be declared in your class:

* public static int totalAccounts = 0
* accounts of type ArrayList< StockAccount >

In your main( ) method, follow these instructions:

* Print: "Welcome to Stock Broker App"
* The menu for this app is provided in the template for StockApp. You can use it by calling the method printMenu( ).

**Option 1**

* Prompt the user for the following

Select the account type:

1. Institutional

2. Retail

* If user selects one, prompt the user for Institution's name and the manager's name. Create a new object and add the newly created object in the ArrayList accounts.

Enter the institution's name:

Enter the manager's name:

* If user enters 2, prompt the user for their name. Create a new object and add the newly created object in the ArrayList accounts.

Enter your name:

* If invalid option, print the following and return to main menu.

Invalid. Returning to main menu.

**Option 2**

* The user has to provide an ID for an account to be deleted. Prompt the user for the ID and remove the account if it exists in the ArrayList.

Enter your account ID:

**Option 3**

* Prompt the user for the account ID.

Enter your account ID:

* If account ID is invalid, print the following and return to main menu

No account with given ID exists.

* If account ID exists, prompt the user for the following and add the newly created Stock object using the addStock( Stock ) method of the Account object.

Enter the name:

Enter the ticker:

Enter number of shares:

Enter current value:

* If stock was added successful (detemined by the return value of addStock( Stock )), print the following

Stock added successfully.

* If it was unsuccessful, this indicates that the newly created Stock was a duplicate and print the following

Stock already exists.

**Option 4**

* Prompt the user for the account ID.

Enter your account ID:

* If account ID is invalid, print the following and return to main menu

No account with given ID exists.

* Prompt the user for the ticker of the Stock. Use the removeStock(Stock) to remove the Stock from the account. To find if the Stock exist, you will need to use loop through the stocksOwned ArrayList and compare the ticker.
  + If the user entered ticker exists, remove the stock from the account (do note Stock can only be deleted if shares are 0).
  + If the ticker does not exist, print the following

Ticker does not exist

**Option 5**

* Prompt the user for the account ID.

Enter your account ID:

* If account ID is invalid, print the following and return to main menu

No account with given ID exists.

- Prompt the user for the ticker of the Stock. If the ticker does not exist, print

Ticker does not exist

* If ticker exists, prompt the user for either buying or selling the shares.

1. Buy shares

2. Sell shares

* If user enters 1, prompt the user for number of shares to buy and use the buyShares( int ) method.

Enter number of shares to buy:

* If user enters 2, prompt the user for number of shares to sell and use the sellShares( int ) method.

Enter number of shares to sell:

**Option 0** - Print and exit

Thanks for using Stock App. Exiting!

* If any other option is entered, print and return to the main menu.

Invalid option entered.

Turning In Procedure

* You are required to submit the all the .java files in Zybooks which will be autograded.
* The automatic grading program is very specific. If you feel you have the correct solution but are not receiving full credit, please
  + Carefully review the output -- you might need to scroll all the way to the right to find what is wrong with a particular output.
  + Verify you have the correct names for the program itself and all methods.
  + Check your calculations by hand: was there a logic error?
  + Review the requirements: did you miss a step? misinterpret a requirement?
  + If all these check out, contact the T.A. for assistance.