Test plan for InvestmentFIrm

public static void defineSector(String sectorName):-

Input Validation Tests

- Null Sector name
- Empty sector name

Boundary Cases

- Duplicate sector name given
- Adding a sector name with maximum length given

Control Flow

- Adding the sectors multiple times with the same name
- Adding the sectors irrespective of the case sensitivity
- Adding a sectors with leading spaces
- Adding a sectors which consists special characters
- Adding first time sector name in table when cash sector is not present
- Adding after cash sector added in the sectors.

Data Flow Tests

- Call this function twice in a row.
- Call this function after adding a stock
- Call this function before doing the transactions
- Call this function after doing the transactions
- Call this function after and before creating new account
- Call this function after and before defining profiles
- Call this function after and before adding clients and advisors

public static void defineStock(String companyName, String stockSymbol, String sector) :-

Input Validation

- Ensure companyName, stockSymbol, and sector are not null or empty.

Boundary Cases

- Add the stock with parameter passed of its maximum length.
- Add a stock which is already present in the database.
- Try to add a stock which consists trailing and leading spaces

Control Flow

- Try to add a stock multiple times which has a same company name
- Try to add a stock for which sector is not defined
- Try to add a stock same stock multiple times
- Try to add a multiple stocks for the same company
- Try to add a stock multiple times which has a different company name
- Try to add a stock for same or different sectors.
- Try to add default price of stock 1 while defining.

Data Flow

- Call this function before defining sectors
- Call this function twice in a row
- Call this function after and before creating acc
- Call this function after and before creating profiles
- Call this function before and after disbursing dividend
- Call this function after and before performing transactions.
- Call this function after and before setting advisors and clients name

public static void setStockPrice(String stockSymbol, double perSharePrice) :-

Input Validation

- Check if stockSymbol is null or empty
- Check if given per share price is a negative number

Boundary Cases

- Try to add a stock's price with maximum number
- Try to add a stock price 0
- Try to add a price for which stock does not exists

Control Flow

- Try to set the price for same stock for which price is already defined
- Try to set the price for the same stock multiple times
- Try to set the price for different stock multiple times
- Try to set the same price for stock for which the same price exists for the stock
- Try to set price for a stock for which stock does not exists

Data Flow

- Call before defining the stocks
- Call after defining the stocks
- Call before and after performing the transactions
- Call before and after creating the acc
- Call the functions twice in a row
- Call this function before and after defining the profiles
- Call this function before and after defining the sectors

public static void defineProfile(String profileName, Map<String, Integer> sectorHoldings) :-

Input Validation

- Check if profile name is empty or null
- Check id sectorHoldings map is null or empty

Boundary Cases

- Test with the maximum length of profileName is given
- Check if one sector for profile consists of 100 percent holding
- Check if one sector in sector holding for profile consists of 0 percent holding
- Check if holding is negative number

Control Flow

- Try to add a profile name which is already defined
- Try to add a sector holdings for profile name for which sector does not exists
- Try to add sector holdings whose total percent is not a 100 percent
- Try to add a sector holding for a profile in which cash sector is not available
- Try to add a sector holdings for a profile in which cash sector is present

Data Flow

- Call this function prior to defining sectors
- Call this function after defining sectors
- Call this function before and after tradings
- Call this function before and after creating account
- Call this function before and after setting clients and advisors
- Call this functions twice in the row
- Call this functions multiple times for the same sectors holdings

int addAdvisor(String advisorName)

Input Validation

- Check if the advisor name is null
- Check if the advisor name is empty

Boundary Cases

- Check if advisor name consist of leading or trailing spaces
- Check if advisor name of maximum length character

Control Flow

- Try to add the advisor name for the same advisor name exists
- Try to add the advisor which does not exists in the database

Data Flow

- Calling this function twice in a row
- Calling this function irrespective of any order before after creating acc likewise

int addClient (String clientName):-

Input Validation

- Check if the client name is null
- Check if the client name is empty

Boundary Cases

- Check if client name consist of leading or trailing spaces
- Check if client name of maximum length character

Control Flow

- Try to add the client’s name for the same client name exists
- Try to add the client which does not exists in the database

Data Flow

- Calling this function twice in a row
- Calling this function irrespective of any order before after creating acc likewise

int createAccount(int clientId, int financialAdvisor, String accountName, String profileType, boolean reinvest):-

Input Validations:

- Check if account name and profile type is not null and empty
- Check if provided clientId and financial advisor is valid integer

Boundary cases :

- Test with maximum account name length
- Test with creating the same account for same client

Control flow :

- Try to create a account with same advisorid
- Try to create a account with different advisorid
- Try to create a account with same clientid
- Try to create a account with different clientid
- Try to create a account with combination of same clientid and account name
- Try to create a account with different combination of same clientid and account name
- Try to create a account for the same and different profiletype
- Try to create a account for the same and different accoutName
- Try to create a account for which client or advisor does not exists

Data flow :

- Call this function before and after clients and advisors are defined
- Call this function twice in a row with same parameter

- Call this function multiple times with different parameters
- Call this functions after trading or before trading the shares
- Call this functions before and after defining the sectors and stocks and for profile as well

tradeShares(int account, String stockSymbol, int sharesExchanged) :-

Input Validations :

- Check if the account passed is a valid integer.
- Check if the stockSymbol is a null or empty

Boundary cases :-

- Check if the stockSymbol is a cash
- Check if the shares exchanged is a positive integer
- Check if the shares exchanged is negative number

Control flow :

- Try to trade the shares for which stockSymbol is cash and then sharesexchanged passed negative
- Try to trade the shares for which account does not exists
- Try to trade the shares for which stock symbol does not present in the stock table
- Try to buy shares for the account if shares exchanged is a positive number
- Try to sell the shares for the account if shares exchanged is a negative number
- Try to update the cash balance for the account after or before trading the shares
- Try to buy shares for the first time for account before adding a cash amount for that account

Data flow:-

- Call this function multiple times
- Call this functions for the account which does not have a sufficient balance
- Call this function before and after creating the account

- Call this function before and after defining stocks
- Call this function immediately after setting or before setting new price of the given stock symbol
- Call this functions after and before disbursing the dividend
- Call this functions before and after adding new clients and advisors and for profiles as well.

changeAdvisor(int accountId, int newAdvisorId) :-

Input Validations :-

- Check account id and newAdvisorId is a valid integer.

Boundary casas:-

- Try to change the advisorId for the account for which advisor not present
- Try to change the advisor with the same advisor passed for this account

Control flow : -

- Try to change the advisor for the account for which account does not exists
- Try to change the advisor that is present in the advisors table for the account that is present in the accounts table

Data Flow :-

- Call this function twice
- Call this function before adding the advisor
- Call this function after adding the advisor
- Call this function before creating the account
- Call this function after creating the account

Reporting on the system

double accountValue(int accountId) : -

Input validations:

- Check if the account id is a valid integer

Boundary cases:

- Test with an account having a cash balance of $0.
- Test with an account having a large cash balance.
- Test with an account having no stocks.
- Test with an account having stocks with various share quantities and prices.

Control flow:

- Try to fetch the account value for the account id that does not exits
- Try to fetch the account value for the account id that exists
- Try to fetch the account value for the account that has a some balance and stocks in it

Data Flow:

- Call this function after setting the new price for the stocks that account hold
- Call this functions before setting the new price
- Call this function before creating the account
- Call this function after creating the account.
- Call this function after and before trading some shares

double advisorPortfolioValue(int advisorId) :-

Input Validations :-

- Check if the passed advisorId is a valid integer

Boundary cases :-

- Test with an advisor managing no accounts.

- Test with an advisor managing multiple accounts.

Control flow :-

- Try to get the portfolio value for a advisor that is not present in the advisors table
- Try to get the portfolio value for the valid advisor that is present and holds the valid accounts

Data flow:

- Call this function before and after the creating the account
- Call this function twice in a row
- Call this function before and after setting the advisors
- Call this function before and after accountValues
- Call this functions after and before trading some shares

Map<Integer, Double> investorProfit(int clientId):

Input validations :

- Check if the clientId is a valid integer

Boundary cases :

- Test with a client having no accounts.
- Test with a client having multiple accounts with no stocks.
- Test with a client having multiple accounts with stocks and various profit scenarios

Control flow :

- Try to get the investorProfit for a client that does not exists
- Try to get the investorProfit for a client that exists
- Try to get the profits for a client holding no accounts
- Try to get the profits for the account which has same average cost per share
- Try to get the profits for the account after setting new price for the stocks
- Try to get the profits for the account that does not have any stocks

Data Flow:

- Call this function twice in a row
- Call this function after setting new price
- Call this function before setting the new price
- Call this function after and before creating the accounts
- Call this functions after and before setting the clients
- Call this functions after and before trading some shares for the accounts.

Map profileSectorWeights(int accountId) :

Input validations:

- Check if account is passed is a valid integer

Boundary cases :

- Test with an account holding no stocks.
- Test with an account holding stocks from different sectors.
- Test with an account holding only cash.

Control flow:

- Try to get the profile sectors weight that has a no cash in the account
- Try to get the profile sectors weight that has investments in the one sector
- Try to get the profile sectors weight that has a investments in all the sectors

Data flow

- Call this function twice in a row
- Call this function before and after trading shares buying or selling
- Call this function before and after immediately setting a new prices

Set divergentAccounts(int tolerance) :

Input validations:

- Check if the tolerance is a valid integer not negative

Boundary cases:

- Check if the tolerance is minimum
- Check if the tolerance is maximum

Control flow:

- Test identifying divergent accounts based on their sector weights compared to target weights with a specified tolerance successfully.

Data flow:

- Call before the profileSectorsWeights
- Call after profileSectorsWeights
- Call before setting the new price
- Call after setting the new price
- Call after trading
- Call before the trading

int disburseDividend(String stockSymbol, double dividendPerShare):

input validations :

- Check for the stockSymbol is null or empty
- Check for the dividend per share for the positive value
- Check for the dividend per share for negative value

Boundary cases:

- Test with a stock symbol that does not exist.
- Test with a stock symbol that exists but has no shareholders.
- Test with a stock symbol that has one or more shareholders.

Control flow:

- Test disbursing dividends for a given stock symbol to all accounts holding the stock successfully.
- Test the distribution of dividends based on whether accounts are set to reinvest dividends or receive them as cash.

Data flow:

- Call this function before setting the stock to the sector
- Call this function after the setting stock to the sector
- Call this function after and before setting the new stock price
- Call this function after and before the making the trading.

Map stockRecommendations(int accountId, int maxRecommendations, int numComparators):-

Input validations:

- Check if the account id is a valid integer
- Check if the maxrecommendation is positive number
- Check if the numcomparator is valid positive number

Boundary cases:

- Maximum Recommendations Reached: Check with a large value of the maximum recommendations you recommend that the function returns calculations within the specified limit range.

- Number of Comparators Exceeds Account Count: Use with comparators that exceed the number of accounts and create no errors through testing.

Control flow:

- Try to recommend the stocks for a account stock vector having all the shares non zero
- Try to recommend the stocks for a account for which has some shares but others vectors does not contain it
- Try to recommend the stocks for a account which does not have any stocks that others account has
- Test with an account having an empty stock vector and ensure it recommends actions based on other account vectors.

Data flow :

- Call this function before adding the stocks
- Call this function after adding a stocks
- Call this function after and before doing some trading

- Call this functions after and before disbursing the dividends

public Set<Set<Integer>> advisorGroups(double tolerance, int maxGroups)

Input validation

- Check if the tolerance is valid i.e. not negative
- Check if the maxGroups is valid positive integer

Boundary cases:

- Find groups if the tolerance is 0
- Find groups if tolerance is 100
- Find the groups for 0 for maxGroups
- Find the groups if maxGroups are passed as a maximum number

Control flow

- Try to get the groups when there is no trade for any advisor
- Try to get the group when the trade is maximum for the advisor
- Try to get the groups of the advisors in which one advisor is associated with almost all the accounts
- Try to get the groups in which the advisors are averagely associated with all the accounts
- Try to get the groups when there is maximum profit for the advisor
- Try to get the groups when there is least profit for the advisor
- Try to get the group of advisors when all the stock holdings of all the account is similar
- Try to get the group of advisors when the stock holding of all the account is moderately similar
- Try to get the group of advisors when the stock holding of all the account is most similar

Data flow

- Call this immediately after changing the advisor
- Call this before doing any trading
- Call this after doing trading
- Call this after defining new multiple advisors
- Call this after changing the stock price
- Call this after defining the new sectors
- Call this just after calculating the profit of the advisor
- Call this just before changing the advisor

- Call this after calling the stock recommendation
- Call this before the stock recommendations