MAD3004\_2

Introduction to Swift Programming

C0676975

Pedro Fernandes

Assignment 1

Requirements:

1. Create a class called StockHolding to represent a stock that you have purchased. It will be a subclass of NSObject. For instance variables, it will have two floats named purchaseSharePrice and currentSharePrice, one int named numberOfShares and one string named companyName. Create accessor methods for the instance variables. Create two other instance methods:

- (float)costInDollars; // purchaseSharePrice \* numberOfShares

- (float)valueInDollars; // currentSharePrice \* numberOfShares

Populate an array with three instances of StockHolding. Display all the stocks sorted by company name in alphabetical order.

***NOTE: Picture below does not show the companyName but you should implement it in your class.***

2. Create a subclass of StockHolding called ForeignStockHolding. Give ForeignStockHolding an additional instance variable: conversionRate, which will be a float. (The conversion rate is what you need to multiply the local price by to get a price in Canadian dollars. Assume the purchasePrice and currentPrice are in the local currency.) Override costInDollars and valueInDollars to do the right thing.

Add at least two instances of ForeignStockHolding to the existing array from part one.

Display all the stocks sorted by company name in reverse alphabetical order.

3. Modify the application so the user can choose how many stocks they want to enter as well as the type of the stock (maximum 8 stocks). The application should accept user data for every stock and after the data has been entered it should display the following outputs:

* Display stock information with the lowest value
* Display stock with the highest value
* Display the most profitable stock
* Display the least profitable stock
* List all stocks sorted by company name (A-Z)
* List all stocks sorted from the lowest value to the highest value

Code:

import UIKit

//StockHolding class represents a stock in the market, this class has the properties purchaseSharePrice, currentSharePrice, numberofShares, companyName.

//Each property has its own accessor methods (gets and setters).

//This class has two constructor one is empty which allows the class to be instantiated with the initial properties values and a constructor accepting all the properties as parameter.

public class StockHolding {

/\*

Private properties created and assigned default values.

\*/

private var \_purchaseSharePrice: Float = 0.0;

private var \_currentSharePrice: Float = 0.0;

private var \_numberOfShares: Int = 0;

private var \_companyName: String = "";

/\*

Getters and Setters for all the properties giving access to the private properties. Following swift pattern of getters and setters.

\*/

var purchaseSharePrice:Float {

get {

return \_purchaseSharePrice

}

set (newVal) {

self.\_purchaseSharePrice = newVal;

}

}

var currentSharePrice:Float {

get {

return \_currentSharePrice

}

set (newVal) {

self.\_currentSharePrice = newVal;

}

}

var numberOfShares:Int {

get {

return \_numberOfShares

}

set (newVal) {

self.\_numberOfShares = newVal;

}

}

var companyName:String {

get {

return \_companyName

}

set (newVal) {

self.\_companyName = newVal;

}

}

//Empty constructor giving the chance to the class be instantiated with the default properties values.

init() {

}

//Contructor that accepts parameters and set the properties value for each one.

init(purchaseSharePrice: Float, currentSharePrice: Float, numberOfShares: Int, companyName: String) {

self.purchaseSharePrice = purchaseSharePrice;

self.currentSharePrice = currentSharePrice;

self.numberOfShares = numberOfShares;

self.companyName = companyName;

}

//As the requirement, costInDollars represents the value of the stock in dollar. This function multiply the number of shares to the purchase price.

public func costInDollars() -> Float {

return self.purchaseSharePrice \* Float(self.numberOfShares)

}

//Value in dollars represents the actual value of the stock in dollar currency.

public func valueInDollars() -> Float {

return self.currentSharePrice \* Float(self.numberOfShares)

}

//Profit calculate how much the company owned buying each share of this stock. Current price less Purchased price.

public func profit() -> Float {

return (currentSharePrice - purchaseSharePrice);

}

//Print values simply print all the properties values.

public func printValues() {

print("Company Name: \(self.companyName) | Purchase Price: \(self.purchaseSharePrice) | Current Price: \(self.currentSharePrice) | Number Of Shares: \(self.numberOfShares) | Profit: \(self.profit())")

}

}

//Required: Populate an array with three instances of StockHolding. Display all the stocks sorted by company name in alphabetical order.

//Instantiate the array of StockHolding

var stockHolderArr : [StockHolding] = [];

//Create three instances of the StockHolding setting fake values to them.

var samsungStock = StockHolding(purchaseSharePrice: 100.50, currentSharePrice: 200, numberOfShares: 5, companyName: "Samsung")

var microsoftStock = StockHolding(purchaseSharePrice: 20, currentSharePrice: 15, numberOfShares: 10, companyName: "Microsoft")

var googleStock = StockHolding(purchaseSharePrice: 200, currentSharePrice: 300, numberOfShares: 50, companyName: "Google")

//Populate the created instances to the array.

stockHolderArr.append(samsungStock);

stockHolderArr.append(microsoftStock);

stockHolderArr.append(googleStock);

//sortArr is a function that accepts the array and a set of parameters to know in which way it should be sorted ASC or DESC and by which property.

func sortArr(arr: [StockHolding], shouldPrint: Bool, isAsc: Bool, byName: Bool, byProfit: Bool, byValue: Bool) {

//Sort the array by the giving parameters. Using the default swift way to sort.

stockHolderArr.sort(by: { (first: StockHolding, second: StockHolding) -> Bool in

if byName {

if isAsc {

return first.companyName < second.companyName

} else {

return first.companyName > second.companyName

}

} else if byProfit {

if isAsc {

return first.profit() < second.profit()

} else {

return first.profit() > second.profit()

}

} else if byValue {

if isAsc {

return first.costInDollars() < second.costInDollars()

} else {

return first.costInDollars() > second.costInDollars()

}

}

return false;

})

//If shouldPrint is true them loop through the array and print each object properties calling the printValues created before.

if shouldPrint {

for stock in stockHolderArr {

stock.printValues()

}

//Print a blank line just to create a space in the console.

print("");

}

}

//Display all the stock sorted by the company name ascending.

print("Stocks sorted by name ASC:")

sortArr(arr: stockHolderArr, shouldPrint: true, isAsc: true, byName: true, byProfit: false, byValue: false)

//Class ForeignStockHolding represents a foreign stock in the market and inherits from StockHolding just created above.

//By doing that it has all the properties and function from the class parent.

//Adding one property do be the conversionRate of the stock from a different currency.

public class ForeignStockHolding: StockHolding {

private var \_conversionRate: Float = 0.0;

//Create accessor method to the property

var conversionRate:Float {

get {

return \_conversionRate

}

set (newVal) {

self.\_conversionRate = newVal;

}

}

//Allow empty constructor.

override init() {

super.init();

}

//Constructor the receives the parameters and instantiate the object setting the values to the properties.

init(purchaseSharePrice: Float, currentSharePrice: Float, numberOfShares: Int, companyName: String, conversionRate: Float) {

super.init(purchaseSharePrice: purchaseSharePrice, currentSharePrice: currentSharePrice, numberOfShares: numberOfShares, companyName: companyName)

self.conversionRate = conversionRate;

}

//CostInDollars should be now multiplied by the conversionRate.

public override func costInDollars() -> Float {

return super.costInDollars() \* self.conversionRate;

}

//ValueInDollars should be now multiplied by the conversionRate.

public override func valueInDollars() -> Float {

return super.valueInDollars() \* self.conversionRate;

}

}

//Question 2)

//Instantiating two ForeignStockHolding and adding them to the already created array of StockHolding.

var alibabaStock = ForeignStockHolding(purchaseSharePrice: 300, currentSharePrice: 350, numberOfShares: 20, companyName: "Alibaba", conversionRate: 2.5)

var instagramStock = ForeignStockHolding(purchaseSharePrice: 240, currentSharePrice: 260, numberOfShares: 2, companyName: "Instagram", conversionRate: 4)

//Since ForeignStockHolding inherits from StockHolding we can use the same array.

stockHolderArr.append(alibabaStock);

stockHolderArr.append(instagramStock);

//Print the array sorted by company name Descending.

print("Stocks sorted by name DESC:")

sortArr(arr: stockHolderArr, shouldPrint: true, isAsc: false, byName: true, byProfit: false, byValue: false)

//Question 3)

//Create arrays of stock properties and loop through them to add the new stocks to the stock array.

var companyNamesArr = ["Company Brasil", "Company India", "Company Canada", "Company USA", "Company Italy"];

var purchaseSharePriceArr = [20.42, 10.52, 52.63, 32.12, 13.44]

var currentSharePriceArr = [22.44, 12.23, 56.54, 43.12, 11.24]

var numberOfSharesArr = [4,2,6,7,8]

var conversionRateArr = [3,4,1,2,3]

//looping through the arrays, instantiating a new stock and adding to the stock array.

for i in 0...companyNamesArr.count - 1 {

var name = companyNamesArr[i];

var purchasePrice = purchaseSharePriceArr[i]

var currentPrice = currentSharePriceArr[i]

var shares = numberOfSharesArr[i]

var convertion = conversionRateArr[i]

var foreignStock = ForeignStockHolding(purchaseSharePrice: Float(purchasePrice), currentSharePrice: Float(currentPrice), numberOfShares: shares, companyName: name, conversionRate: Float(convertion))

stockHolderArr.append(foreignStock)

}

//Sort the array by lowest value and print the stock with the lowest value.

print("Display stock information with the lowest value:")

sortArr(arr: stockHolderArr, shouldPrint: false, isAsc: true, byName: false, byProfit: false, byValue: true)

stockHolderArr[0].printValues()

print("")

//Print the stock with the highest value, since the array is already sorted we just need to get the last position.

print("Display stock information with the highest value:")

stockHolderArr[stockHolderArr.count - 1].printValues()

print("")

//Sort and print the most Profitable stock in the array. To do that, the array is sorted by the profit descending and the first value is used.

print("Display the most profitable stock:")

sortArr(arr: stockHolderArr, shouldPrint: false, isAsc: false, byName: false, byProfit: true, byValue: false)

stockHolderArr[0].printValues()

print("")

//Print the least profitable stock, since the array is already sorted by profit descending, just display the last position of the array.

print("Display the least profitable stock:")

sortArr(arr: stockHolderArr, shouldPrint: false, isAsc: false, byName: false, byProfit: true, byValue: false)

stockHolderArr[stockHolderArr.count - 1].printValues()

print("")

//Sort and Print all the stocks sorted by company name asceding.

print("Stocks sorted by name ASC:")

sortArr(arr: stockHolderArr, shouldPrint: true, isAsc: true, byName: true, byProfit: false, byValue: false)

//Sort and print the stock sorted from the lowest to the highest value.

print("List all stocks sorted from the lowest value to the highest value:")

sortArr(arr: stockHolderArr, shouldPrint: true, isAsc: true, byName: false, byProfit: false, byValue: true)