Introduction to Swift Programming
MAD 3004
Block

Student Name(s): ____Amrit Ghale_____________________

Student ID(s): _____________C0860727_______________

Milestone Check-In @10	Presentation @10	Specification @40	Navigation @10	Design @10	Individual @20	Total

MAD3004

(Teams of up to three students are allowed)

Final Project

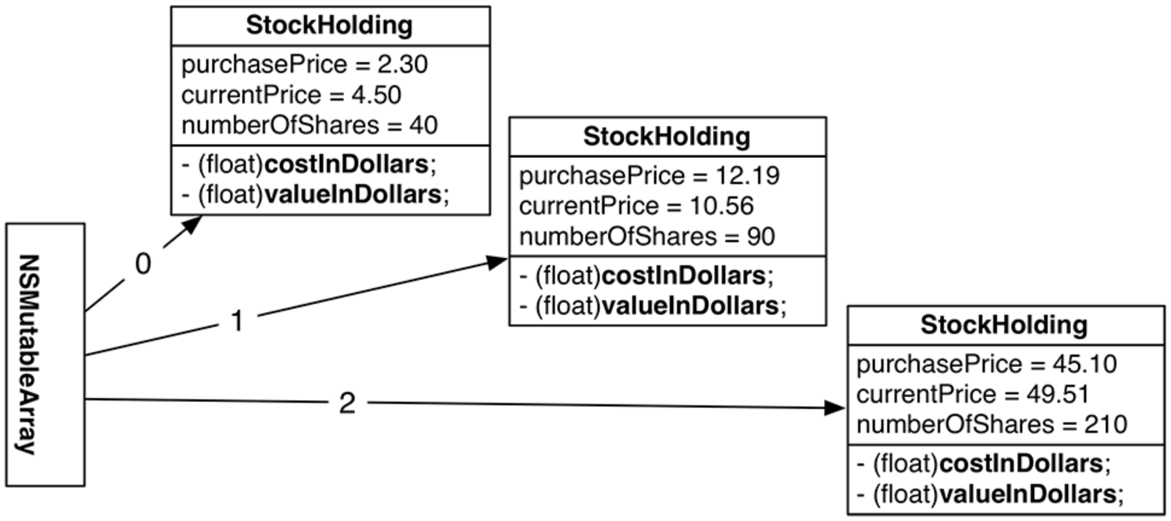
1. Create a class called StockHolding to represent a stock that you have purchased. It will be a subclass of NSObject. For instance variables, it will have two floats named purchaseSharePrice and currentSharePrice, one int named numberOfShares and one string named companyName. Create accessor methods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

- (float)costInDollars; // purchaseSharePrice * numberOfShares

- (float)valueInDollars; // currentSharePrice * numberOfShares

Populate an array with 10 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.



Source Code:

```
import Foundation

public class StockHolding :NSObject
{
    var purchaseSharePrice : Float = 0.0
    var currentSharePrice : Float  = 0.0
    var numberOfShares : Int       = 0
    var companyName : String       = ""

    public init(sharePrice: Float, currSharePrice: Float, numOfShare: Int, compName: 
String)
    {
        self.purchaseSharePrice = sharePrice
        self.currentSharePrice  = currSharePrice
        self.numberOfShares     = numOfShare
        self.companyName        = compName
    }

    public func costInDollars ()-> Float
    {
        return purchaseSharePrice * Float(numberOfShares)
    }

    public func valueInDollars ()-> Float
    {
        return currentSharePrice * Float(numberOfShares)
    }
}
```

//Helper

Source Code:

```
import SwiftUI
import Foundation

func input() -> String {
    let keyboard = FileHandle.standardInput
    let inputData = keyboard.availableData
    let rawString = NSString(data: inputData, encoding:String.Encoding.utf8.rawValue)
    if let string = rawString {
        return string.trimmingCharacters(in: NSCharacterSet.whitespacesAndNewlines)
    } else {
        return "Invalid input"
    }
}

func randomIntBetween(low:Int, high:Int) -> Int {
    let range = high - (low - 1)
    return (Int(arc4random()) % range) + (low - 1)
}
```

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.

Source Code:

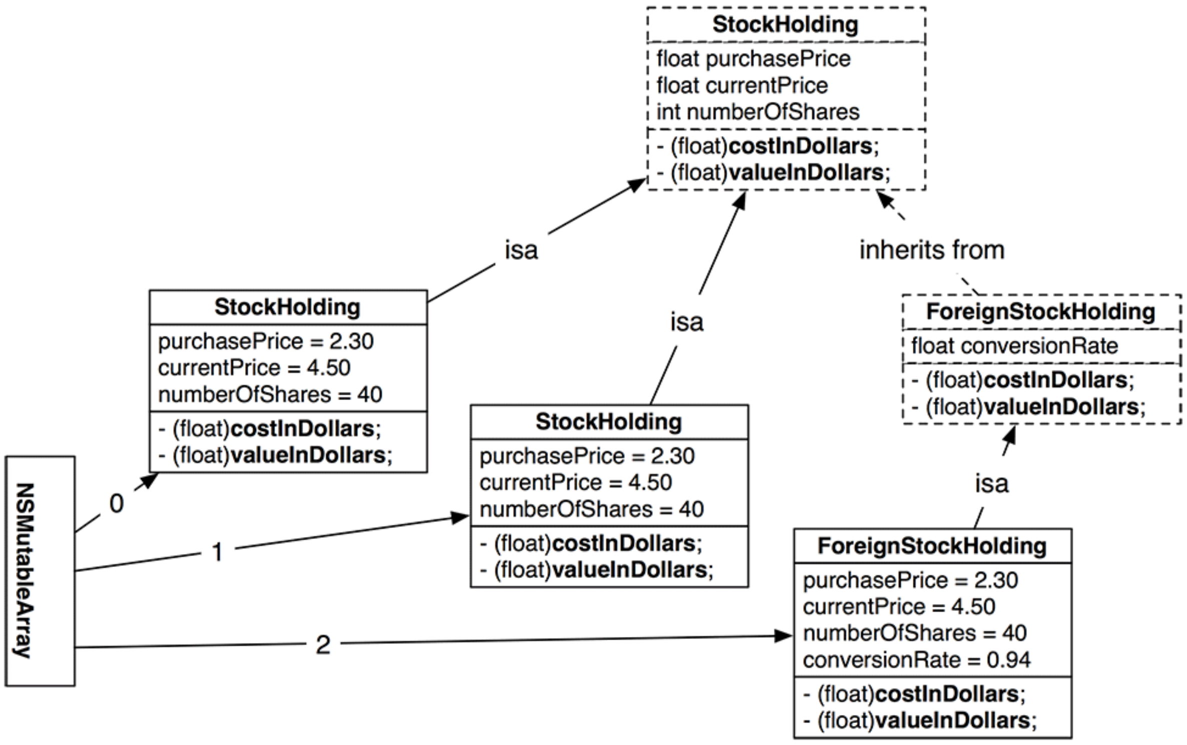
```
import Foundation

 public class ForeignStockHolding: StockHolding
{
    var conversionRate: Float = 0.0

    init(frgnSharePrice: Float, frgnCurrSharePrice: Float, frgnNumOfShare: Int,
fgrnCompName: String, conversionRate: Float)
    {
        self.conversionRate = conversionRate
        super.init(sharePrice: frgnSharePrice , currSharePrice: frgnCurrSharePrice,
numOfShare: frgnNumOfShare, compName: fgrnCompName)
    }

    override public func costInDollars ()-> Float
    {
        return purchaseSharePrice * Float(numberOfShares) * conversionRate
    }

    override public func valueInDollars ()-> Float
    {
        return currentSharePrice * Float(numberOfShares) * conversionRate
    }
}
```



3. Add the following features to the project. User should be able to choose how many stocks they want to enter as well as the type of the stock. The application should accept user data for every stock and after the data has been entered it should display the following menu:

1) Display stock information with the lowest value
2) Display stock with the highest value
3) Display the most profitable stock
4) Display the least profitable stock
5) List all stocks sorted by company name (A-Z)
6) List all stocks sorted from the lowest value to the highest value
7) Exit

All the items in the menu above should be implemented and be fully functional. After each selection from the menu, you should display the corresponding information and allow the user to go back to the main menu.

Source Code:

```
//
//  utilityfunction.swift
//  Stock
//
//  Created by Amrit Ghale on 03/06/2565 BE.
//

import SwiftUI
import Foundation

public var formater = NumberFormatter()
public let localStocks = [1:"Petro Canada",2:"ESSO", 3:"Ryal Bank",4:"Bank of
Montreal",
                          5:"CIBC Bank",6:"No frills",7:"Pizza
Pizza",8:"Subway",9:"Rogers"]
public let forgnStocks = [1:"Samsung", 2:"Air Canada", 3:"Apple", 4:"Google"]

func numberFormat(number: Float) -> String
{
    return formatter.string(from: number as NSNumber)!
}

public func addSpaces(width: Int, str: String, var1: Int)-> String
{
    //var1 is to give an additional feature to have fixed or varaible spaces
    //0- fix width
    //1- add spaces in front

    if var1 == 0
    {
        _ = width - str.count
        let padding = str.padding(toLength: 20, withPad: " ", startingAt: 0)
//          let padding = String(repeating: spaces, count: Character (" "))
        return  padding + str
    }

    else
    {
        return  String(repeating :Character (" "), count: width) + str
    }
}

public func showMenu()
{
print("\nChose option from (1-9) below :")
print("1 : Add Stock")
print("2 : Remove Stock")
print("3 : Display Stock with lowest value")
print("4 : Display Stock with highest Value")
print("5 : Display most profitable Stock")
print("6 : Display least profitable Stock")
print("7 : Display stocks from A-Z")
print("8 : Display stocks lowest to highest Value")
print("9 : Exit")
}
```

```
public func options()->Bool
{
    if (option >= 1 && option <= 9)
    {
        return true
    }
    else{
        print ("Incorrect option please try again !!")
        return false
    }
}

public func displayLocalStocks()
{
    print ("Local Stocks :")
    for i in 1...localStocks.count
    {

        print ("\(i) : \(localStocks[i]!)")
    }
}

public func displayForgnStocks()
{
    print ("Foriegn Stocks :")
    for i in 1...forgnStocks.count
    {

        print ("\(i) : \(forgnStocks[i]!)")
    }
}

public func validOptionFloat(start: Float)-> Float
{
    var flag = Float(input())
    if (flag ?? 0.0 <= start )
    {
        flag = nil
    }
    while (flag == nil)
    {
        print("Incorrect Value entered!! Please enter integer greater than 0")
        flag = Float(input())
        if (flag ?? 0.0 <= start)
        {
            flag = nil
        }
    }
    return flag!
}

public func validOptionString(start: Int,end: Int)->Int
{
    var flag = Int(input())
    if (flag ?? Int(1.1) < start || flag ?? Int(0.0) > end)
    {
        flag = nil
    }
        while (flag == nil)
        {
            print("Incorrect Value entered!! Please enter Integer values from
\(start)-\(end)")
            flag = Int(input())
            if (flag ?? Int(1.1) < start || flag ?? Int(0.0) > end)
            {
                flag = nil
            }
        }
        return flag!
}

public func displayIndividualStock (index : Int)
{
    print("Sr.No"
          ,addSpaces(width: 17, str: "Number of Shares",var1: 0)
          ,addSpaces(width: 17, str: "Purchase Price",var1: 0)
          ,addSpaces(width: 17, str: "Current Price",var1: 0)
          ,addSpaces(width: 17, str: "Cost in Dollars",var1: 0)
          ,addSpaces(width: 17, str: "Value in Dollars",var1: 0)
          ,addSpaces(width: 17, str: "Company name",var1: 0)
          ,addSpaces(width: 17, str: "Share Type",var1: 0) )

    print(String (repeating: Character("="), count :132))
    if (stocklist.isEmpty)
    {
        print("No Stocks available !")
    }
    else
    {
        print (String(format: "%5d%18d%@%@%@%@%@%@",
                (1),
                stocklist[index].numberOfShares,
                      addSpaces(width: 18, str: numberFormat(number:
stocklist[index].purchaseSharePrice),var1: 0),
                      addSpaces(width: 18, str: numberFormat(number:
stocklist[index].currentSharePrice),var1: 0),
                      addSpaces(width: 18, str: numberFormat(number:
stocklist[index].costInDollars()),var1: 0),
                      addSpaces(width: 18, str: numberFormat(number:
stocklist[index].valueInDollars()),var1: 0),
                      addSpaces(width: 18, str: stocklist[index].companyName,var1: 0),
                      addSpaces(width: 8, str: objectOf(stocklist1:
stocklist[index]),var1: 1)
                ))
      }
}

public func diplaystocks()
{
    print("Sr.No"
          ,addSpaces(width: 17, str: "Number of Shares",var1: 0)
          ,addSpaces(width: 17, str: "Purchase Price",var1: 0)
          ,addSpaces(width: 17, str: "Current Price",var1: 0)
          ,addSpaces(width: 17, str: "Cost in Dollars",var1: 0)
          ,addSpaces(width: 17, str: "Value in Dollars",var1: 0)
        ,addSpaces(width: 17, str: "Company name",var1: 0)
        ,addSpaces(width: 17, str: "Share Type",var1: 0) )

    print(String (repeating: Character("="), count :132))
    if (stocklist.isEmpty)
    {
     print("No Stocks available !")
    }
    else
    {
        for count in 0...stocklist.count-1
        {
            print (String(format: "%5d%18d%@%@%@%@%@%@",
                (count+1),
                stocklist[count].numberOfShares,
                          addSpaces(width: 18, str: numberFormat(number:
stocklist[count].purchaseSharePrice),var1: 0),
                          addSpaces(width: 18, str: numberFormat(number:
stocklist[count].currentSharePrice),var1: 0),
                          addSpaces(width: 18, str: numberFormat(number:
stocklist[count].costInDollars()),var1: 0),
                          addSpaces(width: 18, str: numberFormat(number:
stocklist[count].valueInDollars()),var1: 0),
                          addSpaces(width: 18, str: stocklist[count].companyName,var1:
0),
                          addSpaces(width: 8, str: objectOf(stocklist1:
stocklist[count]),var1: 1)
                ))
        }
    }
}
```

Marking Scheme:

Marking of the project will be done according to the following scheme:

Milestone Check-In [10%]: This includes the short presentation halfway through the project to check how the group is performing and the progress of the project.

Presentation [10%]: This includes the group presentation of the project. Individual marks will be awarded for this rubric. Presentation should be professional and without any glitches.

Specification [40%]: This includes all the requirements specified above. Project will be considered for complete marks only if all the specifications are properly implemented. Partial implementation is not awarded any marks.

Navigation [10%]: This rubric includes marks for user interface interaction component (even though this is a command line application, navigation between menus should be implemented properly). The easier it is for the user to navigate the app the higher the mark is. This is somewhat subjective aspect of awarding a mark but for this project, good design practices covered in class will be used.

Design [10%]: This rubric focuses on overall design of the project, which includes ease of application, proper file organization used for the app, classes and class members properly created, proper naming convention used throughout the project.

Individual [20%] – This mark is awarded to each member of the group individually based on their presentation and involvement in the project. Students should be prepared to answer questions by the professor.