# Stock Market

*As a yang investor, you decide to create software, which will help you to keep track of stocks you own.*

## Preparation

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

## Problem description

Your task is to create an investor with a portfolio of different stocks.

### Stock

You’ve been given a C# **class**, called **Stock** with properties:

* **CompanyName: string**
* **Director: string**
* **PricePerShare: decimal**
* **TotalNumberOfShares: int**
* **MarketCapitalization: decimal**

The **constructor**of **the Stock** class should receive **the CompanyName, Director, PricePerShare, and TotalNumberOfShares.  MarketCapitalization** is a calculated property between **PricePerShare** and **TotalNumberOfShares**.

The class should also have the following methods:

* Override **ToString()** method in the format:

**"****Company: {CompanyName}**

**Director: {Director}**

**Price per share: ${PricePerShare}**

**Market capitalization: ${MarketCapitalization}"**

### Investor

The **Investor**class has a **collection**(**portfolio**) of type **Stock** with corresponding **unique** **Company Name**of a **Stock**. The name of the collection should be **Portfolio**. All the entities of the **Portfolio** collection have the **same** properties. The **Investor** has also some additional properties:

* **FullName: string**
* **EmailAddress: string**
* **MoneyToInvest: decimal**
* **BrokerName: string**

The **constructor** of the **Investor** class should receive the **FullName, EmailAddress, MoneyToInvest,** and **BrokerName**.

Implement the coming features:

* Getter **Count** - returns the count of the currently owned stocks.
* Method **void** **BuyStock(Stock stock)** – add a single stock of the given company **if** the stock’s market capitalization is **bigger than $10000** and the **investor has enough money**. Then reduce the **MoneyToInvest**by the price of the stock. If a stock cannot be bought the method should not do anything.
* Method **string** **SellStock(string companyName, decimal sellPrice)** - sell a Stock from the portfolio with the given **company name for the given price**:
  + - * If the company does not exist return: **"{companyName} does not exist."**
      * If the selling price is smaller than the price per **share** return: **"Cannot sell {companyName}."**
      * Upon successful sell, you should remove the company from the portfolio, increase Money to Invest with the selling price, and return: **"{companyName} was sold."**
* Method **Stock** **FindStock(string companyName)** - returns a **Stock** with the given company name. If it doesn't exist, return **null**.
* Method **Stock** **FindBiggestCompany() –** returns the **Stock** with the biggest market capitalization. If there are no stocks in the portfolio, the method should return null.
* Method **string** **InvestorInformation()** - returns information about the Investor and his portfolio in the following format:

**"The investor {fullName} with a broker {brokerName} has stocks:**

**{Stock1}**

**{Stock2}**

**… "**

## Constraints

* Only a single stock of a company could be bought.
* The company name of each Stock in the portfolio will always be unique.
* The PricePerShare of a Stock and the MoneyToInvest of the Investor will always be positive numbers.
* There will not be a case where two Stock has the same CompanyName.
* You will always be given Stock added before receiving the method for its manipulation.

## Examples

|  |
| --- |
| Sample code usage |
| // Sample Code Usage:  // Initialize Investor  Investor investor = new Investor("Peter Lynch", "p.lynch@gmail.com", 2000m, "Fidelity");  // Initialize Stock  Stock ibmStock = new Stock("IBM", "Arvind Krishna", 138.50m, 5000);  // Print a stock  Console.WriteLine(ibmStock.ToString());  // Company: IBM  // Director: Arvind Krishna  // Price per share: $138.50  // Market capitalization: $692500.00  // Buy a stock  investor.BuyStock(ibmStock);  // Sell a stock  Console.WriteLine(investor.SellStock("IBM", 150.00m)); // "IBM was sold."  // Add stocks  Stock teslaStock = new Stock("Tesla", "Elon Musk", 743.17m, 6520);  Stock amazonStock = new Stock("Amazon", "Jeff Bezos", 3457.17m, 8500);  Stock twitterStock = new Stock("Twitter", "Jack Dorsey", 59.66m, 11200);  Stock blizzardStock = new Stock("Activision Blizzard", "Bobby Kotick", 78.53m, 5520);  // Buy more stocks  investor.BuyStock(teslaStock);  investor.BuyStock(amazonStock);  investor.BuyStock(twitterStock);  investor.BuyStock(blizzardStock);  // FindBiggestCompany  Console.WriteLine(investor.FindBiggestCompany());  // Company: Tesla  // Director: Elon Musk  // Price per share: $743.17  // Market capitalization: $4845468.40  // Print investor information  Console.WriteLine(investor.InvestorInformation());  // The investor Peter Lynch with a broker Fidelity has stocks:  // Company: Tesla  // Director: Elon Musk  // Price per share: $743.17  // Market capitalization: $4845468.40  // Company: Twitter  // Director: Jack Dorsey  // Price per share: $59.66  // Market capitalization: $668192.00  // Company: Activision Blizzard  // Director: Bobby Kotick  // Price per share: $78.53  // Market capitalization: $433485.60 |

## Submission

Zip all the files in the project folder except the **bin** and **obj** folders.