### Requirements

1. Provide working source code that will :-
   1. For a given stock,
      1. calculate the dividend yield
      2. calculate the P/E Ratio
      3. record a trade, with timestamp, quantity of shares, buy or sell indicator and price
      4. Calculate Stock Price based on trades recorded in past 15 minutes
   2. Calculate the GBCE All Share Index using the geometric mean of prices for all stocks

### Constraints & Notes

1. Written in one of these languages:
   * Java, C#, C++, Python
2. No database or GUI is required, all data need only be held in memory
3. Formulas and data provided are simplified representations for the purpose of this exercise

### Table1. Sample data from the Global Beverage Corporation Exchange

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stock Symbol | Type | Last Dividend | Fixed Dividend | Par Value |  |
| TEA | Common | 0 |  | 100 |  |
| POP | Common | 8 |  | 100 |  |
| ALE | Common | 23 |  | 60 |  |
| GIN | Preferred | 8 | 2% | 100 |  |
| JOE | Common | 13 |  | 250 |  |

*All number values in pennies*

### Table 2. Formula

|  |  |  |
| --- | --- | --- |
|  | Common | Preferred |
| Dividend Yield |  |  |
| P/E Ratio |  | |
| Geometric Mean |  | |
| Stock Price |  | |

Solution

Class stock{

Private String StockSymbol;

Private String Type;

Private int dividend;

Private int lastDividend;

Private int FixedDividend;

Private int ParValue;

Private double price;

Private String sellBuy;

Private int quantity;

Private int TickerPrice;

Private Date createStock;

Public stock(String \_StockSymbol, String \_Type, int \_dividend, int \_lastDividend, int \_FixedDividend, int \_ParValue, String \_sellbuy, int \_quantity, int \_TickerPrice) {

StockSymbol = \_StockSymbol;

Type = \_Type;

Dividend = \_dividended;

lastDividend = \_lastDividend;

FixedDividend = \_FixedDividend;

ParValue = \_ParValue;

sellBuy = \_sellBuy;

price = \_price;

Tickerprice = \_Tickerprice;

createStock = now();

}

public String getStockSymbol(){

return StockSymbol;

}

public int getTickerPrice(){

return TickerPrice;

}

public Date getCreateStock(){

return createStock;

}

public int getQuantity(){

return quantity;

}

public double dividendYield(){

double result;

if(Type.equalsignorecase(“Common”) {

result = lastDividend/TickerPrice;

} else if(Type.equalsignorecase(“Preferred”) {

result = (FixedDividend \* ParValue)/TickerPrice;

}

return result;

}

public double PERatio(){

return (tickerPrice)/dividend;

}

public int minutesDiff(Date earlierDate, Date laterDate) {

if( earlierDate == null || laterDate == null )

return 0;

return (int)((laterDate.getTime()/60000) - (earlierDate.getTime()/60000));

}

public double StockPrice(Stock [] pStock, String \_StockSymbol){

Double Resulttemp1 = 0;

Double Resulttemp2 = 0;

// Stock Price all last stock in 15 minutes

for (int i=0;i<100;i++){

if ((minnutesDiff(pStock[i].getCreateStock(), now()) <= 15) && (pStock[i].getStockSymbol().equalignorecase(\_StockSymbol.getStockSymbol()))){

Resulttemp1 = Resulttemp1 + (pStock.getTickerprice()\*pStock.getQuantity());

Resulttemp2 = Resulttemp2 + pStock.getQuantity();

}

return (Resulttemp1/Resulttemp2);

}

public double root(double num, double root) {

if (num < 0) {

return -Math.pow(Math.abs(num), (1 / root));

} return Math.pow(num, 1.0 / root);

}

public double geometricMeans(String [] arrayStockSymbols, Stock [] pStock){

int len = arrayStockSymbol.length;

double result = 1;

for (int i=0;i<len;i++){

result = result \* StockPrice(pStock, arrayStockSymbol[i]);

}

return root(result, len);

}

}