|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Section** |  |

**PROJECT OOP in Java – Stock Transactions 50 points**

**Objective** Inheritance example.

***PROJECT DESCRIPTION***

Create stock transactions for XYZ stock for various clients and display a gain or loss on shares based on a given transacted stock price versus an updated stock price.

***Steps to Complete this Project***

**STEP 1** **Create 2 files in Eclipse under a project named Inheritance.**

**Enter all the code as follows (pages 1-3) for your first file called Stock.java**

**import** java.text.DecimalFormat;

**import** java.util.Scanner;

/\*\*

\* The Stock class holds data about a stock.

\* This version of the class has an equals method.

\*/

**public** **class** Stock

{

**private** String symbol; // Trading symbol of stock

**private** **static** **double** *sharePrice*; // Current price per share

**static** **int** *count*=0;

/\*\*

Constructor

**@param** sym The stock's trading symbol.

**@param** price The stock's share price.

\*/

**public** Stock(String sym, **double** price)

{

symbol = sym;

*sharePrice* = price;

}

**public** Stock() //for Subclass call

{

*count*++;

System.*out*.println("New client request count #" + *count*);

}

/\*\*

Copy constructor

**@param** object2 The Stock object to copy.

\*/

**public** Stock(Stock object2)

{

**this**.symbol = object2.symbol;

**this**.*sharePrice* = object2.*sharePrice*;

}

/\*\*

getSymbol method

**@return** The stock's trading symbol.

\*/

**public** String getSymbol()

{

**return** symbol;

}

/\*\*

getSharePrice method

**@return** The stock's share price

\*/

**public** **double** getSharePrice()

{

**return** *sharePrice*;

}

/\*\*

toString method

**@return** A string indicating the object's

trading symbol and share price.

\*/

**public** String toString()

{

// Create a string describing the stock.

String str = "Trading symbol: " + symbol +

"\nShare price: " + *sharePrice*;

// Return the string.

**return** str;

}

/\*\*

The copy method makes a copy of a Stock object.

**@return** A reference to a copy of the calling object.

\*/

**public** Stock copy()

{

// Create a new Stock object and initialize it

// with the same data held by the calling object.

Stock copyObject = **new** Stock(symbol, *sharePrice*);

// Return a reference to the new object.

**return** copyObject;

}

**public** **static** **void** main(String[] args)

{

**int** sharesToBuy; // Number of shares to buy.

// Create a Stock object for the company stock.

// The trading symbol is XYZ and the stock is

// currently $9.62 per share.

Stock xyzCompany = **new** Stock("XYZ", 9.62);

System.*out*.println(xyzCompany);

// Create a Scanner object for keyboard input.

Scanner keyboard = **new** Scanner(System.*in*);

// Create a DecimalFormat object to format numbers

// as dollar amounts.

DecimalFormat dollar = **new** DecimalFormat("#,###.00");

//StockPurchase array of buyers

StockPurchase [] theBuyers = **new** StockPurchase[3];

**for**(**int** x=0;x<3;x++)

{

// Display the current share prices.

System.*out*.println("XYZ Company's stock is currently $"

+ dollar.format(xyzCompany.getSharePrice())

+ " per share.");

// Get the number of shares to purchase.

System.*out*.print("How many shares do you want to buy? ");

sharesToBuy = keyboard.nextInt();

theBuyers[x]= **new** StockPurchase(xyzCompany, sharesToBuy);

// Display the cost of the transaction.

System.*out*.println("Cost of the stock: $"

+ dollar.format(theBuyers[x].getCost()));

}

}//end main

}//end class

**Next enter code as follows for your second file called StockPurchase.java**

**public** **class** StockPurchase **extends** Stock

{

**private** Stock stock; // The stock that was purchased

**private** **int** shares; // Number of shares owned

/\*\*

Constructor

**@param** stockObject The stock to purchase.

**@param** numShares The number of shares.

\*/

**public** StockPurchase(Stock stockObject, **int** numShares)

{

// Create a copy of the object referenced by

// stockObject.

**super**(); //call base class constructor

stock = **new** Stock(stockObject); //call copy constructor

shares = numShares;

}

/\*\*

getStock method

**@return** A copy of the Stock object for the stock

being purchased.

\*/

**public** Stock getStock()

{

// Return a copy of the object referenced by stock.

**return** **new** Stock(stock);

}

/\*\*

getShares method

**@return** The number of shares being purchased.

\*/

**public** **int** getShares()

{

**return** shares;

}

/\*\*

getCost method

**@return** The cost of the stock purchase.

\*/

**public** **double** getCost()

{

**return** shares \* stock.getSharePrice();

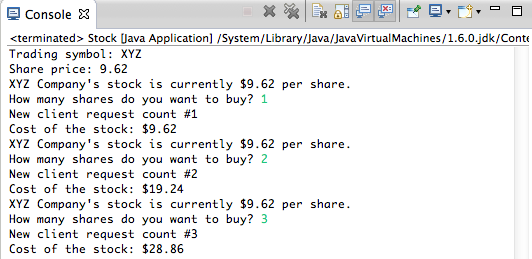
}

}

**STEP 2** **Run and test your code with the following input values 1, 2 & 3 when prompted**

**via the supplied for loop.**

Your display should look exactly like the snapshot below.

****

**STEP 3** **Modify your code to display each client by number, their original stock cost and**

**a gain or loss on shares based on an updated stock price of $20/share. Your**

**outcome should be presented in a table style format similar to the snapshot shown below when you rerun your app.**

Tweak any code necessary to add in extra fields, plus any additive variables, functions, etc., you deem necessary to your class, primarily your Stock class, to allow for a stock price update that’s input from the user *and* a client Number to track.

To display a gain or loss on transactions *after* the stock price has been updated, you want to show the *change* in stock prices \* number of shares purchased per client.

**Example:**

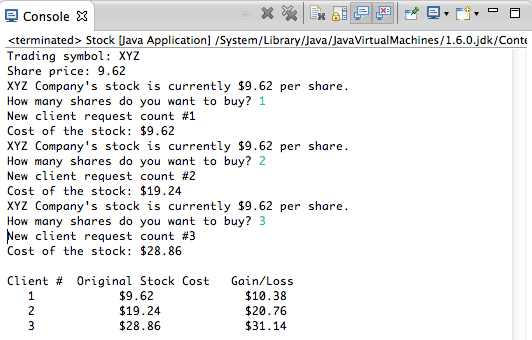
(Stock.*updatedSharePrice*-Stock.*sharePrice*)\*theBuyers[x].getShares()

Check the coding logic and use the functions given especially to help derive at your

outcomes/display.

Sample final run follows…

**Sample final run**

****

**STEP 4** **Turn in your packaged code for the Stock.java and StockPurchase.java files and any snapshot(s) of your input and display(s) as shown above for full credit into Blackboard.**