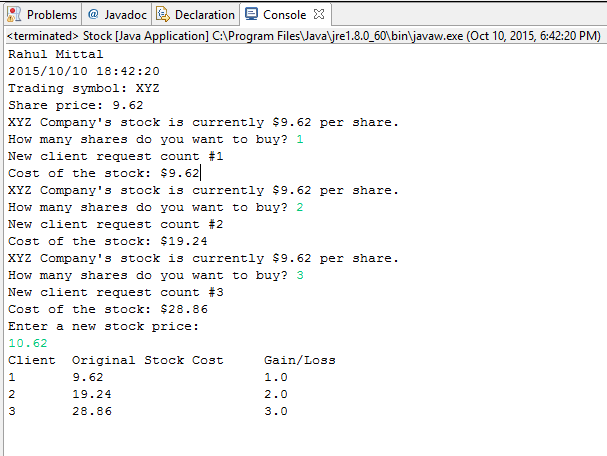
# Stocks Transaction

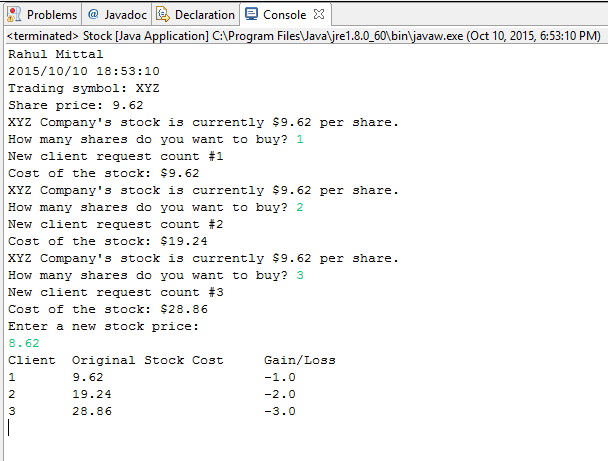
Create stock transactions for XYZ stock for various clients and display a gain or loss on shares based on a given transacted stock price versus an updated stock price.

# Positive Testing:

## For Gain:



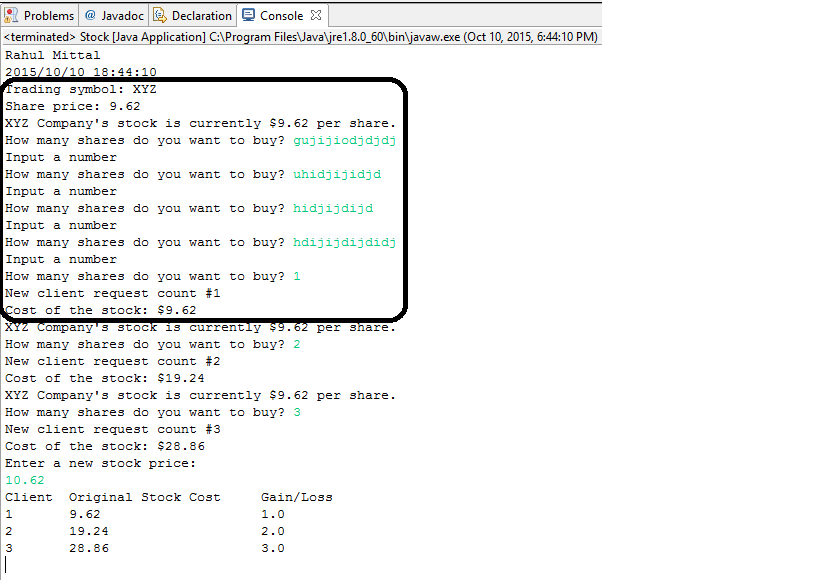
## For Loss:



# Negative Testing:

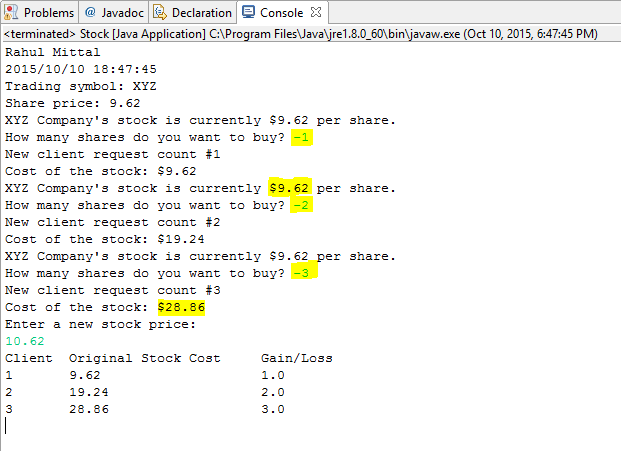
## Entering a share value a string value:

This will run in While loop which will every time checks for the value of number of shares.



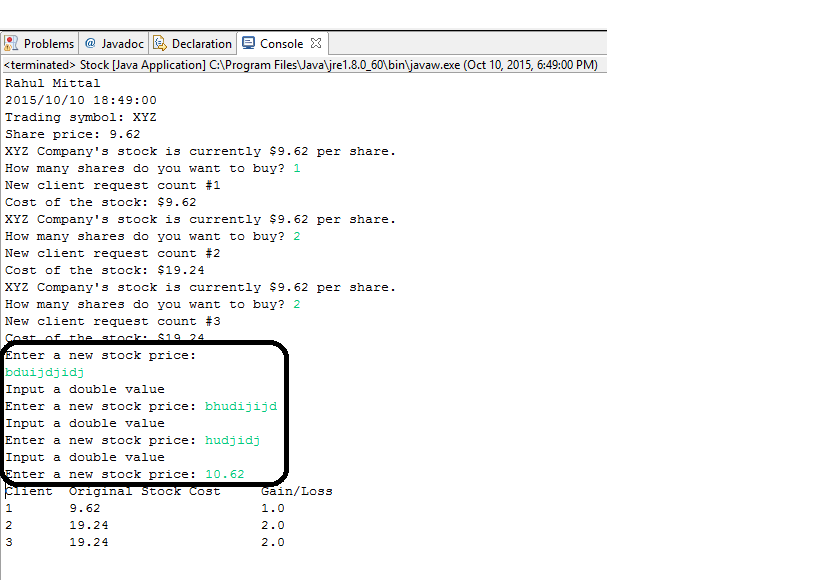
## Enter a Negative value of share

If user enters a negative value, it will take absolute value of the share.



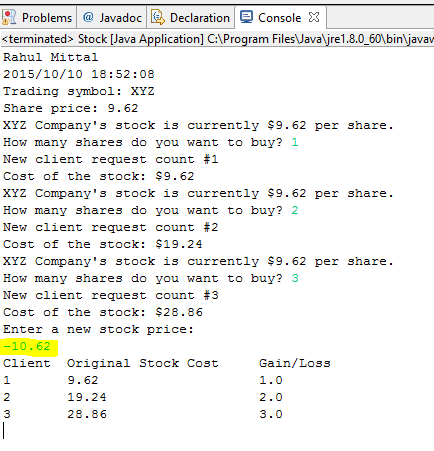
## If new stock price is not entered as a number:

If user enters a string, it will run in a loop until the user will enter a valid number.



## If user enters a negative value of a new stock price:

The program will take absolute value of the new price.



# Code Snippet:

## Stock.java

/\*\*

\* Name: Rahul Mittal

\* Lab Name: ITMD\_510\_LAB\_04

\* Current Date: 10/10/2015

\*/

**import** java.util.Date;

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**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

**private** **static** **double** *newPrice*; //updated Price

**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;

*setSharePrice*(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;

Stock.*setSharePrice*(Stock.*getSharePrice*());

}

/\*\*

getSymbol method

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

\*/

**public** String getSymbol()

{

**return** symbol;

}

/\*\*

getSharePrice method

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

\*/

**public** **static** **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: " + *getSharePrice*();

// 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, *getSharePrice*());

// Return a reference to the new object.

**return** copyObject;

}

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

{

DateFormat dateFormat = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

Date date = **new** Date();

System.***out***.println("Rahul Mittal");

System.***out***.println(dateFormat.format(date)); //2015/09/07 14:19:25

**int** sharesToBuy = 0; // 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(Stock.*getSharePrice*())

+ " per share.");

// Get the number of shares to purchase.

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

//Catch the exception if user has not entered a number

//and if user has entered a negative value, take an absolute value

**while**(keyboard.hasNext())

{

**try**

{

sharesToBuy = Math.*abs*(keyboard.nextInt());

**break**;

}

**catch**(Exception e)

{

System.***out***.println("Input a number");

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

keyboard.next();

}

}

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

// Display the cost of the transaction.

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

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

}

System.***out***.println("Enter a new stock price:");

//Catch the exception if user has not entered a number

//and if user has entered a negative value, take an absolute value

**while**(keyboard.hasNext())

{

**try**

{

*newPrice* = Math.*abs*(keyboard.nextDouble());

**break**;

}

**catch**(Exception e)

{

System.***out***.println("Input a double value");

System.***out***.print("Enter a new stock price: ");

keyboard.next();

}

}

keyboard.close();

System.***out***.println("Client" + "\t" + "Original Stock Cost" + "\t" + "Gain/Loss");

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

{

**int** client = i + 1;

**double** sharePrice = Stock.*getSharePrice*();

**int** shares = theBuyers[i].getShares();

**double** cost = theBuyers[i].getCost();

**double** gain\_loss = ( *newPrice* - sharePrice ) \* shares;

System.***out***.println(client + "\t" + cost + "\t" + "\t" + "\t" + gain\_loss);

}

}//end main

**public** **static** **void** setSharePrice(**double** sharePrice) {

Stock.*sharePrice* = sharePrice;

}

}//end class

## StockPurchase.java

/\*\*

\* Name: Rahul Mittal

\* Lab Name: ITMD\_510\_LAB\_04

\* Current Date: 10/10/2015

\*/

**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*();

}

}