RIT | Golisano College of Computing and Information Sciences
School of Information

Name: _____

ISTE-120
Lab 07: Decisions

Exercise 1 - Validating Stock Information (4 points)

Must be completed during the lab period.

1. Download the file Lab07Starter.zip from myCourses and unzip to obtain two files named `Stock.java` and `TradeStock.java`. The class named `Stock` has the following attributes and methods:

Attribute	Description
`name`	Name of stock as type `String`
`symbol`	Symbol for stock on appropriate stock exchange as type `String`
`price`	Price of one share as type `double`
`shares`	Number of shares as type `int`

Name	Parameters / Return	Description
`Stock`	Two parameters. First parameter is name of stock. Second parameter is symbol for stock. No return type.	Constructor to create a Stock object
`getName`	No parameters. Returns name of stock as type `String`.	Accessor for name attribute.
`getSymbol`	No parameters. Returns symbol of stock as type `String`.	Accessor for symbol attribute.
`getPrice`	No parameters. Returns price of stock as type `double`.	Accessor for price attribute.
`getShares`	No parameters. Returns number of shares as type `int`.	Accessor for shares attribute.
`setPrice`	One parameter with new price of stock as type `double`. No return value.	Mutator for price attribute.
`setShares`	One parameter with new number of shares as type `int`. No return value.	Mutator for shares attribute.

2. The class `TradeStock` allows the user to enter the information for a stock, set the values, and then print them. Run the program a few times to understand how it works.

3. Since the program does not validate any of the attribute values entered by the user, nonsensical inputs may be given. The valid values for each attribute follows:

Attribute	Validation Criteria
`name`	Must have at least one character.
`symbol`	Must be between 3 and 6 characters, inclusively.
`price`	Must be greater than 0.0
`shares`	Must be between 10 and 1000, inclusively and be a multiple of 10.

4. A `Stock` object should contain only valid data; that is, all four attributes must satisfy the validation criteria above. One strategy is to validate the attribute values in the main method and then store the valid values via the constructor and mutators into the object.

   Add if statements to the main method in the class `TradeStock`. If the attribute's value is valid, prompt the user for the value of the next attribute.

5. However, what happens if the value of the attribute is invalid? First, print an error message to indicate that the user entered an invalid input. Include in the message a description of the valid value that was expected by the program. See the first Sample Output below.

6. In a "commercial" program, the program would prompt the user to re-enter the input. However, this requires the use of a loop, which you have not had a lab on yet [next lab]. The only option for now is to terminate the program.

7. How is the program terminated? The `exit()` method in the `System` class terminates the currently running Java Virtual Machine (JVM). See the Java API. The method has one parameter. According to the API, "the argument serves as a status code; by convention, a non-zero status code indicates abnormal termination." Status codes are important for a Windows batch file or Unix script file. Here, use a value of 0 for the status code.

8. Complete testing of a program now requires both valid and invalid data to make sure the program takes the proper action. Especially important is a "boundary" case, that is, a value that separates the valid and invalid data. For example, since the symbol is supposed to have 3 to 6 characters, test with 3 and 6 characters to make sure that both are considered valid. Another valid value is an interior valid value such as 4 characters. Invalid tests should have too few and too many characters.

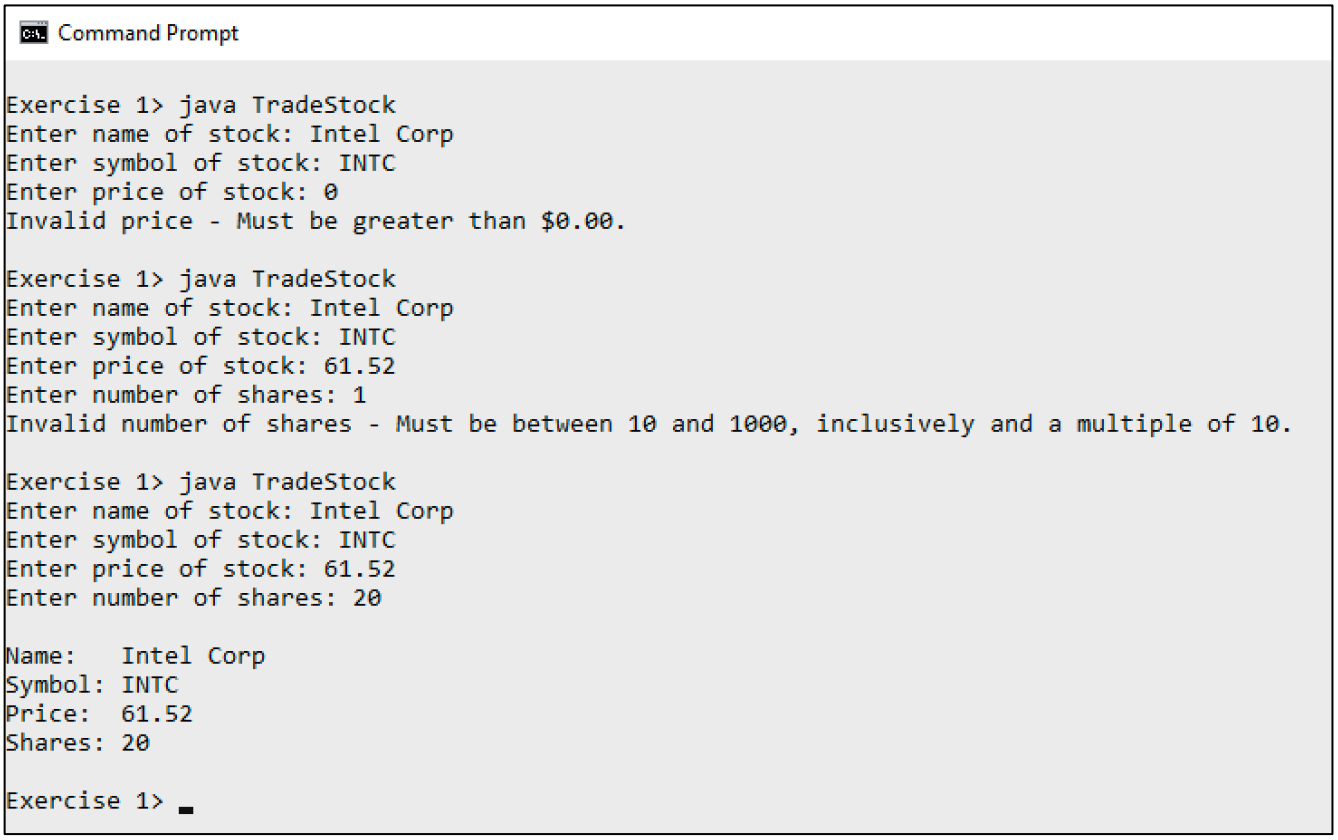
9. In the table on the next page, enter sample valid and invalid data for the "Test Plan" to thoroughly test the program.

Test Plan (copy to a Word document)

Attribute	Valid Test Data	Invalid Test Data
Name of stock		
Symbol of stock		
Price of stock		
Number of shares		

Submit your Test Plan to the Lab07 Assignment folder when completed. Failure to submit a completed Test Plan will result in a grade of 0 for Exercise 1.

Sample Output



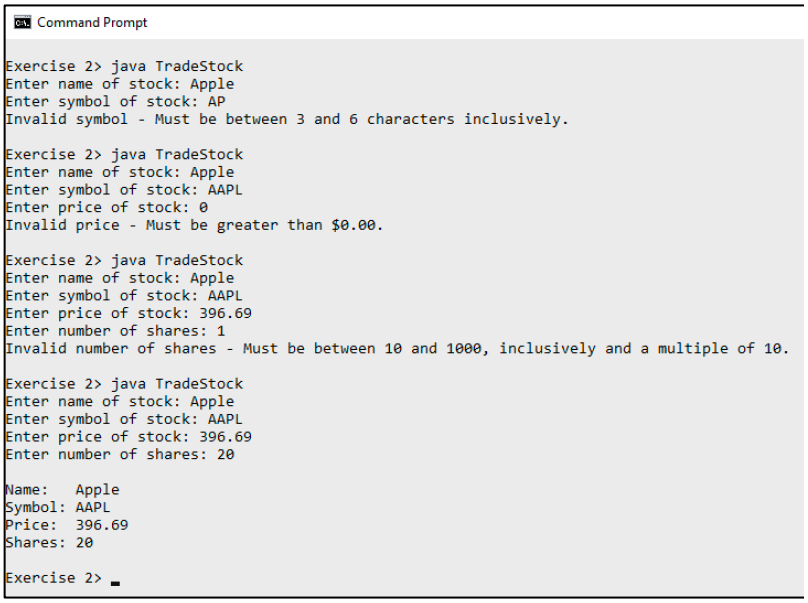
Submit your .java files to the Lab07 Assignment folder when Exercise 1 is working correctly.

Exercise 2 - Validating Stock Information Using Mutators (3 points)

Must be completed during the lab period.

10. Make a copy of files for Exercise 1 into a new folder for Exercise 2.

11. Another strategy to validate user inputs is to test the attribute value in its corresponding mutator method. Use the rules for valid data from Exercise 1. If the value of the attribute is valid, save the new value in the appropriate attribute and return a value of true. If the value is invalid, do NOT change the attribute and return a value of false. Do not print an error message in the mutator method! Modify the mutators for price and number of shares as described above.



12. However, there are no mutators for name and symbol as they are set in the constructor. While a constructor can contain if statements, there is no return value.

13. Add a default constructor to the `Stock` class. Set the value of the name and symbol to the empty `String`.

14. Add mutators for the name and symbol using the rules for valid data from Exercise 1.

15. Modify the code in the main method in the `TradeStock` class to use the **default constructor** to create an empty object.

16. Then use the mutators to validate the user-entered data. Test the return value to determine if the user input is valid. If the return value is false, print the same error messages as in Exercise 1 and terminate the program.

17. The output of the program should be exactly the same as Exercise 1. Remember to test the program thoroughly with valid and invalid data.

Submit your .java files to the Lab07 Assignment folder when Exercise 2 is working correctly.

Exercise 3 – Buy and Selling Stock (3 points)

If not completed during the lab period, finish and submit prior to the beginning of the next lab session.

18. Make a copy of files for Exercise 2 into a new folder for Exercise 3.

19. Now add an option to buy and sell stock. The cost of the shares to be bought or sold is the price multiplied by number of shares.

20. There is a commission for trading stock. The commission for buying or selling shares is 1% of the cost of the shares with a maximum of commission of $500.00.

21. If shares are bought, the commission is added to the cost of the shares to compute the total cost. If shares are sold, the commission is subtracted from the value of the shares to compute the net receipts.

22. Add the following two methods to the Stock class:

Name	Parameters / Return	Description
`calcValue`	No parameters. Return total value of stock as type `double`.	Value is price multiplied by number of shares. Do not include commission.
`calcCommission`	No parameters. Return cost of commission as type `double`.	The commission rate is 1% with a maximum of $500.

23. In the `Stock` class, you must define the commission rate and the maximum commission as **constants** to be used in the `calcCommission` method.

24. At the end of the main method of the `TradeStock` class, add code to prompt the user whether to buy stock, sell stock or exit. The user will enter the character `'B'` [upper or lower case] to buy stock, the character `'S'` [upper or lower case] to sell stock. If any other character is entered, take no action. See the Sample Output for format of input and output.

25. When testing, there is no need to enter invalid attribute values for the stock as this input has been thoroughly tested in the earlier exercises. However, there are still a number of different values of the option to test – `'B'`,`'b'`,`'S'`,`'s'` and others. Each test requires that the stock information be re-entered, which greatly increases the time to test the program.

26. (Optional) A loop can be used here to reduce the amount of input, which reduces the time to test. One very simple loop is an infinite loop, that is, it never stops on its own, but requires manual intervention to halt the loop and the program.

27. (Optional) An infinite loop using the `while` statement is as follows:

```
while (true)
{
   //code to be repeated
}
```

The code to allow the user to enter the stock information occurs before the loop so that the information needs to be entered only once.

To manually terminate the loop in jGRASP, use the End button in the execution window. To terminate the loop in a Terminal window, enter Control-C.

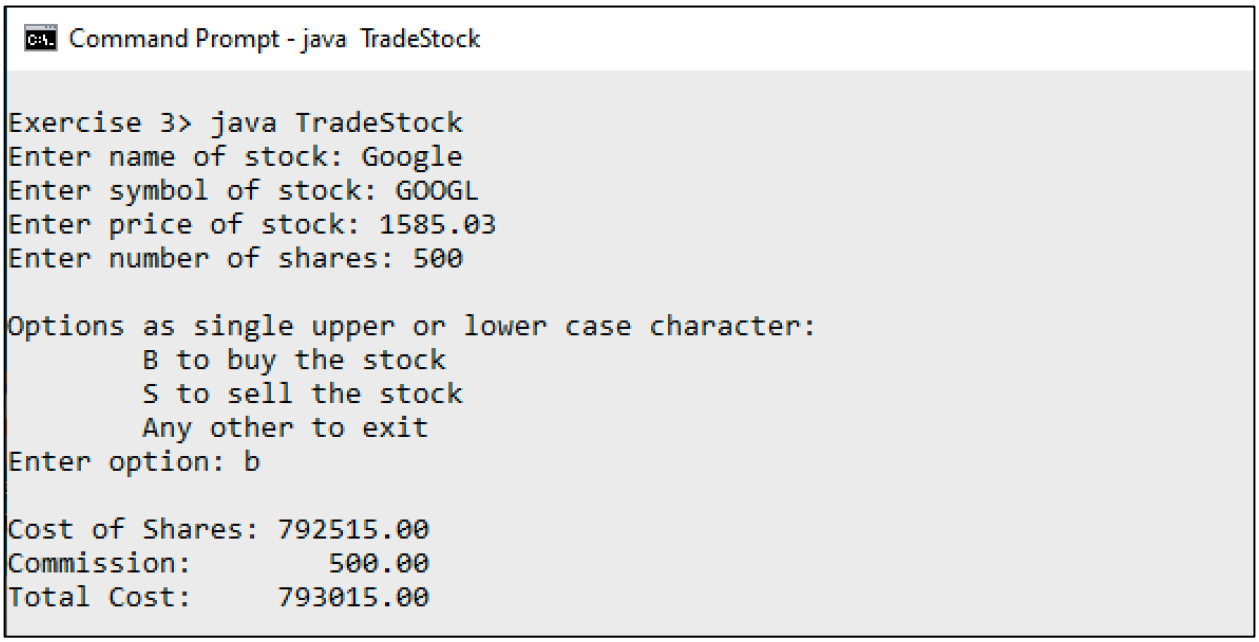
See below for a Sample Output with an infinite loop executed in a Terminal window.

28. (Optional) Use one `printf` statement to print all the information for selling a stock and another `printf` statement to print all the information for buying a stock. Using `printf` ***format specifiers*** to fix the width of the fields makes it easy to align the decimal point. (See the Java API … look up the `PrintWriter` class and scroll down to **Format String Syntax** to see how to use ***format specifiers***).

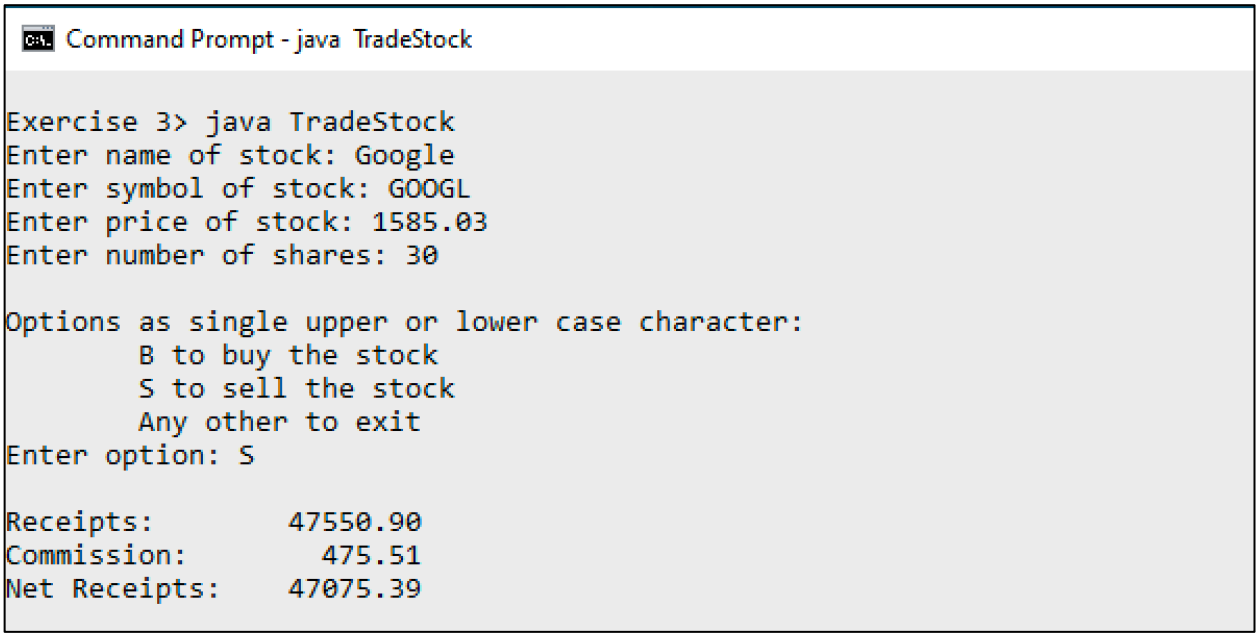
Note that the field width to print the cost of the stock, etc. needs to be fairly large as stocks such as Berkshire Hathaway Inc. have a very high price – in fact, on 7/21/2020 the opening price was $286,720.94!

Sample Output:

One buy transaction



One sell transaction



Multiple transactions using infinite loop

Select Command Prompt

```
Exercise 3> java TradeStock
Enter name of stock: Walmart
Enter symbol of stock: WMT
Enter price of stock: 131.56
Enter number of shares: 10

Options as single upper or lower case character:
        B to buy the stock
        S to sell the stock
        Any other to exit
Enter option: s

Receipts:         1315.60
Commission:         13.16
Net Receipts:     1302.44

Options as single upper or lower case character:
        B to buy the stock
        S to sell the stock
        Any other to exit
Enter option: b

Cost of Shares:   1315.60
Commission:         13.16
Total Cost:       1328.76

Options as single upper or lower case character:
        B to buy the stock
        S to sell the stock
        Any other to exit
Enter option: x
No action taken

Options as single upper or lower case character:
        B to buy the stock
        S to sell the stock
        Any other to exit
Enter option:
Exercise 3>
```

Submit your .java files to the Lab07 Assignment folder when Exercise 3 works correctly.