

Unit 6 Chapter 15 Assignment

Grading Information: This Program is **due** on **Date Specified**.

Comments are **REQUIRED**; flow charts and pseudocode are **NOT REQUIRED**.

Directions	Points
<p>The files must be called <LiFiUnit6Ch15.java> LiFiSaleCheck.java (Sale Checker Class File)</p> <p>The files must be called as specified above, (LiFi = Your Last Initial Your First Initial)</p> <p><i>Proper coding conventions required the first letter of the class start with a capital letter and the first letter of each additional word start with a capital letter.</i></p> <p>Only submit the .java files needed to make the program run. Do not submit the .class files or any other files.</p>	5%
Style Components <p>Include properly formatted prologue, comments, indenting, and other style elements as shown in Chapter 2 starting page 64 and Appendix 5 page 881-892.</p>	5%
Topics covered in chapter <p>Topics with * are covered in this assignment. Ensure you use every item listed below in your completed assignment.</p> <p>*Exceptions and Exception Messages *try / catch *checked and unchecked exceptions *generic catch block *throws</p>	
Basic Requirements <p>Write a program that validates a sale in dollars and cents with a \$ and a .(period).</p> <p>See sample output below.</p>	20%
LiFiUnit6Ch15.java <ul style="list-style-type: none">• Driver class should loop until “q” is entered to quit• If enter is not “q”, then create an instance of the LiFiSaleCheck object passing the entry as an argument• If no error	20%

- Print amount by calling:
 - print numeric from LiFiSaleCheck class
 - print alphabetic from LiFiSaleCheck class
- If error
 - Call the getError() method on this LiFiSaleCheck object and print the error message (see sample)

LiFiSaleCheck.java class

Sales object should store the sale in 2 integer instance variables, dollars and cents, and include a string variable to hold the error. This should be initialized with null.

LiFiSaleCheck Constructor:

- Receive sale as a string
- Perform error checking to ensure amount was entered in proper format to include a period (.) between the dollars and cents and a \$ at the start of the string.
- Use indexOf and substring to separate the sale string into the appropriate instance variables
- Use try/catch to catch format errors of dollars and cents as shown in example
- If an error occurs, change the error instance variable to reflect the error (see sample)
- If more than one error occurs in the format of the dollars and cents, show both.

50%

print numeric method:

- Print in the format \$123.45 using both dollars and cents instance variables (if amounts like \$4.2 is entered, should print \$4.20)

print alphabetic method:

- Print in the format 123 dollars and 45 cents using both dollars and cents instance variables (if amounts like \$4.2 is entered, should print 4 dollars and 20 cents)

getError method:

- returns the error instance variable (String type)

NOTE: Complete your activity and submit it by clicking “Submit Assignment”

Total Percentage

100%

Sample

Your output will vary based on input.

```
Please enter amount of sale in form $#.## <"q" to quit>: 123.45
Invalid sale format missing "$" - 123.45

Please enter amount of sale in form $#.## <"q" to quit>: $123:45
Invalid sale format missing "." - $123:45

Please enter amount of sale in form $#.## <"q" to quit>: $abc.34
Invalid dollar format - For input string: "abc"

Please enter amount of sale in form $#.## <"q" to quit>: $123.de
Invalid cents format - For input string: "de"

Please enter amount of sale in form $#.## <"q" to quit>: $12d.de
Invalid dollar format - For input string: "12d"
Invalid cents format - For input string: "de"

Please enter amount of sale in form $#.## <"q" to quit>: $123.45
$123.45
123 dollars and 45 cents

Please enter amount of sale in form $#.## <"q" to quit>: q
```