**HW04 - Challenge**

* [Description](https://labs.vocareum.com/web/1587618/529558.0/ASNLIB/public/docs/README.html#1)
* [Instructions](https://labs.vocareum.com/web/1587618/529558.0/ASNLIB/public/docs/README.html#2)
* [Testing](https://labs.vocareum.com/web/1587618/529558.0/ASNLIB/public/docs/README.html#3)
* [Submit](https://labs.vocareum.com/web/1587618/529558.0/ASNLIB/public/docs/README.html#4)

**Description**

For this Homework, you will be writing a program called ToySeller that will allow a user to design a custom toy. You will branch off to specific options based on their choices, eventually listing the full price of the item the customer has designed.

Note: 5 points of your Challenge grade is based on Coding Style.  You will need to add documentation to the Starter Code to follow the standards described on Brightspace. Use the "Run" button to check your Coding Style without using one of your 10 Vocareum submissions.

**Instructions**

Download the Starter Code. The class already contains every question prompt you will need. Use these to minimize the risk of spelling mistakes.

**Designing a Solution**

Before describing the implementation requirements, we wanted to provide a quick note on planning your programs.

You may find it useful to use the processing information provided here to create a decision tree, table of inputs-to-outputs, or some other data management tool. Reorganizing the information will allow you to present all possible paths in an organized manner, making it easier to write your solution. We recommend doing this. Practicing program and logic design will be incredibly useful as we move into more complex projects. It can also save you a great deal of time.

**Processing**

Using the questions provided in the Starter Code, determine the brand the user wishes to identify. There are several levels of questions that you will need to process. Each is documented in the diagram available [here](https://labs.vocareum.com/web/1587618/529558.0/ASNLIB/public/docs/ToySellerDiagram.png).

Note:

* All inputs should be accepted on a case-insensitive basis. That is, "Y" and "y" are both valid.

**Testing**

Use the flowchart to verify each path progresses as expected through your program.

Sample Output Test 1

Welcome to the Custom Toy Builder!  
Our store can custom-design 3 different types of toys.  
Which would you like to create? (Enter a number)  
1. Superman figurine  
2. Robotic cyclops  
3. Cat-shaped lamp  
4. EXIT  
[1]  
What size toy would you like? (Enter S, M, or L)  
[M]  
Do you want Superman to talk? (Enter Y or N)  
[Y]  
Do you want to add a cape? (Enter Y or N)  
[Y]  
Your Superman figurine costs $23!  
Thanks for using the Custom Toy Builder!

Sample Output Test 2

Welcome to the Custom Toy Builder!  
Our store can custom-design 3 different types of toys.  
Which would you like to create? (Enter a number)  
1. Superman figurine  
2. Robotic cyclops  
3. Cat-shaped lamp  
4. EXIT  
[2]  
What size toy would you like? (Enter S, M, or L)  
[S]  
Do you want the Robotic Cyclops to move? (Enter Y or N)  
[N]  
What color eye do you want? (Enter 1, 2, or 3)  
1. Red  
2. Yellow  
3. Blue  
[3]  
Your Robotic Cyclops costs $15!  
Thanks for using the Custom Toy Builder!

Sample Output Test 3

Welcome to the Custom Toy Builder!  
Our store can custom-design 3 different types of toys.  
Which would you like to create? (Enter a number)  
1. Superman figurine  
2. Robotic cyclops  
3. Cat-shaped lamp  
4. EXIT  
[3]  
What size toy would you like? (Enter S, M, or L)  
[L]  
What color Cat-shaped Lamp do you want? (Enter 1, 2, 3, or 4)  
1. Gold  
2. Pink  
3. Blue  
4. No Color  
[2]  
Do you want the Cat-shaped Lamp to glow? (Enter Y or N)  
[Y]  
Your Cat-shaped Lamp costs $32!  
Thanks for using the Custom Toy Builder!

Note: Brackets [] indicate input.

Note:  Match this output exactly.  Spelling mistakes will result in lost points.

We have also included a program that will allow you to create and run output tests automatically in the Starter Code. This will make it easier for you to verify that each possible progression through your solution is correct. Take a look at RunLocalTest.java. There are many utility features and tools that you do not need to worry about at the moment, instead, focus on the test case included below:

@Test(timeout = 1000)  
public void testExpectedOne() {  
  
 // Set the input  
 String input = "1\nM\nY\nY\n";  
  
 // Pair the input with the expected result  
 String expected = expectedResultsOne;  
  
 // Runs the program with the input values  
 receiveInput(input);  
 ToySeller.main(new String[0]);  
  
 // Retrieves the output from the program  
 String stuOut = getOutput();  
  
 // Trims the output and verifies it is correct.  
 stuOut = stuOut.replace("\r\n", "\n");  
 stuOut = stuOut.substring(stuOut.indexOf("Your"));  
 assertEquals("Make sure you follow the flowchart and use the given strings for the result!", expected, stuOut);  
}

To use this program, first select which path you wish to test using the flowchart. Determine the inputs for each of the prompts and add them to the input String, with each prompt separated by a \n (simulates pressing enter). Then, create the appropriate output String using the flowchart and given prompts. You can either download the program and run the main method or use the "Run" button on Vocareum to run the test. You can repeat this process for each path.

**Public Test Cases Note**

For many homeworks and projects, we will give you test cases that correspond to several of the ways we will be testing your program. But, we will not give you test cases for ALL of the ways we will be testing your program. You should think of other test cases to use that will fully test every aspect of every feature of your program. Just because your program passes all the test cases we give you does not mean that it is fully correct and will receive a score of 100.

**Submit**

After testing your solution and verifying that it meets the requirements described in this document, you can submit on Vocareum.  You have 10 submission attempts to achieve full points.

Diagram

Description automatically generated

Diagram

Description automatically generated