## IS 2033 – BBCA2 ASSIGNMENT

| **TOTAL POINTS: 100** |
|---|
| **DUE: 03/25/2020 BY 11:59 PM** |
| ▪ **READ CAREFULLY "ALL" THE INSTRUCTIONS!!!**<br>▪ **Do not submit code that doesn't compile**<br>▪ **No assistance given via email.**<br>▪ **If needed, seek help during open tutoring or instructor office hours.** |

**UTSA HONOR CODE:** As a UTSA student, you are **bound** by the honor code, so DO NOT cheat on any of your coursework. ***By submitting this assignment, you are attesting to your own authorship*** based on material from the IS 2033 textbook and/or your professor. Cheating can result in any one, or combination, or all of the following: reduced or failing grade for the assignment, a signed statement of the infraction, reduced or failing grade for the course, reporting of student name to the Department Chair and faculty, Dean's Office and COB faculty, and/or elevation to Student Conduct and Community Standards.

**OBJECTIVE *(this is not the program purpose)*:** Code a program that uses the concepts covered in chapters 1-4 and lecture.

***New instructions are inserted as brown text.***

***NOTE: Make sure your BBCA1 is running with the correct output before attempting BBCA2. Your PAs take alot of time to grade. Your BBCA1 may not be graded before BBCA2 is due; nevertheless, you were given the logic for BBCA1, you know through the sample output and the PA instructions what is expected right down to the formatting (spacing, line advances, etc.), so you don't have to be "told" what you did wrong in BBCA1 to get BBCA2 correct. This is not unlike what you'll be expected to do in industry.***

**Print out your BBCA1 code and the BBCA2 instructions. Read through the BBCA2 instructions thoroughly to get a good idea of the logic, the required control structures, and the changes that have to be made. Insert hand-written changes or cut your BBCA1 code with actual scissors and paste/tape the code in its proper order. After you've done this, open DrJava, copy your BBCA1 code into a BBCA2.java file, then start altering it.**

**PREP WORK:** 1, 2, 3, 4 (includes the instructor's PowerPoint slides); completed labs up through chapter 4.

**GRADING**: You'll be graded on *how well* you follow the program instructions and the ***accuracy of your output as reflected in the prompts, the output specifications, and the sample output***. ***This includes spacing and line advancing.*** **Each line of output can be associated with multiple points in the code!** The instructions, prompts and output are what the user wants. You are **not** at liberty to change anything, but code to these requirements. You'll also be graded on the code given to you by your professor for this PA.

**PROGRAM INSTRUCTIONS:** Create a program that will display/print a sales receipt for customers who purchase from Fairytale Sweets, **but, allow the customer to change the current candy selection**. This program is the same as BBCA1 except for the changes articulated below.

1. *Logical Control Structures:*
   a. Add a boolean variable called ***purchasesMade*** and set it to false. This variable controls whether the sales receipt is printed or not and also the "thank you" messages.

b. Use a sentinel-controlled ***while*** loop to process multiple candy purchases on the same receipt (retain the *while* loop from BBCA1).
   i. Immediately inside the while loop set *purchasesMade* to true. Entry into the *while* loop means that there will be purchases.
   ii. Use a ***nested do-while*** to control any changes made to the current candy selection.
      1. Both the *while* and *nested do-while* are sentinel-controlled loops; therefore, another loop-control variable for the changes is needed.
      2. When capturing input for the candy choice and quantity, check for entries that are not integers. Use the validation *while* loop found in the TestValidInput program coded in class.
      3. Use an ***if-else***
         a) test to see if the candy choice is within the proper range of 1 through 5 – use a logical operator in the test expression;
         b) when the choice is correct, use a ***switch*** in place of the nested if-elses from BBCA1 to figure out
            i) the name of the candy;
            ii) the price of the candy;
         c) use a ***nested if-else*** to format the cost of the first candy item on the sales receipt with a $ sign (same as BBCA1);
         d) otherwise, print the error message **"Invalid candy choice! Try again."** (in BBCA2, there'll be only 1 printf statement for this error message). You'll have to re-initialize the do-while loop-control variable to a value that makes the test expression true.
   iii. The addition to the subtotal, the addition of the candy line item to the sales receipt is outside of the *do-while* loop.

c. After exiting the *while* loop …
   i. Test the boolean variable purchasesMade.
      1. When there are purchases made, print the message: **"Proceeding to checkout."**
      2. Calculate the tax amount and total.
      3. Finalize the sales receipt and print it.
      4. Print the message: **"Thank you for your business! Please come again."**
   ii. Otherwise, print the message: **"Thank you for visiting Fairytale Sweets! Please come again."**

d. **Sales Receipt Recap:** The customer can purchase multiple candies on the same sales receipt. *The customer can also change a candy selection **before** it is added to the sales receipt.* The sales receipt is created in real time. Each candy purchase with quantity, candy name, price and item cost is added. The subtotal is calculated with each purchase. Once there are no more purchases, the sales tax is calculated along with the total and added to the sales receipt which is finalized for printing/displaying.

e. Use printf() with format specifiers where needed.

f. Don't forget to insert the exit statement at the end of main().

g. *The prompts, the final output specs, and the sample output show you in what order to place your code.* To return from these links press Alt then left arrow.

2. ***CODE FOR PROPER OUTPUT ALIGNMENT: where salesReceipt is a String variable that holds the current purchases. Note in the code below the use of right and left justification to align the output regardless of the candy and quantity chosen. Note the use of += which reassigns salesReceipt to the new String object created by String.format().*** **ALL** Java statements are to be typed **exactly** as shown. The following **if-else** prints a $ sign for the first line item; otherwise, there's no $ sign printed. Look at the output specs above or the sample output.

The **trigger** variable determines the first line item. NOTE: There is a **space** between the double quotes.

**/*Java code for Prompt 2*/**

```
System.out.printf("%nFAIRYTALE SWEETS"
                + "%n%n1.  Arabian Nights Chocolate Coins - 1 lb. Bag %5s%,7.2f"
                + "%n2.  Beauty and the Beast Lollipops - 1 lb. Bag %,12.2f"
                + "%n3.  Mad Hatter Jelly Beans - 1 lb. Bag %,20.2f"
                + "%n4.  Pinocchio's Candy Cones - Each  %,23.2f"
                + "%n5.  Sleeping Beauty Caramel Apples - Each %,17.2f"
                + "%n%nEnter your choice:  ", "$", 2.25, 2.50, 1.75, 0.75, 1.25);
```

**/*Java code for Prompt 4*/**

```
System.out.printf("%nDo you want to make any changes to the following "
                + "selection?  "
                + "%n%n%s"
                + "%n     %d @ $%.2f ea.%15s$%,.2f"
                + "%n%nEnter \'Y\' or \'N\':", candy, quantity, price, " ",
                     itemTotal);
```

```
/*After the purchases are successfully completed, this if-else is used to add the first purchase
*to the sales receipt with a $ sign when the formatFirstItem is 1 and no $ sign when the
*formatFirstItem is not. This is now located in the while body before the last prompt. ALL Java
*statements are to be typed exactly as shown. The following if-else prints a $ sign for the first line
*item; otherwise, there's no $ sign printed. Look at the output specs above or the sample output.
*The trigger variable determines the first line item. NOTE: There is a space between the double
*quotes.
*/
if(formatFirstItem == 1)

{   salesReceipt += String.format("%n%s"
                 + "%n    %d @ $%.2f ea. %-24s $%,10.2f%n", candy,
                    quantity, price, " ", itemTotal);

 }  formatFirstItem = 0;
else

{    salesReceipt += String.format("%s"
                  + "%n     %d @ $%.2f ea. %-25s %,10.2f%n", candy, quantity,
                                price, " ", itemTotal);
}//END if formatFirstItem is 1 OR else formatFirstItem NOT 1

/*This is located outside of the while loop in the IF structure
for * purchasesMade.
*/
salesReceipt += String.format("%n%36s %-6s $%,10.2f"
             + "%n%36s %-7s %,10.2f"
             + "%n%n%36s %-6s $%,10.2f%n", "SUBTOTAL:   ", " ",
               subtotal, "TAX @ 8.250%:  ", " ", taxAmount,

                "TOTAL:  ", " ", total);

 System.out.printf("%s", salesReceipt);
```

3. **Work and submit this program on your own** (no partner). Name your program as ***YourLastNameFirstInitialYourSectionNoBBCA2*.**

4. ***Commenting Your Program*:**
   a. In your program, YOU MUST insert a **program purpose** in the first comment box. The content of that first comment box was shown to you in the *Anatomy of a Java Program* lecture for chapter 1.
   b. Use Javadoc comment boxes beginning with /** and ending with */ for your comment boxes.
   c. Insert a Javadoc comment box above your methods explaining what is going on in the method that goes for the main() which is a method.
   d. Line comment the import statements and the variables declared at the class level and/or in any method [including main()].

5. ***Formatting Rules*:** Refer to the *Java Style Guide* PDF posted on Blackboard in IS 2033. Always test your output to validate that your program is functioning properly with the correct output and spacing.

**PROMPTS:** *Code the **bold** from the prompts below in the printf statements that capture data into your program. Once again, the prompts tell you your input variables. Except for the first prompt, all the other ones reside within the while loop.*

*1st Prompt: The value captured from this prompt is the loop-control variable for the sentinel-while loop.*

**Do you want to proceed with your candy purchase? Enter 'Y' or 'N':**

*When the answer is 'N', determine if purchases were made, if **not**, **no** sales receipt prints, but this message is displayed:*

**Thank you for visiting Fairytale Sweets! Please come again.**

*2nd Prompt: Displays when the answer to the 1st **prompt is 'Y'**. This prompt is scoped to the do-while. Link below is to prompt in actual output.*

**FAIRYTALE SWEETS**

| | |
|---|---|
| **1. Arabian Nights Chocolate Coins - 1 lb. Bag** | **$2.25** |
| **2. Beauty and the Beast Lollipops - 1 lb. Bag** | **2.50** |
| **3. Mad Hatter Jelly Beans - 1 lb. Bag** | **1.75** |
| **4. Pinocchio's Candy Cones - Each** | **0.75** |
| **5. Sleeping Beauty Caramel Apples – Each** | **1.25** |

**Enter your choice:**

*While the choice is not an integer keep displaying this message:*

**ONLY enter an integer! Re-enter your choice:**

*If the choice is 0 or less or greater than 5, display this message:*

**Invalid candy choice! Try again.**

*3rd Prompt: If candy choice is 1 through 5, then assign the name of the candy to its variable and the candy's price to a price variable using a **switch** statement, then prompt for the quantity. The Xs is the name of the candy. Code in the IF body of the IF-ELSE that tests the range for the candy choice. The IF-ELSE is scoped to the do-while.*

**Quantity for Xxxxxxxxxxxxx:**

*While the quantity is not an integer keep displaying this message:*

**ONLY enter an integer! Re-enter the quantity:**

*4th Prompt : Where the Xs is the name of the candy, the ZZ9 is the quantity, the $Z.99 is the price of the candy and the $Z,ZZ9.99 is the line item total for that candy purchase. This prompts for the do - while loop control variable. Coded in the IF body of the IF-ELSE that tests the range for the candy choice. The IF-ELSE is scoped to the do-while. When the answer is 'Y', the do-while loop starts over again beginning with prompt 2.*

**Do you want to make any changes to the following selection?**

**Xxxxxxxxxxxxxxxxxxx**
**ZZ9 @ $Z.99 ea. $Z,ZZ9.99**

**Enter 'Y' or 'N':**

*5th Prompt: The value for this prompt is captured in the loop-control variable for prompt 1 (same as prompt 4 in **BBCA1**).*

**Would you like to make another candy purchase? Enter 'Y' or 'N':**

*When the answer is 'N', determine whether purchases were made, if so print this message, then print the sales receipt and a "thank you" message (look in the final output specifications):*

**Proceeding to checkout.**

**Final Output Specifications:** *The Xs is the name of the candy, the ZZ9 is the quantity, the $Z.99 is the price of the candy and the $Z,ZZ9.99 is the line item total for that candy purchase. Z's denote printing of 1-9 in that position (suppression of leading 0's). 9's denote printing 0-9 in that position. The XX for Time indicates AM or PM. Use System.out.printf() and the appropriate format specifiers (look in Appendix I) to properly space the output. **NOTE:** The date is NOT to be hard coded into the header, instead, you will need to capture the system's date (Appendix I). This is so the date will change accordingly. Furthermore, create your receipt as a String which allows you to add the header and date information first, then add the purchase as the customer selects it. Note how the $ sign only prints for the first line item on the receipt. Make sure the content of your receipt is properly aligned. Use left and right justification.*

⬇**Actual Output Specifications**

**FAIRYTALE**
**SWEETS North Star**
**Mall San Antonio, TX**

← 3 Header Lines: First line of header triple-lined advance (2 blank lines before printed line).

**Date: 99/99/99**
**Time: 99:99:99 XX**

← Date & Time Labels

| | |
|---|---|
| **Xxxxxxxxxxxxxxxxxxx** | |
| **ZZ9 @ $Z.99 ea.** | **$Z,ZZ9.99** |
| **Xxxxxxxxxxxxxxxxxxx** | |
| **ZZ9 @ $Z.99 ea.** | **Z,ZZ9.99** |

← Line Items: candy name, quantity, price, line item total.

| | |
|---|---|
| **SUBTOTAL:** | **$Z,ZZ9.99** |
| **TAX @ 8.250%:** | **ZZ9.99** |
| **TOTAL:** | **$ZZ,ZZ9.99** |

← Subtotal: Sum of line item totals.
Tax Amount: .0825 of subtotal.
Total: Sum of subtotal and tax amount.

**Thank you for your business! Please come again.**

*****END OF OUTPUT SPECIFICATIONS*****

**SUBMISSIONS REQUIREMENT:**

1. **Word Document:** Copy your .java code into a Word document and save it with the same name as your program. Upload the document to Blackboard.

1. **Zipping Folders: Your Java files *must be* in a folder.**
   1. Create a folder named for the program (excluding the file extension).
   2. Put your **.java, .class**, **.java~** files in the folder.
   3. To zip the folder, point to it then right click and
      i. **Filzip** if you have it **OR**
      ii. Click **Send To** then click **Compressed (zipped) Folder**
   d. Upload your zipped folder to Blackboard.

3. **Uploading to Blackboard: Make sure your browser is properly configured for Blackboard (see syllabus).**
   a. Your submissions are to be uploaded to Blackboard through **Assignments** only.
   b. **Upload your files no later than the due date by 11:55 pm**; otherwise, you don't have time to recover from any problems and your assignment may not be accepted by Blackboard.
   c. Check to make sure your submission is uploaded. Please **do not ask your instructor** to check whether your assignment has been uploaded. You can do this yourself. Or upload during a tutoring session when someone can help you.

d. If you submit your assignment before the due date, want to make changes or upload additional files, you can **re-upload** your files. Your last submission is the one graded.

**NO ASSIGNMENTS WILL BE ACCEPTED LATE OR VIA E -MAIL. DO NOT UPLOAD PROGRAMS THAT DON'T COMPILE, RUN, or PRODUCE THE CORRECT OUTPUT. Insert the following note in the *Comment* section for the assignment on Blackboard then click Submit: *My program doesn't compile.* OR *My program doesn't run.* OR *My program's output is incorrect.***

*******SAMPLE OUTPUT******* *It is always good to test your code using sample data to see if your program meets the output specifications.* ***Run your program using the data in the following sample output.*** **For the output produced by your program to line -up properly, the FONT in DrJava should be Monospaced or Courier New.** ***Copy and paste the output into a traditional comment box /* */ at the end of your BBCA2.java file. Eliminate the asterisks (*) between the first and the last.*** *The comment box needs to be outside of the close brace for the class.* ***Worth 5 points!*** ***Your output will not print in bold.***

```
Do you want to proceed with your candy purchase? Enter 'Y' or 'N': n

Thank you for visiting Fairytale Sweets! Please come again.

*******OUTPUT WHEN THERE ARE CANDY PURCHASES – THE DOLLAR AMOUNTS IN YOUR
SALES RECEIPT MAY NOT LINE UP EXACTLY – THE MISALIGNMENT CAN RESULT FROM
THE FONT THAT IS USED. TRY MONOSPACED OR COURIER NEW IN DRJAVA*******
Do you want to proceed with your candy purchase? Enter 'Y' or 'N': y

FAIRYTALE SWEETS

1.  Arabian Nights Chocolate Coins - 1 lb. Bag      $  2.25
2.  Beauty and the Beast Lollipops - 1 lb. Bag        2.50
3.  Mad Hatter Jelly Beans - 1 lb. Bag                1.75
4.  Pinocchio's Candy Cones - Each                    0.75

5.  Sleeping Beauty Caramel Apples - Each             1.25

Enter your choice:  0

Invalid candy choice!  Try again.

FAIRYTALE SWEETS
1.  Arabian Nights Chocolate Coins - 1 lb. Bag      $  2.25
2.  Beauty and the Beast Lollipops - 1 lb. Bag        2.50
3.  Mad Hatter Jelly Beans - 1 lb. Bag                1.75
4.  Pinocchio's Candy Cones - Each                    0.75

5.  Sleeping Beauty Caramel Apples - Each             1.25
Enter your choice:  6

Invalid candy choice!  Try again.
```

```
FAIRYTALE SWEETS
1.  Arabian Nights Chocolate Coins  - 1 lb. Bag      $  2.25
2.  Beauty and the Beast Lollipops  - 1 lb. Bag         2.50
3.  Mad Hatter Jelly Beans - 1 lb. Bag                  1.75
4.  Pinocchio's Candy Cones - Each                      0.75

5.  Sleeping Beauty Caramel Apples - Each               1.25

Enter your choice:  l

ONLY enter an integer!  Re-enter your choice:  1

Quantity for Arabian Nights Chocolate Coins:  3

Do you want to make any changes to the following selection?
Arabian Nights Chocolate Coins

     3 @ $2.25 ea.                $6.75

Enter 'Y' or 'N':  y

FAIRYTALE SWEETS
1.  Arabian Nights Chocolate Coins - 1 lb. Bag      $  2.25
2.  Beauty and the Beast Lollipops - 1 lb. Bag         2.50
3.  Mad Hatter Jelly Beans - 1 lb. Bag                  1.75
4.  Pinocchio's Candy Cones - Each                      0.75

5.  Sleeping Beauty Caramel Apples - Each               1.25

Enter your choice:  2

Quantity for Beauty and the Beast Lollipops:  3

Do you want to make any changes to the following selection?
Beauty and the Beast Lollipops

     3 @ $2.50 ea.                $7.50
Enter 'Y' or 'N':  n

Would you like to make another candy purchase?  Enter 'Y' or 'N':  y


FAIRYTALE SWEETS
1.  Arabian Nights Chocolate Coins - 1 lb. Bag      $   2.25
2.  Beauty and the Beast Lollipops - 1 lb. Bag          2.50
3.  Mad Hatter Jelly Beans - 1 lb. Bag                  1.75
4.  Pinocchio's Candy Cones - Each                      0.75

5.  Sleeping Beauty Caramel Apples - Each               1.25
```

```
Enter your choice:  4

Quantity for Pinocchio's Candy Cones:  %

ONLY enter an integer!  Re-enter the quantity:  5

Do you want to make any changes to the following selection?

Pinocchio's Candy Cones
     5 @ $0.75 ea.               $3.75

Enter 'Y' or 'N':  n

Would you like to make another candy purchase?    Enter 'Y' or 'N':  n

Proceeding to checkout.


FAIRYTALE SWEETS
North Star Mall
San Antonio, TX

Date:  02/19/20
Time:  08:44:45 PM

 Beauty and the Beast Lollipops
     3 @ $2.50 ea.                          $     7.50
 Pinocchio's Candy Cones
     5 @ $0.75 ea.                                3.75
                          SUBTOTAL:         $    11.25
                      TAX @ 8.250%:                0.93
                             TOTAL:         $    12.18
Thank you for your business!  Please come again.
```