Bluegrass Community and Technical College

# Programming Requirements Document

## Vendor Purchases

## NARRATIVE DESCRIPTION

Reading for this assignment:

This application will involve calculations which represent money. You should NEVER use floating-point numbers (float or double) for monetary calculations because these data types are not precise. To better understand, this problem (for all programming languages not just Java) read the following:

- http://www.javapractices.com/topic/TopicAction.do?Id=13

It is a bit premature for us to use the BigDecimal class just yet. For now, we will use integers to represent money in pennies. If you are interested in using the BigDecimal class, you may read about it at:
https://docs.oracle.com/javase/10/docs/api/java/math/BigDecimal.html
If you choose to use BigDecimal, email your instructor prior to submitting this assignment to receive permission.

Formatting is the "cosmetics" of input and output. Formatting can provide a pleasing, easy-to-read, and easy-tounderstand interface for the user. From this point forward in the course, you are required to use professional formatting when designing user interfaces (what the user sees). *This will be a standard requirement for all future assignments.*

Working with string data is extremely common in "real" programming environments. String manipulation is a fundamental programming skill.

For this assignment, use only the String methods illustrated in your textbook readings up to this point. Your instructor is assessing your knowledge of specific methods.

During the summer months, Midway, KY, hosts *Midsummer Nights in Midway.* The event offers live music, shopping late at local shops, face painting, kids activities, and food vendors. One vendor sells kettle popcorn ($4.50 a bag) and soft drinks ($1.00 each) on a cash-only basis.

Develop a Java application that will print (display) receipts for popcorn vendor (to give to customers). Since this is a monetary application and we want to be accurate with our calculations, use integers for all monetary values (representing the values in terms of pennies) until the final answer is calculated at which time you can divide by 100 to get the correct floating-point value. For example, $4.50 would be 450 pennies.

Input: Using the Scanner class, the application should input the following:

- date of purchase
  - In format mmddyyyy – for example: 09172016) *
  - Make sure you do not input month, day and year separately
- buyer's first name
- buyer's last name
- the number of bags of popcorn purchased
- the number of soft drinks purchased
- a street vendor's ID (any 5 characters) *

Create any constants you may need for this program.

** <u>Items that contain all digits are not necessarily numeric values.</u> You should declare a data item (variable or constant) as an integer or floating-point value ONLY IF it is reasonable to use that data item in a mathematical calculation. For example, zipcodes are all numeric in the U.S. but we never use them in calculations. We do not total zipcodes, we do not calculate averages of zipcodes, etc. A zipcode should be declared as a String. The same would be true for social security numbers, dates, phone numbers, credit card numbers, student IDs, vendor*
*IDs, etc.*

HINT: If you attempt to read a String (nextLine() method) after reading a number (nextByte(), nextInt(), nextFloat(), or nextDouble() methods), you will first need to:

- Write a nextLine() method to clear the input buffer (this reads the enter from the previously read number
- Write another nextLine() to read the actual String you want

<u>Processing</u>

Generate a 4-digit random number in the range 1000-4999. Make sure 1000 and 4999 are included in the range of numbers that could be generated.

Using String methods and concatenation, build a confirmation "number" for the buyer and store it in a String variable. It should include:

- the string POPCORN followed by
- the character #, followed by
- the buyer's last name, followed by
- the character #, followed by
- the first initial of the buyer's first name (use the substring() method), followed by
- the character #, followed by
- the 4-digit random number generated above (use the Random class methods), followed by
- the character #, followed by
- the street vendor's 5-character ID

An example of a confirmation for Cindy Tucker from vendor AB18A could be:

POPCORN#TUCKER#C#3418#AB18A

Create a String that hold the date with dashes included. For example, 09172018 would become 09-17-2018. <u>Use the substring() method to build the new date.</u>

NOTE: We have not studied selection statements yet so no error checking on data will be done yet. Concatenation allows you to append (add) strings together. You can use + or the concat( ) method for concatenation. The substring( ) method can extract characters from a string to help build new strings.

Calculate the amount the customer owes.

Output

The "receipt" to be printed should use the format below (you can simply display the receipt on the screen rather than print it). When you are ready to display the amount the customer owes, you divide the pennies by 100 and cast to a double variable. Casting is discussed in the textbooks.

Use the DecimalFormat class for formatting currency values. Pay attention to alignment and make sure 1 decimal to the left and 2 decimal places after the decimal point should always be display (make sure you understand 0 vs # in the mask, such as $###,##0.00). For example, do not allow: $207.1 instead of $207.10.

An example of a receipt could be (use this format – the actual data may vary depending upon user input):

```
** Kandy's Kountry Kettle Korn **

Confirmation for Cindy Tucker

Confirmation Number: POPCORN#TUCKER#C#3418#AB18A

Popcorn:        3 @ $4.50 each
Soft Drinks:    5 @ $1.00 each

TOTAL:          $18.50

Thanks for visiting our booth on 09-17-2018
```

## New Concepts Assessed and Illustrated (in addition to any previously learned)

- String Class and methods
- DecimalFormat Class
- Random Class and methods
- Formatting output in a pleasing manner
- Monetary calculation using integers

## Software Requirements (Checklist to Avoid Missing a Feature)

Standard Requirements (expectations for all programming assignments – even when not list in future assignments)

- Include a comment block at the top of your program to clearly identify (1) your name (2) the current date (3) your instructor (4) your class and (5) the purpose of this program (Java class). Significant penalty for missing this requirement and for poor documentation.
- Use the Programming Standards document introduced in Module 1 to apply documentation standards correctly, consistently, and thoroughly in all Java classes submitted for grading. Use "boxed" comments. There is a significant penalty for missing this requirement and for poor documentation.
- Zip your file(s) for grading.
- Submit code that is free of compile-time errors. *No credit is given for assignments with compile errors.*
- Incorporate what you have previously learned into this assignment. For example, all input prompts should be clear to the user as to what they are to key. All output should be clear to the user as to what they are viewing as the results, including headings for the application. Select an appropriate and efficient data type for all variables and constants. Use of constants where appropriate. Etc. Deductions based upon what was overlooked.

Requirements Specific to this Assignment

- Use integers for monetary calculations (pennies) and convert answer to decimal values when you are ready to display answers.
- Use the String methods from the textbook readings. Do not search for other String methods. This assignment is designed to assess your knowledge of the methods discussed in the textbooks.
- Use the Random class to generate a 4-digit number in the proper range.
- Use the DecimalFormat class to format monetary values.
- Format and display a receipt in a prescribed format.

## Security Considerations

Selecting incorrect data types can have security implications. It is very important that you select the correct data type (and size) for each variable and constant you use. We do not want to waste memory and yet we must insure the data type is large enough to hold the data to be stored in memory.

## Special Notes

Data types of float and double are never used for money values. We are using integers to represent numbers (or BigDecimal by permission).