Bluegrass Community and Technical College
CIT 149: Java I
Formatting Output and String Manipulation

Formatting is the "cosmetics" of input and output. Formatting can provide a pleasing, easy-to-read, and easy-to-understand 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).**

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 the textbook in Chapter 3. I am assessing your knowledge of specific methods.

The inaugural running of a $12 million Thoroughbred horse race, the Pegasus World Cup, took place in Florida in January 2017. The Pegasus World Cup, a 1 ⅛ mile dirt-track race, is designed to promote the sport of Thoroughbred racing and is/was expected to attract some of the greats from around the world.

At the race, street vendors sold Pegasus World Cup souvenirs. **Pegasus Glasses** sold glass etched and brass engraved tumblers. They went high-tech for the race and their street vendors carried hand-held mobile devices with attached printers.

When a person purchased a tumbler from the vendor (credit and debit cards only), the vendor gave the customer a printed receipt along with the item(s) purchased. The prices of the tumblers were $37.15 for each glass tumbler and $39.30 for brass tumblers (tax included in the price).

Develop a Java application that will print (display) receipts for **Pegasus Glasses**. 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, $37.15 would be 3715 pennies.

Input: Using the Scanner class, the application should input and store in variables for:

- date of purchase (in format mmddyyyy – for example: 09172016)
- buyer's first name
- buyer's last name
- 16 digit card number (use a String* for this item) – see note on the next page
- the number of glass tumblers purchased
- the number of brass tumblers purchased
- a street vendor's ID (any 5 characters)

Create any constants you may need for this program.

** Items that contain all digits are not necessarily numeric values. 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, phone numbers, credit card numbers, 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 issue a nextLine() to clear the input buffer and then issue another nextLine() to read the String you want.

Processing

Generate a three-digit random number in the range 100-700. Make sure 100 and 700 are included in the range of numbers that could be generated.

Build a confirmation "number" for the buyer and store it in a String variable. It should include:

- the string PEGASUS17 followed by
- the character @, followed by
- the first initial of the buyer's first name, followed by
- the buyer's last name, followed
- the three-digit random number generated above, followed by
- the character @, followed by
- the street vendor's 5-character ID

An example of a confirmation for John Doe could be: PEGASUS17@JDoe521@VEN12

Create a String that hold the date with dashes included. For example 09172016 would become 09-17-2016.

HINTS: 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.

Output

The "receipt" to be printed should use the format on the next page (you can simply display the receipt on the screen rather than actually printing it).

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. 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):

** Pegasus Glasses Receipt **

Confirmation for Allison Browning
Purchased on 09-17-2016

Confirmation Number: PEGASUS17@ABrowning207@VEN12

Glass Tumblers: 2 @ $37.15 each
Brass Tumblers: 3 @ $39.30 each

TOTAL: $192.20

Thanks for visiting us at the Pegasus World Cup

NOTES:
Make sure your program is fully documented and contains a comment block at the beginning with your name, date, and purpose of the program. Use comments throughout the program to explain the code. Use meaningful variable names. Use constants for the prices glass and brass tumblers. Label and format all output appropriately. Grading will reflect quality and effort as well as correctness. As always, **only submit the zipped .java file for grading.**

The program this week did not explore the Math class fully. Make sure you know how to use the methods as you may need these later in the course