

## CSCI165 Computer Science II

### Lab Assignment

Due Friday 2/7/2020 by 9:00 AM

#### Objectives:

- Utilizing variables and types in Java
- Understanding console and file input/output
- Researching the Java API
- Processing String data
- Formatting output with **printf()** and the **DecimalFormat** class
- Using Java arrays

This solution should be created in a source code file named **FoodTruck.java**

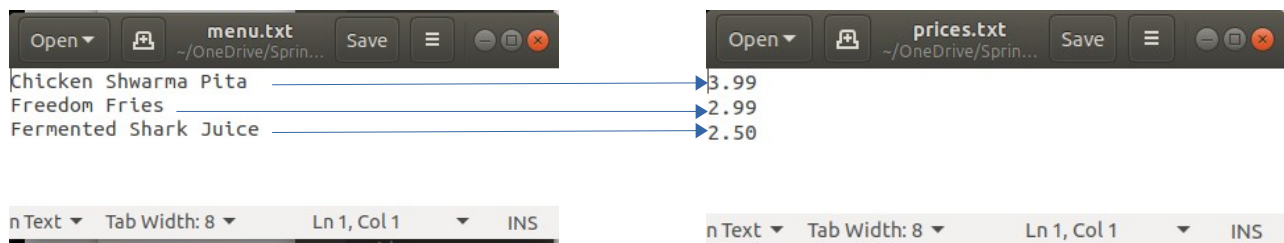
You are creating a point-of-sale program for a food truck that only sells **3 items at a time**. Each week there is a different menu. Here is an example.

1. Chicken Shawarma Pita - 3.99
2. Freedom Fries - 2.99
3. Fermented Shark Juice - 2.50

The menu items are stored in a file called **menu.txt**

The customer orders by entering quantities for each item into the console of a Commodore 64. Prices for the items vary according to the time of day and demand, and are therefore stored in a separate file called **prices.txt**. The prices are in the same order as their corresponding items. This file is updated periodically during the day. The program must read these files each time a customer makes an order.

The item name may contain spaces and multiple words, so keep this in mind when reading the data. Approach this as if you **will not be informed** of how many words an item name contains, so you will have to grab the entire line, including spaces.

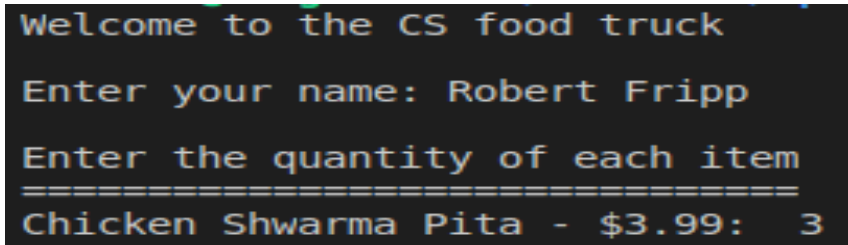


When the program loads, read the contents of

- **menu.txt** into a 3 element array of type String called **menu**
- **prices.txt** into a 3 element array of type double called **prices**

Use a while loop with a counter as an index to accomplish this. Control the loop with a **hasNext** call.

When the program runs it should ask for the customer's full name. **Use a single string variable for the name.** It should then loop through the menu and prices array, displaying the name and price for each item with a prompt for the customer to enter the quantity they would like to order. Here is an example:



Output a receipt with a tax rate of 6.25%. All prices should be output to two decimal places. The **DecimalFormat** class is a Java class that handles all types of number formatting. Use the Java API to research this class.

Include the current date and time in **MM/DD/YYYY** and **HH:MM:SS** format. The date and time should reflect the actual date and time of each program execution. Research the following classes

- `DateTimeFormatter`
- `LocalDateTime`

**The receipt should be output to two locations:**

1. The console window
2. An external file

**Format the bill as a traditional receipt:** The bill should be formatted in columns with 30 characters for the name, 10 characters for the quantity, 10 characters for the price, and 10 characters for the total. Proper alignment must be maintained.

The **printf()** function (print with format) will be quite useful for this operation on the console window. as it allows us to specify formatting conditions as a String argument.

- `System.out.printf("formatting conditions", var_to_format);`

The name and zip will be used to create an **invoice number**.

**The invoice number will be composed on the following**

- the first two initials of the customer's first name, converted to uppercase
- the first two initials of the customer's last name, converted to uppercase

**Create a unique Invoice ID by doing the following**

- Take the Unicode value of the first character of the first name
- Take the Unicode value of the first character of the last name
- Add them together and multiply by the length of the full name
- Concatenate this ID to the initials described above
- Concatenate the day, month, hour and minute with no punctuation

**Example:** Jim Lahey 1/27/2020 11:10

- Initials: JILA
- (J = 74) + (L = 76) => 150
- Length of "Jim Lahey" => 9
- $150 * 9$  => 1350
- Invoice Number: JILA13501271110
- It should go without saying that this process should work for any name, any date and any time

Research the String API for methods to locate spaces (indexOf) and dissecting Strings (substring) and converting to uppercase.

Your code must be built and executed from the command line

**Sample run shown as follows**

```
Welcome to the CS food truck
Enter your name: Jim Lahey
Enter the quantity of each item
=====
Chicken Shwarma Pita - $3.99: 3
French Fries - $2.99: 3
Fermented Shark Juice - $2.50: 3
```

Greeting and Input

```
Invoice Number:  JILA13501271110
Date:           1/27/2020
Time:           11:10:28

Item            Quantity    Price  Total
=====
Chicken Shwarma Pita    3      $3.99  $11.97
French Fries            3      $2.99  $8.97
Grapefruit Juice       3      $2.50  $7.50
=====
Subtotal                                $28.44
6.25% sales tax                        $1.77
Total                                $30.21
```

Output

**Requirement:** This invoice should be displayed on the console window **and** written to a file. The file name should be the invoice number with a .txt extension. Think of clever ways to not have to actually build that invoice twice.

**Methods:** This program could easily benefit from functional decomposition. While it is not required, feel free to experiment with static method definitions. Treat them just like you would functions in Python or C++.

**Submission:** Save your source file into the **lab directory** inside **week-2**. Push to your remote repository

Rubric: 20 points

Requirement	Points
<b>Style:</b> descriptive variable names, appropriate comments and indentation	5
<b>Correctness:</b> Code functions correctly, math is correct, invoice number created, output matches	10
<b>Structure:</b> arrays and loop used appropriately	5