CIS 3090 Project (100 points)

Turn in: In NetBeans, export your NetBeans project to a zip file, name it as **YourLastName_FirstName_project.zip**.

Requirements description:
Assume you work part-time at a Cafe. As the only employee who knows java programming, you help to write an ordering application for the store.

The following is a brief requirement description with some sample output.

1. Selecting Coffee (15 points)
When the program starts, it first shows a list/menu of coffee and their prices, then asks a user to select a coffee by entering an integer number. A sample output is as follows.

```
=== Select Coffee: ===
1 Espresso    $3.00
2 Latte       $4.75
3 Cappuccino  $4.00
4 Cold Brew   $4.00
Select a coffee [1, 4]:
```

You are supposed to validate the input.
If the user **enters a letter or a number not between 1 and 4**, the user will see an error message.

A sample output for invalid number is as follows.

```
Select a coffee [1, 4]: 0
Error! Number must be greater than 0.
Select a coffee [1, 4]:
```

In your program, you can **hard-code** the information for coffee (i.e., coffee names and prices) shown above, such as "`1 Espresso $3.00`" and use the hard-code price, such as 3.00, for calculation of a total price of the order.

2. Selecting Add ons (20 points)

After the user provides a right coffee number for coffee selection, the program asks the user to select Add ons. A sample output is as follows. Again, input validation is needed.

```
=== Select Add ons: ===
1 Extra Espresso $1.50
2 Panna          $1.50
3 Oat Milk       $1.00
4 Almond Milk    $1.00
5 Quit Add on selection
Select Add ons: [1, 5]:
```

You can hard-code the Add ons information as shown above, such as "`1 Extra Espresso $1.50`" and use the hard-code price, such as 1.50, for calculation of the total price of the order.

After the user makes a choice for Add ons, such as 2 for Panna. The program continues asking for selecting an Add on so that the user can have multiple Add ons for one coffee. The user can enter "5" to quit Add ons selection. A sample output is as follows.

```
=== Select Add ons: ===
1 Extra Espresso $1.50
2 Panna          $1.50
3 Oat Milk       $1.00
4 Almond Milk    $1.00
5 Quit Add on selection
Select Add ons: [1, 5]:  2
=== Select Add ons: ===
1 Extra Espresso $1.50
2 Panna          $1.50
3 Oat Milk       $1.00
4 Almond Milk    $1.00
5 Quit Add on selection
Select Add ons: [1, 5]:  5
```

3. Selecting Food (10 points)
After Add ons selection, the program shows food selection. A sample output is as follows.

```
=== Select Food: ===
1 Tuna    Sandwich    $11.00
2 Chicken Sandwich    $9.00
3 Burrito             $10.00
4 Yogurt Bowl         $9.00
5 Avocado Toast       $8.00
Select Food: [1, 5]: 1
```

Input validation is needed and works as before. Unlike Add ons selection which allows selecting multiple Add ons, food selection does not repeat after the user enters a valid number between 1 and 5.

You hard-code the information for food shown above, such as "1 Tuna Sandwich $11.00" and use the hard-code price, such as 11.00, for calculation of the total price of the order.

4. Entering a customer's name (10 points)
After making an Food selection, the program asks for the user's name so that the user can enter text like *John* or *John Smith*.

```
=== Select Food: ===
1 Tuna    Sandwich    $11.00
2 Chicken Sandwich    $9.00
3 Burrito             $10.00
4 Yogurt Bowl         $9.00
5 Avocado Toast       $8.00
Select Food: [1, 5]:  4
Enter customer's name: John Smith
```

5. Displaying and writing order line information (30 points)

After entering a name, the program prints details for this order line in computer monitor. The information includes the six fields: date and time, customer name, coffee, Add on(s), food, and a (formatted) total price, and each field is separated by a comma and a space. A sample output to the monitor is as follows.

```
Enter customer's name: John Smith
```

```
Jul 6, 2016 10:35:43 AM John Smith, Espresso, Extra espresso shot, Almond
Milk, Burrito, $15.50
```

For your reference, the date and time is created by the following statements.

```
Date now = new Date();
DateFormat defaultDate = DateFormat.getDateTimeInstance(DateFormat.MEDIUM, DateFormat.MEDIUM);
String time = defaultDate.format(now);
```

Multiple Add ons are separated by a comma and a space

Besides showing on monitor, the same order line content is written (appended) to a text file named *order.txt* and each order line occupies one line in the text file. Here is a specific example for an order in the order.txt.

```
Jul 6, 2016 10:35:43 AM John Smith, Espresso, Extra espresso shot, Almond
Milk, Burrito, $15.50
```

Your program needs to create the output file, named order.txt, if it doesn't exist yet, and later appends (**not overwrite**) order line information to the file. So next time when your program is executed, new order information will be appended to the order.txt file. You must use a **relative path** when creating an output stream writer object. For simplicity, we don't record any information about the sales clerk and one order includes only one coffee.

Design requirements: (15 points)

You must have at least the following java classes. You may have additional classes if you want.

[1] a **CafeOrder** class to simulate the CafeOrder entity in real world, which has fields of coffee, Add ons, food, and price of the order, customer name, the string value of time stamp (refer to the prior sample code) and corresponding methods. This class also provides a method to append the content of an order's content to the output text file, order.txt. (**CafeOrder.java**)

[2] a java application, named **CafeApp.java**, that contains a main method. This class interacts with CafeOrder class.

Suggestions:

• Start early and start from simple. For example, leaving the multiple add ons selections, validation and writing to file, etc, to a later stage.

• When incrementally improving your programs, make sure to keep a clean copy of your completed program. In case you couldn't complete the whole project in time, you can still submit a working solution of partial project and receive partial points.