CS 123/231 Introduction to Software Engineering **1st Sem 2004-2005**
Programming Assignment #1 **Due: 19 July 2004**

FAST FOOD ORDER COUNTERS

Write Java programs that simulate the activity of point-of-sale counters in a fast food restaurant. Assume that the place sells hamburgers, cheeseburgers, fries, and drinks that come in small and large sizes. In this restaurant, there are 5 counters that accept orders and payments.

Write an application *and* an applet. The application will take input from a text file (named ***orders.txt***) that contains all order data. Each line from the file corresponds to an order and indicates the counter where the order was made as well as the order details, using the following format:

counter <Counter Number>,<Item> <Quantity>,<Item> <Quantity>,…,**end** <Amount Tendered>

where

<Counter Number>	is an integer from 1-5 indicating the counter where the order was made
<Item>	is one of the following strings: **hamburger**, **cheeseburger**, **fries**, **smalldrink**, **largedrink**.
<Quantity>	is an integer that indicates the quantity of the specified item to be included in the order.
<Amount Tendered>	is the amount given by the customer as payment.

Note that the fields are separated either by single commas or single spaces. The following is an example of an order line:

```
counter 4,hamburger 2,fries 1,smalldrink 2,end 100.00
```

The application, as it processes each line, prints the total price of the order and the change returned by the cashier (the difference between the amount tendered and the total price). In the case where the amount tendered is less than the total price, the order is cancelled and therefore not completed (assume that this means that the customer does not have enough money). The line printed for each order will follow one of these two formats:

Counter <Counter Number>, Total price: P<amount>, Change returned: P<amount of change>
Counter <Counter Number>, Total price: P<amount>, Amount tendered is insufficient

The total price is computed by first multiplying the unit prices of ordered items by their corresponding quantities; the sum of all these products is the total price. The price for an item will be determined by invoking a **getPrice()** method on a **PriceChecker** object. The Java program for a **PriceChecker** class will be provided.

After all lines have been processed, the program will print the statistics for each counter. In particular, for each counter, the following information will be printed out, in the specified format:

```
Counter <Counter Number>
   Total orders completed: <Numbers of orders completed>
   Total orders cancelled: <Numbers of orders cancelled>
   Total cash: P<Total amount collected>
```

All output should be sent to a file called ***counters.txt***. All amounts should be written with two decimal places. The following are examples of output lines for an order and for a counter, respectively, where ~ counts as a space.

```
Counter~2,~Total~Price:~P50.40,~Change~returned:~P0.60
```

```
Counter~4
~~~Total~orders~completed:~10
~~~Total~orders~cancelled:~10
~~~Total~cash:~P503.80
```

You are required to specify and use a class called **FastFoodCounter** from which you will create five **FastFoodCounter** objects. The class should encapsulate all information related to a counter. You are *required* to have at least the following public methods for the **FastFoodCounter** class:

void beginOrder();
/* signifies that a new order is being processed */

void add(String prd, int qty);
/* adds items to the current order, in particular, qty units of product prd */

double getTotalPrice();
/* returns the total price for the current order */

void completeOrder();
/* called after an order has been accomplished, i.e., the payment has been made */

void cancelOrder();
/* called to cancel the current order */

int countOrders();
/* returns the total number of completed orders */

double countCash();
/* returns the total cash collected */

You will then write the program that creates the counter objects, processes the input file, and then prints all the specified output onto the output file. You may assume that the input file is in correct format and that the data always makes sense (no error handling needed). The application class should be named **FastFood.**

Write another program, this time an applet, called **FastFoodApplet** that carries out the order processing in an interactive way. That is, instead of processing an input file, use panels, fields, and buttons to carry out an order. Have a special output area that displays the results of a processed order. Have a special button that prints the counter statistics as needed. Of course, I expect you to reuse the **FastFoodCounter** class (and perhaps the **FastFood** class**).**

The **PriceChecker** class and a test input file is available on the course website. A corresponding output file will be ready by Wednesday, 14 July.

The program is due on Monday 11:59pm, 19 July. Email (to jpv@curry.ateneo.net) a zip file containing all source programs by the due date. Submit printouts of all source programs placed in a brown envelope with your name on it, in class on Tuesday. There is no need to submit a diskette.