COMP 1006/1406
Assignment 1 – Carleton Foodorama (Part 1)

Due: Friday, July 14th 2006, before 11h55 pm

In this assignment, you will review basic concepts from COMP 1005. You will get to design a simple text-based interface and a simple GUI window.

Part A (Reading a File):

On the web page, there is a file called menu.txt which contains a menu restaurant. Each line contains an item on the menu as well as its price. It is formatted as below:

```
hot-dog 1.25
hamburger 2.00
cheeseburger 2.75
fries 1.75
poutine 3.75
pizza 10.00
drink 1.50
```

[10 pts] Write a java class called **MenuItem** which represents an item on the menu. A MenuItem has two private instance variables: name (of type String) and price (of type float). You have to write:

- A constructor taking a name and a price as arguments.
- The get() methods for each private variable.
- A toString() method returning the name of the item, followed by a space and its price. The price should be displayed using exactly two decimal digits. In order to do this, you can use the DecimalFormat class of the java API.

[20 pts] Write a java class called Menu which represents a restaurant menu. It has one private instance variable: items (of type ArrayList of MenuItem). You have to write:

- A constructor taking an ArrayList of MenuItem as argument.
- A constructor taking a file name (of type String) as argument. The file is formatted like menu.txt. Each line corresponds to a MenuItem. You can assume that item names do not contain spaces. You have to read the file line by line and add each MenuItem to the ArrayList of items. In order to read the file, you can use the Scanner class of the java API. You have already used that class in COMP 1005
- A print() method which displays the menu on the screen in a command line. Each menu item has to be displayed on a separate line. Both the name and the price should be displayed. Also, menu items should be numbered starting from 1 as shown below:

```
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
```

- A size() method which returns the number of items on the menu.
- A get(int i) method which returns the ith item on the menu.

[25 pts] Write a java class called Order which represents a customer order. An Order has two private instance variables: items (an ArrayList of Menuitems) and quantities (an ArrayList of Integers). You have to write:

- A zero-argument constructor.
- An add() method which takes both a menu item and a quantity. If the item is not already present in the order, both the item and the quantity are appended to the lists. If the item is already present, the quantity should be added to the one that is already there.
- A getSubTotal() method which returns the total amount of the order before taxes.
- A getTaxes() method which returns 15 percent of the sub-total.
- A getTotal() method which returns the sub-total plus the taxes.
- A toString() method which will be used to print the bill. Each item should appear on a separate row, followed by the price and the quantity. The sub-total, the taxes and the total should also appear on separate rows as shown below:

```
hot-dog 1.25 x 3
hamburger 2.00 x 2
fries 1.75 x 1
drink 1.50 x 1
Sub-Total: 11.00
Taxes: 1.65
Total: 12.65
```

Part B (The User Interface):

[20 pts] Write a java class called Cashier which represents a restaurant cashier. A cashier has one private instance variable called menu (of type Menu). You have to write:

- A constructor taking a menu as argument.
- A private printListOfCommands() method which asks the customer about what he wants to do. The first lines correspond to ordering menu items. You should use the print() method you have done for the Menu class. The last line terminates the order. The customer expresses his desire by choosing a command number as shown below:

```
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
```

- A private readNextCommand() method which reads the next custommer command. It must return the integer typed by the user. Once again, you may use the Scanner class of the java API.
- A private askForQuantity() method which asks the customer for the quantity of a given item he

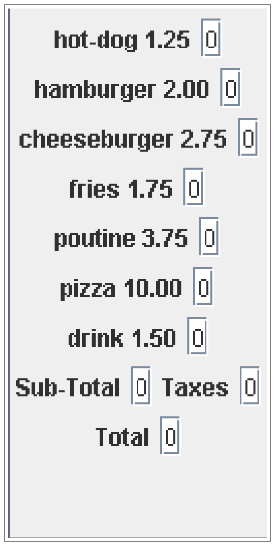
wants. It must return the integer typed by the user. Once again, you may use the Scanner class of the java API.

- A takeOrder() method which takes and **returns** a customer order.
    1. It first says welcome to the customer.
    2. Then, it prints the list of commands the customer can do and reads it.
    3. If he wants to order an item, it asks the customer for the quantity that he wants and add it to the order. It then goes back to step 2.
    4. If the customer is done, the order is printed and the method returns.

[5 pts] Write a java class called Restaurant. A restaurant has no instance variable and no instance method. You have to write:

- A main() method which first create a menu object from the file menu.txt.
- It should then create a Cashier which will use that menu.
- Finally, the cashier should take an order and the program should exit.
- A full example of a program run is provided in Appendix 1.

Part C (Designing a Simple GUI):



hot-dog 1.25 0
hamburger 2.00 0
cheeseburger 2.75 0
fries 1.75 0
poutine 3.75 0
pizza 10.00 0
drink 1.50 0
Sub-Total 0 Taxes 0
Total 0

[15 pts] Write a java class called OrderForm which extends JPanel. This will be the basic component to display the menu. You have to write a constructor taking a menu as argument.

- For each menu item, you have to
    - create a JLabel displaying the name and the price of the item,
    - create a JTextField for the user to write the quantity he wants to order.
- You also have to create three JLabels and three JTextFields for the sub-total, the taxes and the total. For this assignment, those fields **do not have to be operational**.
- For this assignment, **no fancy layout is required**. It is sufficient to have a layout as shown on the right.

[5 pts] Write a java class called OrderFrame which extends JFrame. This will be the main application frame. You have to create:

- A constructor which takes a menu as argument. In the constructor, an OrderFrame is created and added to the frame. The proper size is then set and the frame is displayed.
- A main() method which first create a menu object from the file menu.txt. It then creates an OrderFrame using that menu.

Submission

As usual, you will submit to the Raven system. You should submit an entire directory that contains all of your java files, your class files and any testing/output files that you created.   Also, have a readme.txt file that clearly indicates which files are which.

Good Programming and Good Practice Requirements.

These requirements pertain regardless of what your application is supposed to do (i.e. regardless of the design requirements). These requirements are to ensure that your code is readable and maintainable by other programmers (or TA's in our case), and that your program is robust (It does not crash from bad object references). You will loose 5 marks from your total assignment mark for each of the following requirements that are not satisfied. If you do not satisfy requirement R0 you will get nothing for the assignment.

R0) IMPORTANT Uniqueness Requirement. The solution and code you submit MUST be unique. That is, it cannot be a copy, or be too similar, to someone else's assignment, or other code found elsewhere. A mark of 0 will be assigned to any assignment that is judged by us or the TA's not to be unique.

R1) All of your variables, methods and classes should have meaningful names that reflect their purpose. Do not follow the convention common in math courses where they say things like: "let x be the number of customers and let y be the number of products...". Instead call your variables numberOfCustomers or numberOfProducts. Your program should not have any variables called "x" unless there is a good reason for them to be called "x". (One exception: It's OK to call simple for-loop counters i, j and k etc. when the context is clear.)

R2) All variables in your classes should be private, unless a specific design requirements asks for them to be public (which is unlikely). We will design objects that provide services to others through their public methods. How they store their variables is their own private business.

R3) Robustness Requirements: Your program should never crash because of a "null pointer exception". This exception means that you are using a variable that does not actually refer to an object. We instruct the TA's to try and crash your program for this reason so guard against it.

R4) Comments in your code must coincide with what the code actually does. It is a very common bug in industry for people to modify code and forget to modify the comments and so you end up with comments that say one thing and code that actually does another. (By the way, try not to over-comment your code; instead choose good variable names and method names which make the code more "self commenting".)

R5) Java 1.5 Requirement: Your code should compile with the Java 1.5SDK without warnings.

Appendix 1 (Example for Part A):

```
Welcome to Carleton Foodorama!

What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
1
How many do you want?
3
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
2
How many do you want?
1
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
4
How many do you want?
1
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
```

```
7
How many do you want?
1
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
2
How many do you want?
1
What do you want?
1)  hot-dog 1.25
2)  hamburger 2.00
3)  cheeseburger 2.75
4)  fries 1.75
5)  poutine 3.75
6)  pizza 10.00
7)  drink 1.50
Press 8 if you are done.
8
hot-dog 1.25 x 3
hamburger 2.00 x 2
fries 1.75 x 1
drink 1.50 x 1
Sub-Total: 11.00
Taxes: 1.65
Total: 12.65

Press any key to continue...
```