48024 Applications Programming
Assignment 1: Snacks

Topics: OO design, lists, files **Value:** 30%
Objectives: This assignment supports subject objectives 1 - 3.
Due date: Friday 26 September, 6:00 pm

1. Individual work

All work is individual. You may discuss ideas, approaches, and problems, but you should write every line of code except for code copied from the lecture notes or lecture code. More detail can be found at http://www.gsu.uts.edu.au/rules/student/section-16.html

2. Specification

A fast food store has asked you to build an online ordering system. An order is input as a single string that mixes single digit numbers and snack ids in any order, such as

`b2fc` = burger, 2 fries, coke

f`bfc` = burger, 2 fries, coke

The snacks are identified by the first character of the name. They are in three groups; the groups and cost per serve are shown below. The kitchen has 3 of each snack in stock, so they may run out of a snack. If there are not enough serves of a snack left to fill the order, ask the customer if they want to substitute a similar snack (a snack in the same group); repeat as needed. When an order has been filled, show the filled order and the cost.

Main meal
burger, $3.45
wrap, $4.32
Sides
fries, $4.20
muffin, $2.50
hashBrown, $4.32
Drinks
coke, $1.23
koffee, $2.34
slurpee, $3.45
tea, $4.56
juice, $5.67

The menu has 6 choices; a choice is a single character input:

- o: order
- z: zet (set) favourite order
- u: use favourite order
- s: show the stock left
- f: exit the menu and store to file
- x: exit the menu
- anything else: help

The system always read from file on startup, if a file exists. IO traces are shown in io.pdf (formatted, with explanation) and in io.txt (straight text) in Assignments/1.

2.1 Special conditions

- Use LinkedLists. Do not use arrays, or arrayLists, or Maps, or anything but LinkedList. Each non-LinkedList data structure loses 20 marks.

3. Expected work load

The time to do the tasks worth 60/80 (see Section 9, IO traces 0-6, 12) has been estimated at 15 hours for the average student who has completed all the tutorial and lab exercises. The remaining 20 marks (IO traces 7-11) are harder to get; some students may not get them, so any estimated time may be misleading. These tasks have been given an estimated time of 10 hours.

4. Online support

FAQs (Frequently Asked Questions) and their answers are posted on UTSOnline in **Assignments/1/faq.** If you have a question, check the FAQ first; it may already be answered there. You should read the FAQ at least once before you hand in your solution, but to be safe check it every couple of days. Anything posted on the FAQ is considered to be part of the assignment specification. The FAQ will be frozen (no new entries) two days before the due date; no questions will be answered after it is frozen.

If anything about the specification is unclear or inconsistent, contact me and I will try to make it clearer by replying to you directly and posting the common questions and answers to the FAQ. This is similar to the job experience, where you ask your client if you are unsure what has to be done, but then you write all the code to do the task. Email ***Robert.Rist@uts.edu.au*** to ask for any clarifications or corrections to the assignment.

I cannot tell you how to design, write, or debug your code; that is your task. I cannot answer questions of the form "Is this right?" I cannot "pre-mark" your design, or your code. There is no discussion board; the assignment tests your individual ability to solve the set problem, so it is not appropriate to share solutions. The tutorials and labs provided a forum for discussion and an opportunity for feedback.

5. PLATE marking

Your solution is marked for correctness by PLATE by comparing the output of your system to the output of the model system. You can submit a solution to PLATE many times; I urge you to do this, so you receive credit for your work. The root class must be named `Root` and must include a static `main` method with the standard parameter. The code in the main method should contain only a call to the constructor, as shown below.

```
public class Root()
{   public static void main(String[] args)
    {   new Root();   }
```

I cannot submit for you because PLATE uses the login id to set the owner id. Please do not ask me to submit your solution to PLATE for you after the due date. Please do not ask me to mark an earlier version that worked better. I mark whatever you submit as the final version.

6. Encapsulation marking

Class encapsulation is marked by the MoMa (Model Matcher) tool to ensure a consistent marking scheme. This automatic marking has no flexibility; MoMa simply counts the stated things and gives you a mark for each measure. You submit your solution to PLATE. After the due date, I download all the submissions and run them through MoMa. You cannot submit your solution to MoMa for "pre-marking" before the due date, because the situation is too artificial; you will never have an existing solution when you develop a new solution at work.

Your raw class encapsulation mark is calculated as raw = 20*s + 5*p + 2*g + e, where

- s = number of static members (field or method)
  class `In` is not counted
  `public static void` `main` is not counted
- p = number of public fields
- g = number of getters
- e = number of uses of getters

You want this value to be small.

Here are some examples:

- static members = 8, public fields = 4, getters = 6, use of getters = 18
  raw mark = 20*8 + 5*4 + 2*6 + 18 = 210
- static members = 0, public fields = 4, getters = 5, use of getters = 10
  raw mark = 20*0 + 5*4 + 2*5 + 10 = 40
- static members = 0, public fields = 0, getters = 3, use of getters = 6
  raw mark = 20*0 + 5*0 + 2*3 + 6 = 12
- static members = 0, public fields = 0, getters = 0, use of getters = 0
  raw mark = 20*0 + 5*0 + 2*0 + 0 = 0

Your final encapsulation mark is based on the range of raw CE marks:

```
raw > max, final mark = 0
raw < house CE, final = 20
max > raw >= house, final = (1 - raw/max) * 20
```

Last semester, actual max = 191, used max = 50, and house = 6. Here are the final marks (out of 20) for the examples shown above:

- raw mark = 210, final mark = 0
- raw mark = 40, final mark = (1 – 40/50) * 20 = 0.2 * 20 = 4
- raw mark = 12, final mark = (1 – 12/50) * 20 = 0.76 * 20 = 14.2
- raw mark = 0, final mark = 20

More detail is given in marking.pdf in the Assignments folder on UTSOnline.

The house solution has a CE score of 15: 4 getters, used 7 times.

6.1 Special MoMa requirements

MoMa relies on the standard Java layout and conventions to parse the code. I will fix any small errors, but I cannot re-write your code. Things to avoid are

- anonymous classes
- static blocks

If you have no idea what these are, there is no problem.

7. Spoofing

Your code is marked mainly by software, so you can get a good mark by fooling or spoofing the software. The penalty for spoofing is no marks for that task and a penalty of that task mark; if you spoof a task worth 20 marks, you get no marks for the spoof and lose 20 marks.

8. Submission and return

PLATE shows your mark for correctness when you submit your code. I post a file showing the encapsulation marks shortly after the due date; I will notify you by email at that time.

There is no scheduled late submission period. An extension of up to one week may be given by the subject co-ordinator before the due date; you have to supply documentary evidence of your claim. **An extension CANNOT be given after the due date.**

You may apply for special consideration (SC) either before or after the due date, at *http://www.sau.uts.edu.au/assessment/consideration.html*. You must provide evidence to support your claim, such as a doctor's certificate, a statutory declaration, or a letter from your employer. You can apply for SC before or after the due date.

9. Marking scheme

The final mark is final = correct + design. Design (class encapsulation) is only marked if you do more than half the tasks; that is, if correct > 40/80.

9.1 Correctness (80)

5	Menu and show stock	IO trace 0 in Assignments/1/io.pdf
10	Basic order	IO trace 1
10	Handle one digit numbers	IO trace 2
5	Set and use favourite order (basic)	IO trace 3, 4
10	Substitute once for first in group	IO trace 5
10	Substitute once for any in group	IO trace 6
15	Substitute as needed	IO trace 7, 8, 9, 10
5	Favourite with substitutes	IO trace 11
10	Store and retrieve	IO trace 12

9.2 Design (20)

The detail is given in Section 6 and in Assignments/marking.pdf.

```
raw = 20*s + 5*p + 2*g + e

if (raw > max)
    final mark = 0
else if (raw < house)
    final mark = 20
else
    final mark = (1 - raw/max) * 20
```

The house solution has a CE score of 15: 4 getters, used 7 times.

9.3 Minimum essential requirements

Complete tasks worth half the correctness mark, with an OK class encapsulation.

9.4 Penalties

Spoof	-2*n marks for a task worth n marks
Arrays	-20 marks per array or equivalent

Appendix: House solutions

It is easy to get trapped in a bad solution to this problem, so I outline the House solution structure and process here. You do not have to use any of the code or process described here. This page does not show all the House classes and fields, nor does it show how to handle files.

1. Create an order

I create an order from the input string using these classes and fields:

```
class Order
{   private LinkedList<Item> items;

class Item
{   private Snack snack;
    private int wanted;
    private int filled;

class Snack
{   private String name;
    private int serves;
```

The IO traces show a series of versions that you may wish to follow.

2. Fill an order

I go through each item in turn and try to fill it from the available stock.
I set an item value (`filled`) for the number of items that were filled.
The IO traces show a series of versions that you may wish to follow.

3. Substitute

If an item has not been filled, I try to find a substitute in the same group.
I use a calculation (`wanted - filled`) to find the number of substitutes.
The IO traces show a series of versions that you may wish to follow.