

The codes are written in "BLUEJ".

Topics: Standard patterns, OO design, lists, inheritance

Objectives: This assignment supports subject objectives 1 – 4.

2. System specification

All parts of the specification given in assignment 1 are still valid, unless contradicted by this specification. This specification build on top of the previous specification. The complete solution for the previous assignment is given to you. Your going to build on top of this solution for the new specification. The file is provided, named **base.zip**.

2.1 Domain

The system simulates the operation of the Za pizza shop. The system has three new parts:

1. A new type of pizza can be added
2. A pizza can have any number of toppings
3. Special customers get a discount

The system will be extended in the next assignment.

The Za initially offers three kinds of pizza; a pizza can have any number of toppings. The names of the pizzas, their toppings, and their costs are

pepperoni pepperoni, mushroom, olive \$4.50
marinara prawn, olive \$6.70
vegetarian zucchini, artichoke \$8.90

A customer can order any number of pizzas. A customer has a name that must be unique; choose

your input so each customer has a unique name. The Za starts with \$1,000 in cash; this increases

when a customer pays for his or her order.

The user can add a new type of pizza, by setting the name of the pizza, its toppings, and its price. If the shop does not stock a topping already, then it is added to the list of toppings in stock.

Each topping starts with an arbitrary amount of 20. When a topping is put on a pizza, reduce the

amount on hand by 1. Don't run out of toppings; choose your input so the Za never runs out of toppings. The system does not pay for toppings.

An existing customer becomes a special customer by paying \$10. A special customer gets a credit of 10% of an order, deducted from the next order; a credit has at most two decimal places so it is always a valid amount of money (dollars and cents). Say a customer buys a pepperoni pizza and then converts to a special customer and makes three more orders, each

of
one pepperoni pizza. The charges are:

first order: \$4.50 because this is just a normal customer
<the customer becomes a special customer>

second order: \$4.50 because there is no existing credit; store a credit of \$0.45 ($4.50 * 10\%$)

third order: \$4.05 ($4.50 - .45$) because there is a credit of \$0.45
store a credit of \$0.40 ($4.05 * 10\%$); the 0.5 cents (from \$0.405) is lost

fourth order: \$4.10 ($4.50 - .40$) because there is a credit of \$0.40
store a credit of \$0.41 ($4.10 * 10\%$)

A charge cannot be negative, so if the credit is more than the price then the charge is \$0.00.

2.2 System interface

The system uses the console for all IO; all indents in the output are in steps of four spaces. A full IO trace can be found at the last page of this file.

Main menu

The system is driven from a menu with these choices:

O Order pizzas

C Convert to a special customer

N Define a new pizza

T Show the types of pizza sold by the Za

Z Show the state of the Za: cash and stock on hand

? Show the menu choices

X Exit the system

The prompt for a choice is always preceded by a space line; see the IO trace below. Input may be upper or lower case. The menu input is validated; if the choice is invalid, the system shows the valid choices and asks for a new choice. The system runs until the user decides to exit.

Order pizzas

When the user chooses to make an order, s/he is asked for his or her name, and then the types of pizza. A pizza is requested by name; if the input does not match one of the stored types, then an error message is shown and the user is asked to enter a new name. The order is finished when the user enters "end". The system then shows (with two decimal places) the total charge of that order and increments the Za's cash on hand. A single order looks like:

```
Your choice (O/C/N/T/Z/?/X): o
Your name: fred
Hi fred
```

```
Type of pizza (or end): pepperoni
Type of pizza (or end): caviar
We make pepperoni, marinara, and vegetarian pizzas
Type of pizza (or end): marinara
Type of pizza (or end): end
The charge is $11.20
```

The customer name is unique.

Become a special customer

The system asks for the customer name and finds that customer in the customer records; assume that each customer name is unique. Do not validate the name; just choose your input so that there is always a matching customer to make special. A conversion fee of \$10 is added to the Za's cash on hand. The IO looks like

```
Your choice (O/C/N/T/Z/?/X): C
Your name: fred
    Congratulations fred.
    For just $10 you are now special.
```

Special customer order

When a special customer makes an order, the greeting also shows the amount of credit:

```
Your choice (O/C/N/T/Z/?/X): o
Your name: fred
Hi fred, you have $0.00 credit
Type of pizza (or end): ...
```

Please note that the ellipsis ("...") is not actual output; an ellipsis indicates that the normal output follows, but is not shown here. A special customer gets a credit, so the price charged will be different from normal.

Show the Za

The system shows the cash on hand (with two decimal places), and the stock on hand. New toppings may appear in the stock after a new type of pizza has been added.

2.3 Implementation

All collection attributes (attributes that can store a collection of objects) must be lists, not arrays. Replace any array attributes with lists. Any new collection attributes must be lists. You may have local arrays, but do not use any array attributes. This is marked. Use the class `In` from

the labs to get input. Do not change anything in this class. Please do not ask me if it's OK to change this code; just use the class as it exists.

3. Expected work load

To give you some idea of the code involved, my solution (without class `In`) has about 200 lines of executable code, about 60 lines more than the model solution for assignment 1; this count does not include class and method headers, or attributes. Note that you have to change (and delete) existing code, as well as adding new code.

5. PLATE marking

Your solution is marked for correctness by PLATE (the Programming Learning And Teaching Environment). This ensures correct and consistent marking, but you need to set up your solution exactly as specified here. PLATE has no ability to work out that you really did the task but just didn't get the IO right. It does a character by character comparison of the output of your system and the output of the model system, when given the input in the test files. You can submit a solution to PLATE many times; I urge you to do this, to make sure that your IO format is correct so you receive credit for your work,

5.1 The root class

The root class must be named `Za` and must include a static `main` method for your code to be marked by PLATE. The code in this method contains only a call to the constructor:

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

When you create a jar file, make sure you set the root class in the jar.

5.2 IO format

A sample transcript of the IO is provided **below** or at **Assignments/2/io.txt**. Your IO format must match the transcript format exactly. All indents are made in steps of four spaces. If your IO does not follow the correct format, then PLATE will give you zero for that part. PLATE does not use this file for marking, because PLATE requires a separate file for each mark. Please do not assume that your solution is 100% correct because it echoes `io.txt`; you need to look at your PLATE mark to see how many marks you get for correct code.

6. Style marking

Your system is marked for good design by the MoMa (Model Matcher) tool, as shown in the marking scheme given in lab 2. MoMa compares your code to the model solution using a set of metrics that reflect good design. This was done to ensure a consistent marking scheme, in response to student complaints about inconsistent marking when marking was done by hand.

This means that marking is now completely consistent and has no flexibility; MoMa simply counts the stated things and gives you a mark. It is simply not possible to write an expert system

that has the knowledge, understanding, and flexibility of an expert programmer. If it was possible, we wouldn't need programmers!

The model solution was designed using the design rules and standard code patterns taught in the subject. In many cases, the patterns and rules result in just one solution and there is no choice. In cases where no single solution is mandated, any good solution that obeys the design rules and creates a simple, elegant, reusable solution is accepted.

I understand that you may not agree with or even like the design rules taught in this subject. I didn't invent them! I collected them from many authorities in the field of OO design over many years and re-phrased and systematised them. You may use different design rules in other programming subjects. You may use different design rules at your job, when your company defines a house style to follow. The form of the solution in this assignment, however, should be determined by the correct application of the design rules and standard solutions taught in this subject. This teaches you how to design; it prepares you for other situations where the design rules may be different but must still be correctly applied.

Code reuse

Use the model solution from Assignment 1 as the base for this assignment. This base code is used in two ways by MoMa.

First, MoMa counts the number of lines of code reused without change in your solution and gives you a reuse mark using the normal calculation: s / m , where s is the count of lines reused in your solution, and m is the number of lines reused in the model solution. Do not change the format of this code; the code should be reused without change.

Second, the base code is removed from your solution by MoMa before your solution is marked, so you are marked on new code only.

10. Marking scheme

10.1 Weighting

The final mark is the sum of two components: correctness (c) and good design (d). The two parts are not weighted equally; the correctness mark is worth 40% and the design mark is worth 60%. I reserve the right to change the weightings for the next assignment.

10.2 Correctness (40)

10 Lists (no arrays): any number of ...	10 Add a new pizza
5 Convert to special	15 Special orders

10.3 Good design (60)

5 Classes	5 Attributes
10 Class encapsulation	10 Small methods
10 Functions	10 Reuse
10 Inheritance	

12.4 Penalties

Spoof correctness 0 marks (correctness) for that part

Spoof style -10 (style) each occurrence

Empty methods -1 (style) for each empty method

Unused code -1 (style) for each uncalled method

`show` and `toString` are not counted

IO Trace:

Welcome to Za Pizza

Your choice (O/C/N/T/Z/?/X): Z

The Za has \$1,000.00

The stock on hand is

pepperoni: 20

mushroom: 20

olive: 20

prawn: 20

zucchini: 20

artichoke: 20

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred

Type of pizza (or end): pepperoni

Type of pizza (or end): pepperoni

Type of pizza (or end): marinara

Type of pizza (or end): um

We make pepperoni, marinara, and vegetarian pizzas

Type of pizza (or end): end

The charge is \$15.70

Your choice (O/C/N/T/Z/?/X): c

Your name: fred

Congratulations fred.

For just \$10 you are now special.

Your choice (O/C/N/T/Z/?/X): z

The Za has \$1,025.70

The stock on hand is

pepperoni: 18

mushroom: 18

olive: 17

prawn: 19

zucchini: 20

artichoke: 20

Your choice (O/C/N/T/Z/?/X): n

Pizza name: gold

Topping (or end): gold

Topping (or end): platinum

Topping (or end): diamond

Topping (or end): end

Price: 10000.00

Your choice (O/C/N/T/Z/?/X): t

We make pepperoni, marinara, vegetarian, and gold pizzas

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred, you have \$0.00 credit

Type of pizza (or end): pepperoni

Type of pizza (or end): end

The charge is \$4.50

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred, you have \$0.45 credit

Type of pizza (or end): pepperoni

Type of pizza (or end): end

The charge is \$4.05

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred, you have \$0.40 credit

Type of pizza (or end): gold

Type of pizza (or end): pepperoni

Type of pizza (or end): end

The charge is \$10,004.10

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred, you have \$100.41 credit

Type of pizza (or end): pepperoni

Type of pizza (or end): end

The charge is \$0.00

Your choice (O/C/N/T/Z/?/X): z

The Za has \$2,038.35

The stock on hand is

pepperoni: 14

mushroom: 14

olive: 13

prawn: 19

zucchini: 20

artichoke: 20

gold: 19

platinum: 19

diamond: 19

Your choice (O/C/N/T/Z/?/X): o

Your name: fred

Hi fred, you have \$0.00 credit

Type of pizza (or end): pepperoni

Type of pizza (or end): end

The charge is \$4.50

Your choice (O/C/N/T/Z/?/X): z

The Za has \$2,042.85

The stock on hand is

pepperoni: 13

mushroom: 13

olive: 12

prawn: 19

zucchini: 20

artichoke: 20

gold: 19

platinum: 19

diamond: 19

Your choice (O/C/N/T/Z/?/X): x