

---

Due Date:	By 11:59pm Wednesday November 30, 2016
Evaluation:	5% of final mark (see marking rubric at the end of handout) Late Submission: none accepted
Purpose:	The purpose of this assignment is incorporate arrays of objects and the different types of statements we have seen since the beginning of the course into one more elaborate scenario.
CEAB/CIPS Attributes:	Design/Problem analysis/Communication Skills

---

## Part 1/2

In this part you will implement a class, named **Menu**, that models menus used in text-based menu-driven programs, where the user is presented with a list of options to choose from, similar to menu-based voice interfaces.

For this assignment, the **Menu** objects of interest are required to form their string representation using the pattern shown at right as a template. Thus such **Menu** objects must each represent and maintain at least six attributes: the opening and closing messages, the top and bottom prompts, an option list, and the string “?->”.

Opening Message  
Top Prompt  
(1) Option one  
(2) Option two  
(3) Option three  
(..) ...  
(n) Option n  
Closing Message  
?-> Bottom Prompt

The literal prompt string “?->” provides the minimal string representation for a **Menu** object in that it is always present in the string representation of that object, just before the bottom prompt, whereas any number of the other five *descriptive* attributes can be absent.

For the sake of simplicity, once a **Menu** object is created, the option list maintained by that **Menu** object is required to remain unchanged.<sup>1</sup> However, unless the option list is **null**<sup>2</sup> or empty<sup>3</sup> when a **Menu** object is created, the option list is always present in the string representation of that **Menu** object; that is, you just can’t exclude the option list of a **Menu** object from the string representation of that object.

The public interface of class **Menu** should include the following methods:

---

<sup>1</sup>Possible changes include modifying, reordering, disabling, or enabling the options in the option list, or adding new options to or removing existing options from the option list.

<sup>2</sup>For example, `String[] optionList = null;`

<sup>3</sup>For example, `String[] optionList = new String[0];`

- **Menu(String[] options)**

A normal (non-default) constructor. Takes an array of strings as a parameter to initialize the option list in *this* **Menu**. This supplied array of string options may be **null** or an array of strings of length zero or more. Sets the opening and closing messages to **null**. Sets the top prompt to “**Choose an option:**” and the bottom prompt to “**Enter an option number:**”.

- **Menu()**

A default constructor. Sets all five *descriptive* attributes to null.

- **isEmpty()**

A facilitator. Determines whether the option list in *this* menu is **null** or empty.

- **length()**

A facilitator. Returns zero if the option list in *this* menu is **null**; otherwise, it returns the length of the option list in *this* menu.

- **toString()**

Returns a string formatted using the menu template shown above. Example:

Source code

```
1 String[] drink_options = {"Water", "Soda pop", "Beer"};
2 // create a menu of drink options
3 Menu drinkMenu = new Menu(drink_options); //no messages, default prompts
4 // display the menu
5 System.out.println(drinkMenu); // calls drinkMenu.toString()
```

output

```
1 Choose an option:
2     (1) Water
3     (2) Soda pop
4     (3) Beer
5 ?-> Enter an option number:
```

Note the absence of the top and bottom messages in the output above as they were set to **null** when **drinkMenu** was created on line 3. The source code above may also choose to remove the top and bottom prompts as follows:

#### Source code

```
6 // remove prompts
7 drinkMenu.setTopPrompt(null);
8 drinkMenu.setBottomPrompt(null);
9 // display the menu
10 System.out.println(drinkMenu); // calls drinkMenu.toString()
```

#### output

```
6      (1) Water
7      (2) Soda pop
8      (3) Beer
9      ?->
```

- **readOptionNumber()**

Displays *this* menu and then inputs an integer number from the keyboard.

If the option list is empty, it returns the number entered.

Otherwise, it checks that the input integer is a valid option number: between **1** and **n**, inclusive, where **n** is the number of options in the option list. The method repeatedly displays *this* menu and then inputs a number until a valid option number is entered. After a valid option number has been entered, the method returns that number.

- **Setter (mutator) and getter (accessor) methods** (self-explanatory)

- getBottomMessage, setBottomMessage
- getBottomPrompt, setBottomPrompt
- getTopMessage, setTopMessage
- getTopPrompt, setTopPrompt

Example:

#### Source code

```
11 String[] option_list = {"Water", "Soda pop", "Beer"};
12 Menu full_menu = new Menu(option_list); //no messages, default prompts
13 // introduce messages
14 full_menu.setTopMessage("Quench your thirst with our fine drinks!");
15 full_menu.setBottomMessage("Time to obey your thirst!");
16 // reset prompts
17 full_menu.setTopPrompt("Choose your thirst crusher:");
18 full_menu.setBottomPrompt("Enter a drink number: ");
19 // display the menu and then read an option number
20 int choice = full_menu.getOptionNumber();
21 System.out.println("You entered " + choice );
```

#### output

```
10  Quench your thirst with our fine drinks!
11  Choose your thirst crusher:
12      (1) Water
13      (2) Soda pop
14      (3) Beer
15  Time to obey your thirst!
16  ?-> Enter a drink number: 3
17
18  You entered 3
```

## More Examples

Here is an example of how a default-constructed **Menu** object might be used:

#### Source code

```
22  // create an empty menu
23  Menu m = new Menu(); // no messages, no prompts, no options
24  // read an integer
25  int number1 = m.getOptionNumber();
26  System.out.println("You entered " + number1);
27  System.out.println("-----");
28  // introduce bottom prompt
29  m.setBottomPrompt("Enter an integer for bottom prompt: ");
30  // read an integer
31  int number2 = m.getOptionNumber();
32  System.out.println("You entered " + number2);
33  System.out.println("-----");
```

#### output

```
19  ?-> 15
20  You entered 15
21  -----
22  ?-> Enter an integer for bottom prompt: 16
23  You entered 16
24  -----
```

#### Source code

```
34 // remove bottom prompt
35 m.setBottomPrompt(null);
36 // introduce bottom message
37 m.setBottomMessage("Enter an integer for bottom message: ");
38 // read an integer
39 int number3 = m.getOptionNumber();
40 System.out.println("You entered " + number3);
41 System.out.println("-----");
```

#### output

```
25 Enter an integer for bottom message:
26 ?-> 17
27 You entered 17
28 -----
```

#### Source code

```
42 // introduce top message
43 m.setTopMessage("*****");
44 // introduce top prompt
45 m.setTopPrompt("An integer is even if it is twice another integer");
46 // reset bottom message
47 m.setBottomMessage("*****");
48 // intro bottom prompt
49 m.setBottomPrompt("Enter an even integer: ");
50 // read an integer
51 int number4 = m.getOptionNumber();
52 System.out.println("You entered " + number4);
```

#### output

```
29 *****
30 An integer is even if it is twice another integer
31 *****
32 ?-> Enter an even integer: 18
33 You entered 18
```

The line numbers are printed for reference only.

## Part 2/2

Specialized and successful in selling ice creams in stores, the CreamyIce Company has decided to start an online ice cream store. They have asked that you write a program that will provide their customers with online shopping services. Ideally, their customers would browse through pictures and names of the various ice creams, selecting those they wanted. However, the company knows that you are still learning, so all they want for now is a text-based menu-driven program that will allow customers to prepare and price shopping carts of ice cream orders.

Although you have realized that your task involves implementing “things” such as ice cream orders, shopping carts, and ice cream stores, you head to the newly established IT office at CreamyIce, where you are first presented with an overview of your task and then with the details!

Here is an overview of the what you are required to implement:

1. An **IceCreamOrder** class that represents an order to buy a given ice cream in a given quantity, such as **3** orders of **Single Scoop** of **Vanilla** ice cream in a **cup** each for **\$2.99**.
2. A **ShoppingCart** class that stores **IceCreamOrders** in a cart (a *regular Java array*) and allows ice cream orders to be added to or removed from the cart, or any of the ice cream orders in the cart be listed or revised. To keep it simple, the capacity of the cart is limited to a maximum of **5**<sup>4</sup> ice cream orders.
3. An **IceCreamStore** class that stores a **ShoppingCart** and uses predefined menus to interact with the customer. It provides services that allow the customer to create an order and place it in the shopping cart as well as allowing the customer to manipulate the orders in the shopping cart.

Here are the details:

### 1) Class IceCreamOrder

Represents an ice cream order in terms the following attributes:

- **flavor** name,
- **vessel** (container) name, such as cone, cup, or sundae,
- **amount** name, such as Single Scoop, Double Scoop, Triple Scoop, etc.
- **unit price**, and
- **quantity**, the desired units of this ice cream.

This class should have the following public methods:

---

<sup>4</sup>It really makes no difference what limit is used. We choose 5 as the limit so that we can quickly fill our cart with orders during the testing phase of the program.

### String toString()

Returns a string using this sample pattern: “4 orders of Triple Scoops of Avocado ice cream in a Cup for \$19.96 = 4 x 4.99”.

**double price()** Returns **quantity \* unitPrice**

**Setter and Getter Methods** One pair for each instance variable.

### IceCreamOrder(String flavor, String vessel, String amount, double unitPrice, int quantity)

A normal constructor. Initializes the instance variables with supplied values.

### IceCreamOrder(String flavor, String vessel, String amount, double unitPrice)

A normal constructor. Delegates initialization tasks to the constructor above with quantity set to 1.

### IceCreamOrder()

A default constructor. It initializes all of the instance variables of *this* object based on the input values it receives from the customer in four steps:

1. Display the flavour menu and read user's choice of an ice cream flavour.

#### output

```
1 Placing an order is as easy as ABC, and D.
2 Step A: Select your favorite flavour
3     (1) Avocado
4     (2) Banana
5     (3) Chocolate
6     (4) Hazelnut
7     (5) Lemon
8     (6) Mango
9     (7) Mocha
10    (8) Vanilla
11    (9) Exit this menu
12 ?-> Enter an option number:
```

2. Display the vessel (container) menu and read user's choice of a vessel.

#### output

```
13 Step B: Select a vessel for your ice cream:
14     (1) Cone
15     (2) Cup
16     (3) Sundae
17 ?-> Enter an option number:
```

3. Display the how-much-ice-cream menu and read user's choice for an ice cream

amount.

output

```
18 Step C: How much ice cream?
19     (1) Single Scoop
20     (2) Double Scoop
21     (3) Triple Scoop
22 ?-> Enter an option number:
```

4. Display the how-many-orders menu and read user's choice for a number.

output

```
23 Step D: how many orders of your current selection?
24     (1) One
25     (2) Two
26     (3) Three
27     (4) Four
28     (5) Five
29     (6) Six
30     (7) Seven
31     (8) Eight
32     (9) Nine
33 ?-> Enter how many orders:
```

All that remains to be initialized is the unit price instance variable. The unit price of an order is computed as follows:

Unit Price Table

	Single Scoop	Double Scoop	Triple Scoop
Cup	2.99	3.99	4.99
Cone	3.49	4.49	5.49
sundae	4.25	5.25	6.25

## 2) Class ShoppingCart

Represents and stores **IceCreamOrders** in a cart using a *regular* Java *array* as its underlying storage. To facilitate program testing, the cart capacity is limited to **5 IceCreamOrders**.

This class should have the following public methods:



### **ShoppingCart()**

Creates an empty shopping cart with the maximum capacity allowed.

### **void add(IceCreamOrder order)**

Adds the supplied ice cream order to the cart, keeping track of the number of orders in it. If the list is full, it prints an error message and ignores the request.

### **void remove(int position)**

Removes an order at a specified position from the cart. If the list is empty or the specified position is out of range, it prints an error message and ignores the request.

### **String toString()**

Returns a string representation of all ice cream orders in the cart separated by new line characters.

### **boolean isEmpty()**

Determines whether this cart is empty.

### **boolean isFull()**

Determines whether this cart is full.

### **IceCreamOrder get(int position)**

Returns the order placed at the supplied position in the cart. If the specified position is out of range, it prints an error message and ignores the request.

### **int size()**

Returns the number of orders currently in the cart.

## **3) Class IceCreamStore**

Stores a **ShoppingCart** and interacts with the customer through a set of predefined menus. The main menu displayed by **IceCreamStore** provides options that allow customers to print, price, or revise the ice cream orders in their cart, or to add new orders to or remove existing orders from their cart.

Ideally, the main menu would display all of the options in its option list with some of the options possibly disabled or enabled depending on the size of the cart.

However, since our **Menu** objects are not required<sup>5</sup> to provide services to enable or disable menu options, **IceCreamStore** eliminates the need for those services by directly pointing out the options in the main menu the user may or may not select<sup>6</sup>, based on the number of orders in the cart:

---

<sup>5</sup>because of the increasing complexity involved.

<sup>6</sup>It is of course possible to use three different menus, one for when the cart is empty, one for when the cart is full, and one for when the cart neither full nor empty. However, users tend to not appreciate related menus that use different option numbers for the same options.

1.

#### Main menu when the shopping cart is neither empty nor full

```
1 Your shopping cart contains 2 ice cream orders
2 What would you like to do?
3     (1) Place an order
4     (2) Delete an order
5     (3) Price the cart
6     (4) List the cart
7     (5) Proceed to checkout
8     (6) Exit this menu
9 ?-> Enter an option number:
```

2.

#### Main menu when the shopping cart is full

```
1 Your Shopping Cart is full with 5 ice cream orders.
2 Cannot place orders! what would you like to do?
3     (1) Place an order
4     (2) Delete an order
5     (3) Price the cart
6     (4) List the cart
7     (5) Proceed to checkout
8     (6) Exit this menu
9 Please select option 2, 3, 4, 5, or 6
10 ?-> Enter an option number:
```

3.

#### Main menu when the shopping cart is empty

```
1 Your Shopping Cart is empty.
2 You have only two options: 1 or 6
3     (1) Place an order
4     (2) Delete an order
5     (3) Price the cart
6     (4) List the cart
7     (5) Proceed to checkout
8     (6) Exit this menu
9 Please enter 1 or 6
10 ?-> Enter an option number:
```

The public interface of class **IceCreamStore** should include the following methods:

**void placeOrder()**

Creates an order using **IceCreamOrder**'s default constructor and places that order in the cart.

**void deleteOrder()**

Using a menu, displays a list of all of the orders in the cart as options, prompts for and reads the option number associated with the order to be deleted, and then removes that order from the cart. To avoid forcing the user to having to remove an order, the method always adds an *exit* option at the end of the option list displayed by the menu.

**double computeTotalPrice()**

Returns the total price of all the items in the cart.

**void printTotalPrice()**

Prints the total price of all the items in the cart.

**void reviewOrders()**

Prints a complete list of all orders in the cart.

**void checkout()**

Ideally, collects payment and arranges for pickup or delivery. In this assignment, however, it first calls **reviewOrders()** and then **printTotalPrice()**.

**void run()**

Repeatedly displays the main menu and performs the user selected actions.

**Note**

Feel free to add your own **private** methods to any of the classes above to facilitate your task.

## Sample Run of Program

To reduce the amount of printout in the following sample run to below 400 lines, the cart capacity in **ShoppingCart** was set to **MAX\_ORDERS = 3**. For the purpose of clarity, the user inputs are shown in **red**.

### Driver Program

```
1 public class IceCreamStoreOnline
2 {
3     public static void main(String[] args)
4     {
5         IceCreamStore shop = new IceCreamStore();
6         shop.run();
7     }
8 }
```

### Output of a sample run of the program with MAX\_ORDERS = 3

```
1 Your Shopping Cart is empty.
2 You have only two options: 1 or 6
3     (1) Place an order
4     (2) Delete an order
5     (3) Price the cart
6     (4) List the cart
7     (5) Proceed to checkout
8     (6) Exit this menu
9 Please enter 1 or 6
10 ?-> Enter an option number: 1
11
12 Placing an order is as easy as ABC, and D.
13 Step A: Select your favorite flavour
14     (1) Avocado
15     (2) Banana
16     (3) Chocolate
17     (4) Coffee
18     (5) Hazelnut
19     (6) Lemon
20     (7) Mango
21     (8) Mocha
22     (9) Vanilla
23     (10) Exit this menu
24 ?-> Enter an option number: 6
25
```

### Output of a sample run of the program with MAX\_ORDERS = 3

```
26 Step B: Select a vessel for your ice cream:
27     (1) Cone
28     (2) Cup
29     (3) Sundae
30 ?-> Enter an option number: 3
31
32 Step C: How much ice cream?
33     (1) Single Scoop
34     (2) Double Scoop
35     (3) Triple Scoop
36 ?-> Enter an option number: 2
37
38 Step D: how many orders of your current selection?
39     (1) One
40     (2) Two
41     (3) Three
42     (4) Four
43     (5) Five
44     (6) Six
45     (7) Seven
46     (8) Eight
47     (9) Nine
48     (10) Ten
49 ?-> Enter how many orders: 1
50
51 Your shopping cart contains 1 ice cream order
52 What would you like to do?
53     (1) Place an order
54     (2) Delete an order
55     (3) Price the cart
56     (4) List the cart
57     (5) Proceed to checkout
58     (6) Exit this menu
59 ?-> Enter an option number: 2
60
61 You have selected to remove an order from your cart
62 What would you like to do?
63     (1) 1 order of Double Scoop of Lemon ice cream in a Sundae for
        $4.99 = 1 x 4.99
64     (2) Exit this menu
65 ?-> Enter an option number: 1
66
67 The order you selected was deleted
68
```

## Output of a sample run of the program with MAX\_ORDERS = 3

```
69 Your Shopping Cart is empty.
70 You have only two options: 1 or 6
71     (1) Place an order
72     (2) Delete an order
73     (3) Price the cart
74     (4) List the cart
75     (5) Proceed to checkout
76     (6) Exit this menu
77 Please enter 1 or 6
78 ?-> Enter an option number: 5
79
80 Your Shopping Cart is empty.
81 You have only two options: 1 or 6
82     (1) Place an order
83     (2) Delete an order
84     (3) Price the cart
85     (4) List the cart
86     (5) Proceed to checkout
87     (6) Exit this menu
88 Please enter 1 or 6
89 ?-> Enter an option number: 1
90
91 Placing an order is as easy as ABC, and D.
92 Step A: Select your favorite flavour
93     (1) Avocado
94     (2) Banana
95     (3) Chocolate
96     (4) Coffee
97     (5) Hazelnut
98     (6) Lemon
99     (7) Mango
100    (8) Mocha
101    (9) Vanilla
102    (10) Exit this menu
103 ?-> Enter an option number: 5
104
105 Step B: Select a vessel for your ice cream:
106     (1) Cone
107     (2) Cup
108     (3) Sundae
109 ?-> Enter an option number: 2
110
111 Step C: How much ice cream?
112     (1) Single Scoop
113     (2) Double Scoop
114     (3) Triple Scoop
115 ?-> Enter an option number: 3
```

## Output of a sample run of the program with MAX\_ORDERS = 3

```
116
117 Step D: how many orders of your current selection?
118     (1) One
119     (2) Two
120     (3) Three
121     (4) Four
122     (5) Five
123     (6) Six
124     (7) Seven
125     (8) Eight
126     (9) Nine
127     (10) Ten
128 ?-> Enter how many orders: 4
129
130 Your shopping cart contains 1 ice cream order
131 What would you like to do?
132     (1) Place an order
133     (2) Delete an order
134     (3) Price the cart
135     (4) List the cart
136     (5) Proceed to checkout
137     (6) Exit this menu
138 ?-> Enter an option number: 1
139
140 Placing an order is as easy as ABC, and D.
141 Step A: Select your favorite flavour
142     (1) Avocado
143     (2) Banana
144     (3) Chocolate
145     (4) Coffee
146     (5) Hazelnut
147     (6) Lemon
148     (7) Mango
149     (8) Mocha
150     (9) Vanilla
151     (10) Exit this menu
152 ?-> Enter an option number: 1
153
154 Step B: Select a vessel for your ice cream:
155     (1) Cone
156     (2) Cup
157     (3) Sundae
158 ?-> Enter an option number: 2
159
160 Step C: How much ice cream?
161     (1) Single Scoop
162     (2) Double Scoop
163     (3) Triple Scoop
164 ?-> Enter an option number: 3
```

## Output of a sample run of the program with MAX\_ORDERS = 3

```
165
166 Step D: how many orders of your current selection?
167     (1) One
168     (2) Two
169     (3) Three
170     (4) Four
171     (5) Five
172     (6) Six
173     (7) Seven
174     (8) Eight
175     (9) Nine
176     (10) Ten
177 ?-> Enter how many orders: 4
178
179 Your shopping cart contains 2 ice cream orders
180 What would you like to do?
181     (1) Place an order
182     (2) Delete an order
183     (3) Price the cart
184     (4) List the cart
185     (5) Proceed to checkout
186     (6) Exit this menu
187 ?-> Enter an option number: 1
188
189 Placing an order is as easy as ABC, and D.
190 Step A: Select your favorite flavour
191     (1) Avocado
192     (2) Banana
193     (3) Chocolate
194     (4) Coffee
195     (5) Hazelnut
196     (6) Lemon
197     (7) Mango
198     (8) Mocha
199     (9) Vanilla
200     (10) Exit this menu
201 ?-> Enter an option number: 9
202
203 Step B: Select a vessel for your ice cream:
204     (1) Cone
205     (2) Cup
206     (3) Sundae
207 ?-> Enter an option number: 3
208
209 Step C: How much ice cream?
210     (1) Single Scoop
211     (2) Double Scoop
212     (3) Triple Scoop
213 ?-> Enter an option number: 1
```



## Output of a sample run of the program with MAX\_ORDERS = 3

```
214
215 Step D: how many orders of your current selection?
216     (1) One
217     (2) Two
218     (3) Three
219     (4) Four
220     (5) Five
221     (6) Six
222     (7) Seven
223     (8) Eight
224     (9) Nine
225     (10) Ten
226 ?-> Enter how many orders: 6
227
228 Your Shopping Cart is full with 3 ice cream orders.
229 Cannot place orders! what would you like to do?
230     (1) Place an order
231     (2) Delete an order
232     (3) Price the cart
233     (4) List the cart
234     (5) Proceed to checkout
235     (6) Exit this menu
236 Please select option 2, 3, 4, 5, or 6
237 ?-> Enter an option number: 4
238
239 Your current selections of our scrumptious ice creams
240 -----
241 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
242 $19.96 = 4 x 4.99
243 4 orders of Triple Scoop of Avocado ice cream in a Cup for
244 $19.96 = 4 x 4.99
245 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
246 $23.94 = 6 x 3.99
247 -----
248
249 Your Shopping Cart is full with 3 ice cream orders.
250 Cannot place orders! what would you like to do?
251     (1) Place an order
252     (2) Delete an order
253     (3) Price the cart
254     (4) List the cart
255     (5) Proceed to checkout
256     (6) Exit this menu
257 Please select option 2, 3, 4, 5, or 6
258 ?-> Enter an option number: 1
```

## Output of a sample run of the program with MAX\_ORDERS = 3

```
256
257 Your Shopping Cart is full with 3 ice cream orders.
258 Cannot place orders! what would you like to do?
259     (1) Place an order
260     (2) Delete an order
261     (3) Price the cart
262     (4) List the cart
263     (5) Proceed to checkout
264     (6) Exit this menu
265 Please select option 2, 3, 4, 5, or 6
266 ?-> Enter an option number: 3
267
268 -----=
269 Total price of all your orders in the cart: $63.86
270 -----=
271
272 Your Shopping Cart is full with 3 ice cream orders.
273 Cannot place orders! what would you like to do?
274     (1) Place an order
275     (2) Delete an order
276     (3) Price the cart
277     (4) List the cart
278     (5) Proceed to checkout
279     (6) Exit this menu
280 Please select option 2, 3, 4, 5, or 6
281 ?-> Enter an option number: 2
282
283 You have selected to remove an order from your cart
284 What would you like to do?
285     (1) 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
286 $19.96 = 4 x 4.99
287     (2) 4 orders of Triple Scoop of Avocado ice cream in a Cup for
288 $19.96 = 4 x 4.99
289     (3) 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
290 $23.94 = 6 x 3.99
291     (4) Exit this menu
292 ?-> Enter an option number: 4
293
294 Your Shopping Cart is full with 3 ice cream orders.
295 Cannot place orders! what would you like to do?
296     (1) Place an order
297     (2) Delete an order
298     (3) Price the cart
299     (4) List the cart
300     (5) Proceed to checkout
301     (6) Exit this menu
302 Please select option 2, 3, 4, 5, or 6
303 ?-> Enter an option number: 4
```

### Output of a sample run of the program with MAX\_ORDERS = 3

```
301
302 Your current selections of our scrumptious ice creams
303 -----
304 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
    $19.96 = 4 x 4.99
305 4 orders of Triple Scoop of Avocado ice cream in a Cup for
    $19.96 = 4 x 4.99
306 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
    $23.94 = 6 x 3.99
307 -----
308
309 Your Shopping Cart is full with 3 ice cream orders.
310 Cannot place orders! what would you like to do?
311     (1) Place an order
312     (2) Delete an order
313     (3) Price the cart
314     (4) List the cart
315     (5) Proceed to checkout
316     (6) Exit this menu
317 Please select option 2, 3, 4, 5, or 6
318 ?-> Enter an option number: 2
319
320 You have selected to remove an order from your cart
321 What would you like to do?
322     (1) 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
        $19.96 = 4 x 4.99
323     (2) 4 orders of Triple Scoop of Avocado ice cream in a Cup for
        $19.96 = 4 x 4.99
324     (3) 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
        $23.94 = 6 x 3.99
325     (4) Exit this menu
326 ?-> Enter an option number: 2
327
328 The order you selected was deleted
329
330 Your shopping cart contains 2 ice cream orders
331 What would you like to do?
332     (1) Place an order
333     (2) Delete an order
334     (3) Price the cart
335     (4) List the cart
336     (5) Proceed to checkout
337     (6) Exit this menu
338 ?-> Enter an option number: 4
339
```

## Output of a sample run of the program with MAX\_ORDERS = 3

```
340 Your current selections of our scrumptious ice creams
341 -----
342 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
    $19.96 = 4 x 4.99
343 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
    $23.94 = 6 x 3.99
344 -----
345
346 Your shopping cart contains 2 ice cream orders
347 What would you like to do?
348     (1) Place an order
349     (2) Delete an order
350     (3) Price the cart
351     (4) List the cart
352     (5) Proceed to checkout
353     (6) Exit this menu
354 ?-> Enter an option number: 3
355
356 -----=
357 Total price of all your orders in the cart: $43.90
358 -----=
359
360 Your shopping cart contains 2 ice cream orders
361 What would you like to do?
362     (1) Place an order
363     (2) Delete an order
364     (3) Price the cart
365     (4) List the cart
366     (5) Proceed to checkout
367     (6) Exit this menu
368 ?-> Enter an option number: 5
369
370 Your current selections of our scrumptious ice creams
371 -----
372 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
    $19.96 = 4 x 4.99
373 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
    $23.94 = 6 x 3.99
374 -----
375 -----=
376 Total price of all your orders in the cart: $43.90
377 -----=
378
```

### Output of a sample run of the program with MAX\_ORDERS = 3

```
379 Your Shopping Cart is empty.
380 You have only two options: 1 or 6
381     (1) Place an order
382     (2) Delete an order
383     (3) Price the cart
384     (4) List the cart
385     (5) Proceed to checkout
386     (6) Exit this menu
387 Please enter 1 or 6
388 ?-> Enter an option number: 6
389
390 Cheers!
391
```



## Evaluation Criteria for Assignment 4 (20 points)

<b>Class Menu</b>	<b>pts</b>
Constructors, isEmpty, length	1
Getters/Setters	1
toString	2
getOptionNumber	1
<b>Class IceCreamOrder</b>	
toString, price, 2 normal constructors	1
Setter and Getter Methods	1
default constructor	3
<b>class ShoppingCart</b>	
add, remove,	3
toString	1
isEmpty, get, size, isFull	1
<b>class IceCreamStore</b>	
run	3
deleteOrder	1
placeOrder, printTotalPrice, checkout, reviewOrders	1