Concordia University
COMP 248 – Fall 2016
Assignment 4

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.[1] However, unless the option list is **null**[2] or empty[3] 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:

[1]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.

[2]For example, `String[] optionList = null;`

[3]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

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

```java
// remove prompts
drinkMenu.setTopPrompt(null);
drinkMenu.setBottomPrompt(null);
// display the menu
System.out.println(drinkMenu); // calls drinkMenu.toString()
```

output

```
     (1) Water
     (2) Soda pop
     (3) Beer
?->
```

- **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

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

output

```
Quench your thirst with our fine drinks!
Choose your thirst crusher:
     (1) Water
     (2) Soda pop
     (3) Beer
Time to obey your thirst!
?-> Enter a drink number: 3

You entered 3
```

More Examples

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

Source code

```
// create an empty menu
Menu m = new Menu(); // no messages, no prompts, no options
// read an integer
int number1 = m.getOptionNumber();
System.out.println("You entered " + number1);
System.out.println("-----------------------");
// introduce bottom prompt
m.setBottomPrompt("Enter an integer for bottom prompt: ");
// read an integer
int number2 = m.getOptionNumber();
System.out.println("You entered " + number2);
System.out.println("-----------------------");
```

output

```
?-> 15
You entered 15
-----------------------
?-> Enter an integer for bottom prompt: 16
You entered 16
-----------------------
```

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 CearmyIce, 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 **IceCreamOrder**s 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**[4] 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:

[4]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

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

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

output

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

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

amount.

output

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

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

output

```
Step D: how many orders of your current selection?
     (1) One
     (2) Two
     (3) Three
     (4) Four
     (5) Five
     (6) Six
     (7) Seven
     (8) Eight
     (9) Nine
?-> 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 **IceCreamOrder**s in a cart using a *regular* Java *array* as its underlying storage. To facilitate program testing, the cart capacity is limited to **5 IceCreamOrder**s.

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[5] 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[6], based on the number of orders in the cart:

[5]because of the increasing complexity involved.

[6]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

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

2. Main menu when the shopping cart is full

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

3. Main menu when the shopping cart is empty

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

```java
public class IceCreamStoreOnline
{
   public static void main(String[] args)
   {
      IceCreamStore shop = new IceCreamStore();
      shop.run();
   }
}
```

Output of a sample run of the program with MAX_ORDERS = 3

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

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

```

Output of a sample run of the program with MAX_ORDERS = 3

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

Step C: How much ice cream?
    (1) Single Scoop
    (2) Double Scoop
    (3) Triple Scoop
?-> Enter an option number: 2

Step D: how many orders of your current selection?
    (1) One
    (2) Two
    (3) Three
    (4) Four
    (5) Five
    (6) Six
    (7) Seven
    (8) Eight
    (9) Nine
    (10) Ten
?-> Enter how many orders: 1

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

You have selected to remove an order from your cart
What would you like to do?
    (1) 1 order of Double Scoop of Lemon ice cream in a Sundae for
$4.99 = 1 x 4.99
    (2) Exit this menu
?-> Enter an option number: 1

The order you selected was deleted

```

Output of a sample run of the program with MAX_ORDERS = 3

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

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

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

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

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

Output of a sample run of the program with MAX_ORDERS = 3

```

Step D: how many orders of your current selection?
    (1) One
    (2) Two
    (3) Three
    (4) Four
    (5) Five
    (6) Six
    (7) Seven
    (8) Eight
    (9) Nine
    (10) Ten
?-> Enter how many orders: 4

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

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

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

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

Output of a sample run of the program with MAX_ORDERS = 3

```

Step D: how many orders of your current selection?
    (1) One
    (2) Two
    (3) Three
    (4) Four
    (5) Five
    (6) Six
    (7) Seven
    (8) Eight
    (9) Nine
    (10) Ten
?-> Enter how many orders: 4

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

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

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

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

Output of a sample run of the program with MAX_ORDERS = 3

```

Step D: how many orders of your current selection?
    (1) One
    (2) Two
    (3) Three
    (4) Four
    (5) Five
    (6) Six
    (7) Seven
    (8) Eight
    (9) Nine
    (10) Ten
?-> Enter how many orders: 6

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

Your current selections of our scrumptious ice creams
-----------------------------------------------------
4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
$19.96 = 4 x 4.99
4 orders of Triple Scoop of Avocado ice cream in a Cup for
$19.96 = 4 x 4.99
6 orders of Single Scoop of Vanilla ice cream in a Sundae for
$23.94 = 6 x 3.99
-----------------------------------------------------

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

Output of a sample run of the program with MAX_ORDERS = 3

```

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

-------------------------------------=
Total price of all your orders in the cart: $63.86
-------------------------------------=

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

You have selected to remove an order from your cart
What would you like to do?
    (1) 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
$19.96 = 4 x 4.99
    (2) 4 orders of Triple Scoop of Avocado ice cream in a Cup for
$19.96 = 4 x 4.99
    (3) 6 orders of Single Scoop of Vanilla ice cream in a Sundae for
$23.94 = 6 x 3.99
    (4) Exit this menu
?-> Enter an option number: 4

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

Output of a sample run of the program with MAX_ORDERS = 3

```

Your current selections of our scrumptious ice creams
-----------------------------------------------------
4 orders of Triple Scoop of Hazelnut ice cream in a Cup for $19.96 = 4 x 4.99
4 orders of Triple Scoop of Avocado ice cream in a Cup for $19.96 = 4 x 4.99
6 orders of Single Scoop of Vanilla ice cream in a Sundae for $23.94 = 6 x 3.99
-----------------------------------------------------

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

You have selected to remove an order from your cart
What would you like to do?
    (1) 4 orders of Triple Scoop of Hazelnut ice cream in a Cup for $19.96 = 4 x 4.99
    (2) 4 orders of Triple Scoop of Avocado ice cream in a Cup for $19.96 = 4 x 4.99
    (3) 6 orders of Single Scoop of Vanilla ice cream in a Sundae for $23.94 = 6 x 3.99
    (4) Exit this menu
?-> Enter an option number: 2

The order you selected was deleted

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

```

Output of a sample run of the program with MAX_ORDERS = 3

```
Your current selections of our scrumptious ice creams
-----------------------------------------------------
4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
$19.96 = 4 x 4.99
6 orders of Single Scoop of Vanilla ice cream in a Sundae for
$23.94 = 6 x 3.99
-----------------------------------------------------

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

-------------------------------------=
Total price of all your orders in the cart: $43.90
-------------------------------------=

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

Your current selections of our scrumptious ice creams
-----------------------------------------------------
4 orders of Triple Scoop of Hazelnut ice cream in a Cup for
$19.96 = 4 x 4.99
6 orders of Single Scoop of Vanilla ice cream in a Sundae for
$23.94 = 6 x 3.99
-----------------------------------------------------
--------------------------------------------=
Total price of all your orders in the cart: $43.90
--------------------------------------------=

```

Output of a sample run of the program with MAX_ORDERS = 3

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

Cheers!

```

Evaluation Criteria for Assignment 4 (20 points)

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