Assignment 11: Reading Assignment and Programming Project
due 11/24 (Late deadline extended to 12/2)

For this week you have to complete a Reading Assignment, programming practice and a programming project.

The programming project is larger than usual and you have more time to work on it. However, you should start developing it as soon as possible to complete it on time.

Reading Assignment

Read Sections 5.1-5.3, 5.5 from Chapter 5, More about Objects and Methods completing self-check exercises.
Review Handouts 11 and 12.

Programming Practice

Create a project for the DepartmentApp application. That application consists of three java files Employee.java, Department.java and DepartmentApp.java.

Place all three java files in the same project folder and compile them together. Examine the code and structure of the classes carefully, trace the execution of the DepartmentApp.java stepping into each method and make sure you understand well what is going on.

The programming project that you are assigned for this week resembles this program, as it also requires implementation of a container class. In DeartmentApp application Department is the container class - each object of this class contains a set of Employee objects.

Programming Assignment

Programming Project

ShoppingCart: Items in a Shopping Cart
due 11:00 p.m. on Thursday, 11/24
worth 12 points

In this assignment you will create an application that implements an electronic shopping cart.

First, define a class ShoppingCartItem to represent a shopping item. Each item has a name, e.g. Clock, the quantity being purchased, e.g. 2, and the price per one item in dollars (we will assume the price of each item is an integer number), e.g. 20. You must include a constructor that is passed the name of the item and the price, an accessor method for each instance variable as well as methods for setting the quantity being purchased, and a method that returns the total price for purchasing that quantity of items. You may include other methods as you find necessary.
Then, define a class **ShoppingCart** that represents an electronic shopping cart capable of storing up to 10 different shopping cart items (items are considered same if their name and their price are the same).

The cart must be created initially empty. Your class definition must define public instance methods

- **AddNewItem(itemName, price, quantity)** that adds an item to the shopping cart. The parameters passed describe the name, price and quantity of the item being purchased.
  
  If the shopping cart already contains 10 different items, the method should not take any action except for printing a message **Shopping cart is full** and should return **false**.
  
  If the shopping cart already contains the item with the same name and the same price, the previous quantity of that item should be increased by **quantity**.

- **HowManyOf(itemName, itemPrice)** that returns the quantity of the item with the name **itemName** and price **itemPrice**. If the item with this name and price is not in the shopping cart - it should return 0.

- **ComputeTotalCost()** to compute the total cost for the purchase.

- **PrintItemList()** to print the list of all items. Each line of the printout must contain the name of the item, its price and quantity as shown in the sample interaction below. Items must be printed in the order they were entered.

Finally, create a class **Shopping** with a main method that uses a **ShoppingCart** object to do the following.

The main method should contain a loop that allows the user to enter one of the 4 following commands

1. A for Adding a new item to the shopping cart,
2. C for Checking the quantity for an item with certain name and price,
3. P for Printing out the contents of the entire shopping cart,
4. S for Stop.

If the user enters something other than A, C, P or S no action should be taken and the next command should be read.

For each command an appropriate data should be requested from the user and the appropriate action taken. When the user enters S (stop) - the program must display the total amount due and terminate.

Here is a sample interaction. Note the format of the output from the C, P and S commands.

Greetings! You may start shopping now.
The commands are as follows: A for Adding an item, C for Checking the quantity of an item, P for Printing the contents and S for Stopping and displaying the total cost.

Please enter the next command (A,C,P or S): A
What is the name of this item? CD-Player
What is the price of one item? 90
What is the quantity purchased? 1
Added to the shopping cart.

Please enter the next command (A,C,P or S): A
What is the name of this item? CD-Player
What is the price of one item? 120
What is the quantity purchased? 2
Added to the shopping cart.

Please enter the next command (A,C,P or S): A
What is the name of this item? Clock-Radio
What is the price of one item? 20
What is the quantity purchased? 2
Added to the shopping cart.

Please enter the next command (A,C,P or S): C
Check quantity of which item? CD-Player
What is the price of this item? 120
Quantity is #2

Please enter the next command (A,C,P or S): A
What is the name of this item? Clock-Radio
What is the price of one item? 20
What is the quantity purchased? 3
Added to the shopping cart.

Please enter the next command (A,C,P or S): C
Check quantity of which item? Clock-Radio
What is the price of this item? 20
Quantity is #5

Please enter the next command (A,C,P or S): C
Check quantity of which item? Zanzibar-Brush
What is the price of this item? 10
Quantity is #0

Please enter the next command (A,C,P or S): P
Contents of the shopping cart:
CD-Player 90 1
CD-Player 120 2
Clock-Radio 20 5

Please enter the next command (A,C,P or S): S
The total cost of these items is #430

How to go about developing this application

Develop the ShoppingCartItem class first. Test it thoroughly. Then develop the ShoppingCart class and test every method of that class. Then, I suggest you start developing the main method of ShoppingCart class with getting the loop that just lets the user enter the command while the command is not S, and stops when S is entered. Then, add appropriate actions to each of the commands.