# Homework 4 – LAST HOMEWORK OF THE SEMESTER!

## Directions

* Your work must be your own. Do not collaborate. I require unique answers. Copying code or solutions from websites or textbooks and claiming it as your own is plagiarism. Don’t do it.
* For each question, write well-formed, *executable* Java or C# code for the following scenarios. Your code *must* compile, be thoughtfully named (no variables like X), and have inline comments to explain your work.
* Put all code into a single text file. I must be able to copy/drop your code into an IDE to compile, execute, and examine it. Do not submit a PDF, Word document, compiled file, etc.
* If your code does not compile, your maximum score will be 50.

### Scenario

You are building a shopping cart application for a customer. The customer’s CTO specified that the cart must implement the State Machine pattern.

You will need a shopping cart object that stores the items, prices, and quantities, plus the payment method details.

The shopping cart machine states are:

1. **AddItems**
   1. Ask the user to enter the name of the item, quantity, and price. The name must be at least 5 characters, price must be greater than 0, and the quantity must be greater than 0. *Do not* allow invalid input to be added to the cart.
   2. Store each item’s name, quantity, and price in the shopping cart.
   3. This state requires at least one 1 item in the cart before going to the next state.
   4. Customers either say “done” to move to the payment state or continue entering items.
2. **ChoosePaymentMethod**
   1. Choose “cash” or “check”
   2. Conditionally change to the AcceptPaymentCash or AcceptPaymentCheck state.
   3. If input is *invalid*, prompt the user again.
3. **AcceptPaymentCash**
   1. Show total cost of items in the cart (sum of item cost \* quantity).
   2. Accept payment equal to the total cost. Less or more is invalid.
   3. Store the payment amount, type and payment type (case).
   4. Once *valid* payment has been made, change to the ShowReceipt state.
4. **AcceptPaymentCash**
   1. Show total cost of items in the cart (sum of item cost \* quantity).
   2. Accept payment equal to the total cost. Less or more is invalid.
   3. Accept a check number. The check number must be greater than 100.
   4. Store the payment amount, type, payment type (check), and check number.
   5. Once *valid* payment has been made, change to the ShowReceipt state.
5. **ShowReceipt**
   1. Dump the entire contents of the cart object to the console.
   2. Ask the user if he/she wishes to make another purchase. If “yes”, reset the cart and change state to Add Items and restart the cart process.
   3. Otherwise, exit.
6. **50 POINTS EXTRA CREDIT - OPTIONAL:**
   1. Use the decorator pattern to display the receipt. You must include the following decorator components:
      1. Separator  
         *This is a dashed line ------------------------*
      2. Line items  
         *Number, Item Name, Price, Quantity, and Extended Price  
         For example  
         #1) Large Sneakers, $25, 2, 2\*$25 = $50*
      3. Total Price  
         *For example, Total = $225*
      4. Payment Details  
         *All details captured when cash or check was stored*.
      5. Return Policy  
         *Make this up – just a bunch of text*
      6. Thank you  
         *Make this up – just a bunch of text*.
   2. Using the decorator pattern, output the receipt when the state machine goes to ShowReceipt like the following:
      1. Separator
      2. Line Items
      3. Total Price
      4. Separator
      5. Payment Details
      6. Separator
      7. Return Policy
      8. Separator
      9. Thank You
      10. Separator

### Hint

If you don’t know where to get started, see the state machine code I demonstrated on Thursday 10/25 in class.

### Instructions

1. Carefully follow the instructions in the **Scenario** section. If you’ve lost points in the past for missing requirements, print them out and strike them off as you achieve them. Recheck your work.
2. Comment your code as appropriate and use excellent OO design methods. Bad variables, awful method names, etc. will lose points.
3. The Extra Credit portion is optional. I *highly* recommend doing it if your Homework Average is less than 80%.
4. Submit **a single zipped** file of your Visual Studio or Intellij project.
   1. If your project is C#, name the Visual Studio project zip file ***yourname*\_ homework4\_charp.zip**.
   2. If your project is Java, name the Intellij project zip file ***yourname*\_ homework4\_java.zip**.

*To avoid losing no less than 10 points, ensure that your zip file can be opened and executed with no additional effort.*

1. If you have lost points for code that didn’t compile in the past, I recommend opening your zip file in a new project and verifying it works. Never submit broken code or hard to open projects. The biggest part of your job is creating code that works. If your project fails to compile, the best grade you can hope for is **50**. If you are concerned I will have difficulty running your project, include build instructions. Always err on the side of convenience for the person reviewing your work.
2. When you submit, let me know if you attempted the Extra Credit portion in the D2L submission notes.
3. If you are struggling with the homework, you need to get more involved during lecture and after class. I stay after class every Tuesday night helping students, plus I can optionally stay late on Thursday after class with a bit of notice. You should also available yourself of the demo code I post and step through it carefully.