**Workout 2020-10-09, Template Pattern #2 Name: Jeffrey Mosteller**

Instructions: Answer the following questions in this document and save it in the root directory of your program for Question 3. This workout is due at the beginning of class Wednesday and counts as 2 workouts.

I’m intentionally leaving some of the specifications flexible. Every program submitted will look different based on the design decisions you make. Your goal by the end of class today should be to at least complete step 1 below. I’m here to help.

# Problem

Gilbarco manufactures most, if not all, pay at the pump gas stations. Gas stations have slightly different processes for pay at the pump, but all processes have basically the same sequence of actions. The ideas of the template pattern are the same as in your last workout; however, the template method will be more complex. To simplify, I’ve limited transactions to credit card purchases.

1. Carefully think through what you do when you purchase gas at different stations. Develop an abstract list of steps that describe a generalized process that can fit most any gas station. I’ll get you started:
   1. Initiate purchase process
   2. Swipe or insert card
   3. Validate cards
      1. Each station will have its specific list of cards accepted.
      2. Validation sometimes requires additional data like zip code.
      3. Cancel purchase if card not accepted.
   4. Select carwash (if available at the station)

You design the rest of the algorithm through printing a receipt. Don’t feel like you must account for anything that can go wrong in purchasing gas, but show enough to make the program interesting.

1. Given your list of steps in the process and using the Template pattern. Draw the UML for your solution. I suggest creating default implementations for most steps in the process so that creating the concrete gas stations will be easier

----Ask for a card to swipe/insert

----Validate card

--------If not valid cancel transaction

----If valid, proceed

----Credit or Debit?

----How many gallons?

----Car Wash (yes or no)?

------Add to total if yes, don’t add to total if no.

----Print total on receipt

----Print farewell greeting.

Diagram

Description automatically generated

1. Now write the code for your solution and create two concrete gas stations. For most steps you will simply print a statement saying what happens, but some steps will require some user input and some will return values.  
   Gas stations need configuration data (such as name of the gas station, what credit cards are accepted, whether it has a car wash and the cost of the car wash, etc.). This data should be passed to the gas station constructor and stored as fields in the class.  
   Include a Client class with main that instantiates two concrete gas stations and perform two transactions with each station. Let one out of three card verifications fail so you can test that code. Include print statements that clearly separate and briefly describes what station is being used when you start a gas purchase.