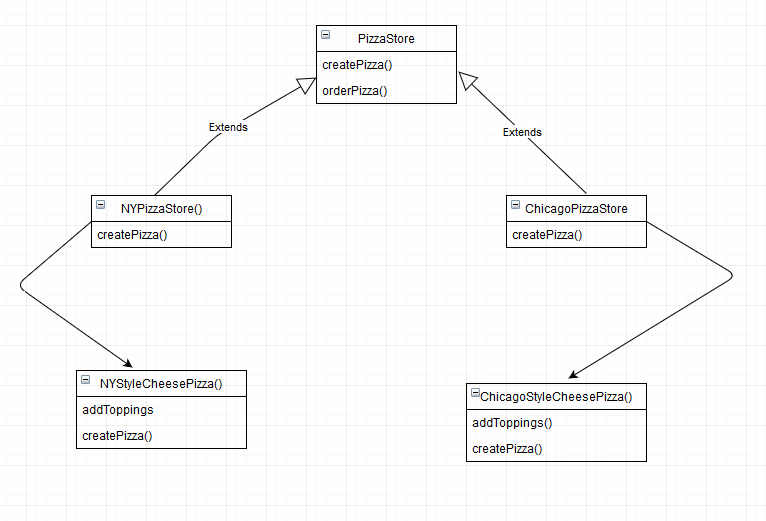
# Workout – 2019-09-25 – Command Pattern

You are to work in pairs on this assignment. The pairs are assigned in ASULearn. You may not finish the code during class, I encourage you to create a shared Google Directory and finish the program together outside of class. Make sure both name are at the top of this document and in a comment at the top of RemoteLoader.

Create a workout directory. Edit this document with answers to questions 1-10 and place it in the workout directory. Then place all your code in that directory. When you are finished with the workout, zip up the workout directory and submit the zip file for this assignment.

1. Match each Diner Example component to the appropriate Command Pattern concept (Command, execute(), Client, Invoker, Receiver, setCommand()):
   * Waitress - **Invoker**
   * Short Order Cook - **Receiver**
   * orderUp() - **execute()**
   * Order - **Command**
   * Customer - **Client**
   * takeOrder() - **setCommand()**
2. Draw the UML for the generic Command Pattern. Export the result as a graphic and paste it into your document.



1. What does the client need to know about specific command objects?
   * It is responsible for creating the command objects.
2. What does the Invoker need to know about command objects?
   * Only the functions the command objects have i.e Execute()
3. Does the answer to the last two questions exemplify “loose” coupling? Explain.
   * Yes, as the client and invoker only know the functions in the command. The command is responsible for the inner logic.
4. What is the NoCommand object? Why is it often useful?
5. In the context of the Command Pattern, define the following terms:
   * Invoker - **Invokes the command passed to it and optionally does bookkeeping about the command execution.**
   * Receiver - **Knows how to perform the work needed to carry out the request.**
   * Command - **has a receiver object and invokes a method of the receiver in a way that is specific to that receiver’s class.**
   * execute - **The function hosting the logic for the object to do when invoked by the invoker.**
   * Client - **The client contains the decision making about which commands to execute at which points. To execute a command, it passes the command object to the invoker object.**
6. Match each Home Automation Example component to the appropriate Command Pattern concept (Command, execute(), Client, Invoker, Receiver, setCommand()):
   * Remote Control - **Invoker**
   * Light - **Receiver**
   * LightOnCommand - **setCommand()**
   * RemoteLoader - **Client**
7. In relation to the Design Principle, "Encapsulate what varies.", what is being encapsulated in the Command Pattern?
   * The command objects.
8. How does the Design Principle, "Program to interfaces, not implementations.", apply in the Command Pattern?
   * This pattern only implements the command interface which is used by the invoker to call each of the functions that the command interface implements.
9. The source code for the Remote Control project in the textbook is available in a zip file in this section of our ASULearn course. If any file you need for the exercises below has a package statement, remove it. Make the following changes to the code.

* Change the Hottub controls so that the user can increment and decrement the temperature.
  + Add a getter to Hottub for temperature. Add code to the Hottub so that the temperature can never go outside the range 80 to110 (make those values constants in HotTub and modify all methods that change the temperature to enforce this rule).
  + When the Hottub is turned on, always make the initial temperature 105.
  + Write two command classes: IncrementHottubTemperature and DecrementHottubTemperature that correspondingly add 1 or subtract 1 to/from the temperature. In addition to changing the hottub, make each command displays a message stating what the command did along with the resulting hottub temperature. These commands should do nothing if the hottub is off.
* Modify the RemoteLoader class adding a 5thwith a pair of commands on/off for the Hottub. Make the 6th pair of commands incrementing and decrementing the Hottub temperature.
* Add the following remoteControl commands to the end of main:
  + Turn on the hottub
  + Increment the hottub temperature 6 times
  + Decrement the hottub temperature
  + Turn off the hottub