**Salad Chef Test Notes:**

In this application, I wanted to show several methods of inter-object communication including “SendMessage()”, Singleton direct calls, and the basic “m\_object.publicValue = 42” direct assignment. I also wanted to show game flow, UI, audio, and animation and how they could be implemented using those techniques. I also wanted to show how to handle frequent allocations and deallocations of fixed size objects—this to combat memory leaks over time.

The main application’s state machine proceeds as such:

**Init** Merely connects some pointers and resets the player(s)

**Start** Player names, One/Two player mode select, Start

**Game** The Game

**Hi-Scores** List of high scores. This will return to Idle above on “Retry” or “Quit” the app completely.

*No “Pause” functionality*

**Here are some things to look for in the code**

1. All reusable objects manage their own allocations. Namely in Order.cs, Ingredient.cs, Main.cs. “Instantiate()” called once. All objects that allocate, are responsible for freeing that memory. The do not pass allocated objects along in the hopes another object will manage it.
2. Message passing using “SendMessageUpwards()” from peripherals for game events “OnPlayerScored()” is a good example that comes from game logic. I also then for UI, animation, and collision. This is to show an OO approach. Also
3. Singletons are used as well. These are not OO but they are clean. Look at “Ingredient.Grab()” or “Waiter.SubmitPlate()”. The whole Audio system, as seen by the game, is a singleton.
4. There is also some valid but not recommended uses of public member variables such as “”m\_playerVitals.m\_timer = waitTime;”. I use these internally for only the class, “Player”, that needs them.
5. Anything requiring a frequent search is contained in a “Dictionary<>”. Any objects that require frequent iteration are in “List<>”
6. I used extremely verbose member and method names. This is for self-documenting the code. Rare but their might be some “int i” or “int iter” decls in there. Just a few.
7. Both the player and main application use state machines.

I tested it but there are still a few bugs I fixed at the 11th hour. The kind that only occur after playing all the edge cases.

I think with a week or two of love, you could actually submit this to the app store. It is “fun” for a few minutes but gets boring quickly. The “love” part could add other challenges including ingredient shortages and the players battling for them. Something…It would be cool.