Dear Hypothetical Programmer:

I really appreciate you asking me to review your Joke Generator application. It was an interesting exercise.

First I would like to talk about the things that you did right.

Use APIs for Retrieving Data

It was a very good idea to use the ChuckNorris and the privServ APIs to get jokes and random names. By offloading this functionality to established third parties you save on development time and end up with a solution that is superior to what you could have created within the given time.

Now the bad ☹

There are several areas where improvements need to be made. You can review my solution to see how this could be done.

All development work should make extensive use of automated testing. Using xUnit, nUnit or any other test framework actually makes development easier and saves time. Tests should cover individual processes (like the queries against the api) as well as integration processes (like the flow of the application).

When I was refactoring the application I usually started with the test cases. First you write the test case for the failure, and then you fix the code to make the test case pass. For example, when I wrote the functions to access the PrivServ API and return a first and last name, I first wrote the test case expecting the return to be in that format. Obviously this test failed until I had changed the code. When it succeeded I had both the confidence that I had coded the module correctly, as well as knew that if I made changes in the future breaking this functionality, it would be easy to spot.

You would see how test cases help by looking at some of the functionality that you wrote that just did not work. For example, users asking for a number of jokes, did not receive the number they asked for. A test case would have ensured that this worked correctly, and kept working correctly as you made revisions.

Another issue in the original application is the extensive use, of global variables and reusing them repeatedly. For example in your JsonFeed.cs file you have the following variable declaration:

static string \_url = "";

You then use that variable for both the Chuck Norris API as well as the PrivServe API. I know I was confused trying to figure out what url you were using.

Another example is the entire ConsolePrinter.cs module. I understand the purpose of wrapping functionality up to make it easier to manage but this module as written neither made logic easier to follow, nor did it make it easy to modify. Have a look at how I created virtual Print and Input functions in my flow module. These functions handled all of the reading and writing to the console. They were able to be easily modified when I tried cleaning up the interface with things like spacing and command prompts. They also were easy to override for the purposes of creating tests simulating the console commands.

In general, it is considered bad practice to put significant logic within the Program.Main(string[] args) function. You will notice how I moved the logic to a separate Flow module. Within that module I further broke the question flow down so that it would be easier to read.

Reviewing that you will also see how much of the strings I moved to resource files. Both the questions as well as the urls for the api were moved. This makes finding and modifying this data much easier. It also makes it easier to have an external party verify copy or handle changes to endpoints for various environments.

One last point, even for a console application, improving readability and usability is important. A simple thing like spacing, formatting and just rewording questions to be more polite is important.

Anyways, please have a look at my new solution. It has many areas where it has improved. Its usability is significantly improved, it is going to be much easier to maintain and expand, and it should be far more reliable.