Github:<https://github.com/iraspopo/CSharp-Challenge-by-Geotab>

**Requirements:**

**Basic** (extracted from initial code):

* User shall be able to get Chuck Norris jokes based on provided jokes category
* User shall be able to replace Chuck Norris name in the joke with a randomly generated name.
* User shall be able to select between 1 and 9 jokes in total

**Usability (**added to improve user experience (usability):

User input/interaction to obtain above info shall be streamlined to minimum.

* Removed need for user to enter “?” to receive initial input selection options.

Following initial options/greeting will be shown by default :

Lets get some Chuck Norris jokes!

Press 1 - Get one Chuck Norris joke

Press 2 - Make customized Chuck Norris joke/s

Press x - Exit

* Provided option to get single joke with single selection (to minimize number of inputs user will need to do if they just want single joke)
* Provided selection to customize joke/s with following guided set of selections:
  + - Want to use a random name y/n
    - Want to specify a category? y/n
    - How many jokes do you want? (1-9)
* Provide users with category options every time they are asked to specify a joke category. Initial implementation offers an option to get jokes categories but if the user did not use it prior to the customized joke workflow it will not be aware of the options.
* If user makes mistake in typing the category it will be notified and presented with options again
* All other user input reacts on single key input (user does not need to press enter)
* All user input is validated and the user is represented with selection again if not valid.
* Users are provided with an option to exit the program (with option x).
* Users are provided with some sort of progress bar indicator while waiting for all jokes to be collected.

**Robustness**

* All user input is validated and the user is represented with selection again if not valid. This prevents unexpected crashes due to invalid user input.:
* Replacement of Chuck Norris name is made more robust as it will ignore string case
* Replacement will be done for any number of occurrences of the Chuck Norirs name and not just the first occurrence in the joke.
* Adding basing error handling for failed HttpClient requests

**Efficiency**:

* Search for Chuck Norris name is done not against the whole json response but only against the Value field (where the actual joke is). This should increase search speed.
* Use of Replace and Regex API in a single call will replace any number of Chuck Norris (first or last name occurrences) in the joke regardless of the case sensitivity. This should be faster than using the existing IndexOf API.

**Bugs and design deficiencies**:

* Missing url endpoints (to get categories)
* Missing key input handlers (use of key ‘2” for example)
* JsonFeed - Deserialise to proper type (example sting[]) when handling retrieval of categories
* Extensive use of global/static variables (names, results, ..)
* ConsolePrinter class ToString() method returns null ?
* Missing default constructors
* Missing HttpClient error hanling

**Design** : Decouple code to classes with fairly specialized responsibilities/concerns

* Handling user input (**Program** class)

* Printing user prompts for selection or results of the selection(**ConsolePrinter**)
* Controller that is responsible handling logic of how to get and return data for a given selection. (**ChuckNorrisController**)
* Fetching data from the back-end (server) (**JsonFeed**)

Sequence diagram and component interactions:



Missing but identified improvements for the future:

* Add tests
* Consider using [System.Threading.Tasks.Parallel](https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.parallel) to execute multiple GET calls for jokes in parallel