

Joke Generator

Joke generator is a joke generating console application written using .NET Core and C#. The application gets random Chuck Norris jokes and can substitute other people's names into the joke instead of Chuck Norris'. This application uses different API to get a random name and jokes.

Review from the existing code:

- **No User Input Validation**
 - Sometimes users may not enter what the prompt expects. So, there should be something to validate the user's input.
 - There is no validation whether the user has entered a positive number or negative number in case of a number of jokes or in case of any other character inputs when the user input is asked.
- **No Exception Handling**
 - Even if the user input validation is there, there should always be some code catching exceptions.
 - Not able to find any try-catch statement in code. If an unexpected error occurs, it will be bad user experience.
- **Poor usability and UX**
 - Sometimes it's only required to press a single key, and sometimes it required to enter a value and then press enter this should be made clear.
 - No UX to denote the progress of any selected action. E.g. fetching the jokes from API the user knows nothing whether the application is processing, or it's crashed!
 - If the user enters 'n' when asked, "Want to use a random name? y/n" then the program doesn't ask for their choice of name to use in the jokes.
 - After entering the program and press '?' no way to exit!
- **No comments/documentation or proper naming convention**
 - Naming convention is not followed e.g. Getnames() instead of GetNames()
 - No proper namespace name, e.g. ConsoleApp1
 - The only comment in the whole project was for the method Getnames() and it is unable to describe what exactly it does.
- **Some features not working**
 - Random name feature is throwing error.
 - Error while fetching categories.
 - Only one joke is printed every time, even after selecting a number greater than 1.
- **No Unit Tests**
 - A lot of the coding errors could have been prevented by having some unit tests written.
 - It would have helped to structure code, find errors within code and test API responses.
- **Other**

- Logic and UI are dependants on the same file 'Program.cs'. While 'Program.cs' should act as a central node that contains information on how to build all layers together to do the required job and each layer should be separate.
- The file overall had some inconsistent indentations, this could lead to misreading the code.
- No production standards or any design pattern were used
- No scope of extension and reusability

Improvements:

- **Completely new UX and user interface:**

Main Screen:

The screenshot shows a terminal window with a dark background and green text. At the top, it says "Welcome to the Joke Factory" between two lines of asterisks. Below this, it displays the "Current Joke Configuration:" with three lines: "Selected Name: Chuck Norris", "Selected Category: None", and "Jokes To View: 1". A green arrow points from this configuration to a yellow box labeled "Default Configuration". Further down, it prompts the user to "Press one of the following keys to configure your joke:" followed by a list of options: J (To generate a joke), A (To choose the number of jokes), C (To select a category), N (To provide the name you wish), R (To generate a random name), and Q (To quit). A green arrow points from this list to a yellow box labeled "Options to configure jokes".

```

*****
Welcome to the Joke Factory
*****

Current Joke Configuration:
Selected Name: Chuck Norris
Selected Category: None
Jokes To View: 1

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
  
```

- **Functionalities:**
 - Added default functionality so whenever the user comes by default, they can at least view one joke with the default configuration as a demo.
 - Additionally, the "Current Joke Configuration" functionality offers the user a dynamic experience by keeping them updating with the latest configuration values for the selected name, selected category and number of jokes to view. Through this, user can anytime look at the current configuration and perform customization very easily without memorizing the configuration in their head.
 - For configuration, it provides six options through a single keystroke, which is described in the following sections.
- **To generate your joke:**

```
*****
                                     Welcome to the Joke Factory
*****


Current Joke Configuration:

Selected Name: Chuck Norris
Selected Category: None
Jokes To View: 1

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
j

Working on your jokes.....
```



```
*****
                                     Joke(s):
*****

=> Jesus can walk on water, but Chuck Norris can walk on Jesus.

Press a key to go back to the main menu
```

- **Functionalities:**
 - Press key 'j' from the keyboard to view the configured joke.
 - It shows the joke building is in progress, this helps user to understand that the application is processing their request and is not crashed or there are no errors.
- **To chose number of jokes:**
 - One can press key 'a' when they want to change the number of jokes to view
 - Once they enter their selection e.g. 5 the application will show the updated value in the "Current joke configuration" area.

```
*****
Welcome to the Joke Factory
*****


Current Joke Configuration:

Selected Name: Chuck Norris
Selected Category: None
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
a

How many jokes do you want? (1-9)
```



Asks user to enter
number of jokes one want

- **To select a category:**

- To view the available categories and select one from them user can press key 'c'
- User can either specify their category ('y') or if they don't want to specify a category, they can simple enter 'n' for no
- Once user press 'y' the application will prompt the user to enter their selected category.
- Once the user presses enter the category is changed in the "Current joke configuration"

```
*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Chuck Norris
Selected Category: None
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
c

List of categories:

    animal |    career |  celebrity |    dev |
  explicit |    fashion |    food |    history |
    money |    movie |    music |  political |
  religion |    science |    sport |    travel |

Want to specify a category? y/n
y

Enter a category:
[Press enter]
animal
```

Shows available categories to chose

Asks users if they want to specify their choice of category

User specified category

```
*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Chuck Norris
Selected Category: animal
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
```

- **To provide the name you wish:**
 - User can also specify their choice of name instead of default “Chuck Norris” by pressing key ‘n’
 - The functionality will ask for user’s first name and last name. User can skip entering either of the names by simply pressing enter.

- The example is shown with both the first name and last name.

```
*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

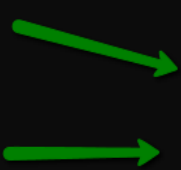
Selected Name: Chuck Norris
Selected Category: animal
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
n

Please enter a first name
[Press enter]
Saurav

Please enter a last name
[Press enter]
Agarwal
```



Provides option to change the first name and last name

Once the user presses enter the “Selected name” will change from “Chuck Norris” to “Saurav Agarwal”

```
*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Saurav Agarwal
Selected Category: animal
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
```

If then user presses key ‘j’ to view the jokes, the user will be able to see their custom name in 5 jokes of animal category.

```

*****
Joke(s):
*****

=> Saurav Agarwal once rode a bull, and nine months later it had a calf.

=> Saurav Agarwal once kicked a horse in the chin. Its descendants are known today as Giraffes.

=> Saurav Agarwal once kicked a horse in the chin. Its descendants are known today as Giraffes.

=> They say curiosity killed the cat. This is false. Saurav Agarwal killed the cat. Every single one of them.

=> Saurav Agarwal once kicked a horse in the chin. Its descendants are known today as Giraffes.

Press a key to go back to the main menu

```

- **To generate a random name:**

- If user does not want to enter explicit name, then one can press key 'r' to generate a random name (first name + last name) for the jokes
- Once pressed the system will update the newly considered random name in the "Current joke configuration"

```

*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Veronica Părvulescu
Selected Category: animal
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit

```



Press key 'j' to view the jokes for random name

```

*****
Joke(s):
*****

=> They say curiosity killed the cat. This is false. Veronica Părvulescu killed the cat. Every single one of them.

=> Veronica Părvulescu once kicked a horse in the chin. Its descendants are known today as Giraffes.

=> Veronica Părvulescu can set ants on fire with a magnifying glass. At night.

=> Veronica Părvulescu can set ants on fire with a magnifying glass. At night.

=> Veronica Părvulescu once kicked a horse in the chin. Its descendants are known today as Giraffes.

Press a key to go back to the main menu

```

- User input validation

```

*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Veronica Pârvulescu
Selected Category: animal
Jokes To View: 5

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
z
Options to respond to the question(s) are: j,c,r,n,a,q
Please Try Again

```

If keys other then the specified key is entered the application will throw validation error

```

*****
Welcome to the Joke Factory
*****

Current Joke Configuration:

Selected Name: Chuck Norris
Selected Category: None
Jokes To View: 1

Press one of the following keys to configure your joke:

| J | To generate a joke
| A | To choose the number of jokes
| C | To select a category
| N | To provide the name you wish
| R | To generate a random name
| Q | To quit
a
How many jokes do you want? (1-9)
j
Please enter a positive numeric value.
Please Try Again

```

If user enters values other then the numeric values the application will throw validation error.

The application will only give three attempts to the user. If the user does not enter the correct input, then the application will quit.

- Refactoring

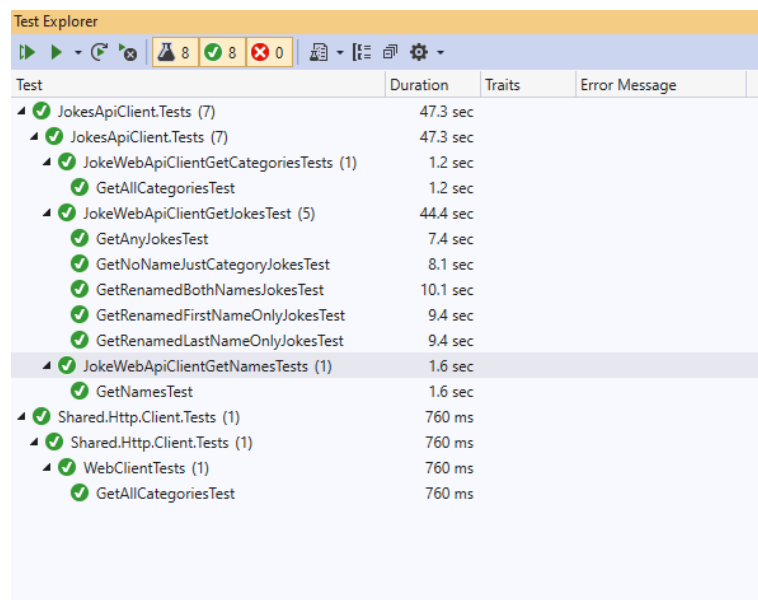
- Complete refactoring of existing code so that there is no dependency on any layer.
- Main idea of refactoring this project is to separate UI presentation from user activities. 'ApplicationService' contains the logic of user behavior but it communicates with

interface 'IPresentationBehavior' and knows nothing about any information of the implementation related to console.

- Now easily other features and implementation can be added without depending on existing code. Class was decoupled for proper implementation of getting data by 'IJokeWebApiClient'.
- The 'Program.cs' is a host. This is just one place which knows how to build all layers together to do required job.
- Separated into different project to decouple and for easy extension and future maintenance
- Earlier application was having dependency on some external functionality like data was provided by HTTP. So, I have extended a separate project for external API.
- Taking API name from Appsetting to decouple dependency from code.
- Separate folder for application behavior and console presentation.
- Used different design patterns for easy extension and reusability.

• Unit Test

- Added different unit test cases to test API call based on different scenarios

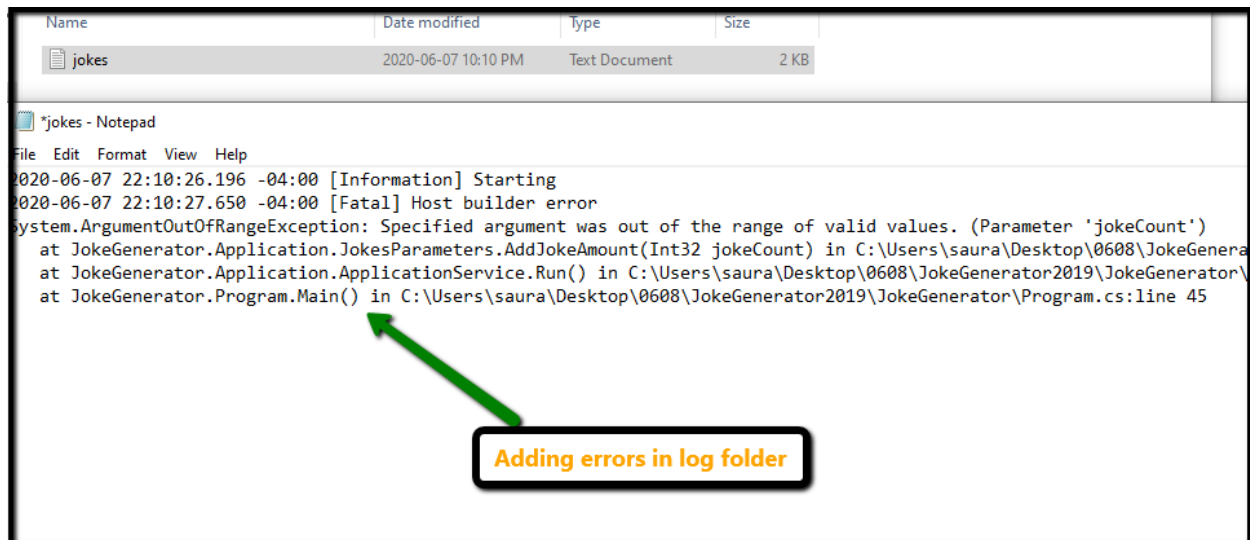


Test	Duration	Traits	Error Message
✓ JokesApiClient.Tests (7)	47.3 sec		
✓ JokesApiClient.Tests (7)	47.3 sec		
✓ JokeWebApiClientGetCategoriesTests (1)	1.2 sec		
✓ GetAllCategoriesTest	1.2 sec		
✓ JokeWebApiClientGetJokesTest (5)	44.4 sec		
✓ GetAnyJokesTest	7.4 sec		
✓ GetNoNameJustCategoryJokesTest	8.1 sec		
✓ GetRenamedBothNamesJokesTest	10.1 sec		
✓ GetRenamedFirstNameOnlyJokesTest	9.4 sec		
✓ GetRenamedLastNameOnlyJokesTest	9.4 sec		
✓ JokeWebApiClientGetNamesTests (1)	1.6 sec		
✓ GetNamesTest	1.6 sec		
✓ Shared.Http.Client.Tests (1)	760 ms		
✓ Shared.Http.Client.Tests (1)	760 ms		
✓ WebClientTests (1)	760 ms		
✓ GetAllCategoriesTest	760 ms		

• Exception handling

- Added exception handling at each step to provide information to user, developer and tester.
- Whenever any error occur, proper message will be provided to the user.
- For developer/tester I have added logger as a new feature
 - Through error logger whenever any error occurs then developer or tester can check the log file to investigate the details of error to solve the issue easily.

- Path for error log folder:
`\JokeGenerator\bin\Debug\netcoreapp3.1\Logs\jokes.log'`



- **Implementation Details:**

I have created two different projects for this application one in Visual Studio 2017 using .Net Core 2.1 and another project in Visual Studio 2019 using .Net Core 3.1.

Project 1:

- **Prerequisites:**
 - **Microsoft Visual Studio 2019 (.Net Core 3.1)**
- **Build and Run Locally:**
 - Open the JokeGenerator.sln file in Visual Studio.
 - To build the project, choose Build Solution from the Build menu. The Output window shows the results of the build process.
 - To run, on the menu bar, choose Debug, start debugging/Start without debugging or by pressing F5. This will open the command prompt and run the project and will display the welcome menu.
- **Running the tests:**
 - To run tests manually in MS Visual Studio:
 - Test -> Run All Tests
 - Test -> Run -> All Tests
- **Unit Test Case Result:**

Test Explorer

Test	Duration	Traits	Error Message
✓ JokesApiClient.Tests (7)	33.1 sec		
✓ JokesApiClient.Tests (7)	33.1 sec		
▶ ✓ JokeWebApiClientGetCategoriesTests (1)	1.5 sec		
✓ JokeWebApiClientGetJokesTest (5)	30.7 sec		
✓ GetAnyJokesTest	5.8 sec		
✓ GetNoNameJustCategoryJokesTest	5.5 sec		
✓ GetRenamedBothNamesJokesTest	6.9 sec		
✓ GetRenamedFirstNameOnlyJokesTest	5.6 sec		
✓ GetRenamedLastNameOnlyJokesTest	6.9 sec		
▶ ✓ JokeWebApiClientGetNamesTests (1)	932 ms		
✓ Shared.Http.Client.Tests (1)	820 ms		
✓ Shared.Http.Client.Tests (1)	820 ms		
▶ ✓ WebClientTests (1)	820 ms		

Group Summary

JokesApiClient.Tests

Tests in group: 7

🕒 Total Duration: 33.1 sec

Outcomes

✓ 7 Passed

- **Module Implementation details:**

JokeGenerator:

- Main idea of refactoring this project is to separate UI presentation from user activities. 'ApplicationService' contains the logic of user behavior but it communicates with interface 'IPresentationBehavior' and knows nothing about any information of the implementation related to console.
- Now easily other features and implementation can be done without depending on existing code. Class was decoupled for proper implementation of getting data by 'IJokeWebApiClient'.
- The 'Program.cs' is a host. This is just one place which knows how to build all layers together to do required job.

Implemented Shared.Http.Client project:

Reason: To separate code related to the communication by WebApi from client side.

- This project doesn't connect with anything related to "Jokes".
- The project can be reused by any application communicating via WebApi.
- It can be extended easily. One can easily add additional functionality like PUT, POST, REMOVE commands.

Any URL contains this parts:

- a) Base url
- b) Path part
- c) Parameters part

Example:

Url: [https://api.chucknorris.io/jokes/random/?name=Isabel Phillips&category=animal](https://api.chucknorris.io/jokes/random/?name=Isabel%20Phillips&category=animal)

Base address: <https://api.chucknorris.io>

Path: jokes/random

Parameters:

1. name=Isabel Phillips
2. category=animal

Any URL (Even not related with the Joke's API can be configured like that)

- Earlier application was having dependency on some external functionality like data was provided by HTTP. So, I have extended a separate project for external API. This functionality became remote facade. It means you can work with remote functionality and still won't realize that the data is coming from remote server.
Reason: It's to separate logic of communication with external system in one place. There is an interface `IJokeWebApiClient`, so by that you just don't know the data takes from remote server.
- Any thread is an expensive resource and if we don't block any then it gives us a way to improve the app. That's why I have used asynchronous code

Project 2:

- **Prerequisites:**
 - **Microsoft Visual Studio 2017 (.Net Core 2.1)**
- **Build and Run Locally:**
 - Open the `JokeGenerator.sln` file in Visual Studio.
 - To build the project, choose Build Solution from the Build menu. The Output window shows the results of the build process.
 - To run, on the menu bar, choose Debug, start debugging/Start without debugging or by pressing F5. This will open the command prompt and run the project and will display the welcome menu.
- **Running the tests:**
 - To run tests manually in MS Visual Studio:
 - Test -> Windows -> Test Explorer -> All Tests
 - Test -> Run -> All Tests
- **Unit Test Case Results:**

Test Explorer

Search

Run All | Run... | Playlist : All Tests

📁

JokeGenerator (7 tests)

▶

✔ JokeApiClient.Tests (7)

32 sec

▶

✔ JokeApiClient.Tests (7)

32 sec

▶

✔ GetJokesCategoriesTests (1)

1 sec

▶

✔ GetJokesTests (5)

30 sec

▶

✔ GetAnyRandomJokesTest

6 sec

▶

✔ GetBothNamesJokesTest

5 sec

▶

✔ GetJokesWithFirstNameOnlyTest

5 sec

▶

✔ GetJokesWithLastNameOnlyTest

6 sec

▶

✔ GetJustCategoryDefaultNameJokesTest

5 sec

▶

✔ GetNamesTests (1)

984 ms

▶

✔ GetNamesTest

984 ms

Summary

✔

Last Test Run Passed (Total Run Time 0:00:17.4375882)

✔

7 Tests Passed