Documentation for OldPhonePad Simulation

Project Overview

The OldPhonePad simulation project is a C# implementation designed to replicate the behavior of an old mobile phone keypad. The goal is to translate a sequence of key presses into the corresponding text output, following specific rules such as handling spaces for pauses, backspaces (\*), and terminating input (#).

The implementation showcases my approach to writing clean, efficient, and well-documented code. It demonstrates how I plan and implement solutions, ensuring functionality and quality while maintaining clarity and modularity.

Solution Design

Key Considerations

Key Mapping:

Each numeric key (2 to 9) is mapped to a string of corresponding alphabetic characters (A-Z).

The solution uses a Dictionary to efficiently map each key press to its corresponding set of characters, allowing quick lookup and manipulation.

Input Parsing:

The function iterates through each character in the input string, identifying special characters (\* for backspace, # for end of input, and space for pauses).

Based on these inputs, it builds the output string character by character.

Special Characters Handling:

\* Removes the last character from the accumulated result if any characters exist.

A space ( ) indicates a pause, resetting the sequence, so multiple characters from the same button can be typed.

# is used as the termination character, signaling the end of input and stopping further processing.

Method Implementation

<https://github.com/replyravi/C-Coding-Challenge-Software-QA-Manager/blob/main/Method%20Implementation.cs>

Explanation

Dictionary Setup: The KeyMap dictionary is initialized to map each key (from 2 to 9) to its respective string of characters.

Iterative Parsing: The function parses each character in the input string and adjusts the result based on the character’s value:

1. #: Terminates processing, returning the accumulated result.
2. \*: Removes the last character if present.
3. Space: Allows for consecutive characters from the same key by resetting the count.
4. Number Keys (2-9): Adds or cycles through characters based on the number of consecutive presses.

Testing Strategy

To validate the code, I developed a set of test cases covering a variety of scenarios, from simple inputs to more complex cases involving spaces and backspaces. Each test case is designed to check different aspects of the method’s functionality, ensuring that it meets all requirements and handles edge cases.

### Test Cases Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** |
| TC001 | Single key pressed multiple times | 33# | E |
| TC002 | Mixed digit sequence with pauses | 4433555 555666# | HELLO |
| TC003 | Sequence with backspace character | 227\*# | B |
| TC004 | Complex sequence with multiple characters | 8 88777444666\*664# | TOOL |
| TC005 | Edge case: no input | # | "" (empty string)` |
| TC006 | Multiple letters from same key with pause | 666 6# | MN |
| TC007 | Continuous backspace characters | 777\*77\*# | P |

Each test case verifies that the method correctly handles numeric key presses, pauses, and special characters, ensuring robustness and correctness.

### Documentation and Commenting

* **Inline Comments**: Each section of the code is thoroughly commented to explain the logic, making it easy for other developers or reviewers to follow the flow.
* **XML Documentation**: The OldPhonePad method is documented using XML comments to provide an overview of the parameters, return type, and behavior of the function.
* **README**: The project includes a comprehensive README file explaining the setup, usage, test cases, and expected outputs.

### Challenges and Edge Case Handling

During the development of this project, I encountered challenges related to handling consecutive key presses and integrating the backspace functionality. The solution required precise management of state variables like lastKey and pressCount to ensure that characters were correctly added or removed based on user input. Proper testing of these cases ensured the method was robust and performed as expected.