Roles Clean Code

As of my knowledge cutoff in September 2021, Robert C. Martin's book "Clean Code: A Handbook of Agile Software Craftsmanship" outlines a set of 13 rules to follow in order to write clean and maintainable code. These rules are not rigid, but rather guidelines to help developers produce better code. Below are the 13 Clean Code rules:

1. **Keep Functions Small**: Functions should be short and focused on doing one thing well. A function should ideally fit on one screen without scrolling.
2. **Do One Thing**: Functions and classes should have a single responsibility. They should do one thing and do it well.
3. **Use Meaningful Names**: Choose descriptive and meaningful names for variables, functions, and classes. Code should be self-documenting and easy to understand.
4. **Function Arguments (Ideal: 0, Acceptable: 1)**: Functions should ideally have zero arguments. If a function needs to take more than one or two arguments, it may indicate a need for refactoring.
5. **Avoid Side Effects**: Functions should not have hidden side effects that modify the state of the system or other objects. Functions should be predictable and not change external state unexpectedly.
6. **Avoid Global State**: Minimize the use of global variables and mutable shared state. Global state can lead to unexpected interactions and make code harder to reason about.
7. **Don't Repeat Yourself (DRY)**: Avoid duplicating code and extract common functionality into reusable functions or classes.
8. **Structured Programming Over GOTO**: Avoid using the GOTO statement and favor structured control flow using loops and conditionals.
9. **Error Handling**: Use exceptions (or their equivalent in your programming language) for error handling rather than error codes or error-prone control flow.
10. **Unit Tests**: Write unit tests to validate the correctness of your code. Well-tested code is more maintainable and less prone to bugs.
11. **Expressive Code over Comments**: Strive to write code that is expressive and self-explanatory. Rely on clear code rather than adding extensive comments.
12. **Formatting**: Follow a consistent and readable code formatting style. Consistent formatting makes it easier for developers to read and understand the codebase.
13. **Limit Class Size**: Keep classes small and focused on a single responsibility. Large classes with many methods can be difficult to comprehend and maintain.