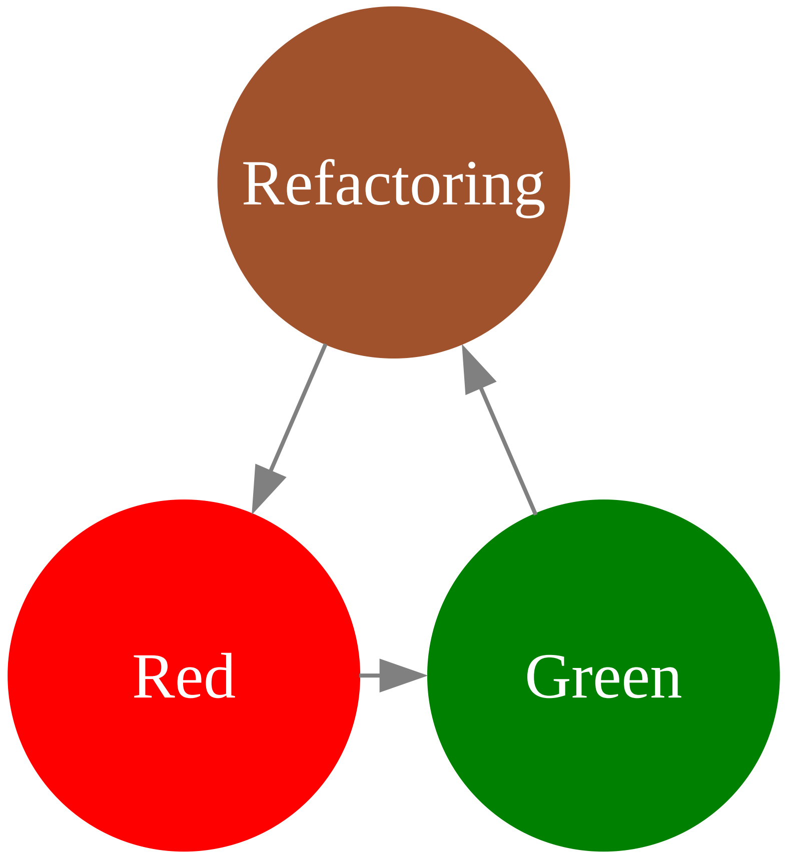
# TDD (Test-Driven Development)

## What is TDD?

- A development style where software is implemented by intorducing test cases first
- States of software development
    - Red: All the tests are not passed (some tests are failed)
    - Green: All the tests are passed
    - Refactoring: The codes should be refactored



## Basic Steps of TDD

- Step 1: Create a test case for a function. The software status is Red because the function is not implemented yet.
- Step 2: The minimum implementation is done to pass the test case, and the software status becomes Green.
- Step 3: Consider the refactoring of codes while the status of software is kept Green.

Repeat Step 1 though Step 3.

## Concept

- Creation of both body and test codes for the system
- Created tests become minimum check programs to ensure the system to run
- The test code is regarded as **the definition of done**.

## Best Practice

- The software under test should be available any time anyone. ➡️ The time for Red is to be as short as possible.
- The test code is also to be tested. ➡️ The status is to be Red once.
- The design of codes is considered while the status is Green. ➡️ The test codes behave as a safety function.

## Benefits of TDD

- Development starts from clear and testable specifications.
- Design becomes more modular and robust.
- Code is implemented with testing in mind.
- Test cases serve as up-to-date documentation.
- Tests are maintainable, reusable, and easy to understand.
- "Definition of done" becomes clearer through test coverage.

## Summary

- TDD encourages reliable, testable, and clean code.
- Development follows a simple cycle: **Red → Green → Refactor**.
- The **Red** state means tests fail (expected after writing a new test).
- The **Green** state means all tests pass.
- **Refactoring** is done while the code is Green to improve structure.
- Test code defines the **"Definition of Done"**.
- Tests themselves should be **maintainable and clear**.