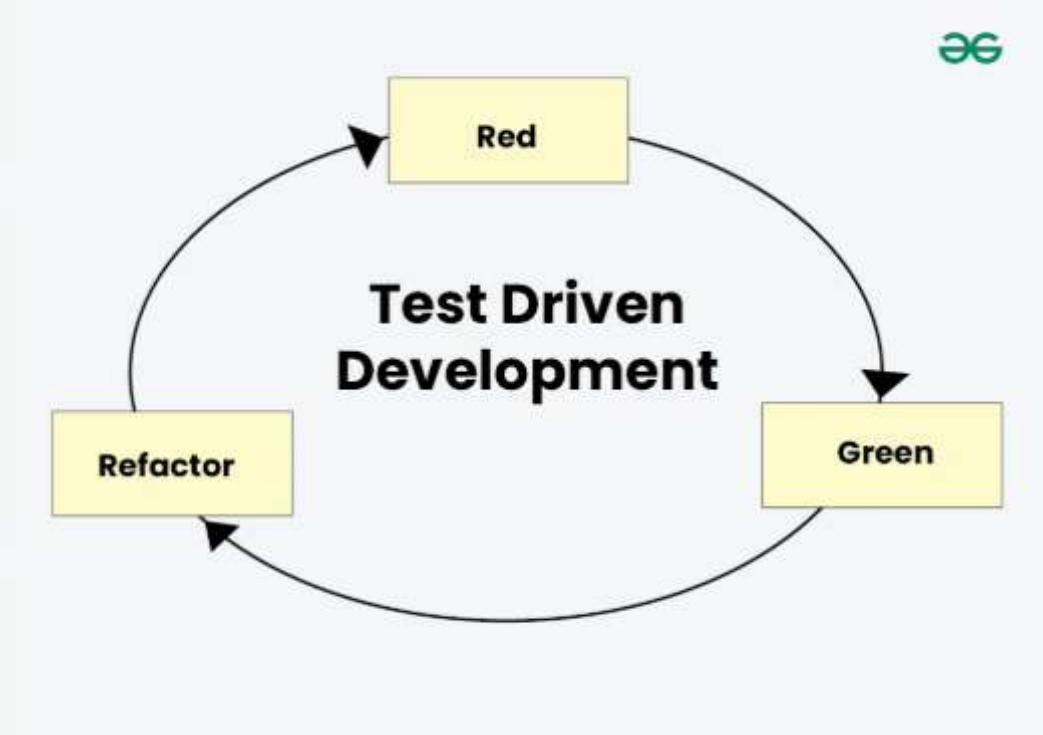
Cập nhật tháng 8 năm 2024

[Bài đọc] Tổng quan TDD, Mô hình và vòng vận hành TDD

1. **TDD là gì?**
   - **TDD (Test-Driven Development)** là **phương pháp phát triển phần mềm hướng kiểm thử**, trong đó việc viết **test được thực hiện trước khi viết code chính thức**
   - Mục tiêu: Đảm bảo chất lượng phần mềm từ đầu, giảm lỗi, giúp code sạch, dễ bảo trì
   - Nguyên lý cốt lõi: **"Test First – Code Later"** (Viết test trước, code sau)

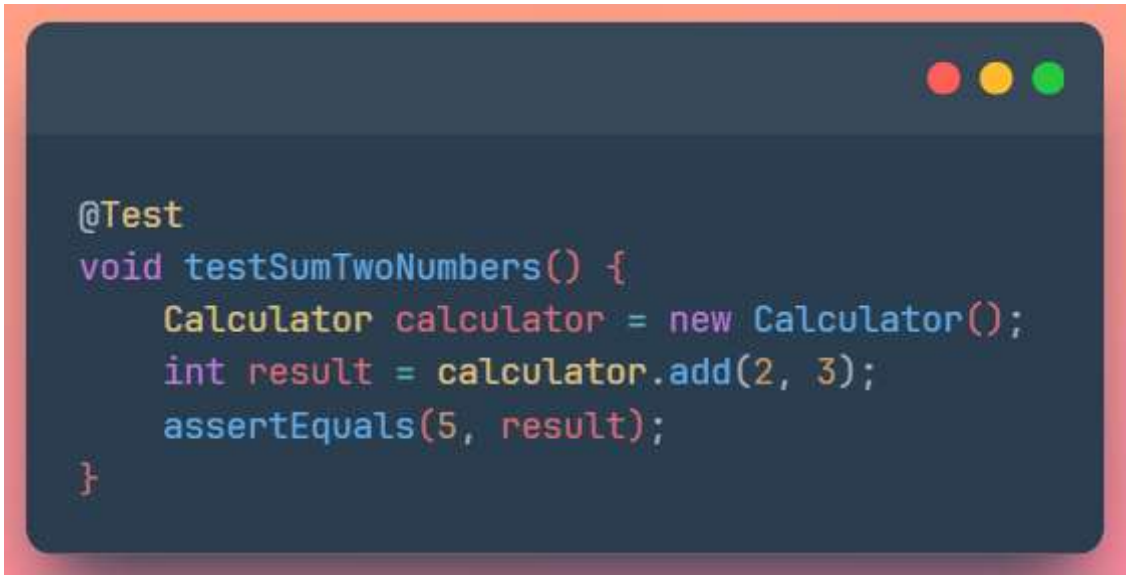
2. **Lợi ích của TDD**
   - Code sạch, rõ ràng, dễ hiểu
   - Giảm lỗi ngay từ đầu
   - Dễ bảo trì, dễ mở rộng
   - Đẩy nhanh quá trình refactoring mà không lo phá vỡ chức năng
   - Hệ thống luôn được kiểm thử tự động liên tục

3. **Mô Hình và Vòng Đời Vận Hành TDD**
   - TDD vận hành theo chu kỳ lặp đi lặp lại với **3 bước chính**, gọi là **Red – Green – Refactor**

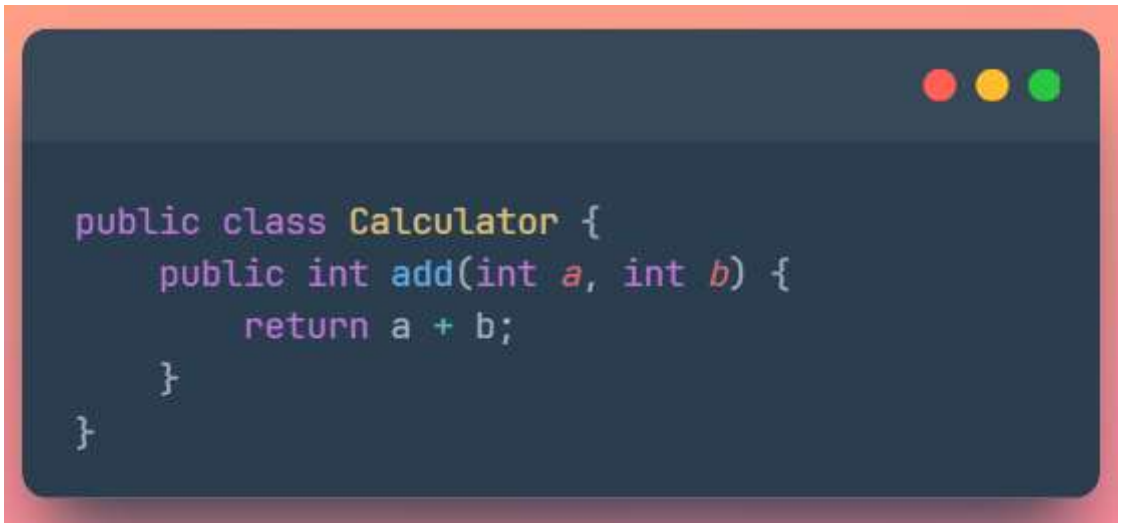


   - **Red - Viết Test thất bại**
     - Viết một test cho chức năng bạn muốn xây dựng
     - Chạy test và chắc chắn nó **thất bại** (do chưa có code hiện thực)
     - Đây là bước đảm bảo test đúng ý nghĩa

- Ví dụ:



- **Green - Viết Code để Pass Test**
  - Viết code đơn giản nhất để test vượt qua
  - Chỉ tập trung làm cho test **pass**, chưa cần tối ưu
  - Ví dụ:



- **Refactor - Cải tiến Code**
  - Sau khi test pass, cải thiện chất lượng code:
    - Loại bỏ lặp lại
    - Tối ưu cấu trúc
    - Đảm bảo vẫn pass toàn bộ test
  - Ví dụ:
    - Nếu trước đó hardcode giá trị, thì giờ refactor lại để code tổng quát hơn
    - Hoặc tách logic phức tạp ra thành các hàm nhỏ hơn

4. **Vòng Lặp TDD Lặp Đi Lặp Lại:**

- Mỗi chức năng mới hoặc thay đổi đều theo quy trình:
  - **Viết test (Red)**
  - **Viết code (Green)**
  - **Refactor (Yellow)**
- Rồi quay lại bước 1 cho chức năng tiếp theo

5. **Quy Trình TDD Cụ Thể Trong Java**

Bước	Hành động
1	Viết test unit cho chức năng
2	Chạy test và thấy thất bại
3	Viết code đơn giản để test pass
4	Kiểm tra test pass
5	Refactor code (nếu cần)
6	Lặp lại cho chức năng tiếp theo

6. **Công Cụ Hỗ Trợ TDD Trong Java**
   - **JUnit:** Framework test unit phổ biến nhất
   - **Mockito:** Mocking để kiểm thử độc lập từng thành phần
   - **IDE hỗ trợ mạnh:** IntelliJ IDEA, Eclipse, VS Code
   - **Maven / Gradle:** Tích hợp test tự động khi build

Tài nguyên đọc thêm: https://www.geeksforgeeks.org/software-engineering/test-driven-development-tdd