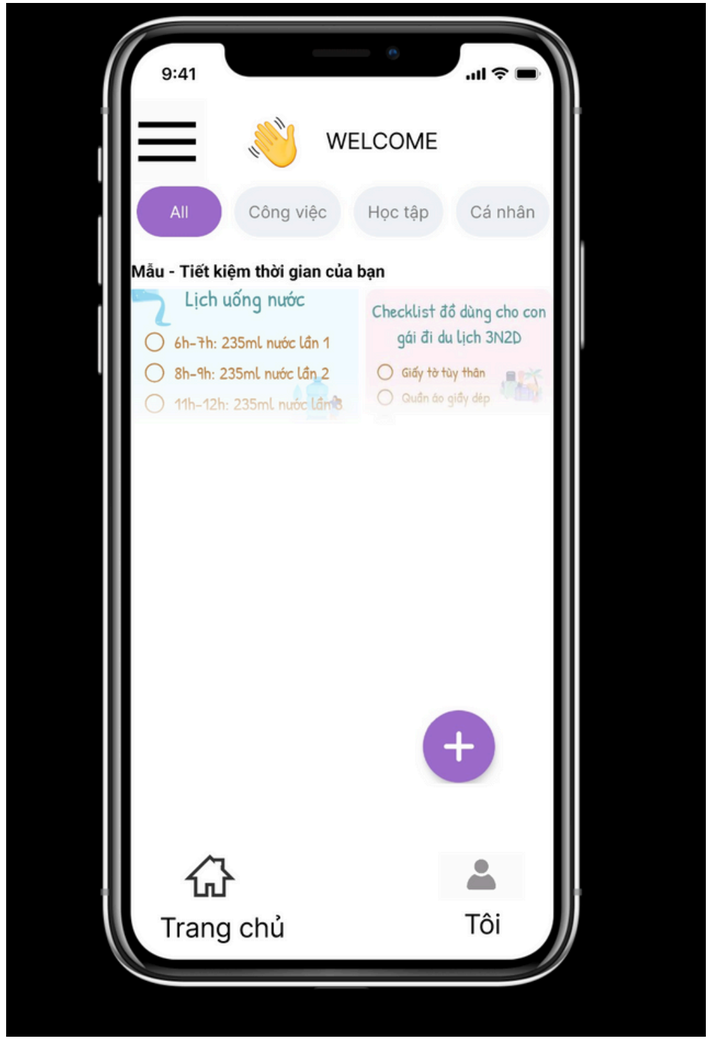
Ứng Dụng Quản Lí Thời Gian

Nhóm 11

Hồ Nguyễn Thế Vinh : 102220046
Phạm Quốc Vũ : 102220049

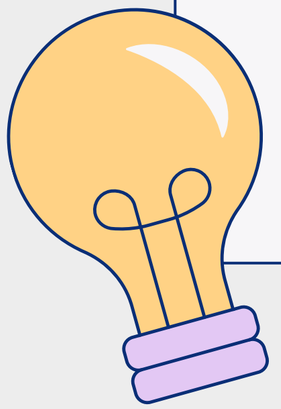


Giới thiệu về sản phẩm

https://www.figma.com/design/Xz5GP9MGuDInoPN0ggTeMS/CNPM-N11?node-id=9-2&t=S30WCnLFbL4IQ7dC-0

I. Giới thiệu Unit Test

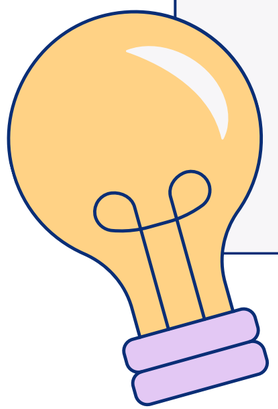
1. Định nghĩa



Unit test là một phương pháp kiểm thử phần mềm tập trung vào việc kiểm tra các đơn vị nhỏ nhất của mã nguồn, thường là các hàm, phương thức, hoặc lớp, để đảm bảo rằng chúng hoạt động đúng cách.

I. Giới thiệu Unit Test

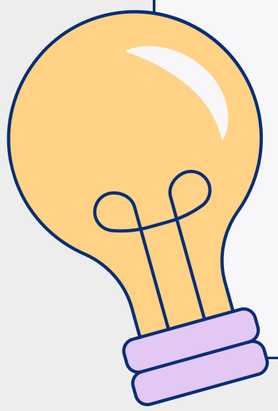
2. Mục tiêu



Mục tiêu của unit test là phát hiện và khắc phục lỗi ở giai đoạn phát triển sớm nhất có thể, giúp tiết kiệm thời gian và công sức về sau.

II. Thành phần chính

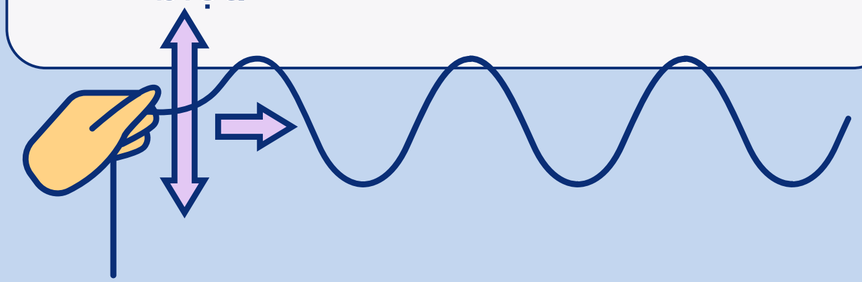
- **Assertion**: Phát biểu mô tả kiểm tra tính đúng đắn của mã.
- **Test Case & Test Point**: Tập hợp kiểm thử cho mỗi tính năng cụ thể.
- **Test Suite**: Nhóm các test case cho mỗi module.
- **Regression Testing**: Kiểm thử tự động ngăn lỗi cũ tái phát



III. Vòng đời Unit Test

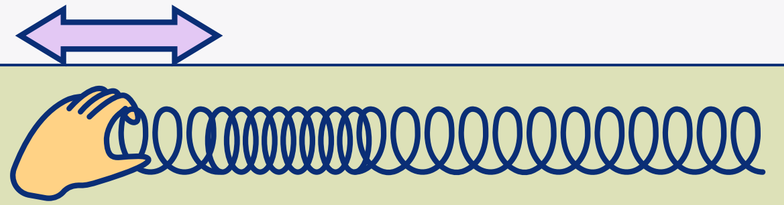
Trạng thái

- -Fail (trạng thái lỗi)
- -Ignore (tạm ngừng thực hiện)
- -Pass (trạng thái làm việc)
- Toàn bộ Unit Test được vận hành trong một hệ thống tách biệt.



Quy trình

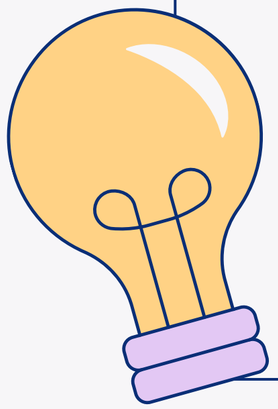
- -Được vận hành lặp lại nhiều lần
- -Tự động hoàn toàn
- -Độc lập với các Unit Test khác.



IV. Thiết kế Unit Test

Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyêncần thiết, xây dựng các dữ liệu giả...Triệu gọi các phương thức cần kiểm tra.Kiểm tra sự hoạt động đúng đắn của các phương thức.Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

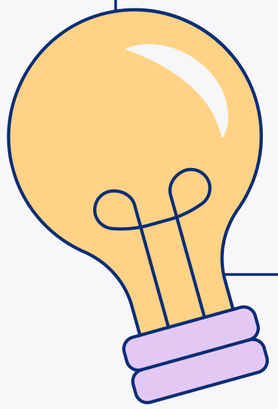
V. Ứng dụng Unit Test



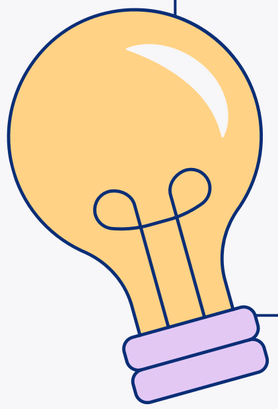
-Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.
-Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.
-Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc(workflow – tập hợp của nhiều quy trình)

VI. Lợi ích của việc áp dụng Unit Test

- -Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn code nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.
- -Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
- -Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
- -Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
- -Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.
  -



VII. Hiệu quả trong coding



- Chuẩn bị cho các tình huống xấu: Xử lý ngoại lệ, lỗi kết nối.
- Phương châm: Khởi đầu bằng Fail, kết thúc bằng Pass.
- Thực thi thường xuyên: Tự động hóa để phát hiện sớm lỗi.

II. Git/GitHub

GitHub là gì?

- Nền tảng lưu trữ và quản lý mã nguồn dựa trên Git.
- **Lịch sử và phát triển:**
- Thành lập vào năm 2008 bởi Tom Preston-Werner, Chris Wanstrath, PJ Hyett, và Scott Chacon.
- **Chủ sở hữu hiện tại:**
- Microsoft mua lại vào năm 2018 với giá 7.5 tỷ USD.

II. Git/GitHub

Tính năng chính của GitHub

- Kho lưu trữ (Repositories):
- Lưu trữ mã nguồn và tài liệu dự án.
- Làm việc trên các tính năng hoặc bản sửa lỗi riêng biệt mà không ảnh hưởng đến mã chính.
- Đề xuất thay đổi và hợp nhất mã.
- Theo dõi lỗi, tính năng mới và các nhiệm vụ khác.
- Hành động GitHub (GitHub Actions):
- Tự động hóa quy trình công việc (CI/CD).

II. Git/GitHub

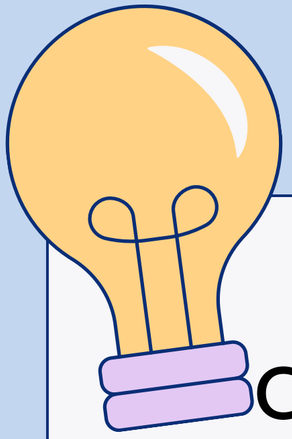
Các Trường Hợp Sử Dụng GitHub

- Dự án nguồn mở (Open source projects)
- Dự án doanh nghiệp và hợp tác nhóm
- Các cộng đồng lập trình viên

II. Git/GitHub

Tại sao lập trình viên nên sử dụng GitHub

- Quản lý mã nguồn hiệu quả
- Hợp tác dễ dàng
- Tích hợp và Tự động hóa
- Bảo mật
- Dễ dàng chia sẻ mã nguồn



Code Style

Code Style là gì?

Code style là tập hợp các quy tắc và hướng dẫn để viết mã nguồn một cách nhất quán và dễ đọc. Việc tuân thủ code style giúp cải thiện tính dễ đọc, bảo trì và hợp tác trong các dự án phần mềm.

Code Style

Tên File

- Không sử dụng dấu cách khi đặt tên file, dùng - hoặc _ để phân tách các từ
- Sử dụng chữ viết thường
- Đặt tên có nghĩa tương ứng

Ví dụ

```
my_document.txt
user_profile.jpg
data_processing.py
product_catalog.json
```

Tên Object

- Sử dụng dâu gạch dưới _ để phân tách các từ
- Đặt tên có ý nghĩa nhưng ngắn gọn
- Không sử dụng tên đã bị lặp lại
- Đặt tên có nghĩa tương ứng

Ví dụ

```
user_profile
product_catalog
customer_order
employee_record
```

White Space

Thụt đầu dòng

- 1 tab = 1 đơn vị thụt đầu dòng.
- Dòng code cùng cấp cách nhau 1 tab.

Ví dụ

```
def function():
    if condition:
        do_something()
```

Dòng trống

- Gom dòng code liên quan thành block.
- Giữa hai block cách nhau 1 dòng trống.
- Khoảng trắng sau dấu phẩy, chấm phẩy và xung quanh toán tử.

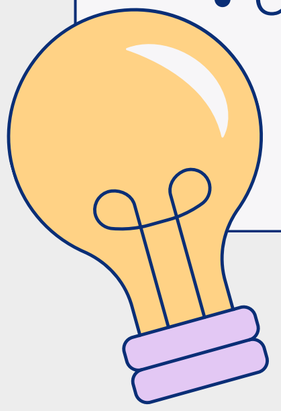
Ví dụ

```
def function_one():
    # Code của function_one
    pass


def function_two():
    # Code của function_two
    pass
```

Comment

- Comment đơn giản, rõ ràng.
- Endline comment canh lề như nhau.
- Viết comment trong quá trình code.
- Chỉ viết comment khi code phức tạp.



Ví dụ

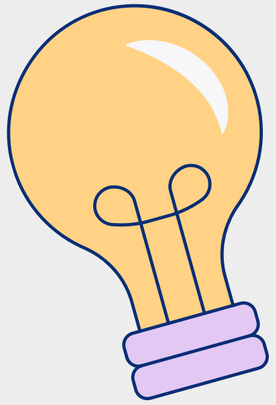
```python
# Đây là bình luận một dòng
def calculate_sum(a, b):
    """Tính tổng hai số.

    Args:
        a: Số thứ nhất.
        b: Số thứ hai.

    Returns:
        Tổng của a và b.
    """
    return a + b
```

Quy tắc viết hoa

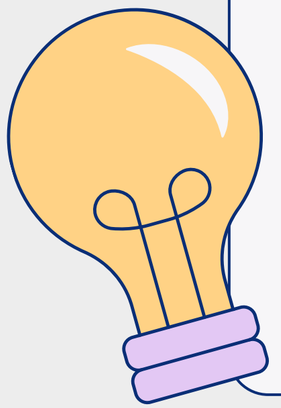
- CamelCase: Bắt đầu bằng chữ thường, sau đó mỗi từ sau đó sẽ bắt đầu bằng chữ in hoa mà không có khoảng trắng hoặc dấu gạch dưới. Ví dụ: myVariable, calculateInterestRate, getUserInfo.



- PascalCase: Tương tự như CamelCase, nhưng bắt đầu bằng chữ in hoa. Thường được sử dụng cho tên lớp, interface, enum, v.v. Ví dụ: MyClass, CalculateInterest, GetUserInfo.

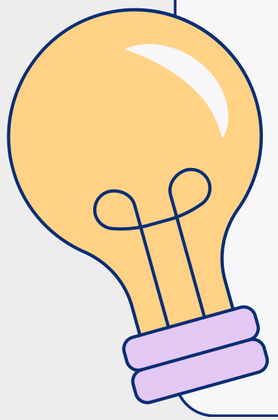
Đặt tên class, interface, abstract class

- Sử dụng danh từ hoặc cụm danh từ: SinhVien, FormSinhVien.
- Dùng Pascal Case: SinhVien, FormSinhVien.
- Hạn chế viết tắt gây khó hiểu:
- Sai: FormSV
- Đúng: FormSinhVien
- Không dùng tiền tố khi đặt tên lớp:
- Sai: ISinhVien
- Đúng: SinhVien



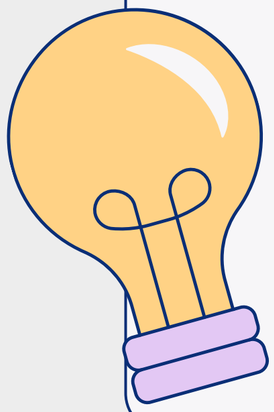
Viết phương thức hiệu quả

- Gom code lặp thành phương thức.
- Tách code phức tạp ra phương thức riêng.
- Khai báo tham số vừa đủ.
- Mỗi phương thức chỉ thực hiện 1 chức năng.
- Phương thức dài 50-150 dòng.



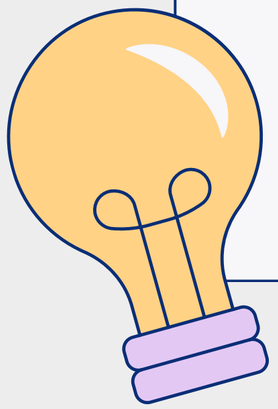
Biến

- Sử dụng Camel Case để đặt tên biến, ví dụ: int diemTrungBinh, String hoTen.
- Không dùng tiền tố, ví dụ:
- Đúng: String address
- Sai: String strAddress
- Tên biến gợi nhớ, tránh viết tắt gây khó hiểu, ví dụ:
- Đúng: String address
- Sai: `String



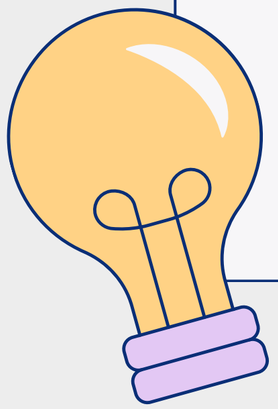
Sử dụng biến

- Tránh khai báo biến không dùng.
- If, while, for không lồng nhau quá 3 bậc.



Import thư viện

- Chỉ import cần thiết: import java.util.List; thay vì import java.util.*;



Xin cảm ơn!