# I. *Giải thích một số thuật ngữ kiểm thử phần mềm – SOFTWARE TESTING TERMS/TERMINOLOGIES*

## 1. Ý NGHĨA CỦA TỪNG TỪ BUG/DEFECT/ERROR/MISTAKE/FAILURE: XEM CÂU HUYỀN THOẠI TRONG CHEAT SHEET

* Tạm hiểu phần mềm được viết ra sẽ hk có những thứ hk như mong đợi

- Chạy/xử lí hk ổn, kết quả hk chính xác như mong đợi
- Việc hiện thực cài đặt code/implement hk theo như thiết kế ban đầu, app hk dùng được trong thực tế.

## 2. ĐỊNH NGHĨA KIỂM THỬ PHẦN MỀM LÀ GÌ?

- ***SỰ SO SÁNH giữa EXPECTED vs ACTUAL***

- Kì vọng, phần mềm thực tế, nhưng thực tế
- Đáng ra phải như thế này nó là thế này

Expected == actual -> okie
Expected != actual -> bug

VD1: Môn Java WEB, phần Login

- Login thành công, màn hình sau login phải chào Hello <fullname>
- Chạy app, login account hoangnt -> thành công, expected lời chào
  Phải là: Hello Hoàng Ngọc Trinh
  Thực tế khi chạy app (code bạn viết) mà nó lại chào Hello hoangnt
  VD2: Test thử phần mềm Calculator
- Test thử tính năng chuyển đổi hệ 10 -> Hex
  Nếu ta đưa vào 15 -> app phải trả về/ expected: F
  250 expected: FA
  Thực tế khi chạy đưa 15 về . F -> đúng
  . ≠ F -> bug

- ***SO SÁNH XEM APP ĐƯỢC IMPLEMENT, CÀI ĐẶT, HIỆN THỰC CÓ GIỐNG NHƯ THIẾT KẾ HAY KHÔNG?***

- Quy trình làm phần mềm thì Requirements, Design (UI, DB, Architecture – MVC, IoC, DAO,...) được gom vào trong 1 tài liệu/document được gọi là SPECIFICATION bản đặc tả phần mềm.
- Ta căn cứ theo bản SPECIFICATION để xem nhóm Developer có cài đặt các màn hình của app đúng như thiết kế hay hk: nút nhấn, vị tri button/ô nhập, màu sắc kích thước, layout, luồng màn hình, phân quyền,...

- ***SO SÁNH XEM APP CÓ ĐẢM BẢO ĐƯỢC CÁC TIÊU CHÍ HAY HIỆU NĂNG VẬN HÀNH HAY HK? ĐẢM BẢO ĐƯỢC CÁC NON-FUNCTIONAL REQUIREMENTS HAY HK?***
    - Kiểm thử trên các NON_FUNCTIONAL REQS, đo xem hiệu năng vận hành của app
    - Tốc độ xử lí đủ nhanh?
    - Khả năng chịu tải/ khả năng xử lí bao nhiêu reqs, bao nhiêu users trong 1 thời điểm bất kì.
    - >>>>>>>>>> PERFORMANCE TESTING!!
    - >>>>giả lập được (như thật) lượng tải (1000 users, 5000 request...) lên app/server
    - >>>>xài tool: JMETER (giả lập n user, gửi n request, nhận về response cho từng request, đo thời gian phản hồi, vẽ biểu đồ...)

[NGOẠI TRUYỆN – HỌC CHI TIẾT HƠN Ở BÊN SWR]
SOFEWARE REQUIREMENTS (có 2 loại lớn, có 2 loại reqs phổ biến)
*FUNCTIONAL REQUIREMENTS – YÊU CẦU CHỨC NĂNG
- VD:
+ App hỗ trợ việc lưu hồ sơ, bệnh án của bệnh nhân (app QL bệnh viện).
+ App cung cấp cho customer tính năng theo dõi trang thái đơn hàng (app BH).
+ App cung cấp tính năng tích điểm cho người mua hàng ở quầy thu ngân (7-E).
- Câu phát biểu về tính năng/tên màn hình/chức năng của app cung cấp cho người dùng làm được 1 việc gì đó thì gọi là FUNCTIONAL REQS.
VERB + OBJECT

* NON-FUNCTIONAL REQUIREMENTS – YÊU CẦU PHI CHỨC NĂNG (yêu cầu mà hk focus vào 1 chức năng cụ thể nào đó)
- Ví dụ:
+ App phải dễ dùng màu sắc hài hòa (mơ hồ).
+ các xử lí của app (nhấn nút bấm) thì phản hồi kết quả trong vòng <=3s.
+ App có khả năng hỗ trợ tối đa 10k người dùng cùng lúc trong giờ cao điểm X với thời gian phản hồi 1 request từ 1...15s.
(VD: . khi săn sale sập sàn voucher 0dd
. cao điểm nhận requet và xử lí: ngày/giờ Bộ giáo dục và đào tạo công bố điểm thi TN THPT 800k thí sinh cùng xem điểm trong vòng 10p đầu tiên)
- Câu phát biểu liên quan đến cảm xúc, trải nghiệm khi xài app, khi xài 1 tính năng bất kì (tui hk focus vào tính năng cụ thể - focus vào trải nghiệm app nói chung - NON)
ADJECTIVE/ADVERB: từ diễn ta tính chất!!
App phải dễ dùng, xài đã, tiện lợi, an toàn, bảo mật, an tâm , đẹp xấu to nhỏ

## 3. AI THAM GIA VÀO QUÁ TRÌNH KIỂM THỬ PHẦN MỀM?

### *3.1. DEVELOPER*

[NGOẠI TRUYỆN]

- Coder - làm theo chỉ đạo thiết kế có sẵn.
- Programmer - giống coder có thể refactor code.
- Developer - xây dựng các interface để hiện thực hóa code/tối ưu code, dùng các Design Pattern (DAO Singleton).
- Software Engineer – xây dựng kiến trúc cho 1 cái app toàn là chơi với interface.

### *3.2. TESTER/QC/QA*

- QA: ...ghi nhận, làm việc với tất cả các phòng ban...
- QC: ...phòng phát triển phần mềm
- .....

### *3.3. TEST MANAGER*

- Sếp của dân QC/Tester, thường đi lên từ QC/Tester
- Nhiệm vụ của Test Manager là:
  - phối hợp với PM (Project Manager) để cùng tham gia vào các công đoạn kiểm thử phần mềm, để lên kế hoạch kiểm thử app
  - phân bổ nhân lực, QC/Tester nào sẽ test dự án/module/chức năng nào
  - đề xuất mua những trang thiết bị/tài nguyên cần thiết hỗ trợ cho việc kiểm thử
  - bàn thảo luận với PM về hiện trạng bug

### *3.4. USER, END-USER*

- Người dùng cuối là chốt chặn cuối cùng của công việc kiểm thử phần mềm. Họ sẽ dùng thử/thật app và cho feedback/phản hồi rằng app xài được cho công việc của họ hoặc app hk giúp ích gì hoặc app nên cải tiến thêm tính năng gì để dùng phù hợp
- Việc người dùng cuối/user dùng app rồi chấp nhận nó đưa nó vào trong công việc hàng ngày – còn gọi là NGHIỆM THU SẢN PHẨM – UAT (USER ACCEPTANCE TESTING)

- **Có 2 loại app (nhìn theo góc độ phổ biến cho user):**
  - ***GENERIC APP*** (app dành cho số đông/app phổ thông, tool tiện ích)

    VD: Game, VLC, IDM, CocCoc, Firefox, Word, Excel, Acrobat Reader, Photoshop,...

    + Có phiên bản download trên mạng hoặc dùng thử online

    + Ai/bất kì ai muốn trải nghiệm thử/dùng thử/để sau này có thể dùng thật thì download về, cài đặt, đăng ký dùng thử

    + USER LÀ BÁ TÁNH/TOÀN THIÊN HẠ

+ Hãng phần mềm thường có nhiều cách để nhờ thiên hạ test giùm app, test qua việc dùng thử trải nghiệm sớm để chửi, feedback, send bug report về chính hãng.
+ Đưa ra nhiều phiên bản khác nhau tùy từng giai đoạn làm app để collect bug/feedback
Alpha, Beta, RC (Release Candidate), Preview, Stable/LTS
Nighty Build, Development Build.. -> phiên bản thử nghiệm của dân dev công bố sớm cho thiên hạ

- ***CUSTOM APP/CUSTOMIZED APP/BE-SPOKE APP*** (app được viết riêng cho nhu cầu của ai đó, app đặc chế cho ai đó)
  VD: App quản lý ngân hàng TPBank, app quản lý tồn kho, bán hàng của TGDD,
  + Không có download, viết app xong bàn giao ngay cho chính chủ đặt hàng
  + Dev team nghĩ rằng đã đến lúc cài đặt cho khách hàng dùng thử
  + User chính là nhân viên của các công ty đặt hàng làm app sẽ xài thử trong công việc của họ
  + Cô thu ngân, dùng thử tính năng tính tiền..
  + Cô giao dịch viên, dùng thử tính năng quản lý sổ tiết kiệm tiền gửi...
  + Cho feedback...

---

## *II. 7 VIÊN NGỌC RỒNG – 7 NGUYÊN LÍ CỦA KIỂM THỬ PHẦN MỀM – 7 PRINCIPLES OF SOFTWARE TESTING*

- **NL1: TESTING SHOWS THE PRESENCE OF DEFECTS**
  - Kiểm thử phần mềm là tìm bug/tìm sai sót của app của quá trình làm app
  - Kiểm thử phần mềm hk có nghĩa vụ chứng minh/gáy/tuyên bố rằng app HẾT BUG
  - Hệ quả (hiểu rộng thêm):
    - Bug luôn tồn tại trong app dù test kỹ cỡ nào!!! Có app là có bug/cong bug
    - Nhiệm vụ của kiểm thử là tìm mọi cách/cố gắng lôi ra càng nhiều bug càng tốt, càng nhiều bug nghiêm trọng càng tốt!!!
  - Lập trình viên phải có trách nhiệm viết code tử tế
  - QC phải test với trách nhiệm cao nhất để hk xảy ra bug nghiêm trọng vì ta phải giữ uy tín với khách hàng/với user

- **NL2: EXHAUSTIVE (vắt kiệt) TESTING IS NOT POSSIBLE**
  - Dân QC hk thể test hết các khả năng/các tình huống xài app của user

- Dân QC hk thể mô phỏng/giả lập/dự đoán hết các hành vi sử dụng app của user để ngăn chặn những bug có thể gặp trong tương lai khi xài app do bug này đã được dân QC report trước rồi khi đóng vai khách hàng xài app.

- Ví dụ:
  - Xét tính năng login của 1 app bất kì – login dùng username/pass riêng của app có những tình huống nào user có thể tương tác với login???
    Dân QC, dân BA, dân Designer, dân viết code sẽ tính toán các phương án xài login như sau:
    - TH1 – Case 1: hk nhập username và pass và lại nhấn [login]
      Expected: chửi hk được để trống

      Case 2: nhập sai username, hk care pass, và nhấn [login]

      Expected: chửi account hk tồn tại, suggest SIGN-UP, REGISTER

      Case 3: nhập đúng username, hk nhập pass, và nhấn [login]

      Expected: pass hk để trống

      Case 4: nhập đúng username, nhập sai pass, và nhấn [login]

      Expected: sai pass, suggest: RESET

      Case 5: nhập đúng username, nhập sai pass, và nhấn [login], làm cái này 5 lần

      Expected: khóa account!!!

      Case 6: nhập đúng username, nhập đúng pass, và nhấn [login]

      Expected: vào dashboard đúng role

      Case 7: nhập đúng username, nhập đúng pass, và nhấn [login] và hk muốn làm lại điều này

      Expected: remember

- Ví dụ 2: TEST APP CALCULATOR CỦA WINDOWS
  TEST CHỨC NĂNG + - * /
  + cần phải test:
  - 2 số nhỏ coi có đúng hk?
  - 2 số 1 số nhỏ 1 số lớn coi có đúng hk?
  - 2 số mà có khả năng tràn biên (int -> max ~ 2 tỷ 2, long: 18 số 0)
  - Số dương + số âm
  - 2 số âm
  - +-*/ phối hợp trong biểu thức

DO HK TEST HẾT CÁC KHẢ NĂNG XÀI APP DO ĐÓ VỀ LÍ THUYẾT VẪN CÓ THỂ TIỀM ẨN BUG DO RƠI VÀO TÌNH HUỐNG TA CHƯA TEST KỊP, MÀ ĐỂ TEST HẾT THÌ TÍNH BẰNG TRIỆU NĂM

→ PHẢI CÓ KỸ THUẬT NÀO ĐÓ, CÁCH THỨC NÀO ĐÓ KO TEST HẾT CÁC TÌNH HUỐNG XÀI APP CỦA USER MÀ VẪN KẾT LUẬN/VẪN CÓ THỂ RELEASE SẢN PHẨM

MỤC 5. TESTING TECHNIQUES CỦA CÂY BẢN ĐỒ TƯ DUY CHỈ CÁC CÁCH DỪNG TEST

- Quy nạp – Phân vùng tương đương
- ...

- **NL3: EARLY TESTING – KIỂM THỬ CÀNG SỚM CÀNG TỐT, THẬM CHÍ NGAY CẢ KHI CHƯA VIẾT CODE THÌ DÂN QC CŨNG ĐÃ PHẢI THAM GIA VÀO QUÁ TRÌNH KIỂM THỬ BẰNG CÁCH REVIEW CÁC TÀI LIỆU REQUIMENT, DESIGN**
  - Dân QC sẽ giao tiếp luôn với dân BA để hiểu hệ thống từ sớm, có thể đề xuất...
    SWR: 56% và 82%
  - Thêm 1 góc nhìn của người ngoài sẽ giúp việc phân tích requirements, design thêm chính xác

- **NL4: DEFECTS CLUSTERING – SỰ PHÂN BỐ/SỰ TẬP TRUNG CỦA BUG**
  - Bàn về sự phân bố/sự tập trung của 1 thứ gì đó, người ta hay nhắc đến nguyên lí 80/20 – nguyên lí Pareto – bàn về sự tập trung, tỉ trọng của những thứ quan trọng – 20% thời gian bỏ ra mà đạt được 80% kết quả -> tốt về mặt quản lí quan trọng: 20% thời gian bỏ ra mà đạt được 80% kết quả => tốt về mặt quản lí
  - Nhìn trong kiểm thử phần mềm cũng có nguyên lí này:
    + Bug thường tập trung hay xuất hiện nhiều ở 1 số chỗ, xuất hiện ít ở chỗ khác
    + App mà có sử dụng các thiết bị ngoại vi: camera, sensor, máy đọc thẻ/barcode
    + App mà có kết nối với app khác, ví dụ Lazada kết nối với MoMo và các ví thì thường những chỗ kết nối này hay có bug, hay chạy bị trục trặc
    - Do tính ổn định, tương thích, khả năng hoạt động chính xác của thiết bị
    - Timeout do đường truyền và độ trễ xử lí của app bên ngoài

  Những tính năng của nội tại app, xử lí trong app, trong database của app ví dụ như CRUD user, CRUD product, dễ làm, ít bug!!!

  Phân bổ nhân lực và thời gian cho những chỗ có nhiều BUG: kết nối bên ngoài!!!

- **NL5: PESTICOD PARADOX – NGHỊCH LÍ THUỐC TRỪ SÂU!! HIỆN TƯỢNG KHÁNG THUỐC**
    - Phun thuốc trừ sâu để tiêu diệt, nhưng sâu hk chết!!!
    - Nhờ dân QC test app, nhưng lại để sót bug, bug nghiêm trọng, bug ngớ ngẩn vẫn tồn tại, QC hk đóng vai trò gác cổng chất lượng
      Khi nào xảy ra hiện tượng sót bug:
        - Qc chủ quan, do làm, đi làm lại, test đi test lại 1 cái app, 1 chức năng quen -> chủ quan, ví dụ khi tuần trước đã test phần CRUD Product rồi, ổn, tuần này test chức năng Order, do dân dev có sửa tùm lum code, nhưng chủ quan nghĩ rằng phần Product đã ổn, hk cần test lại, chỉ tập trung test Order, chủ quan này có thể phải trả giá phần Product
          ➔ Giải pháp: Dân QC nên được bố trí, hoán đổi công việc/project/module kiểm thử trong 1 khoảng thời gian nào đó
            - Test Mobile App 6 tháng, 6 tháng sau chuyển qua test Web App vẫn cùng app nhưng khác môi trường, khác môi trường thì tạo sự tò mò khám phá... tránh được nhàm chán lặp lại công việc

- **NL6: CONTEXT DEPENDENT – KIỂM THỬ PHỤ THUỘC NGỮ CẢNH**
    - Loại app khác nhau (web, mobile, desktop)
    - Loại môi trường/platform/OS khác nhau
    - Thiết bị khác nhau

    ➔ Thì phải có cách test khác nhau!!!

  Khi test Mobile App -> app nằm ngang/đứng layout bị ảnh hưởng thế nào
        - Desktop Ap chạy trên PC -> chỉ test màn hình nằm ngang
        - Android và iOS khác nhau về chính sách bảo mật, quyền truy xuất tài nguyên máy
        - Chạy đa nền: file thực thi phải chạy được trên các nền tảng OS thực , OS ảo
        - Chạy đa trình duyệt: Firefox, CocCoc, Opera.. (w3school có khuyến cáo tag nào với trình duyệt nào!!)
        - Camera: môi trường ánh sáng tốt, mạnh, yếu
        - Máy barcode: máy có cơ chế sửa lỗi cái dòng barcode!!!

- **NL7: ABSENCE OF ERRORS FALLACY - ẢO TƯỞNG/QUAN NIỆM SAI LẦM VỀ VIỆC APP ÍT BUG**
    - Không gáy về việc app ít bug. App ít bug hk phải là đáng tự hào!!!
    - Đừng ảo tưởng rằng app ít bug là app ngon!!!
    - Việc viết app, làm app phải nhắm đến MỤC TIÊU TỐI THƯỢNG:

- APP LÀM HÀI LÒNG KHÁCH HÀNG/USER, KHÁCH HÀNG/USER PHẢI RẤT RẤT THÍCH DÙNG APP
- DÙNG NÓ CHO CÔNG VIỆC HỌC ĐANG LÀM
- APP PHẢI ĐẠT ĐƯỢC LƯỢNG USER ĐÔNG ĐẢO ĐÓ MỚI LÀ MỤC TIÊU LÀM APP
- CÒN APP ÍT BUG THÌ ĐÓ LÀ ĐIỀU MẶC ĐỊNH, HK CẦN PHẢI GÁY, PHẢI BÀN

App được người dùng yêu thích sử dụng cho công việc của họ -> PASSED CÁI GỌI LÀ UAT – USER ACCEPTANCE TESGTING – KIỂM THỬ CHẤP NHẬN

- **CHỐT HẠ CHO 7 VIÊN NGỌC RỒNG:**
  - 7 nguyên lí kiểm thử giúp định hướng các hoạt động của dân QC
  - Biết 7 nguyên lí này thì khi đó dân QC biết được rằng:
    + Kiểm thử là tìm bug thôi, ráng tìm càng nhiều bug càng tốt, kh thể tìm hết bug, nếu ráng tìm hết thì hk đủ thời gian để release sản phẩm
    + ko thể test hết các tình huống sử dụng app của user
    + Ko nên làm mãi 1 công việc/ thao tác kiểm thử mà nên hoán đổi
    + Kiểm thử phu thuộc vào bối cảnh, mobile khác desktop... nên phải thay đổi cách thức kiểm thử dựa theo sự khác nhau của app//bối cảnh sử dụng app
    + Làm app phù hợp nhu cầu sử dụng của người dùng LÀ MỆNH LỆNH TỐI THƯỢNG

**[NGOẠI TRUYỆN]**

Bàn về những con số gắn với kiến thức lập trình

- OOP: 4 + 5
  AEIP SOLID
- AGILE: 4 + 12
  Tuyên ngôn Nguyên lí
- DATABASE: 3 loại SQL:
  - DDL (Data Definition Language): create/drop
  - DML (Data Manipulation Language): select/update/delete
  - DCL (Data Control Language): grant/revoke

  3 loại DẠNG CHUẨN PHỔ BIẾN:
  - 1NF, 2NF, 3NF – Normalization Form – kĩ thuật đánh giá việc thiết kế ERD/Table tốt hay hk? Tốt theo nghĩa là tính trùng lặp dữ liệu khả năng mở rộng thêm bớt data có dễ dàng?

- DESIGN PATTERNS: 23 THIẾT KẾ CHUẨN VỀ CÁC CLASS CẦN CÓ CHO NHỮNG BÀI TOÁN QUEN THUỘC NÀO ĐÓ!!!
- SOFTWARE TESTING: 7
- SWR302: 3 GÓC NHÌN

---

## *III. TESTING LEVELS – 4 MỨC ĐỘ KIỂM THỬ*

- LEVEL: mức, mức độ, 1 thứ gì đó đạt đến 1 giá trị/ngưỡng nào đó
  Mức nước lũ, mức gió bão, mức nước hồ bơi

Quá trình: viết app bắt đầu từ REQUIREMENTS

... giai đoạn khác

IMPLEMENTATION giai đoạn viết code làm app

XÉT RIÊNG GIAI ĐOẠN VIẾT CODE – LÀM APP = IMPLEMENTATION CÁC REQUIREMENTS

THÌ VIỆC HOÀN THIỆN APP, VIẾT CODE SẼ CHIA LÀM 4 MỨC!!!

- ***LEVEL 1 – MỨC 1:*** developer vừa viết xong hàm, hoặc class (chứa hàm/method bên trong)
  - Class UserDTO, class DBUtil

→ MỨC ĐƠN VỊ - UNIT LEVEL

Code vừa viết xong mức đơn vị, tức là hàm/class vừa xong, PHẢI ĐẢM BẢO HÀM CLASS XỬ LÍ NGON, CHÍNH XÁC THÔNG TIN

- hàm() phải được test xem nhận đầu vào xử lí trả kết quả có đúng hk
- class phải được test xem nhận đầu vào qua constructor, setter, trả về kết quả có đúng hk qua getter, toString() và các lệnh return

VIỆC KIỂM THỬ CÁC HÀM, CLASS CHẠY ĐÚNG HAY HK GỌI LÀ KIỂM THỨC ĐƠN VỊ - UNIT TESTING LEVEL

- ***LEVEL 2 – MỨC 2:*** developer bắt đầu xây dựng các class phức tạp – phối hợp các class khác nhau + frond-end để hình thành nên các chức năng đơn lẻ
  - Class UserDAO
    - Xài UserDTO: map với cột của tabe User
    - Xài DBUtil để lấy connect tới DB
    - Method ()
    - checkLogin(username/pass)
    - updateProfile() update thông tin user xuống DB

→ MỨC TÍCH HỢP – INTERGRATION LEVEL

- ***LEVEL 3 – SYSTEM LEVEL***

- ***LEVEL 4 – ACCEPTANCE LEVEL***

---

## *IV. UNIT TESTING LEVEL – LÀ CÁCH MÀ DÂN DEV SẼ TEST CODE CỦA MÌNH!!!*

---

TEST HÀM VÀ CLASS

LAM SAO/KĨ THUẬT NÀO ĐỂ TEST CODE/TEST HÀM/TEST CLASS CỦA MÌNH?

CÓ NHỮNG KĨ THUẬT SAU ĐỂ LÀM UNIT TEST

1. **<u>IN KẾT QUẢ XỬ LÍ CỦA HÀM/METHOD RA MÀN HÌNH</u>**
   - System.out.println(gọi hàm ở đây()); //Java
   - Cosole.writeln(gọi hàm ở đây()); //C#
   - ..tương tự cho ngôn ngữu khác
     - BÌNH LUẬN CÁCH KIỂM THỬ CODE=IN RA MÀN HÌNH
   - Ưu điểm: dễ làm, chỉ việc gọi hàm

2. **<u>IN KẾT QUẢ XỬ LÍ CỦA HÀM RA LOG FILE, .TXT FILE</u>**

3. **<u>POP-UP LÊN DESKTOP, TÌNH DUYỆT</u>**

4. **<u>NGẦU NHẤTl DÙNG 1 BỘ THƯ VIỆN TRỢ GIÚP QUÁ TRÌNH KIỂM THỬ HÀM/CLASS MÀ HK CẦN IN RA KẾT QUẢ XỬ LÍ CỦA HÀM, CHỈ DÙNG 2 TÍN HIỆU XANH-ĐỎ DÙNG 1 BỘ THƯ VIỆN PHỤ TRỢ - CÒN GỌI LÀ UNIT TEST FRAMEWORK</u>**
   - Unit Test, Unit Test framework liên quan đến 1 vài KHÁI NIỆM QUAN TRỌNG SAU:
     - CI (là 1 phần đầu của tiến trình CI/CD/DevOps) – Continuos Integration – Tích hợp liên tục
     - TDD – Test Driven Developement
     - DDT – Data Driven Testing
   -

---

- 10/02/2023

- UNIT TEST NÓI CHUNG, JUNIT NÓI RIÊNG DÙNG MÀU SẮC XANH ĐỎ ĐỂ QUY ƯỚC CHO HÀM ĐÚNG HAY SAI

---

**[NGOẠI TRUYỆN]**

- Câu hỏi phỏng vấn: em hãy cho tui biết, phân biệt cho tui 2 khái niệm THƯ VIỆN – LIBRARY vs. FRAMEWORK

---

- 10/02/2023

- NHẬP MÔN CI – CONTINUOS INTEGRATION NÓ LÀ PHẦN KHỞI ĐẦU CỦA QUY TRÌNH CI/CD/DEVOPS
  - Chuẩn bị sẵn GitJub account, url nên sửa toàn là chữ thường

    QUY TRÌNH ĐÓNG GÓI APP! -> .jar

---

14/02/2023

Biến trong lập trình?

- Tên gọi cho 1 value: int a=10;
- Toán học cho x=100 -> delta=b^2-4ac
- Windows, OS, khác:
  - Do OS cung cấp sẵn, đặt tên sẵn ta xài, thay đổi value:

  - Do app tự đặt ra

Biến môi trường – Environment Variable:

- Là tên gọi cho những giá trị mà biến này được xài cho các app khác nhau
- Luu info, các app xài

Clean and Build: ktra xem cú pháp có lỗi hay hk

* Linus Torvalds: LINUX & GIT

* SCM: source control management system
* VCS: version control system

---

17/02/2023

NHẬP MÔN GIT/GITHUB

THỰC HÀNH CÂU LỆNH GIT ĐỂ ĐỒNG BỘ CODE TỪ LOCAL LÊN SERVER TRÊN MẠNG

1. NHỮNG CÂU LỆNH NÀY CHỈ LÀM 1 LẦN DUY NHẤT CHO 1 MÁY
   NHỮNG CÂU LỆNH NÀY ĐỂ THIẾT LẬP/CẤU HÌNH THÔNG SỐ GH Ở MÁY LẬP TRÌNH VIÊN
   - Sẽ gõ lại nếu cài lại Windows
   - Sẽ gõ lại nếu đổi user/password của GitHub
   - Sẽ gõ lại nếu mượn máy của bạn để xài GH

   • Đứng ở đâu gõ 2 lệnh này cũng được

```
git config –global user.name <nick-github-của-bạn>
git config –global user.email <email-github-của-bạn>
```

   • password sẽ bị hỏi sau khi ta gõ lệnh đồng bộ code. Máy tính sẽ remember account để sau này ko cần gõ lại cho đến khi đổi info!

2. NHỮNG LỆNH SẼ LÀM 1 LẦN DUY NHẤT CHO MỖI PROJECT MỚI TINH
   - Khi có project mới được tạo lần đầu tiên đồng bộ thì gõ lệnh này
   - Lệnh này dành cho chủ dự án, người tạo dự án
   - Anh em team member được mời vào project để code chung thì hk gõ lệnh này
   - Nếu anh em lại có dự án riêng, thì anh em phải gõ lệnh này
   - Một lần duy nhất cho 1 dự án mới. Cũ ko cần làm

   ❖ BẮT BUỘC PHẢI ĐỨNG Ở THƯ MỤC PROJECT, NẾU KO ĐỨNG SAI PROJECT. UPLOAD ĐỒNG BỘ SAI ĐỒ!

   - Khi lệnh này gõ xong, GH client tự động tạo ra 1 thư mục ẩn trong thư mục project tên là .git; CẤM TUYỆT ĐỐI XÓA SỬA THƯ MỤC NÀY VÌ NÓ DÙNG ĐỂ THEO DÕI VÀ LƯU TRỮ LỊCH SỬ THAY ĐỔI CODE CỦA TOÀN BỘ TEAM

```
git init        //khởi động thư mục chứa code ở local thành KHO CHƯA CODE LOCAL REPO
                // theo dõi hết hành động sửa code của nhóm, lưu vào trong thư mục
                //mới được tạo ra
```

3. NHÓM CÂU LỆNH LÀM 1 LẦN DUY NHẤT CHO LẦN ĐỒNG BỘ ĐẦU TIÊN
   - Chỉ dùng cụm lệnh này nếu đây là lần đàu tiên upload code lên server
   - Những lần sau là sửa code, 3 lệnh huyền thoại nhóm 4
   - PHẢI ĐỨNG Ở ĐÚNG PROJECT, MƠI MÀ TA GÕ LỆNH GIT INIT

```
git add .  //scan qua toàn bộ thư mục project đẻ biết tập tin nào đã có thay đổi ở local và đưa vào danh sách chuẩn bị đồng bộ
           //chấp nhận cả thằng mà tên có chữ đầu tiên là dấu .
           //.gitinore em đi lên server. Ta cần cái này vì lát hồi team member cùng down cả .gitinore về máy họ!
```

```
git add * ///scan qua toàn bộ thư mục project đẻ biết tập tin nào đã có thay đổi
          ở local và đưa vào danh sách chuẩn bị đồng bộ
          //KHÔNG chấp nhận cả thằng mà tên có chữ đầu tiên là dấu .
                    //.gitinore em ở lại local

git add <tập tin nào đó muốn đưa lên server>
```

❖ CHUYỂN HƯỚNG PHIÊN BẢN CODE CHÍNH CHUẨN BỊ ĐI LÊN

```
git branch -M main

//KHAI BÁO ĐƯỜNG DẪN TỚI KHO Ở XA, SERVER, REMOTE REPO
git remote add origin https://github.com/

//ĐƯA CODE LÊN, CẦN WIFI
//SẼ BỊ POPUP 1 MÀN HÌNH HỎI PASSWORD, CHỌN OPTION TRÌNH DUYỆT, NHẬP PASS
//BÁO AUTHEN THÀNH CÔNG!!

//WIN TỰ ĐỘNG REMEMBER PASS
```

4. NHÓM NHỮNG CÂU LỆNH HUYỀN THOẠI, LÀM MỖI NGÀY MỖI NGFAY CỨ SỬA CODE, LƯU CODE, CẤT CODE, SUBMIT CODE KO QUAN TÂM SỐ LẦN

```
git add *
git commit -m “lí-do-nội-dung-sửa-đổi-code-ghi-tóm-tắt-here”
git push
```

21/02/2023

ML: Martap Language – Từ ngữ để mô tả về ... nào đó
<h1>: mô tả về data
<h2>: tag, ML dùng ngôn ngữ ghi chú cho 1 điều khác
<b><a><img>...: HTML

MD: Mark down (README.md)

ML khác:

- Tập hợp các lệnh cần làm được viết theo cú pháp

Pwd: print working directering

24/02/20223

| BUILD TOOL – công cụ đóng gói app | | |
|---|---|---|
| ANT | MAVEN | GRADLE |
| Project java | Project java | |
| Folder\<br>Math-ultil\<br>Src\<br>Conf\ | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

POM (Project Object Model): cách maven quản lý mọi thứ trong project trên HDD/SSD theo style object

---

03/03/2023

- SELENIUM (1 bộ thư viện 1 dependency, rất nhiều .jar)
  - Test Automatic (tự động hóa 1 web app)
  - Hệ quả 1: Viết tool cày view
  - Hệ quả 2: CÀO DATA CRAWLER

## *HOMEWORK: LÀM UNIT TEST (CHO PHÉP CHỌN THƯ VIỆN VÀ NGÔN NGỮ)*

- unit test framework for javascript

CV:

Skills:

- SoftSkill
- Technical Skills:
  - HTML/CSS: 24 months hoặc Beginner/Immediate
  - TDD, DDT: basic

- Search trong ChatGPT, Google keyword: “Unit Test framework for JavaScript”
- Sự khác nhau của IDE (Intercreater Develope Environment) & Editor (VS code, Atom, Subline) – tool chỉ để trình bày code
- Maven, Ant, GRAILDL: Build Tool