**Mô tả JUnit**

Link video: https://www.youtube.com/watch?v=HzU65DxapLQ

**Giới thiệu:**

* JUnit là một framework đơn giản dùng cho việc tạo các unit testing tự động là một chuẩn trên thực tế cho unit testing trong Java. JUnit tránh cho người lập trình phải làm đi làm lại những việc kiểm thử nhàm chán bằng cách tách biệt mã kiểm thử ra khỏi mã chương trình, đồng thời tự động hóa việc tổ chức và thi hành các bộ số liệu kiểm thử.  Được sử dụng để viết và chạy thử nghiệm.

Cung cấp chú giải để xác định các phương pháp thử.

Cung cấp Khẳng định cho kết quả thử nghiệm dự kiến.

Cung cấp thử nghiệm chạy cho chạy thử nghiệm.

* **Các phương thức mô tả:**  
  assertEquals() So sánh 2 giá trị để kiểm tra bằng nhau. Test sẽ được chấp nhận nếu các giá trị bằng nhau  
  assertFalse() Đánh giá biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức sai  
  assertNotNull() So sánh tham chiếu của một đối tượng với null. Test sẽ được chấp nhận nếu tham chiếu đối tượng khác null  
  assertNotSame() So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến các đối tượng khác nhau  
  assertNull() So sánh tham chiếu của một đối tượng với giá trị null. Test sẽ được chấp nhận nếu tham chiếu là null  
  assertSame() So sánh địa chỉ vùng nhớ của 2 tham chiếu đối tượng bằng cách sử dụng toán tử ==. Test sẽ được chấp nhận nếu cả 2 đều tham chiếu đến cùng một đối tượng  
  assertTrue() Đánh giá một biểu thức luận lý. Test sẽ được chấp nhận nếu biểu thức đúng  
  fail() Phương thức này làm cho test hiện hàĐây là một thiết kế không tốt vì mỗi phương thức assertXXX() đang kiểm tra phần không liên quan của chức năng. Nếu phương thức assertEquals thất bại, phần còn lại của test sẽ không được thi hành. Khi xảy ra điều này thì chúng ta sẽ không biết các test khác có đúng chức năng hay không.nh thất bại, phương thức này thường được sử dụng khi xử lý các biệt lệ
* **Ví dụ:**  
  assertEquals(employeeA, employeeB);  
  assertEquals(“Employees should be equal after the clone() operation.”, employeeA, employeeB).

Mỗi unittest chỉ nên kiểm tra phần cụ thể của một chức năng nào đó. Chúng ta không nên kết hợp nhiều test không liên quan với nhau lại vào trong một phương thức testXXX()

* **Ta có một lớp Game như sau:**

public class Game {  
private Map ships = new HashMap();  
  
public Game() throws BadGameException {  
}  
  
public void shutdown() {  
// dummy method  
}  
  
public synchronized Ship createFighter(String fighterId) {  
Ship s = (Ship) this.ships.get(fighterId);  
if (s == null) {  
s = new Ship(fighterId);  
this.ships.put(fighterId, s);  
}  
return s;  
}  
  
public boolean isPlaying() {  
return false;  
}  
}  
  
public class BadGameException extends Exception {  
public BadGameException(String s) {  
super(s);  
}  
}

* **Sau đó ta viết một đoạn test sau đây:**

public void testGame() throws BadGameException{  
Game game = new Game();  
Ship fighter = game.createFighter(“001”);  
assertEquals("Fighter did not have the correct identifier", "001", this.fighter.getId());  
Ship fighter2 = this.game.createFighter("001");  
assertSame("createFighter with same id should return same object", fighter, fighter2);  
assertTrue("A new game should not be started yet", !this.game.isPlaying());  
}

Đây là một thiết kế không tốt vì mỗi phương thức assertXXX() đang kiểm tra phần không liên quan của chức năng. Nếu phương thức assertEquals thất bại, phần còn lại của test sẽ không được thi hành. Khi xảy ra điều này thì chúng ta sẽ không biết các test khác có đúng chức năng hay không.

Và chúng ta có thể tham khảo nhiều hơn tại website: Junit.org