65

Hôm nay là thứ 7 ngày 29/07/2018, 1 ngày sau sinh nhật mình. Mình đang ngồi tại coffee house Bình Thạnh, gần hàng xanh.

Mình nhiều công việc quá, không biết bắt đầu từ đâu và làm cái gì, cái gì cũng dở dở ương ương.

Report chưa làm tới đâu.

In3D chưa làm tới đâu.

Việc công ty thì cũng chưa gọi là pro.

Kinh nghiệm code rừng code chùa đã qua, giờ cần phải định hình lại code đẹp là thế nào.

Chứ code kiểu le\*d như cái đám rừng, đã ngu còn k tiếp thu -> thua

Mr.L\*i thì thích thể hiện, cái team như cờ cờ.

Tại sao dùng JPA?  
JPA cung cấp một mô hình các table/các mối quan hệ giữa các table trong database sang các class/mối quan hệ giữa các object.

Ví dụ: table Users với các column (Id, name, age…) sẽ tương ứng với class Users.java với các field Id, name, age… từ đó mỗi khi truy vấn table hay các column ta sẽ truy vấn trực tiếp trên các class, các field của class mà không cần quan tâm tới việc đang dùng loại database nào, dữ liệu database ra sao

# DAO

Tham khảo cách đặt tên ở đây: <https://docs.spring.io/spring-data/jpa/docs/1.5.0.RC1/reference/html/jpa.repositories.html>

\_ List<Employee> findAll

\_ Employee findById

\_ **int** save(Employee student)

\_ **int** update(Employee student)

# Switch

Mỗi khi gặp switch, nó thường khá phức tạp vì mình phải xét từng case trong đó, vì thế nên tách riêng nó ra thành private method.

**while**(rentals.hasMoreElements()){  
 **double** thisAmount = 0;  
 Rental each = (Rental) rentals.nextElement();  
*//determine amounts for each line* **switch** (each.getMovie().getPriceCode()) {  
 19  
 **case** Movie.REGULAR:  
 thisAmount += 2;  
 **if** (each.getDaysRented() > 2)  
 thisAmount += (each.getDaysRented() - 2) \* 1.5;  
 **break**;  
 **case** Movie.NEW\_RELEASE:  
 thisAmount += each.getDaysRented() \* 3;  
 **break**;  
 **case** Movie.CHILDRENS:  
 thisAmount += 1.5;  
 **if** (each.getDaysRented() > 3)  
 thisAmount += (each.getDaysRented() - 3) \* 1.5;  
 **break**;  
 }

Sẽ đổi lại thành private method tính tiền cho từng case:

**while** (rentals.hasMoreElements()) {  
 **double** thisAmount = 0;  
 Rental each = (Rental) rentals.nextElement();  
 thisAmount = amountFor(each);  
*// add frequent renter points* frequentRenterPoints ++;  
*// add bonus for a two day new release rental* **if** ((each.getMovie().getPriceCode() == Movie.NEW\_RELEASE) &&  
 each.getDaysRented() > 1) frequentRenterPoints ++;  
*//show figures for this rental* 20  
 result += **"\t"** + each.getMovie().getTitle()+ **"\t"** +  
 String.*valueOf*(thisAmount) + **"\n"**;  
 totalAmount += thisAmount;  
 }

**private double** amountFor(Rental each) {  
 **double** thisAmount = 0;  
 **switch** (each.getMovie().getPriceCode()) {  
 **case** Movie.REGULAR:  
 thisAmount += 2;  
 **if** (each.getDaysRented() > 2)  
 thisAmount += (each.getDaysRented() - 2) \* 1.5;  
 **break**;  
 **case** Movie.NEW\_RELEASE:  
 thisAmount += each.getDaysRented() \* 3;  
 **break**;  
 **case** Movie.CHILDRENS:  
 thisAmount += 1.5;  
 **if** (each.getDaysRented() > 3)  
 thisAmount += (each.getDaysRented() - 3) \* 1.5;  
 **break**;  
 }  
 **return** thisAmount;  
}

**Refactoring changes the programs in small steps. If you make a mistake, it is easy to**

**find the bug.**

**Any fool can write code that a computer can understand. Good programmers write**

**code that humans can understand.**

# Replace temp with query

<https://refactoring.guru/replace-temp-with-query>

Purpose của phương pháp này là:

\_ 1 method chỉ nên thực hiện 1 chức năng.

Thông comment: Sau này business có đổi để tính basePrice chẳng hạn thì chỉ cần thay đổi bên trong cái hàm tính basePrice thôi là nó tự apply cho tất cả những thằng đang dùng nó.

Còn đưa trực tiếp vào tính toán như này thì đổi 1 cái là ngồi lục từng chỗ đang sử dụng công thức đó mà thay.