Bài test kiểm tra năng lực

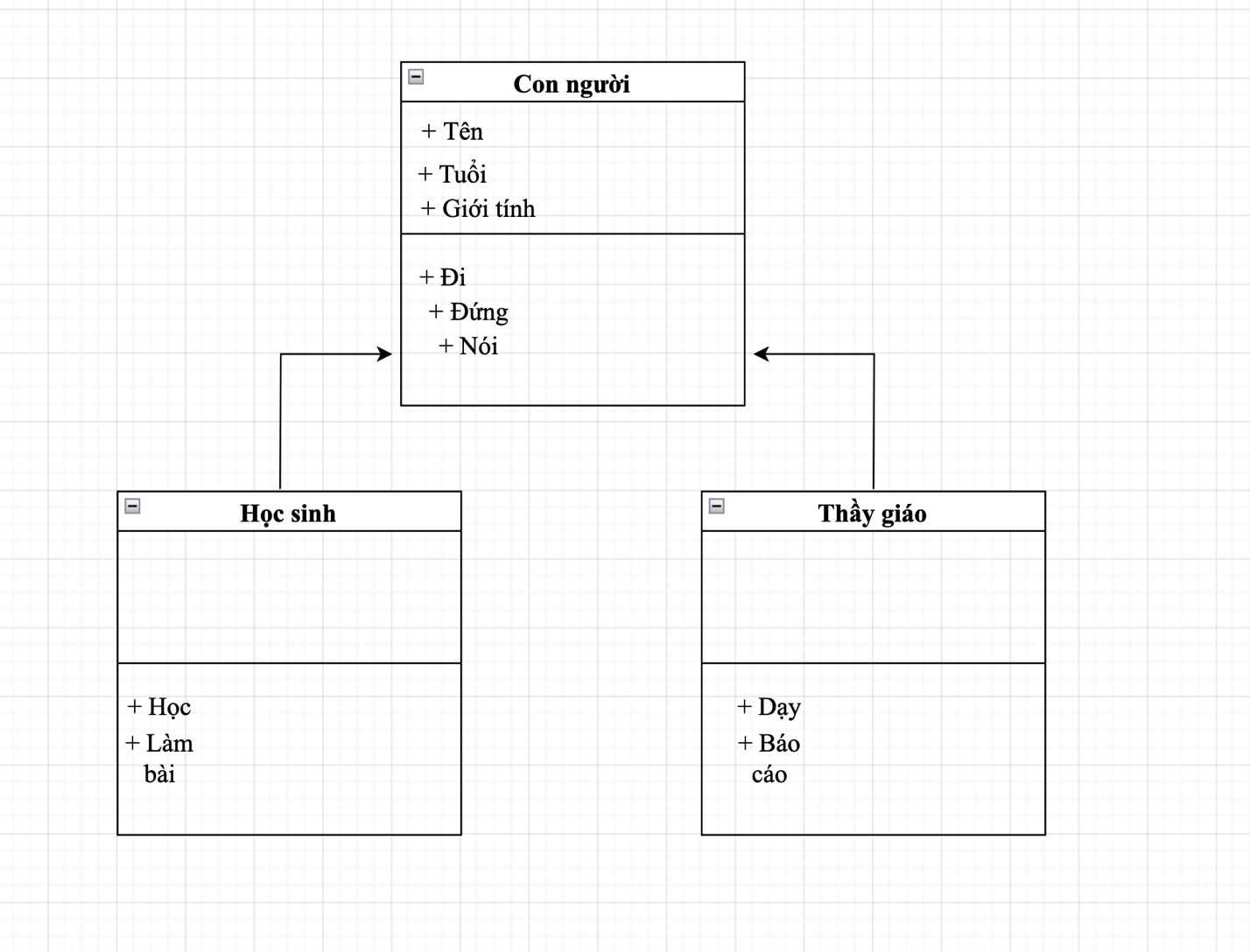
Câu 1 - Lập trình hướng đối tượng (OOP) là một kiểu lập trình mà trong đó, các lập trình viên đóng gói các tính chất, thuộc tính nhằm “trừu tượng hóa” các đối tượng trong code. Trước khi OOP được ra đời, người ta sử dụng phương thức code theo dạng thủ tục, đi từ trên xuống dưới. Tuy nhiên, cách này sẽ khiến file code trở nên rất dài và khó bảo trì. Và OOP sinh ra để khắc phục những nhược điểm mà Lập trình hướng cấu trúc có.    
Trong lập trình hướng đối tượng, các object sẽ có 2 thứ cần quan tâm : Thuộc tính và phương thức. Thuộc tính chỉ các đặc điểm ví dụ như Chó thì có: Lông, mắt, mũi , đuôi. Phương thức chỉ những hành vi mà đối tượng đó có, ví dụ như: Đi, nằm, ngồi, sủa, ăn

Các objects có nhiều đặc tính chung sẽ được gộp vào với Class ( lớp). Ví dụ như xe toyota, honda, mercedes,etc, đều có các điểm chung như 4 bánh, khung, đèn, kính. Ta gom lại vào 1 class là Xe hơi. Và bất cứ khi nào ta nói về 1 trong các xe trên, ta hiểu rằng nó là xe hơi.

4 tính chất của lập trình hướng đối tượng đó là: Tính đóng gói, Tính kế thừa, Tính đa hình và Tính trừu tượng.

1 - Tính đóng gói: Đây là 1 điểm mạnh lớn của OOP khi data có liên quan đến nhau sẽ được đóng gói lại và che giấu thông tin, giúp đảm bảo sự vẹn toàn của dữ liệu, Nếu muốn truy cập vào data của 1 đối tượng, ta phải thông qua hàm getter setter.

2 - Tính kế thừa: Cho phép lập trình viên dựa vào những đặc tính của class đã có để xây dựng class mới với những tính chất mà class trước đó sở hữu. Ví dụ bằng sơ đồ sau:



Dựa theo diagram trên, có thể thấy, đối tượng **Học sinh** và T**hầy giáo** đều kế thừa các thuộc tính chung như tên, tuổi, giới tính, các phương thức như đi, đứng, nói. Và Học sinh có them 2 phương thức riêng là Học và Làm bài, trong khi đó đối tượng Thầy giáo có thêm 2 phương thức đó là dạy và báo cáo.

3 – Tính đa hình: Trong lập trình hướng đối tượng, đa hình chỉ về hai hoặc nhiều class có phương thức giống nhau nhưng cách thực thi có thể khác nhau. Ví dụ: Chó và mèo, chúng đều thuộc class Động vật và các thuộc tính giống nhau như 4 chân, có lông, có đuôi và đều kêu. Tuy nhiên hành vi kêu của mỗi loại, lại thực hiện theo cách khác nhau

4 – Tính trừu tượng: Khi lập trình, người ta trừu tượng hóa các đối tượng mà không cần quan tâm đến các tính chất không cần thiết, để khi người khác chỉ cần nghe là hiểu đối tượng đó là gì, không quan tâm đến quá trình hay đặc điểm sâu của đối tượng. Ví dụ, chúng ta quan tâm đến oto ở các mặt như: Hãng, màu, tính năng, chân ga, chân phanh. Nhưng chúng ta không cần quan tâm quá sâu như cách phối màu, chân ga/chân phanh hoạt động cấu trúc như thế nào.

Câu 2 – Trong swift có các loại type sau

Data types:

+ Int: kiểu số nguyên, độ rộng của kiểu int có thể kéo dài từ 8 đến 64 bít

+ Float: Kiểu số thực độ rộng 32 bits

+ Double: Kiểu số thực độ rộng cao hơn, 64 bits

+ String: Kiểu chuỗi kết hợp nhiều ký tự, ví dụ “Hello World”

+ Boolean: Kiểu dữ liệu logic Đúng/Sai

+ Optional: Kiểu dự liệu optional nếu muốn sử dụng sẽ phải unwrap bằng if-let, guard-

let, ??. Hoặc nếu chắc chắn có dữ liệu truyền vào, sử dụng dấu !

+ var: Dùng để khai báo biến

+ let: Dùng để khai báo hằng số

Naming types:

+ Đối với tên class: Viết hoa chữ cái đầu, ví dụ MyClass

+ Đối với tên biến: tuân theo kiểu đặt tên lạc đà, ví dụ myClass, thisClass

Ngoài ra còn một số kiểu lưu trữ dữ liệu như Enum, Mảng 1 chiều, Mảng 2 chiều, Dictionary.

Câu 3 – Ngoài kiểu lập trình hướng thủ tục và lập trình hướng đối tượng, chúng ta có thêm khái niệm về lập trình hướng giao thức. Protocol Oriented Programming. Trong Swift, lập trình hướng giao thức cho phép 1 class kế thừa nhiều protocol khác nhau. 1 ví dụ dễ hiểu như sau: Có 2 phân loại xe hơi đó chính là: Xe chạy nhiên liệu hóa thạch (GasCar) và xe chạy điện (ElectricCar). 2 kiểu xe này có các đặc trưng riêng biệt như GasCar thì chạy xăng hoặc dầu, tiếp nhiên liệu bằng cách đổ xăng, ElectricCar thì chạy điện và nạp “nhiên liệu” bằng cách sạc. Tuy nhiên, lại có 1 dòng xe lai giữa 2 kiểu này đó chính là HybridCar, có đầy đủ tính chất của xe hơi nói chung, và các đặc tính riêng biệt của 2 dòng trên. Và chúng ta sử dụng protocol để nhặt ra những khía cạnh của các object có sẵn, tạo ra 1 object mới. Ví dụ như sau:

Class Vehicle {

Let numberOfTires = 4

Let drivingTool = “gear”

}

Protocol GasCar {

Let fuel = “gas”

}

Protcol ElectricCar: {

Let fuel = “electric”

}

Class HybridCar: Vehicle {}

Extension HybridCar: GasCar, ElectricCar {}

Câu 4:

Câu 5: Trên một giao diện màn hình, chúng ta sẽ có các elements như UIButton, UILabel, UIView,etc và chúng ta sẽ cố gắng đặt các thành phần trên theo đúng tính năng và vị trí. Tuy nhiên, có nhiều kiểu máy, kích thước màn hình với độ phân giải khác nhau, nên nếu set giá trị dài – rộng cho từng thành phần như vậy, sẽ sinh ra hiện tượng thành phần đó trở nên quá bé ở màn hình rộng hơn, hay quá to ở màn hình bé hơn, và tỷ lệ giãn lề bị thay đổi, ảnh hưởng đến trải nghiệm sử dụng. Để khắc phục điều này, chúng ta sử dụng AutoLayout để tạo ra các ràng buộc (constraint) về kính thước, từ đó máy sẽ tự giãn cách kích thước của các elements theo đúng tỷ lệ. Trong lập trình iOS, để constraint một thành phần, chúng ta cần quan tâm 1 số định nghĩa sau:

+ Căn trên: Top

+ Căn dưới: Bottom

+ Căn trái: Leading

+ Căn phải: Trailing

+ Constant: tỷ lệ

+ Căn giữa: Center

+ Priority: Độ ưu tiên

Khi làm việc trong file .xib hoặc file code, thành phần được khai báo sau, sẽ được constraint dựa theo thành phần chứa nó hoặc liền kề với nó. Hoặc, ta hoàn toàn có thể áp dụng công thức sau trong code: item1.attribute = multiplier x item2.attribute + constant

Câu 6 – Khi thiết kế giao diện, lượng thông tin có thể vượt qua độ dài của một UIView hoặc độ dài của màn hình. Vậy nên chúng ta sử dụng UIScrollView để cuộn và xem được trọn vẹn nội dung của màn đó. Một ScrollView có thể cuộn được khi chúng ta set ContentSize vượt qua độ dài/rộng của UIView/Màn Hình. ScrollView có thể được cuộn chiều dọc, chiều ngang hoặc cả hai.

Câu 7 -