**[Introducing Protocol-Oriented Programming in Swift 3](https://www.raywenderlich.com/148448/introducing-protocol-oriented-programming" \o "Introducing Protocol-Oriented Programming in Swift 3)**

Hãy tưởng tượng bạn đang phát triển game đua xe. Bạn có thể lái xe ô tô, xe máy hoặc máy bay. Thì bạn nên nghĩ đến thiết kế các đối tượng, nhưng bình thường thì chúng ta sẽ tạo một lớp base rồi tất cả các class khác sẽ kế thừa.

Nhưng nó cũng đem lại một số khó khăn. Ví dụ, nếu bạn muốn tạo một cái phương tiện mà yêu cầu chạy bằng gas hoặc bất kể thứ gì mà bạn muốn sử dụng lại game logic, thì khá là khó khan để tách các thành phần của các phương tiện để có thể tái sử dụng.

Với trường hợp này bạn có thể nghĩ đến protocol.

Swift luôn cho phép bạn chỉ rõ ra interface của class, struct và enum bằng cách sử dụng protocols, Swift 2 đã cho phép bạn extend protocols và với Swift 3 cung cấp cho bạn them các toán tử cơ bản và sử dụng chúng để có thể tạo ra các protocols mới giựa trên các thư viện tiêu chuẩn. Trong bài này sẽ tìm hiểu cách tạo và sử dụng protocol để giúp code của bạn dễ mở rộng.

Bắt đầu

Bạn cần tạo một playground, File\New\Playground… và đặt tên là SwiftProtocols. Bạn viết các đoạn mã sau:

|  |
| --- |
| protocol Bird {  var name: String { get }  var canFly: Bool { get }  }    protocol Flyable {  var airspeedVelocity: Double { get }  } |

Nó định nghĩa một protocol có tên là Bird với các thuộc tính là name và canFly cũng như protocol có tên Flyable nó định nghĩa thuộc tính airSpeedVelocity.

Chúng going như khi bạn lập trình như trước thì protocol Flyable như là một base class và sau đó bạn định nghĩa ra những loài nào có thể bay ví dụ như máy bay. Nhưng hãy chú ý là ở đây tất cả thao tác với protocol. Bạn sẽ thấy các đoạn mã bạn viết ra sẽ mềm dẻo hơn khi bạn định nghĩa ra các kiểu ở phần tiếp theo.

Định nghĩa protocol-conforming types

Thêm một struct sau:

|  |
| --- |
| struct FlappyBird: Bird, Flyable {  let name: String  let flappyAmplitude: Double  let flappyFrequency: Double  let canFly = true    var airspeedVelocity: Double {  return 3 \* flappyFrequency \* flappyAmplitude  }  } |

Ở đây tôi định nghĩa một struct mới FlappyBird, nó tuân thủ theo Bird và Flyable protocols. Vận tốc bay của nó được đính toán như trên. Và thuộc tính canFly = true nghĩa là nó có thể bay. Tiếp theo chúng ta sẽ định nghĩa thêm 2 structs:

struct Penguin: Bird {

let name: String

let canFly = false

}

struct SwiftBird: Bird, Flyable {

var name: String { return "Swift \(version)" }

let version: Double

let canFly = true

*// Swift is FASTER every version!*

var airspeedVelocity: Double { return version \* 1000.0 }

}

Dĩ nhiên chim cánh cụt là một loại chim nhưng nó không thể bay. May quá ở đây chúng ta không sử dụng kế thừa, việc sử dụng protocol cho phép chúng ta chỉ rõ ra đối tượng nào phù hợp với protocol đó.