

Pinkoi iOS Interview

參考題目所提供之範例程式，優化、擴充使其達成問題目標。

背景

設計一個架構能支援不同類型的 Data Model 採取不同類型的儲存方式。

- 需考慮擴充性
- 可使用任何 Design Pattern 或 Swift 特性
- 不需考慮儲存方式實作細節 (e.g. CoreData API)
- 盡可能完善單元測試

給定以下 User Model 及 CoreDataService:

```
struct User: Codable {
    let id: String
    let name: String
    let email: String
    let isDesigner: Bool
}

protocol CoreDataTable: Codable {
    func toData() -> Data
}

extension CoreDataTable {
    func toData() -> Data {
        return try! JSONEncoder().encode(self)
    }
}

protocol CoreDataService {
    func store(_ table: CoreDataTable)
    func fetchAll<T: CoreDataTable>(_ table: T.Type, completion: @escaping ([T]) -> Void)
}

final class CoreDataStorer {
    private let coreDateService: CoreDataService

    init(coreDateService: CoreDataService) {
        self.coreDateService = coreDateService
    }
}

extension CoreDataStorer {
    struct UserTable: CoreDataTable {
        let id: String
        let field_name: String
        let field_email: String
        let field_is_designer: Bool
        let created_at: Date
    }
}

extension CoreDataStorer {
    enum CoreDataStorerError: Error {
        case notFound
    }
}
```

Question 1. 請完成 CoreDataStorer 中的實作及補上單元測試：

```
extension CoreDataStorer {
    // TODO:
    // Step 1. Convert User to UserTable
    // Step 2. Save UserTable to CoreDataService (with .store(UserTable))
    func save(user: User) {
        }

        // TODO:
        // Step 1. Fetch user with specifically User.id
        // Step 1-1. if User.id is not exists in coreDateService than completion with
        notFound Failure
        // Step 1-2. if User.id is exists in coreDateService than convert it to User and
        completion with User Success
        func fetch(user: User, completion: @escaping (Result<User, CoreDataStorerError>)
-> Void) {
            }

    }

// Unit Test:
import XCTest

class CoreDataStorerTests: XCTestCase {

}

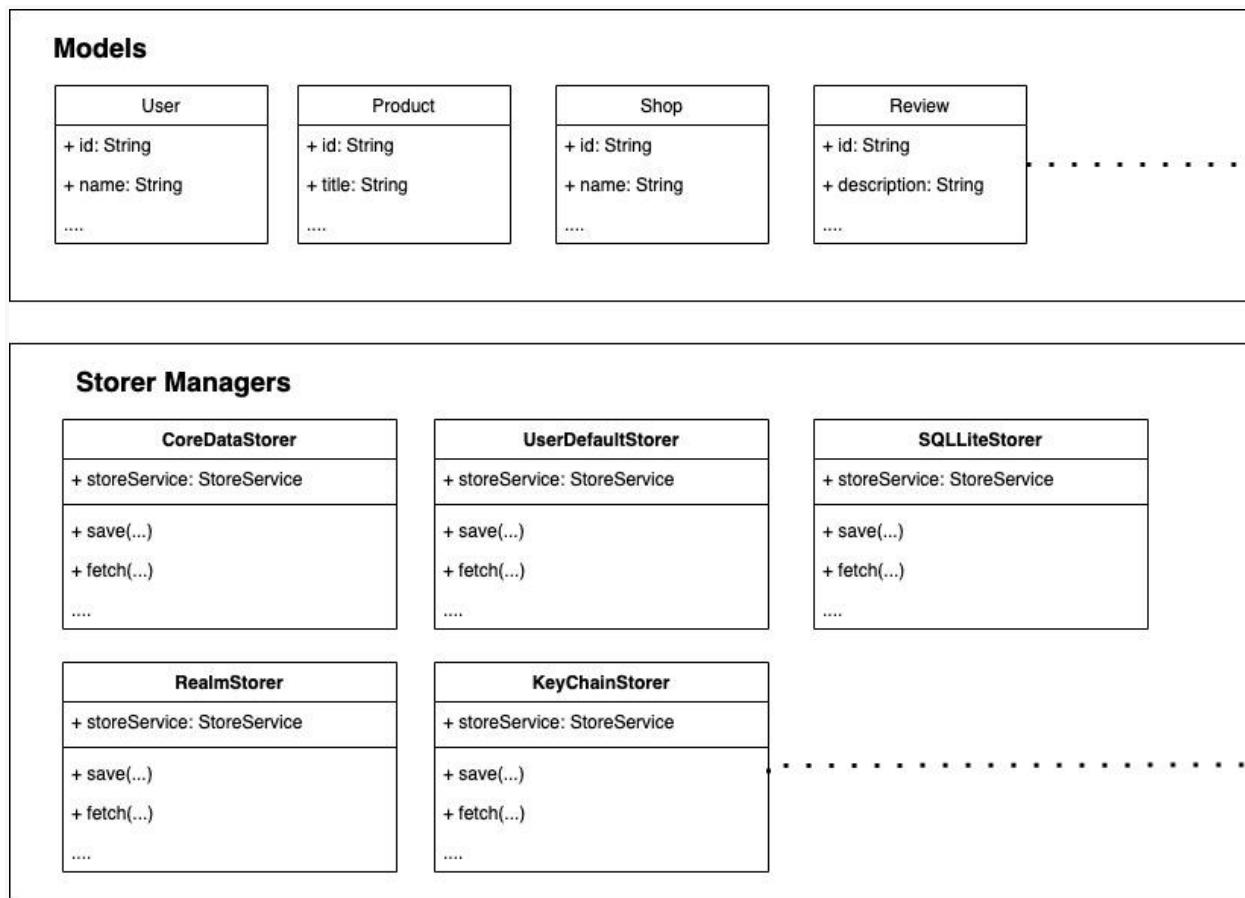
// 如果有其他物件的單元測試，也一並往下寫：
```

Question 2. 承上題，請重構題目給定的 **CoreDataService**, **Models** 程式使其能支援 **Save & Fetch** 不同 **Models**, 至少需多支援以下 **Product Model**。(最好能考量日後 **Models** 疊多的擴充性及閱讀性問題)

```
struct Product: Codable {
    let id: String
    let title: String
    let description: String
    let price: Decimal
    let user: User
}

extension CoreDataStorer {
    struct ProductTable: CoreDataTable {
        let id: String
        let field_title: String
        let field_description: String
        let field_price: Decimal
        let created_at: Date
        let user: UserTable
    }
}
```

Question 3. 假設未來有更多類型的儲存方式且不同的 Model 可能會使用不同方式儲存，您會怎麼做架構設計？



- 至少 50+ 個 Models 和至少 5 種 Storer 儲存方式
- 不同的 Model 可能會使用不同方式儲存：User 使用 KeyChain 存取、但 Product 使用 CoreData
- 需考慮閱讀性、擴充性、可測試性
- 可只畫架構圖表示，無需真的實作