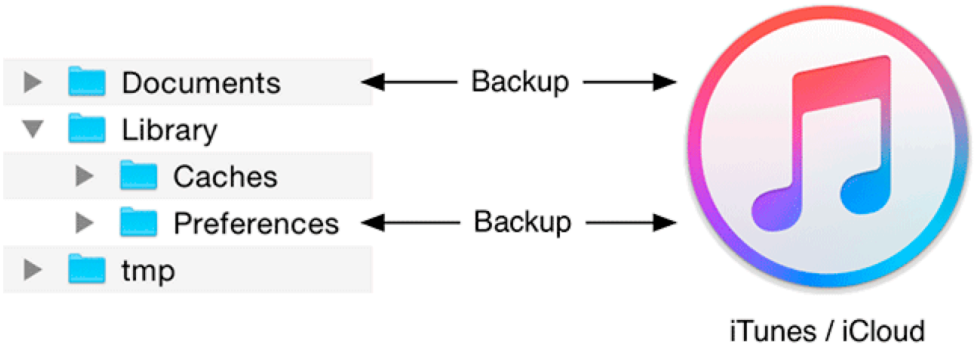
File System

แผนผัง Directory



		ถูกลบอัตโนมัติ	Back up ตอน Sync	User เข้าถึงได้
AppName.app	ไว้อ่าน	No	No	No
Documents	ไว้เขียนข้อมูล	No	Yes	Yes
Documents/Inbox	ไว้รับข้อมูลจาก App อื่น	No	No	Yes
Library/Caches	ไว้เขียนข้อมูลที่จะอยู่ข้ามการปิดเปิด	เมื่อ Disk เต็ม	No	No
Library/Preferences	จัดการอัตโนมัติผ่าน NSUserDefaults	No	Yes	No
tmp	ไว้เขียนข้อมูลชั่วคราว	เมื่อปิด App	No	No

การเข้าถึงไฟล์ทำผ่าน URL โดยสร้าง URL ตั้งต้นจาก

```
FileManager.default.urls(for: .documentDirectory, in: .userDomainMask)
```

แล้วใช้ appendPathComponent เพื่อขยายต่อ

ตัวอย่าง

```
let rootURL = FileManager.default.urls(for: .documentDirectory,
                                       in: .userDomainMask)[0]
let dirURL = rootURL.appendingPathComponent("grade", isDirectory: true)
let fileURL = dirURL.appendingPathComponent("file.txt")
```

การเข้าถึงและจัดการไฟล์จะทำผ่าน FileManager.default เช่น

```
let fm = FileManager.default
fm.fileExists(atPath: dirURL.path)
do {
  try fm.createDirectory(at: dirURL,
              withIntermediateDirectories: true, attributes: nil)
} catch {
    print("Problem with directory")
}
```

createDirectory throws exception จึงต้องมี try catch

Saving App State

ทำ TableView Worksheet ก่อน

เราสามารถ Save Object ได้ถ้า Object รองรับ NSCoding protocol

```
protocol NSCoding {
   func encode( with aCoder: NSCoder)
   init?( coder aDecoder: NSCoder)
}
```

ตัวอย่างการทำ NSCoding

```
class Data:NSObject{
    var height:Float
    var name:String

    func init(){
        self.height = 0
        self.name = "N/A"
    }
}
```

เป็น

```
class Data:NSObject,NSCoding{
    var height:Float
    var name:String

    override init(){
        self.height = 0
        self.name = "N/A"
    }

    //MARK:NSCoding
    enum VKey:String {
        case height = "height"
        case name = "name"
    }
    required init?(coder aDecoder: NSCoder) {
        self.height = aDecoder.decodeFloat(forKey: VKey.height.rawValue)
        self.name =
            aDecoder.decodeObject(forKey: VKey.name.rawValue) as! String
    }

    func encode(with aCoder: NSCoder) {
        aCoder.encode(self.height, forKey: VKey.height.rawValue)
        aCoder.encode(self.name, forKey: VKey.name.rawValue)
    }
}
```

เราใช้ enum:VKey เพื่อลดโอกาสในพิมพ์ผิด (ใช้ชื่ออื่นนอกจาก VKey ได้) และให้ใช้ method ของ NSCoder ที่ตรงกับ Data type. Array ถือเป็น Object ชนิดหนึ่ง

Task 1. เปิด Project จากงาน TableView

Task 2. ให้ Student ให้รองรับ NSCoding Protocol (เฉลย ท้ายเอกสาร)

Task 3. ใน StudentData ให้สร้าง private field fileURL เพื่อเก็บ URL ของไฟล์ที่จะเอาไว้เก็บข้อมูล field นี้จะถูกตั้งค่าตอน init (เฉลย ท้ายเอกสาร) ลอง run ดูเพื่อดูว่า Simulator เก็บไฟล์ไว้ที่ไหน

Task 4. สร้าง function save (ที่เป็น private) ใน StudentData เพื่อ save ข้อมูลเวลามีการเปลี่ยนแปลง การ save ใช้ static method archiveRootObject ของ NSKeyedArchiver (Hint ข้อมูลคือ field ที่ต้องการ save) (เฉลย ท้ายเอกสาร)

Task 5. แก้ไข method ต่างๆของ StudentData ให้เรียก function save เมื่อมีการเปลี่ยนแปลงข้อมูล

Task 6. เพิ่มการ Load ข้อมูลจาก Save ใน init ของ StudentData เพื่อให้ load data เมื่อมีการสร้าง StudentData ใหม่เมื่อ App ถูกปิดแล้วเปิดใหม่ โดยใช้ static method unarchiveObject ของ NSKeyedUnarchiver (เฉลย ท้ายเอกสาร)

อย่าลืม comment ส่วนที่สร้าง Student สุ่มๆใน ViewController ออกก่อนทดสอบ

ทดสอบ ใส่ข้อมูลแล้วปิด App แล้วเปิดใหม่ดูว่าข้อมูลยังอยู่ไหม

เขียนอ่านไฟล์รูป

ทำ Camera Worksheet ก่อน

Task 7. เปิด Project จากงาน Camera

เราสามารถสั่งให้ Object เขียนข้อมูลลงใน File โดยตรงได้ถ้า Object มี method

write(to url: URL, options: Data.WritingOptions = default) throws

Class ที่รองรับเช่น Data, String, etc

การเขียนรูปจะแปลงให้ UIImage เป็น Data โดย UIImagePNGRepresentation หรือ UIImageJPEGRepresentation (Default SDK รองรับแค่ jpeg,png และ pdf)

Task 8. หลังถ่ายรูปแล้วให้เขียนรูปลงใน file image.jpg ใน directory Documents (เฉลย ท้ายเอกสาร)

การโหลดรูปทำได้โดยสร้าง UIImage จาก URL

Task 9. ใน viewDidLoad ให้ load รูปล่าสุดที่ถ่ายไว้มาแสดงถ้ามี (เฉลย ท้ายเอกสาร)

ในงานที่ต้อง Load รูปเข้าออกบ่อยๆ ควรใช้ NSCache เพื่อช่วยจัดการหน่วยความจำเพื่อความเร็ว

Answer

Task 2

ส่วนที่เพิ่มมา

```
class Student:NSObject, NSCoding{
    var firstName:String
    var lastName:String
    var year:Int

    //MARK:NSCoding
    enum VKey:String {
        case firtName = "FirstName"
        case lastName = "LastName"
        case year = "Year"
    }

    required init?(coder aDecoder: NSCoder) {
        self.firstName = aDecoder.decodeObject(forKey: VKey.firtName.rawValue) as!
            String
        self.lastName = aDecoder.decodeObject(forKey: VKey.lastName.rawValue) as!
            String
        self.year = aDecoder.decodeInteger(forKey: VKey.lastName.rawValue)
    }

    func encode(with aCoder: NSCoder) {
        aCoder.encode(self.firstName, forKey: VKey.firtName.rawValue)
        aCoder.encode(self.lastName, forKey: VKey.lastName.rawValue)
        aCoder.encode(self.year, forKey: VKey.year.rawValue)
    }
```

Task 3

```
private let fileURL:URL!

override init(){
    let fm = FileManager.default

    let rootURL = fm.urls(for: .documentDirectory,
                                                in: .userDomainMask)[0]
    let dirURL = rootURL.appendingPathComponent("Data", isDirectory: true)
    self.fileURL = dirURL.appendingPathComponent("datastore.archive")
    print(self.fileURL.absoluteString)
    // Check and create directory
    if(!fm.fileExists(atPath: dirURL.path)){
        do {
            try fm.createDirectory(at: dirURL, withIntermediateDirectories: true, attributes: nil)
        } catch {
            print("Problem with directory")
        }
    }
}
```

Task 4

```
private func save(){
    print(NSKeyedArchiver.archiveRootObject(self.data, toFile: self.fileURL.path))
}
```

Task 6

```
private let fileURL:URL!

override init(){
    let fm = FileManager.default

    let rootURL = fm.urls(for: .documentDirectory,
                                                in: .userDomainMask)[0]
    let dirURL = rootURL.appendingPathComponent("Data", isDirectory: true)
    self.fileURL = dirURL.appendingPathComponent("datastore.archive")
    print(self.fileURL.absoluteString)
    // Check and create directory
    if(!fm.fileExists(atPath: dirURL.path)){
        do {
            try fm.createDirectory(at: dirURL, withIntermediateDirectories: true, attributes: nil)
        } catch {
            print("Problem with directory")
        }
    }
    // Load data up if exists
    if let d = NSKeyedUnarchiver.unarchiveObject(withFile: self.fileURL.path) {
        self.data = d as! [Student]
    }else{
        self.data = []
    }
}
```

Task 8

```
// Turn image into JPEG data
if let data = UIImageJPEGRepresentation( image, 0.5) {
     // Write it to full URL
     let _ = try? data.write( to: self.imageURL!,
         options: [.atomic])
}
```

Task 9

```
let rootURL = FileManager.default.urls(for: .documentDirectory,
                                       in: .userDomainMask)[0]
self.imageURL = rootURL.appendingPathComponent("image.jpg")
print(self.imageURL?.path ?? "")

if let image = UIImage(contentsOfFile: (self.imageURL?.path)!){
    self.imageView.image = image
}else{
    // Does nothing
}
```