iOS行動程式基礎開發上架

swift：錯誤處理

本堂教學重點

1. **處理錯誤**
2. **描述和丟出錯誤**
   - 使用function丟出錯誤
   - 使用do - catch處理錯誤
   - 轉換錯誤成為可nil值
   - 關閉錯誤向上傳遞
3. **指定function最後處理的動作**

1.處理錯誤

```
enum VendingMachineError: Error {
    case invalidSelection
    case insufficientFunds(coinsNeeded: Int)
    case outOfStock
}
```

2.描述和丟出錯誤

使用function丟出錯誤1

```
struct Item {
    var price: Int
    var count: Int
}
```

```
class VendingMachine {
    var inventory = [
        "Candy Bar": Item(price: 12, count: 7),
        "Chips": Item(price: 10, count: 4),
        "Pretzels": Item(price: 7, count: 11)
    ]
    var coinsDeposited = 0

    func vend(itemNamed name: String) throws {
        guard let item = inventory[name] else {
            throw VendingMachineError.invalidSelection
        }

        guard item.count > 0 else {
            throw VendingMachineError.outOfStock
        }

        guard item.price <= coinsDeposited else {
            throw VendingMachineError.insufficientFunds(coinsNeeded: item.price - coinsDeposited)
        }

        coinsDeposited -= item.price

        var newItem = item
        newItem.count -= 1
        inventory[name] = newItem

        print("Dispensing \(name)")
    }
}
```

2.描述和丟出錯誤

使用function丟出錯誤2

```
    let favoriteSnacks = [
        "Alice": "Chips",
        "Bob": "Licorice",
        "Eve": "Pretzels",
    ]
    func buyFavoriteSnack(person: String, vendingMachine: VendingMachine) throws {
        let snackName = favoriteSnacks[person] ?? "Candy Bar"
        try vendingMachine.vend(itemNamed: snackName)
    }


    struct PurchasedSnack {
        let name: String
        init(name: String, vendingMachine: VendingMachine) throws {
            try vendingMachine.vend(itemNamed: name)
            self.name = name
        }
    }
```

2.描述和丟出錯誤

使用do - catch處理錯誤1

```
do {
    try expression
    statements
} catch pattern 1 {
    statements
} catch pattern 2 where condition {
    statements
} catch {
    statements
}
```

2.描述和丟出錯誤

使用do - catch處理錯誤2

```
var vendingMachine = VendingMachine()
vendingMachine.coinsDeposited = 8
do {
    try buyFavoriteSnack(person: "Alice", vendingMachine: vendingMachine)
    print("Success! Yum.")
} catch VendingMachineError.invalidSelection {
    print("Invalid Selection.")
} catch VendingMachineError.outOfStock {
    print("Out of Stock.")
} catch VendingMachineError.insufficientFunds(let coinsNeeded) {
    print("Insufficient funds. Please insert an additional \(coinsNeeded) coins.")
} catch {
    print("Unexpected error: \(error).")
}
// Prints "Insufficient funds. Please insert an additional 2 coins."
```

2.描述和丟出錯誤

使用do - catch處理錯誤3

```
func nourish(with item: String) throws {
    do {
        try vendingMachine.vend(itemNamed: item)
    } catch is VendingMachineError {
        print("Invalid selection, out of stock, or not enough money.")
    }
}

do {
    try nourish(with: "Beet-Flavored Chips")
} catch {
    print("Unexpected non-vending-machine-related error: \(error)")
}
// Prints "Invalid selection, out of stock, or not enough money."

```

2.描述和丟出錯誤

轉換錯誤成為可nil值

```
func someThrowingFunction() throws -> Int {
    // ...
}

let x = try? someThrowingFunction()

let y: Int?
do {
    y = try someThrowingFunction()
} catch {
    y = nil
}


func fetchData() -> Data? {
    if let data = try? fetchDataFromDisk() { return data }
    if let data = try? fetchDataFromServer() { return data }
    return nil
}
```

2.描述和丟出錯誤

關閉錯誤向上傳遞

```
let photo = try! loadImage(atPath: "./Resources/John Appleseed.jpg")
```

3.指定function最後處理的動作

```
func processFile(filename: String) throws {
    if exists(filename) {
        let file = open(filename)
        defer {
            close(file)
        }
        while let line = try file.readline() {
            // Work with the file.
        }
        // close(file) is called here, at the end of the scope.
    }
}
```