do try catch
쓰려고 하는 이유

drake

목차

1. do catch 란?
2. 기존코드의 문제점
3. 지향점
4. 결론

1. do catch 란?

- return 이외의 방법으로 반환값을 던질수 있음
- throw 로 미리 선언해 놓은 에러를 던질수 있음
- 함수 선언부에서 throws 를 사용해야 throw 사용가능
- 그냥 return 값으로 메세지 던지면 되지 뭐하러?
- 객체를 분리해보니 필요할 때가 생김

1. do catch 란?

```
func canThrowErrors() throws -> String
```

```
var vendingMachine = VendingMachine()
vendingMachine.coinsDeposited = 8
do {
    try buyFavoriteSnack("Alice", vendingMachine: vendingMachine)
} 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.")
}
```

- 함수 선언부에 throws 추가
- 내부에서 에러 throw
- try 로 받아서 catch 로 처리

출처)https://blog.asamaru.net/2015/10/05/swift-2-dot-1-try-catch/

2. 기존 코드의 문제점

- 현재 구조 : 자판기 객체 - 메인함수 - OutputView
- 메인함수에서 자판기객체 생성해서 자판기의 함수를 직접 사용
- 메인함수 내부에 자판기의 출력메세지를 사용하는 함수 증가
- 메세지 출력 함수를 분리하기 어려움

2. 기존 코드의 문제점

```
/// 돈 추가를 선택시 진행순서
func insertMoneyStep(){
    if let money = inputView.insertMoney() {
        vendingMachine.plusMoney(money: money)
    }
    else {
        outputView.printMessage(message: OutputView.errorMessage.notNumeric.rawValue)
    }
}
```

```
/// 음료선택시 재고가 남아있는지 체크후 해당 음료의 정보를 가저온다
func getInventoryDetail(drinkNumber:InputView.DrinkNumber)->InventoryDetail?{
    // 메뉴번호-1 이 실제 배열임
    guard let drinkDetail : InventoryDetail = vendingMachine.getAllInventory()[drinkNumber.rawValue-1] else {
        // 음료재고가 0이면 에러
        outputView.printMessage(message: OutputView.errorMessage.notEnoughDrink.rawValue)
        return nil
    }
    return drinkDetail
}
```

- 메인함수 내부의 함수에서 print 기능을 사용. 분리가 힘듬

2. 기존 코드의 문제점

```
/// 음료 선택후 구매 진행과정
func buyingDrink(drinkNumber:InputView.DrinkNumber){
    /// 구매가 가능한지 체크한다
     // 구매하려는 음료가 잔고가 있는지
    guard let drinkDetail = getInventoryDetail(drinkNumber: drinkNumber)
        // 원하는 수량이 >0 인지
        , let orderCount = receiveOrderCount(drinkName: drinkDetail.drinkName)
        // 재고수량 >= 주문수량 인지
        , isEnoughDrinkCount(drinkDetail: drinkDetail, orderCount: orderCount)
        else {
            // 하나라도 잘못되면 단계 취소
            return ()
}
// 돈 계산
if vendingMachine.calculateMoney(drink: drinkDetail,orderCount:orderCount) == false {
    return ()
}
// 입력받은 음료 번호를 음료 타입으로 변환
let drinkType = drinkNumberToType(drinkNumber: drinkNumber)
// 인벤토리->주문내역 으로 음료 이동
let result = vendingMachine.orderDrinks(drinkType:drinkType, drinkCount: orderCount)
// 결과를 출력
outputView.printMessage(message: outputView.buyingResult(drinkDetail: result))
```

guard 선언
각각의 함수내부에서
print 함수 사용중

guard 이후에도
각각의 함수내부에서
print 함수 사용중

- 객체로 분리시킬 경우 print 못씀 + 출력 구조 단일화 필요

3. 지향점

```
func runUserMode(_ vendingMachine: UserAvailable & VendingMachineUserModePrintable) throws {
    let outputView = OutputView(vendingMachine)
    OutputView.printAllStock(vendingMachine.readAllStock())
    while true {
        outputView.printBalance()
        outputView.printBuyableProducts()
        outputView.printMenu()
        let (menu, option) = InputView.selectMenu()
        switch menu {
        case INSERTCOIN:
            vendingMachine.insertMoney(option)
        case BUYBEVERAGE:
            let buyables = vendingMachine.readBuyableProducts()
            OutputView.printSoldBeverage(try vendingMachine.buy(buyables[option - 1]))
        case EXITTOSELECTMODE:
            return
        default:
            continue
        }
    }
}
```

메세지 출력 함수 단일화

단일화된 함수 내부에서 조건문으로 메세지 처리

각각의 함수들은 에러메세지를 throw 만.

-출처:myssun0325/VendingMachine

4. 결론

- 메인함수가 너무 많은 함수를 사용중
- 분리해야 하지만 직접 error print 하는 함수가 많음
- 해당 함수들의 에러메세지를 throw 로 변경
- 에러메세지 출력 함수들을 모아서 하나의 함수로 생성
- 생성된 함수 내부에서 do catch 로 에러출력 단일화
- 함수 객체로 분리