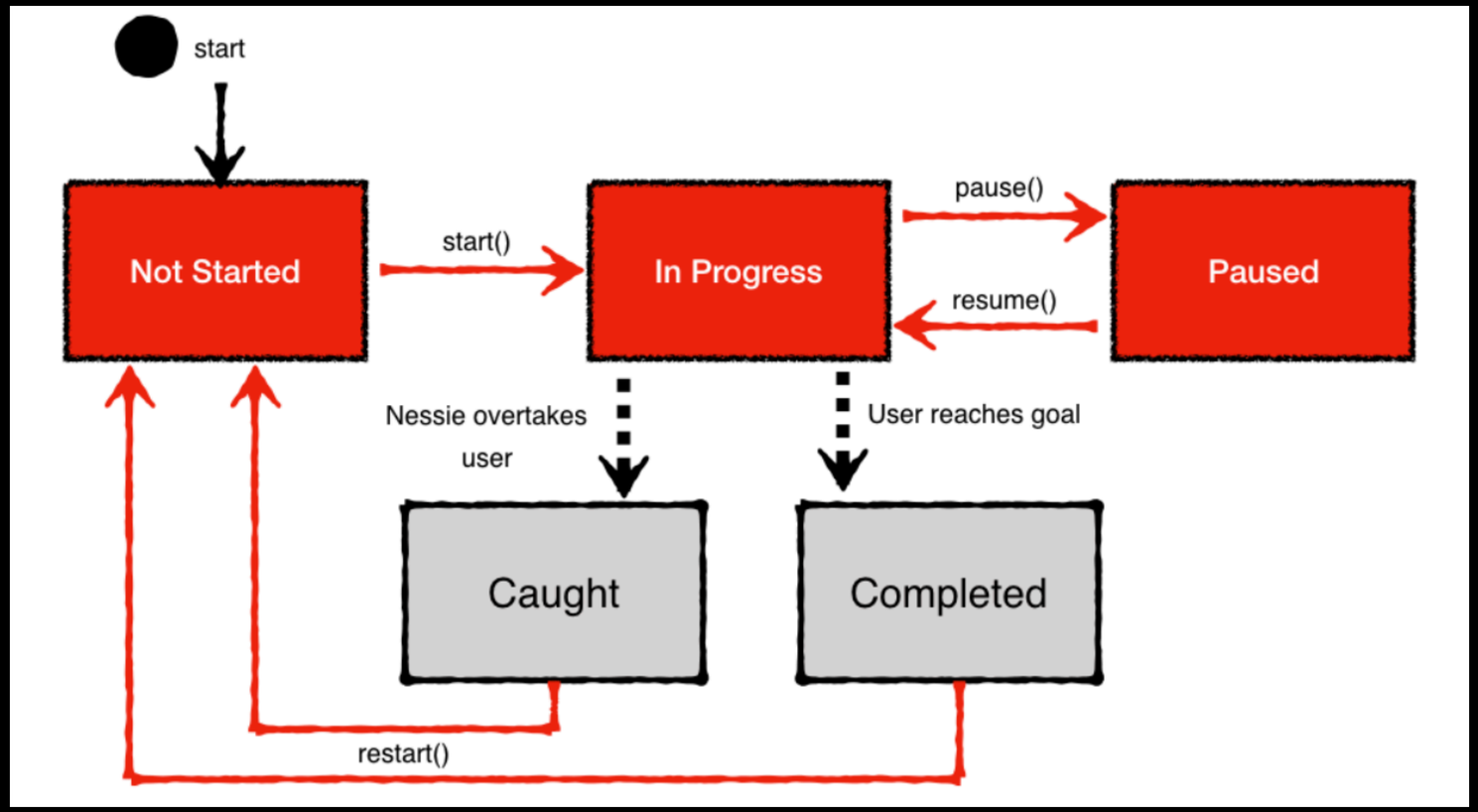
Test-Driven Development

Chapter 5: Expectations

김련호

Expectation

- 직접적인 제어의 외부에서 일어나는 이벤트에 대한 테스트(비동기 상황)



Expectation / Waiter

XCTest expectation

- Expectation - 나중에 fulfill 할 수 있는 객체
- Waiter - expectation이 fulfill 하거나, 일정 시간이 지날때까지 대기

비동기 테스트 작성

- *Closure - callback* 방식에서 이용
- ~~Delegate~~
- *Observing - notification* 방식에서 이용

비동기 테스트 작성 - AppModel

```
179   // MARK: - State Changes
      func testAppModel_whenStateChanges_executesCallback() {
181     // given
182     givenInProgress()
183
184     var observedState = AppState.notStarted
185
186     // expectation 객체 생성
187     let expected = expectation(description: "callback happened")
188
189     // callback 할당
190     sut.stateChangedCallback = { model in
191       // 상태 변경
192       observedState = model.appState
193
194       // expectation fulfill 호출
195       expected.fulfill()
196     }
197     // 상태 변경
198     sut.pause()
199
200     // fulfill 혹은 timeout 대기
201     wait(for: [expected], timeout: 1)
202
203     // 테스트
204     XCTAssertEqual(observedState, .paused)
205   }
206 }
```

진정한(?) 비동기 테스트 - StepCountControllerTests

expectation을 fulfill하는 observer 이용

```
11  class ButtonObserver: NSObject {
12    var expectation: XCTestExpectation?
13    weak var button: UIButton?
14
15    // observer 셋팅
16    func observe(_ button: UIButton, expectation: XCTestExpectation) {
17      self.expectation = expectation
18      self.button = button
19
20      button.addObserver(self, forKeyPath: "titleLabel.text", options: [.new], context: nil)
21    }
22
23    // callback(notification?)
24    override func observeValue(forKeyPath keyPath: String?, of object: Any?, change:
        [NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {
25      expectation?.fulfill()
26    }
27
28    deinit {
29      button?.removeObserver(self, forKeyPath: "titleLabel.text")
30    }
31
32  }
```

진정한(?) 비동기 테스트 - StepCountControllerTests

expectation을 fulfill하는 observer 이용

```
184     func testController_whenCaught_buttonLabelIsTryAgain() {
185       // given
186       givenInProgress()
187
188       let exp = expectation(description: "button title change")
189
190       // observer 생성
191       let observer = ButtonObserver()
192
193       // 버튼 감시
194       observer.observe(sut.startButton, expectation: exp)
195
196       // when, 상태 변경
197       whenCaught()
198
199       // then, 대기
200       waitForExpectations(timeout: 1)
201       let text = sut.startButton.title(for: .normal)
202
203       // 테스트
204       XCTAssertEqual(text, AppState.caught.nextStateButtonLabel)
    }
```

Alert 테스트 - AlertCenterTests

Single notification

- 게임이나 액티비티 앱에서는 alert를 유저에게 전달하는것이 매우 중요
- XCTestExpectation 을 사용하기에 적합

```
47      func testPostOne_generatesANotification() {
48        // given
49        // notification이 post될 때 fulfill되는 expectation 생성
50        let exp = expectation(forNotification: AlertNotification.name, object: sut, handler: nil)
51        let alert = Alert("this is an alert")
52
53        // when
54        sut.postAlert(alert: alert)
55
56        // then
57        wait(for: [exp], timeout: 1)
58      }
    }
```

Alert 테스트 - AlertCenterTests

Multiple notification

```
    func testPostingTwoAlerts_generatesTwoNotifications() {
80    //given
81    let exp = expectation(forNotification: AlertNotification.name, object: sut,
        handler: nil)
82
83    // 복수의 expectation이 있음. wait할 때 2번의 fulfill이 호출되지 않으면 실패
84    exp.expectedFulfillmentCount = 2
85    let alert1 = Alert("this is the first alert")
86    let alert2 = Alert("this is the second alert")
87
88    // when
89    // 삭제하면 테스트에 실패함
90    sut.postAlert(alert: alert1)
91    sut.postAlert(alert: alert2)
92
93    // then
94    wait(for: [exp], timeout: 1)
95  }
```

일어나지 않아야하는 상황에 대한 테스트 - AlertCenterTests

alert이 예상치 않게 두 번 발생하는 상황에 대한 테스트

```
98    func testPostDouble_generatesOnlyOneNotification() {
99      //given
100     let exp = expectation(forNotification: AlertNotification.name, object: sut, handler: nil)
101     exp.expectedFulfillmentCount = 2
102
103     // 성공여부를 반대로 설정, true일 때 테스트가 성공하면 실패로 처리
104     exp.isInverted = true
105
106     let alert = Alert("this is an alert")
107
108     // when
109     sut.postAlert(alert: alert)
110     sut.postAlert(alert: alert)      Fulfilled inverted expectation "Expect notification 'Alert' from FitNess.Aler
111
112     // then
113     wait(for: [exp], timeout: 1)
    }
```

notification의 부가 정보 이용 - AlertCenterTests

notification을 이용하여 Custom 데이터 전달

```
146   func testNotification_whenPosted_containsAlertObject() {
147     // given
148     let alert = Alert("test contents")
149     let exp = expectation(forNotification: AlertNotification.name,
150                           object: sut,
151                           handler: nil)
152     var postedAlert: Alert?
153
154     sut.notificationCenter.addObserver(forName: AlertNotification.name, object: sut, queue:
          nil) { notification in
155
156       // 추가정보
157       let info = notification.userInfo
158       postedAlert = info?[AlertNotification.Keys.alert] as? Alert
159     }
160
161     // when
162     sut.postAlert(alert: alert)
163
164     // then
165     wait(for: [exp], timeout: 1)
166     XCTAssertNotNil(postedAlert, "should have sent
167     XCTAssertEqual(alert,postedAlert, "should have
    }
```

```
48    func postAlert(alert: Alert) {
49      guard !alertQueue.contains(alert) else { return }
50
51      alertQueue.append(alert)
52      let notification = Notification(name: AlertNotification.name,
53                                      object: self,
54                                      userInfo: [AlertNotification.Keys.alert: alert])
55       notificationCenter.post(notification)
56    }
```

Data Model로부터의 alert - DataModelTests

데이터가 특정 목표값이 되면 발생하는 alert 테스트

```
154   func testWhenStepsHit25Percent_milestoneNotificationGenerated() {
155     // given
156     sut.goal = 400
157
158     // handler가 expectation의 fulfill할지 결정
159     let exp = expectation(forNotification: AlertNotification.name, object: nil) { notification -> Bool in
160       return notification.alert == Alert.milestone25Percent
161     }
162
163     // when
164     sut.steps = 100
165
166     // then
167     wait(for: [exp], timeout: 1)
    }
```

각 경우에 따라 각각 테스트(25, 50, 75, complete)

```
35    var steps: Int = 0 {
36      didSet {
37        updateForSteps()
38      }
39    }
40
41    // MARK: - Updates due to distance
42    func updateForSteps() {
43      guard let goal = goal else { return }
44      if Double(steps) >= Double(goal) * 0.25 {
45        AlertCenter.instance.postAlert(alert: Alert.milestone25Percent)
46      }
47    }
```

Testing for multiple expectations - DataModelTests

한번에 모두 테스트(25, 50, 75, 100)

```
    func testWhenGoalReached_allMilestoneNotificationsSent() {
206   // given
207   sut.goal = 400
208   let expectations = [
209     givenExpectationForNotification(alert: .milestone25Percent),
210     givenExpectationForNotification(alert: .milestone50Percent),
211     givenExpectationForNotification(alert: .milestone75Percent),
212     givenExpectationForNotification(alert: .goalComplete)
213   ]
214
215   // when
216   sut.steps = 400
217
218   // then
219   wait(for: expectations, timeout: 1, enforceOrder: true)  Asynchronous wait failed: Exceeded timeout of 1 seconds,
220 }
```

Refining Requirements - DataModelTests

```
    func testWhenStepsIncreased_onlyOneMilestoneNotificationSent() {
230   // given
231   sut.goal = 10
232   let expectations = [
233     givenExpectationForNotification(alert: .milestone25Percent),
234     givenExpectationForNotification(alert: .milestone50Percent),
235     givenExpectationForNotification(alert: .milestone75Percent),
236     givenExpectationForNotification(alert: .goalComplete)
237   ]
238
239   // clear out the alerts to simulate user interaction
240   let alertObserver = AlertCenter.instance.notificationCenter
241     .addObserver(forName: AlertNotification.name,
242                  object: nil, queue: .main) { notification in
243                   if let alert = notification.alert {
244                     AlertCenter.instance.clear(alert: alert)
245                   }
246   }
247
248   // when
249   for step in 1...10 {
250     self.sut.steps = step
251     sleep(1)
252   }
253
254   // then
255   wait(for: expectations, timeout: 20, enforceOrder: true)
256   AlertCenter.instance.notificationCenter.removeObserver(alertObserver)
257 }
```