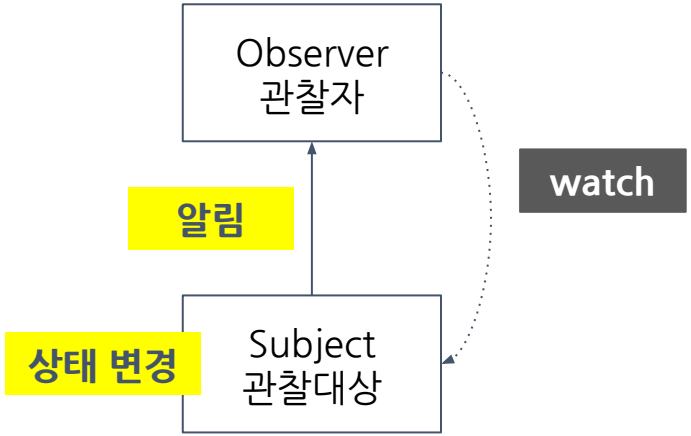
4 w.

Observer Pattern

Ivy

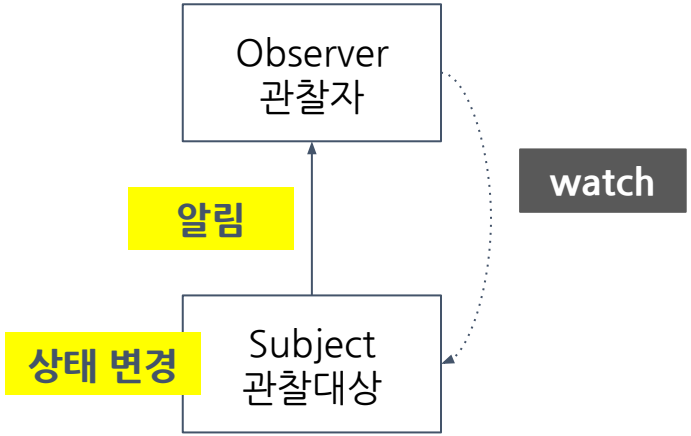
Exercise.

요구사항 1. StateSubject 구현



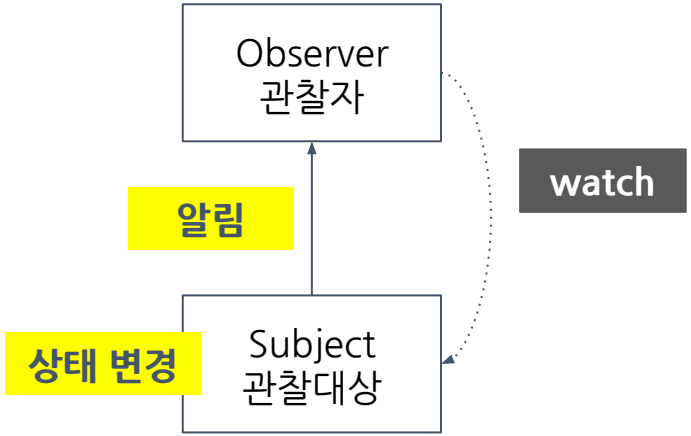

```kotlin
interface StatusObserver {
    fun onAbnormalStatus(status: Status)
}

enum class Status {
    LOADING,
    SUCCESS,
    ERROR
}
```

요구사항 2. StateChecker 구현




```
abstract class StatusSubject {
    class StatusChecker : StatusSubject() {

    }
}
```

요구사항 3. StateEmailSender 구현



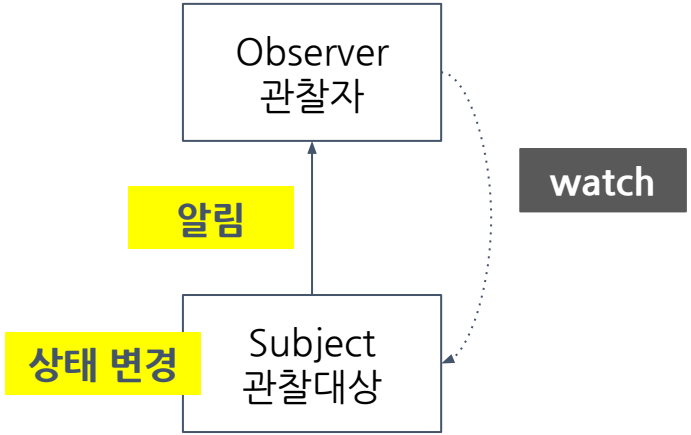

```kotlin
interface StatusObserver {
}

class StatusEmailSender : StatusObserver {
    override fun onAbnormalStatus(status: Status) {

    }
}
```

구현

```
abstract class StatusSubject {
    class StatusChecker : StatusSubject() {

        fun check() {
            val status = loadStatus()
            if (status != Status.SUCCESS) {
                notifyStatus(status)
            }
        }

        private fun loadStatus() = Status.LOADING
    }
}
```

```
fun main() {
    val checker = StatusChecker()
    val emailSenderObserver = StatusEmailSender()
    checker.add(emailSenderObserver)
    checker.check()
}
```

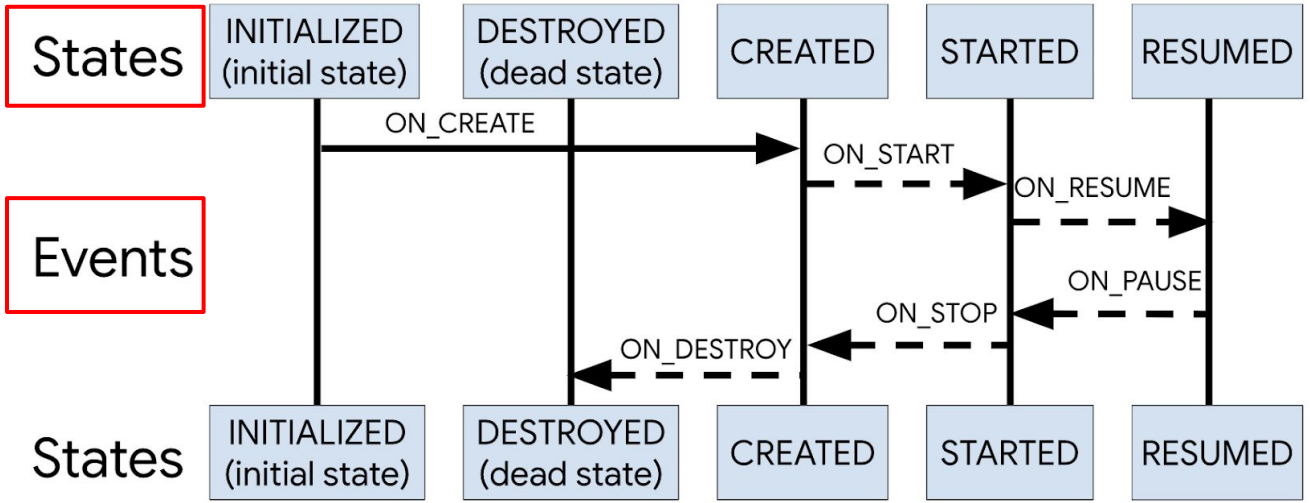
```
interface StatusObserver {
class StatusEmailSender : StatusObserver {
    override fun onAbnormalStatus(status: Status) {
        sendEmail(status)
    }

    private fun sendEmail(status: Status) {
        println("sendEmail status=${status}")
    }
}
```



LiveData

Lifecycle-aware 컴포넌트

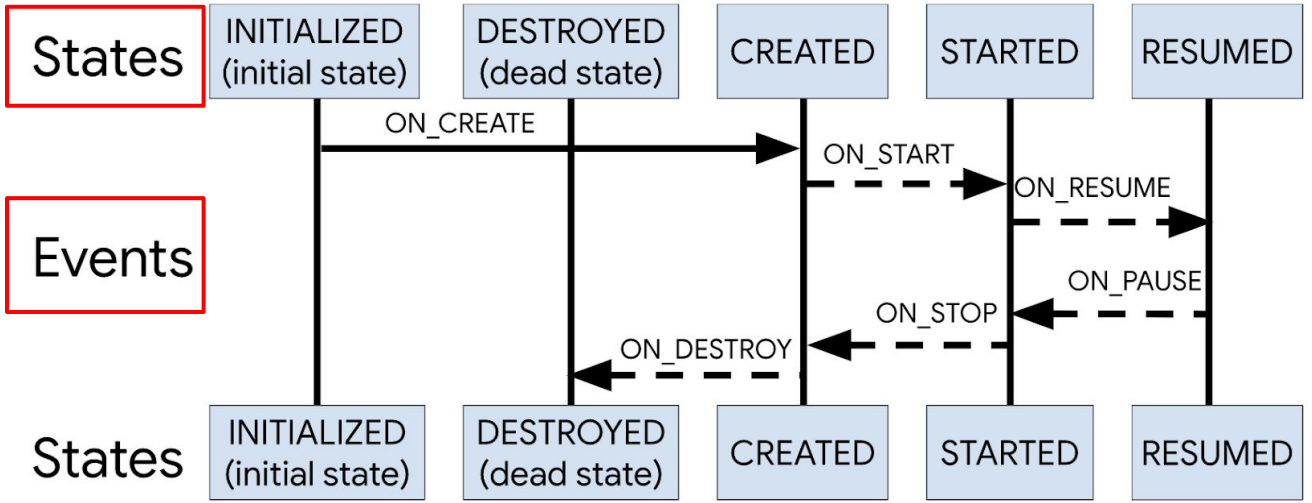
- Activity, Fragment와 같은 컴포넌트의 생명 주기 상태가 변경에 따라, 동작을 조정할 수 있는 클래스와 인터페이스를 제공합니다.



https://developer.android.com/topic/libraries/architecture/lifecycle

Lifecycle-aware 컴포넌트

- States : 그래프의 노드, Events를 edge로 생각하자.
- Events : Lifecycle 클래스로부터 얻는 생명주기 이벤트.
  - 두 개의 노드 사이를 이동하는 사건



LifecycleOwner

- Lifecycle의 소유권을 추상화하는 인터페이스
- AppCompatActivity, Fragment는 이미 LifecycleOwner가 구현되어 있음
- LifecycleOwner의 생명주기를 관찰하려면 LifecycleObserver를 구현해야 함
  - LifecycleOwner : 생명 주기 정보 제공
  - 등록된 LifecycleObserver : 생명 주기 변화 관찰

```java
public interface LifecycleOwner {
    Returns the Lifecycle of the provider.
    Returns: The lifecycle of the provider.
    @NonNull
    Lifecycle getLifecycle();
}
```

```kotlin
class LifecycleDetector(
    private val TAG: String
) : LifecycleObserver {

    @OnLifecycleEvent(Lifecycle.Event.ON_CREATE)
    fun printOnCreate() {
        Log.d(TAG, msg: "onCreate")
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    fun printOnStart() {
        Log.d(TAG, msg: "onStart")
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    fun printOnResume() {
        Log.d(TAG, msg: "onResume")
```

```kotlin
lifecycle.addObserver(LifecycleDetector( TAG: "MainActivity"))
```

Q. LifecycleOwner의 생명주기 변경은 누가 알려주는가?

```
class MyActivity : Activity(), LifecycleOwner {

    private lateinit var lifecycleRegistry: LifecycleRegistry

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        lifecycleRegistry = LifecycleRegistry(this)
        lifecycleRegistry.markState(Lifecycle.State.CREATED)
    }

    public override fun onStart() {
        super.onStart()
        lifecycleRegistry.markState(Lifecycle.State.STARTED)
    }

    override fun getLifecycle(): Lifecycle {
        return lifecycleRegistry
    }
}
```

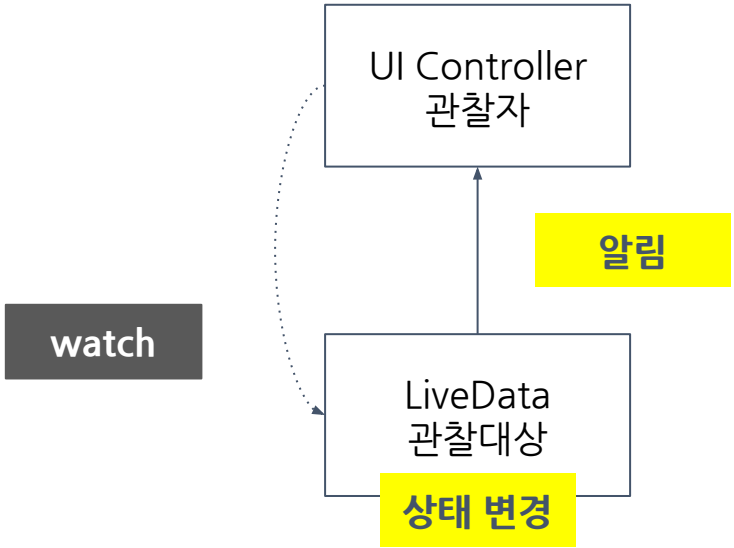
- LifecycleRegistry 클래스를 사용해,
  직접 이벤트를 전달해야 한다.

https://developer.android.com/topic/libraries/architecture/lifecycle#implementing-lco

LiveData + Observer pattern

Lifecycle을 통해 생명 주기를 인식하는 Observable한 data holder class다.

- LiveData는 LifecycleObserver로 UI Controller의 생명주기를 observe 한다.
- UI Controller는 LiveData 값의 변경을 observe 한다.




```
viewModel.newTaskEvent.observe(viewLifecycleOwner, EventObserver {
    navigateToAddNewTask()
})
```

Observer 누가 등록하고 있을까요?