

2019
DROID KNIGHTS

Android TDD 적용기

Speaker. 김성일



이메일 입력

비밀번호 입력

[비밀번호 찾기](#)

로그인

굿닥에 처음 오셨나요? [회원가입](#)



페이스북



카카오톡



네이버

- TDD & Android Test 알아보기
- TDD를 위한 준비
- Business logic - TDD로 개발하기
- UI 테스트 작성

Goal

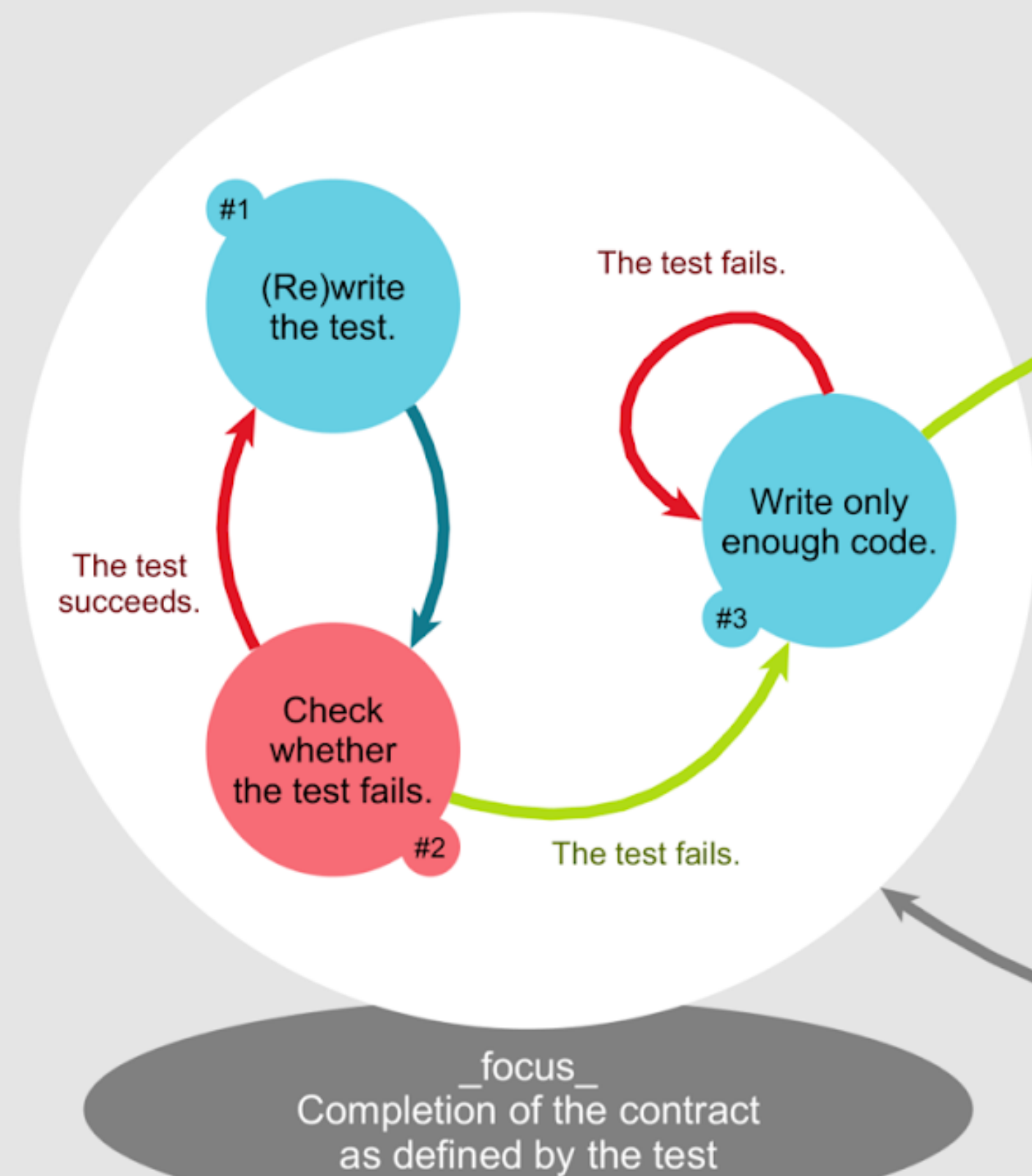
어..?! TDD.. 나도 한번 해볼까..?

What is TDD?

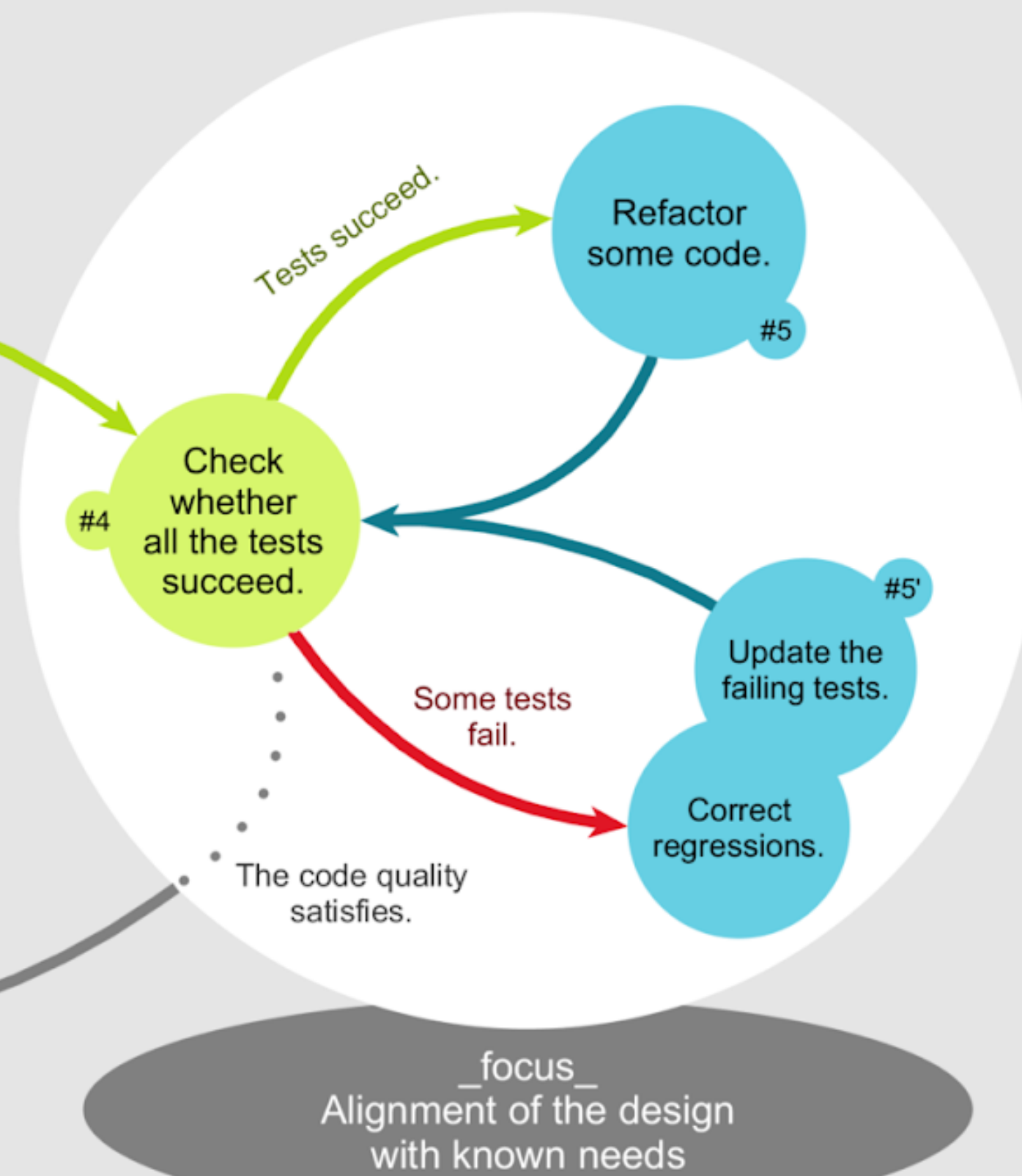
- **Test Driven Development**
- **Software development process**
- **Not same Unit test**

What is ^{TDD}TDD?

CODE-DRIVEN TESTING

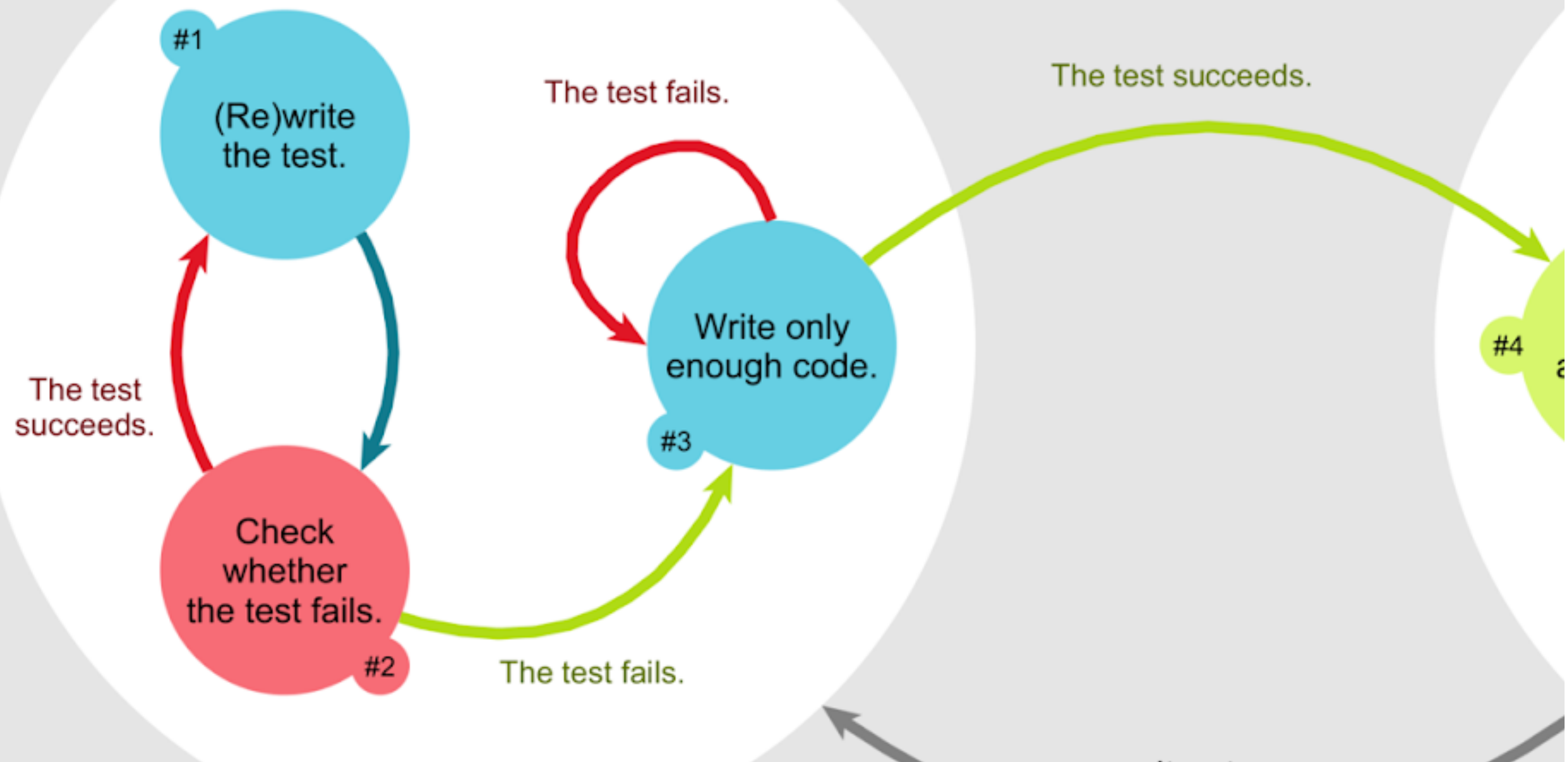


REFACTORING

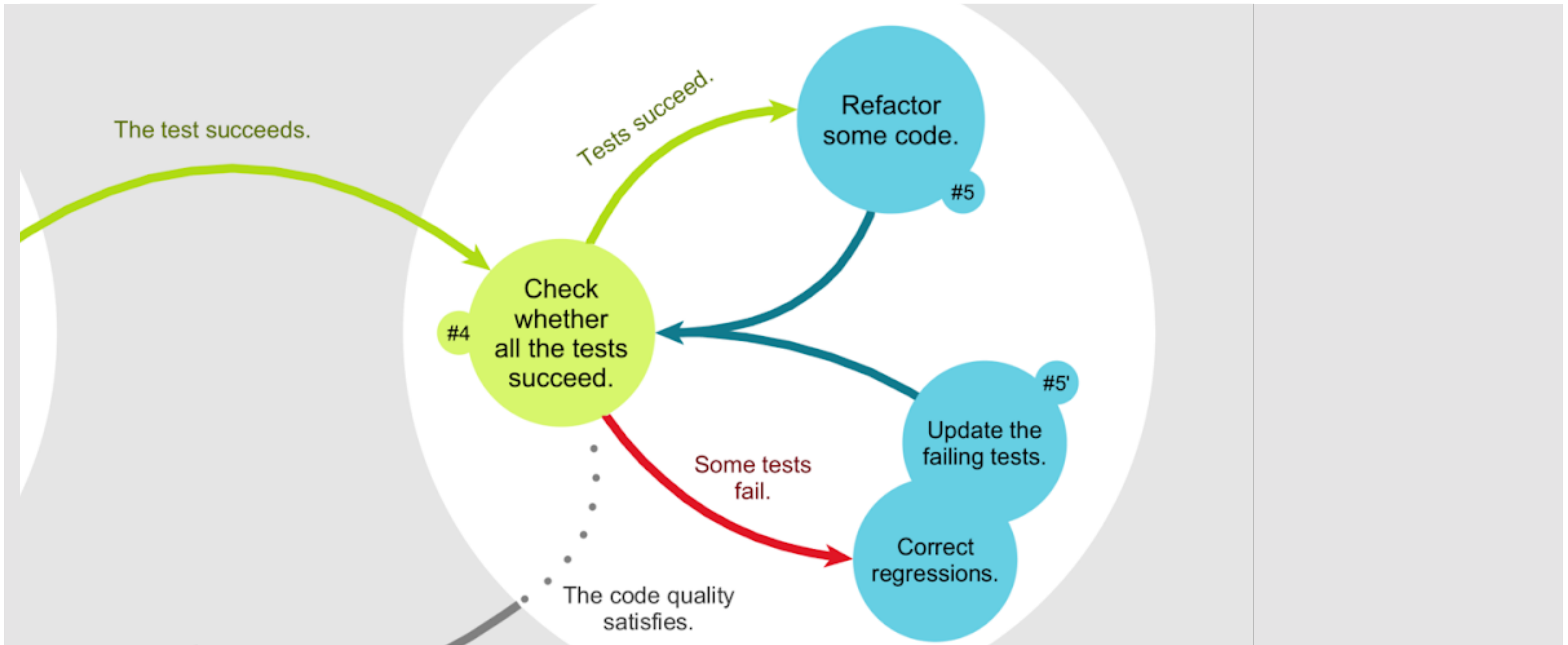


Iterate

What is TDD?

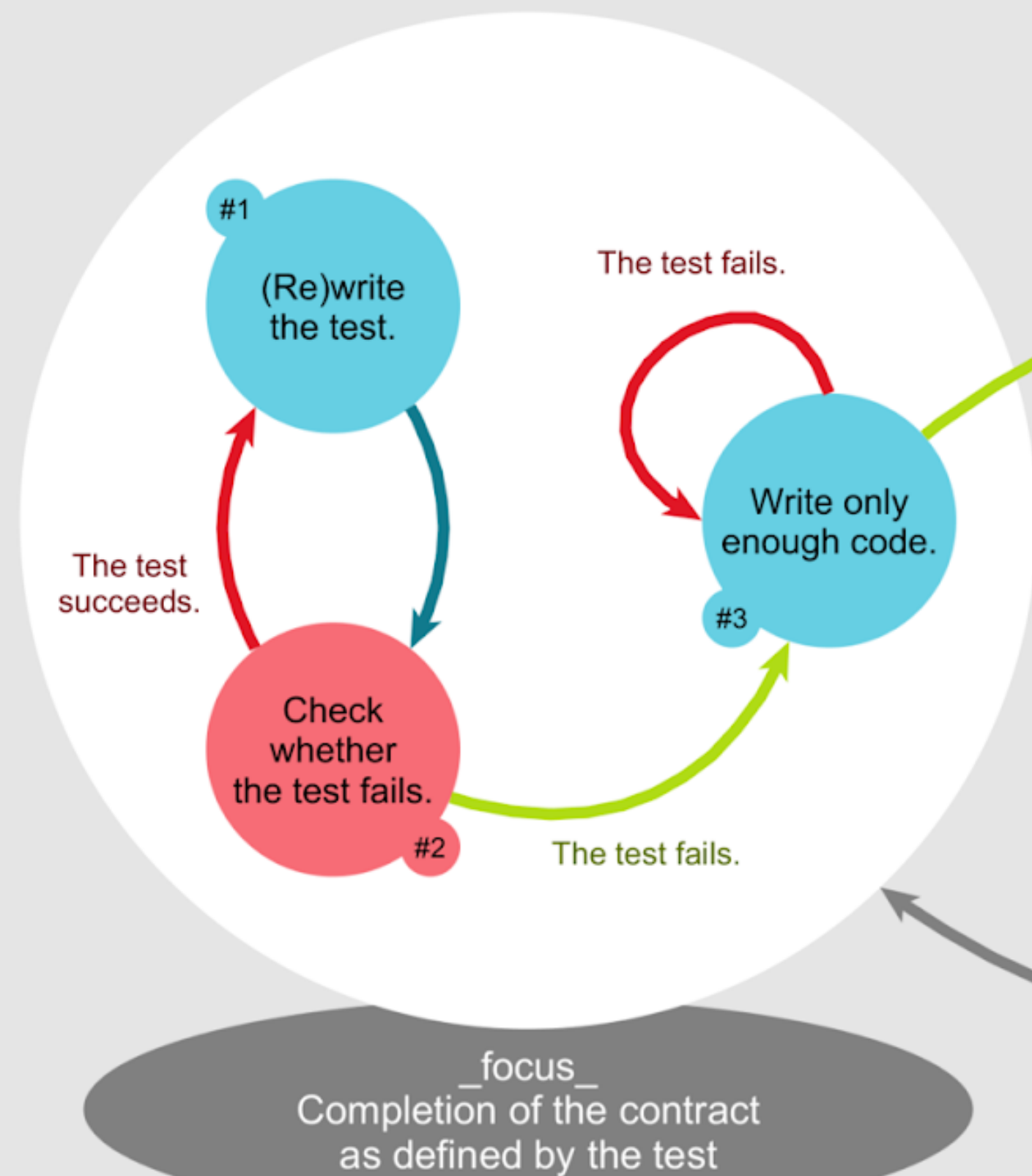


What is TDD?

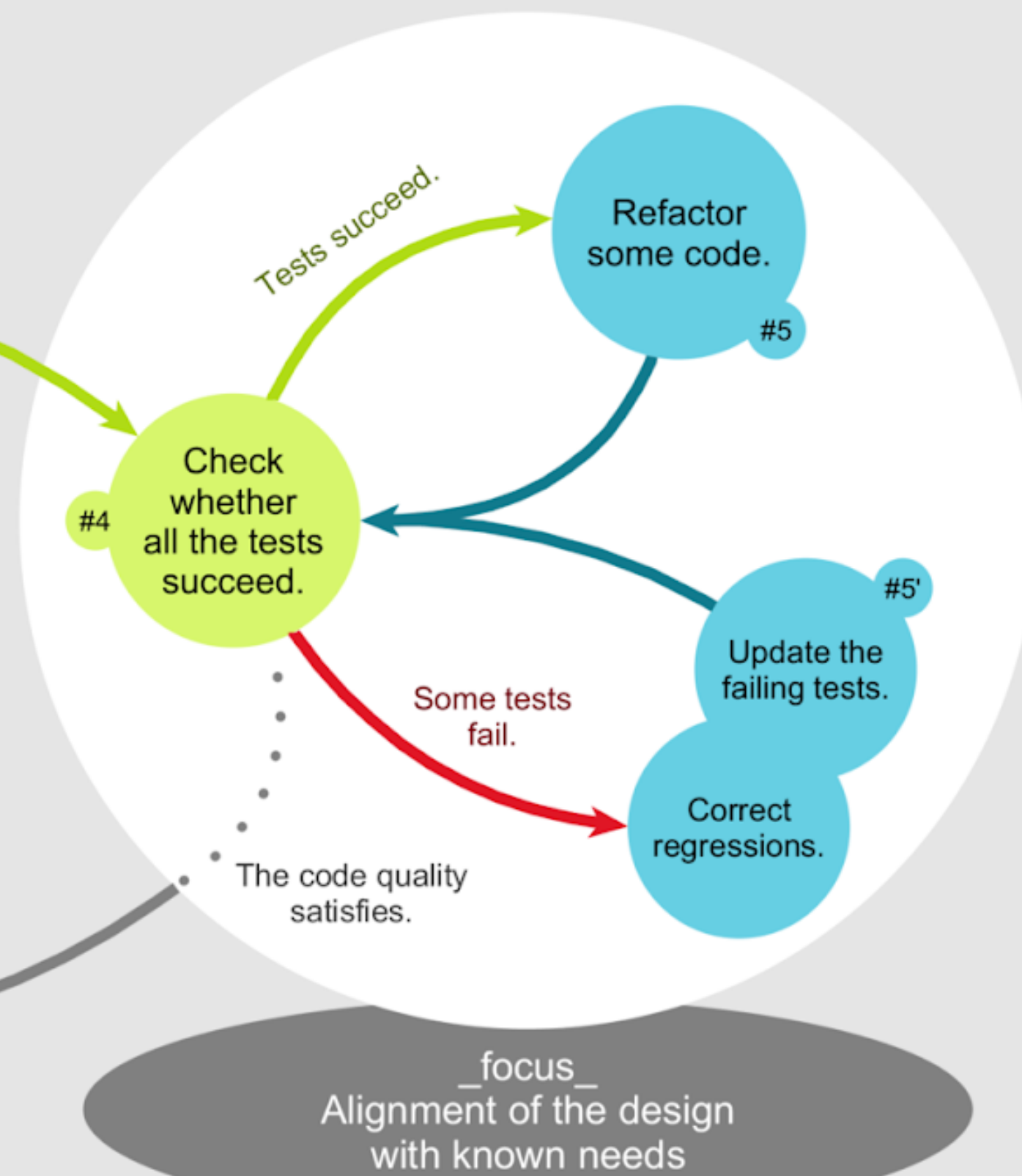


What is TDD?

CODE-DRIVEN TESTING

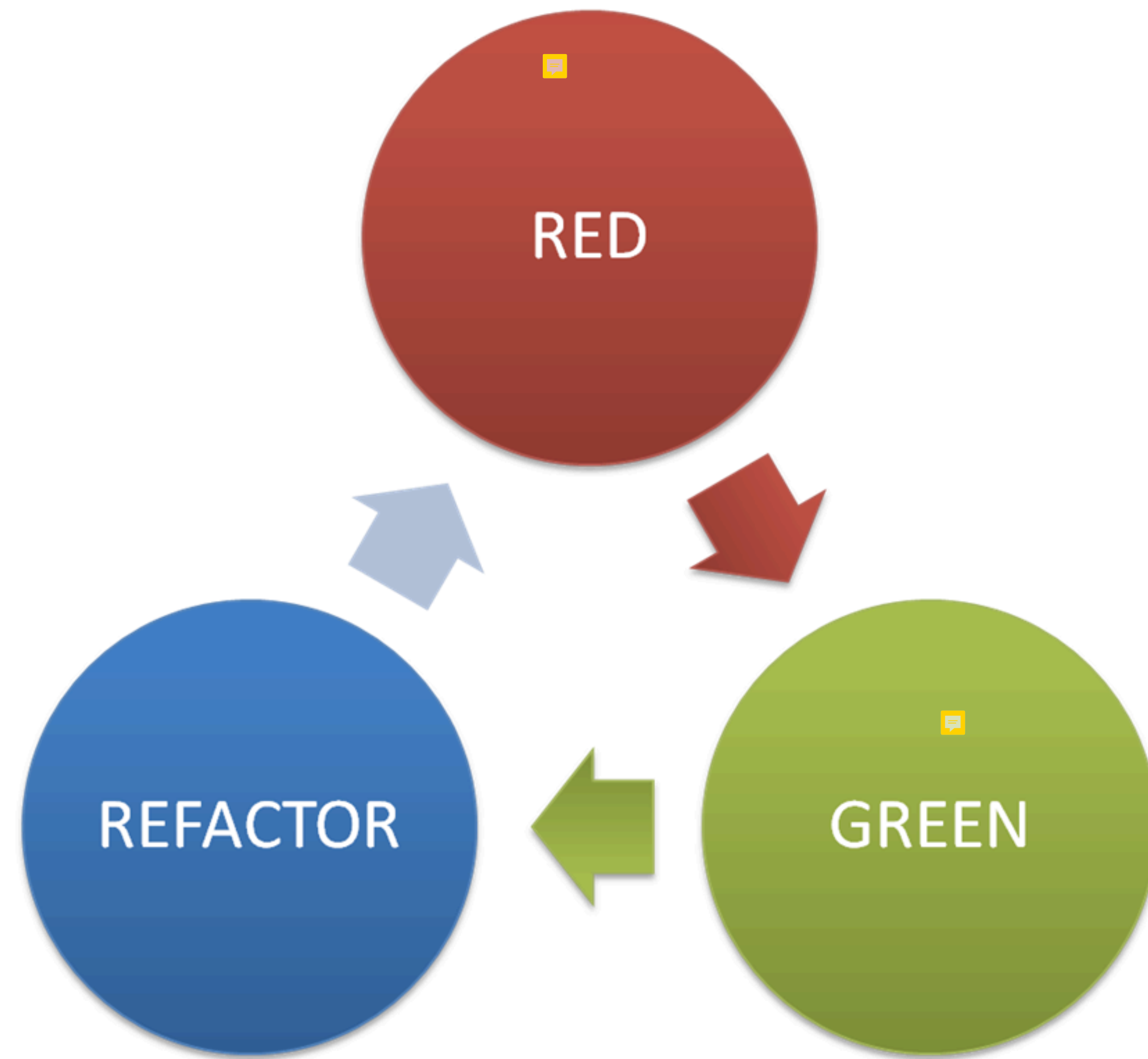


REFACTORING

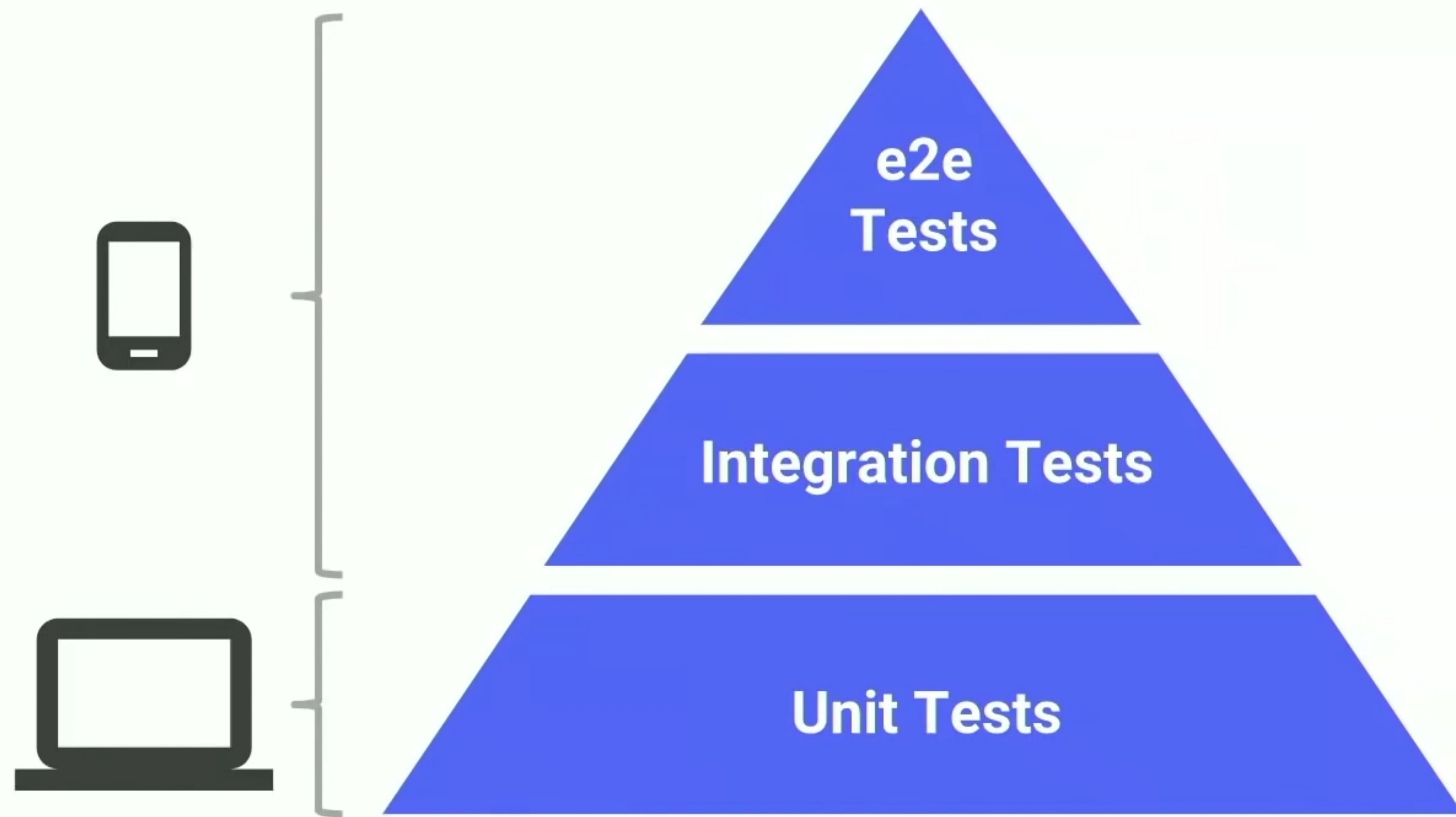


Iterate

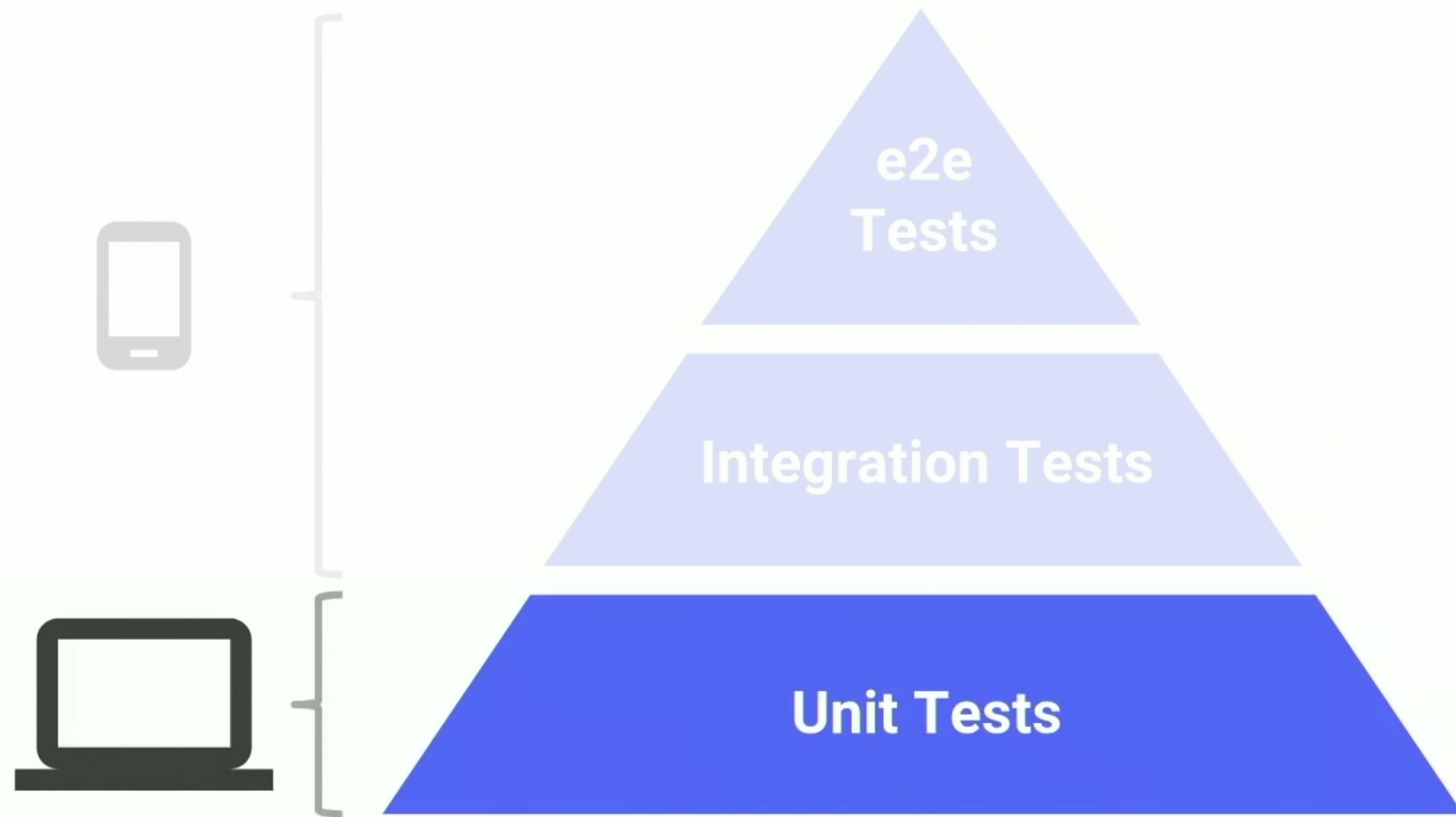
What is TDD?



Android Test

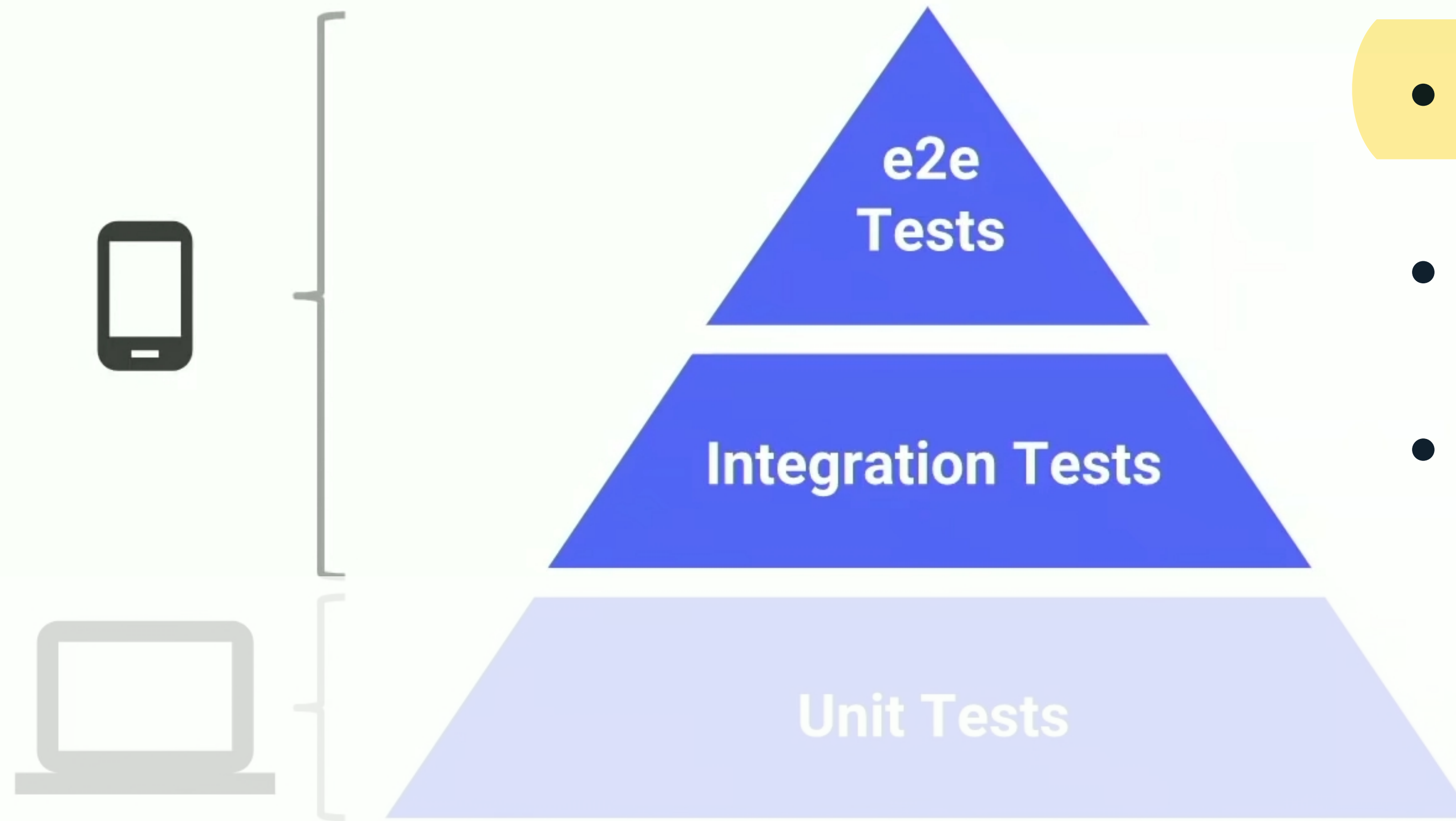


Android Test



- Class / Method
- Local Test (JVM)
- Fast

Android Test

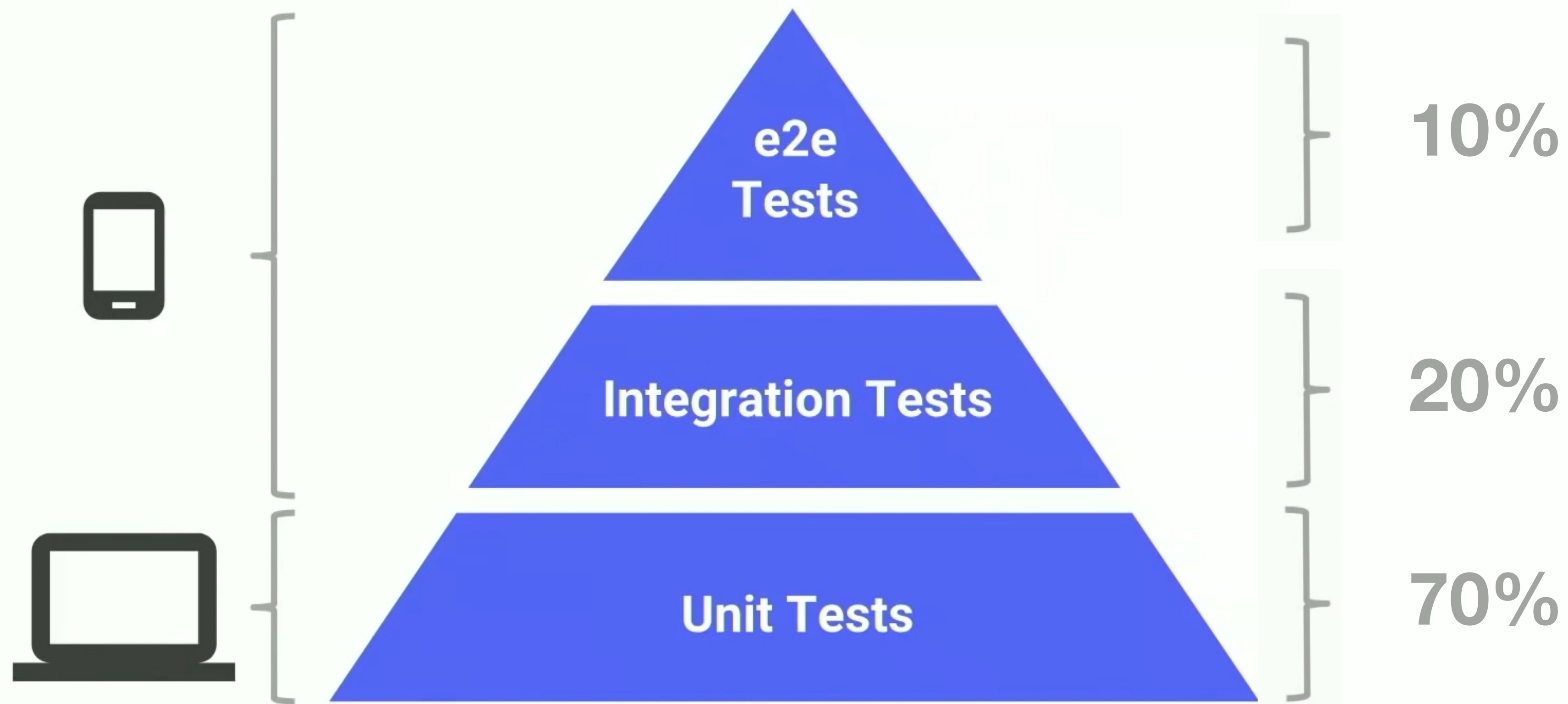


- **Slow**

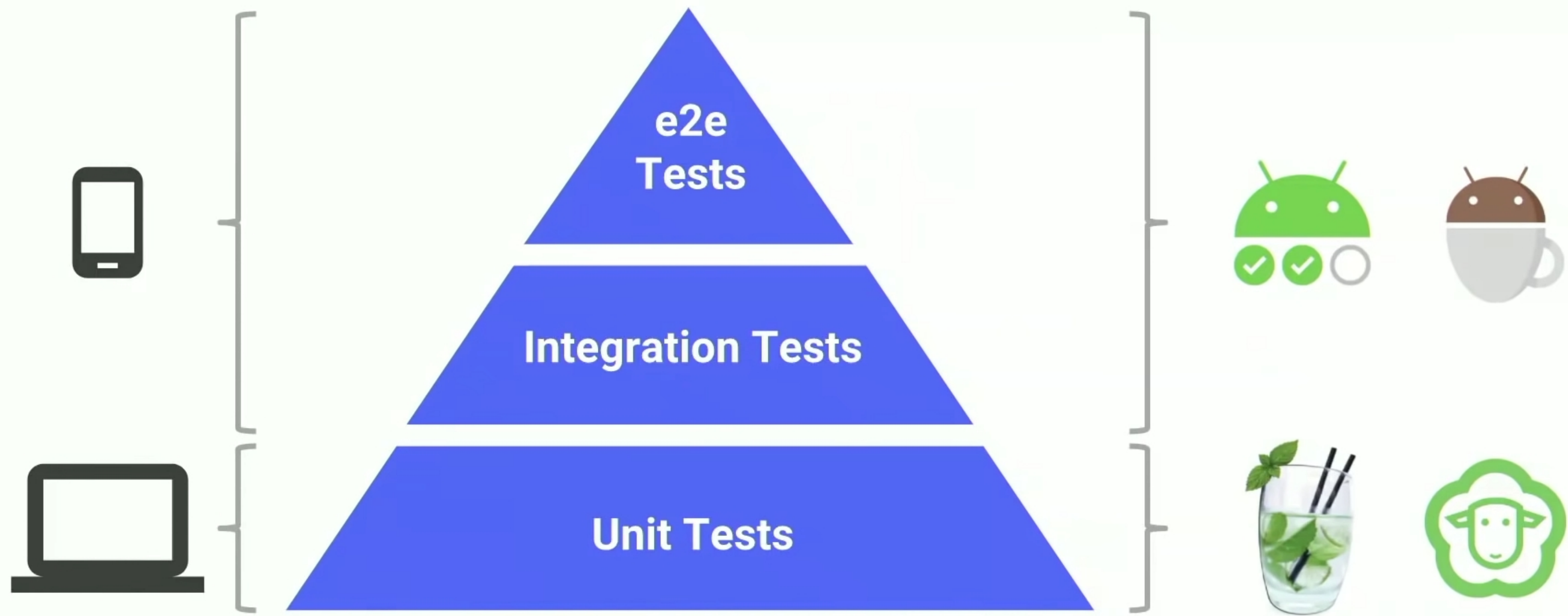
- **Real / Virtual Device**

- **On Android Framework**

Android Test



Android Test



What is first?

What is first?

Architecture

What is first?

MVC

Clean Architecture

MVP

Architecture

MVVM

MVI

What is first?

MVC

Clean Architecture

Testable Architecture MVP

MVVM

MVI

What is testable architecture?

테스트 가능한 아키텍처의 핵심 아이디어는
응용 프로그램의 부분을 분리하여
개별적으로 유지 관리하고 테스트 할 수 있도록
하는 것입니다.

What is testable architecture?

테스트 가능한 아키텍처의 핵심 아이디어는
응용 프로그램의 부분을 분리하여
개별적으로 유지 관리하고 테스트 할 수 있도록
하는 것입니다.

Model - View - Presenter (MVP)

View

Presenter

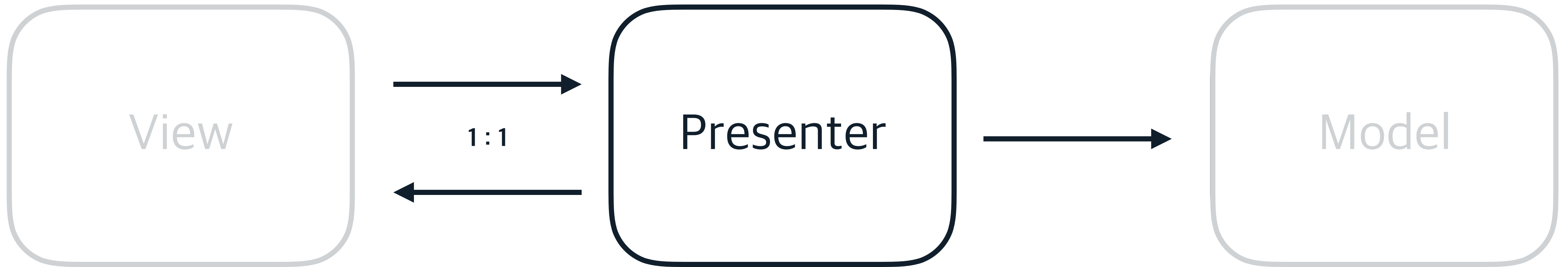
Model

Model - View - Presenter (MVP)



- Handle the display of data
- Forward user action to presenter

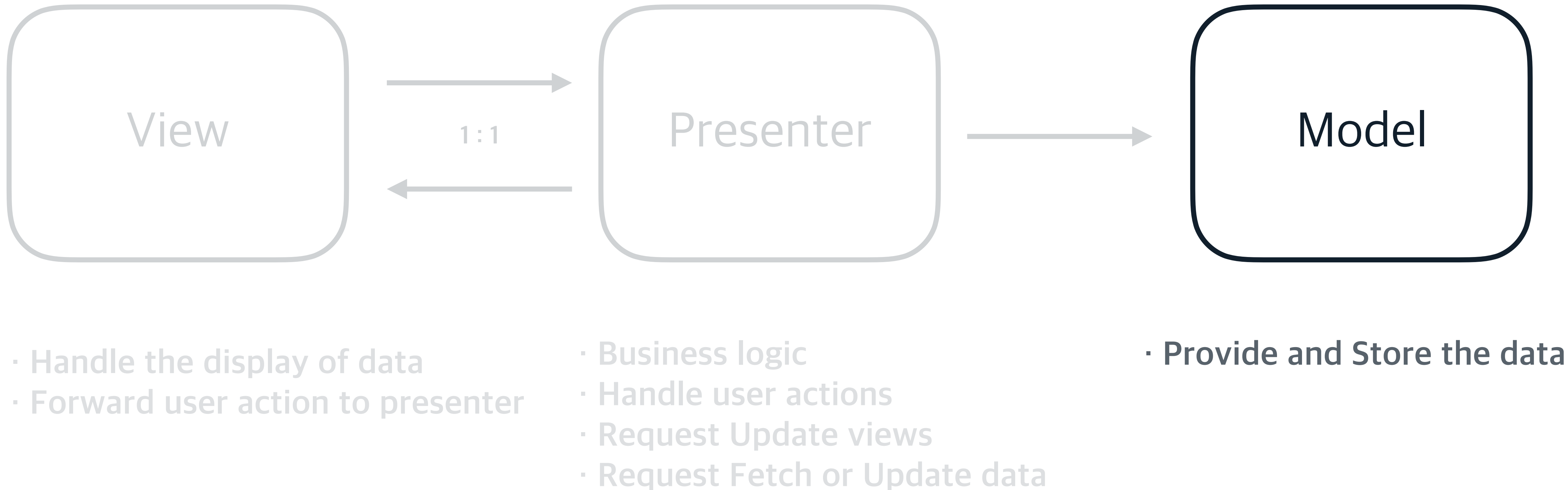
Model - View - Presenter (MVP)



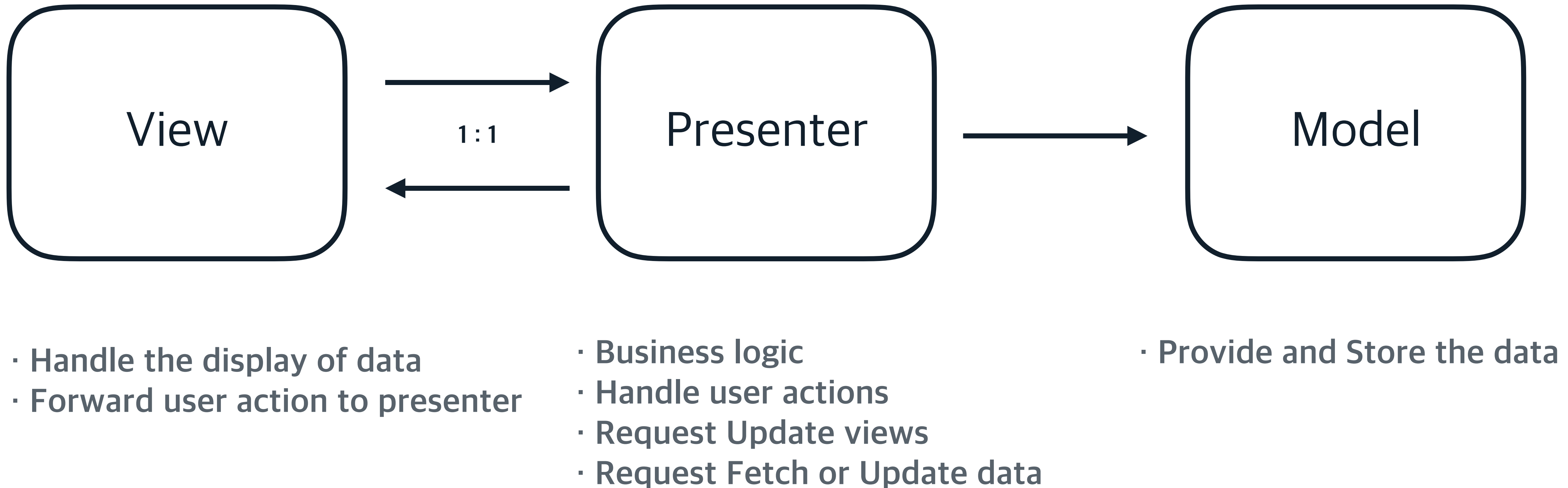
- Handle the display of data
- Forward user action to presenter

- **Business logic**
- Handle user actions
- Request Update views
- Request Fetch or Update data

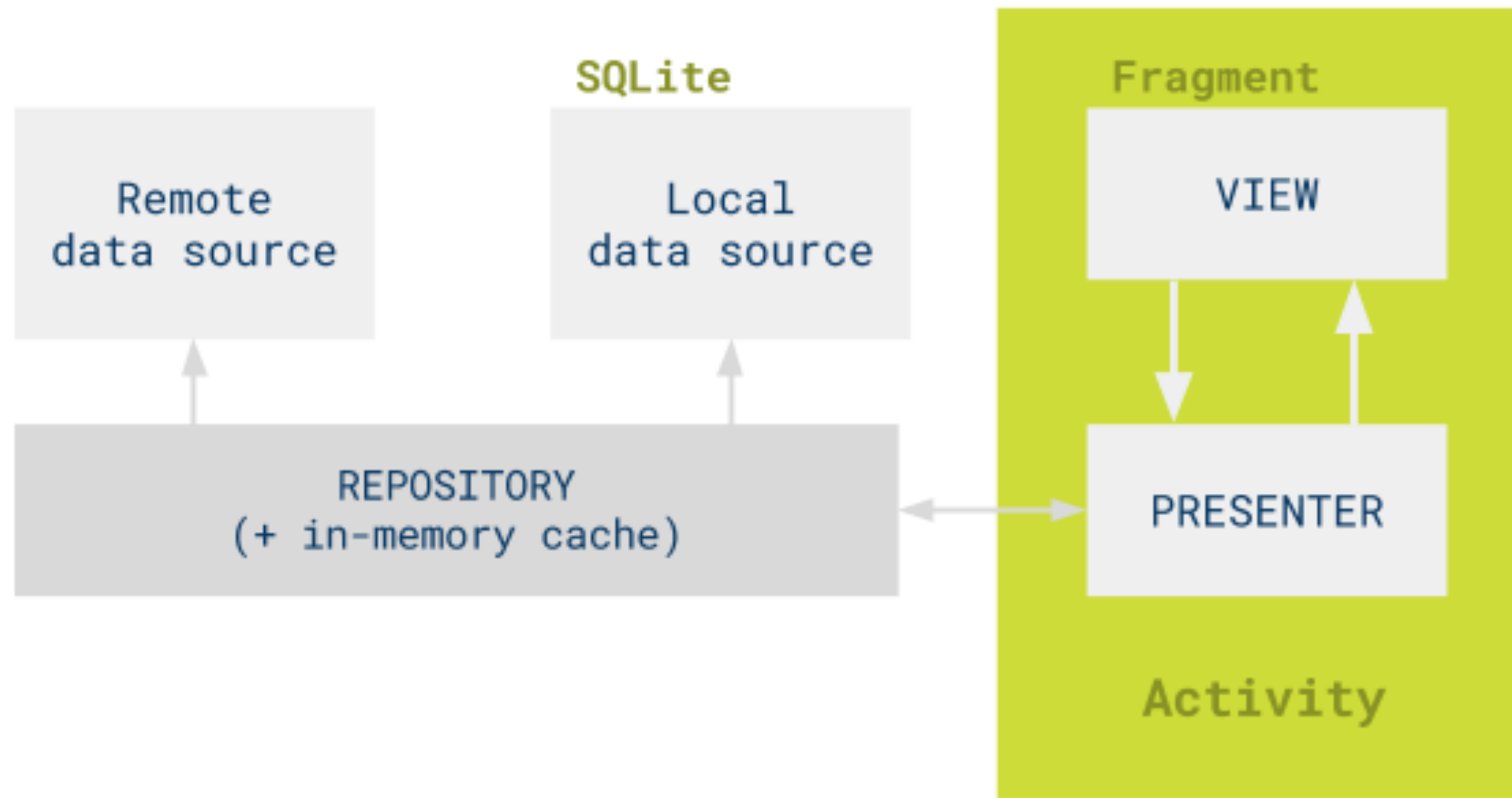
Model - View - Presenter (MVP)



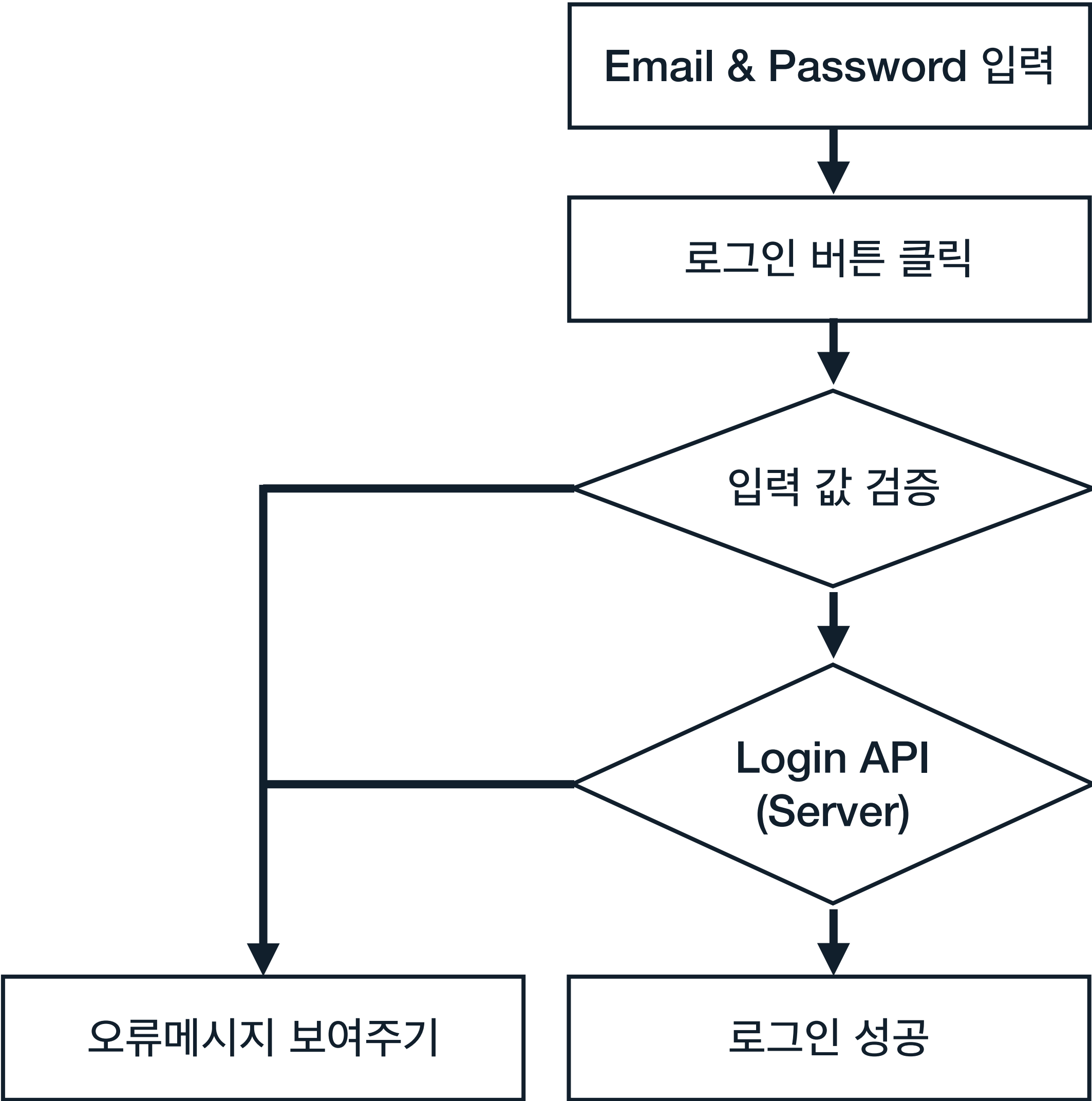
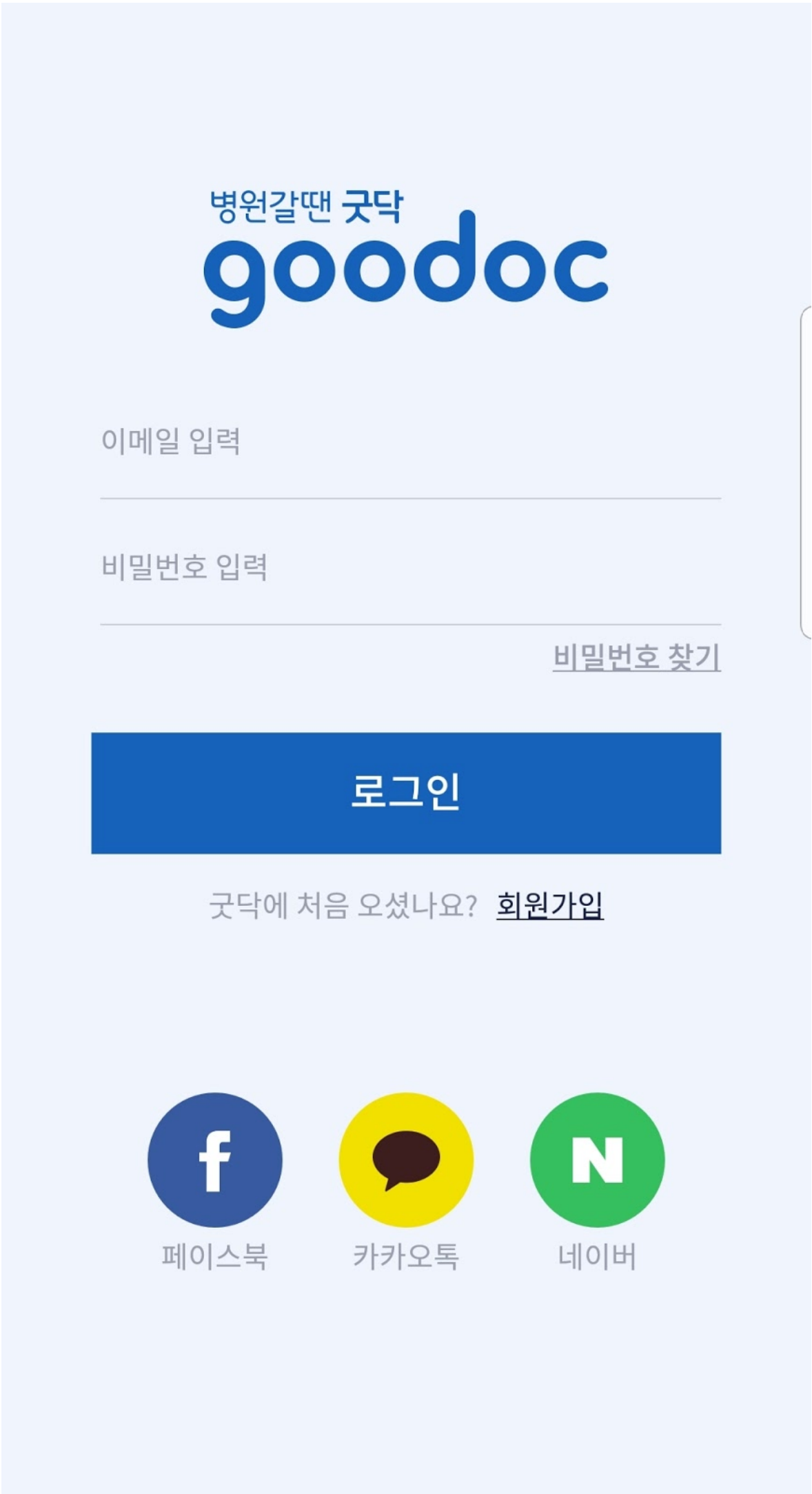
Model - View - Presenter (MVP)



Model - View - Presenter (MVP) + Repository



Develop email login with TDD



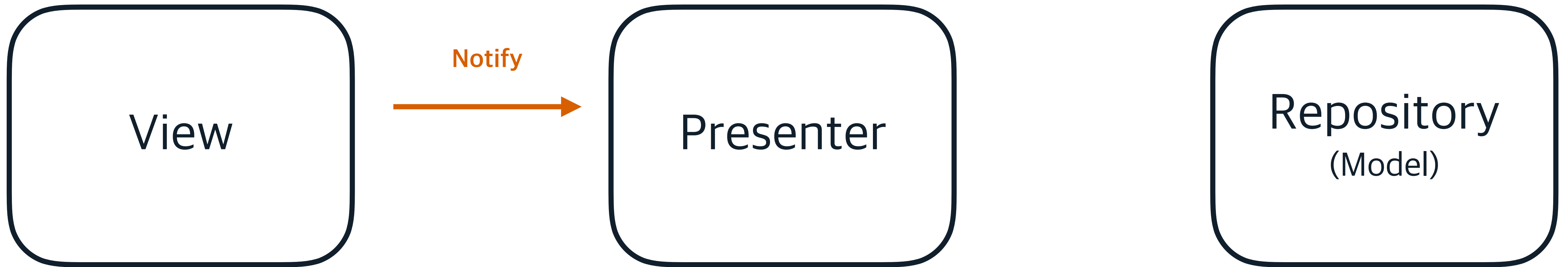
Develop email login with TDD

View

Presenter

Repository
(Model)

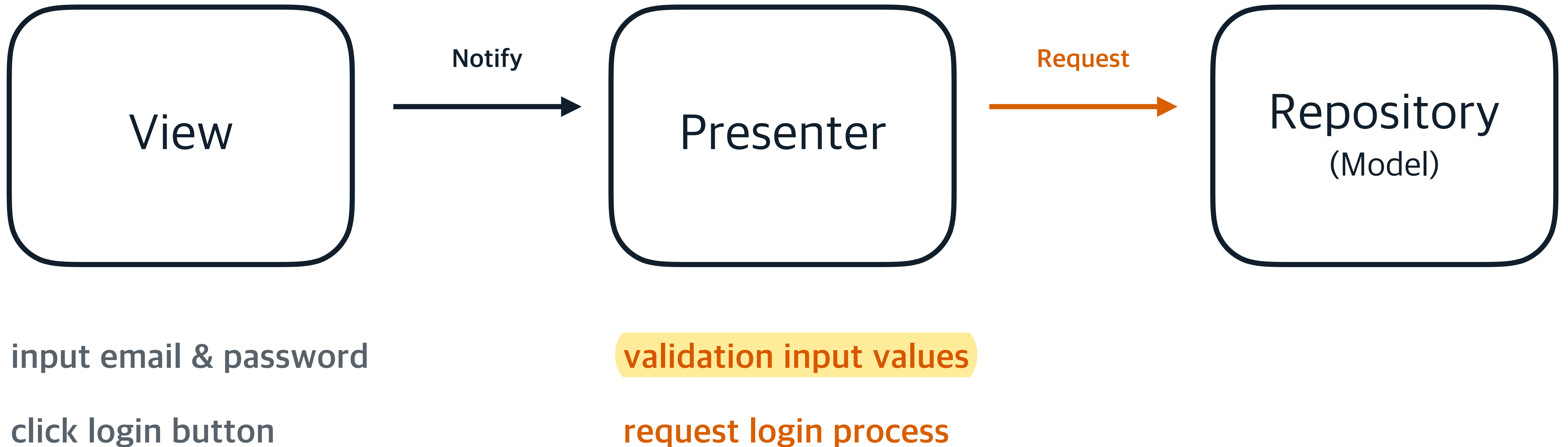
Develop email login with TDD



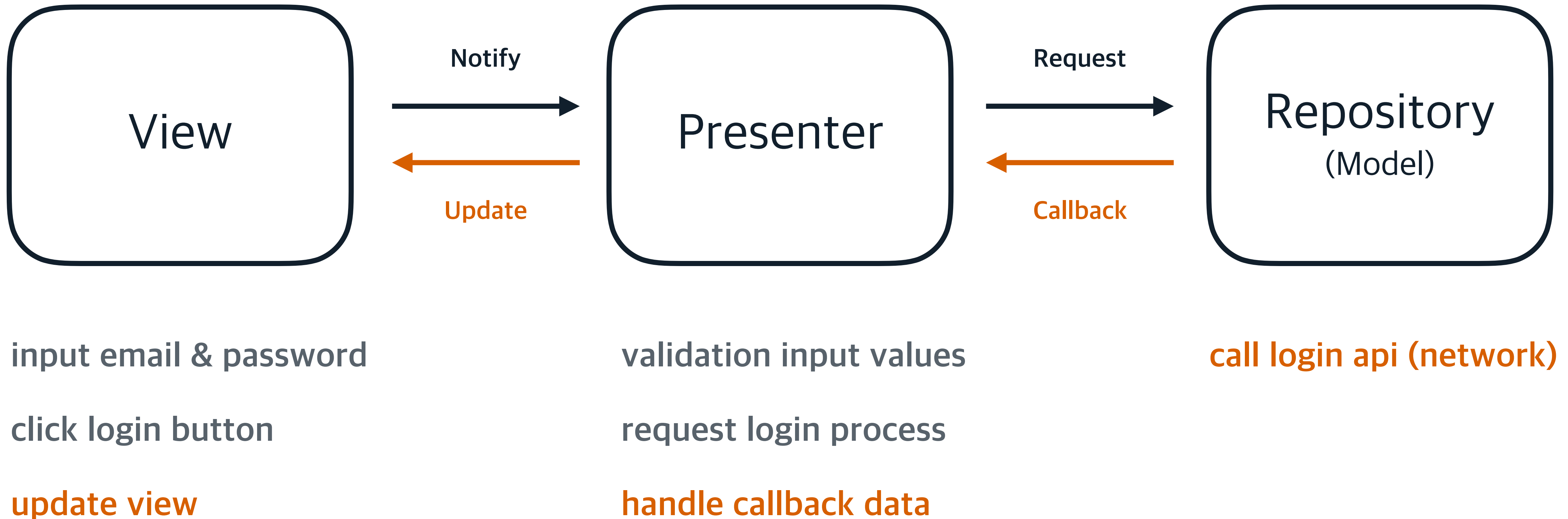
input email & password

click login button

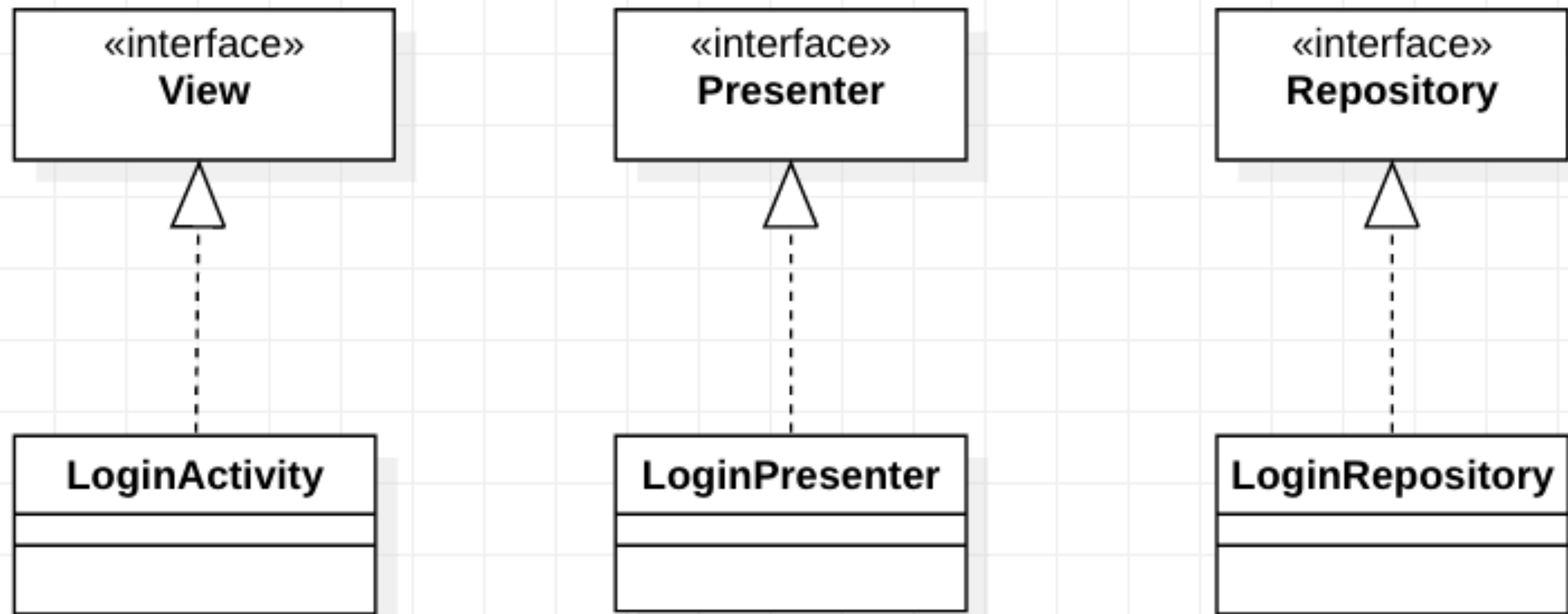
Develop email login with TDD



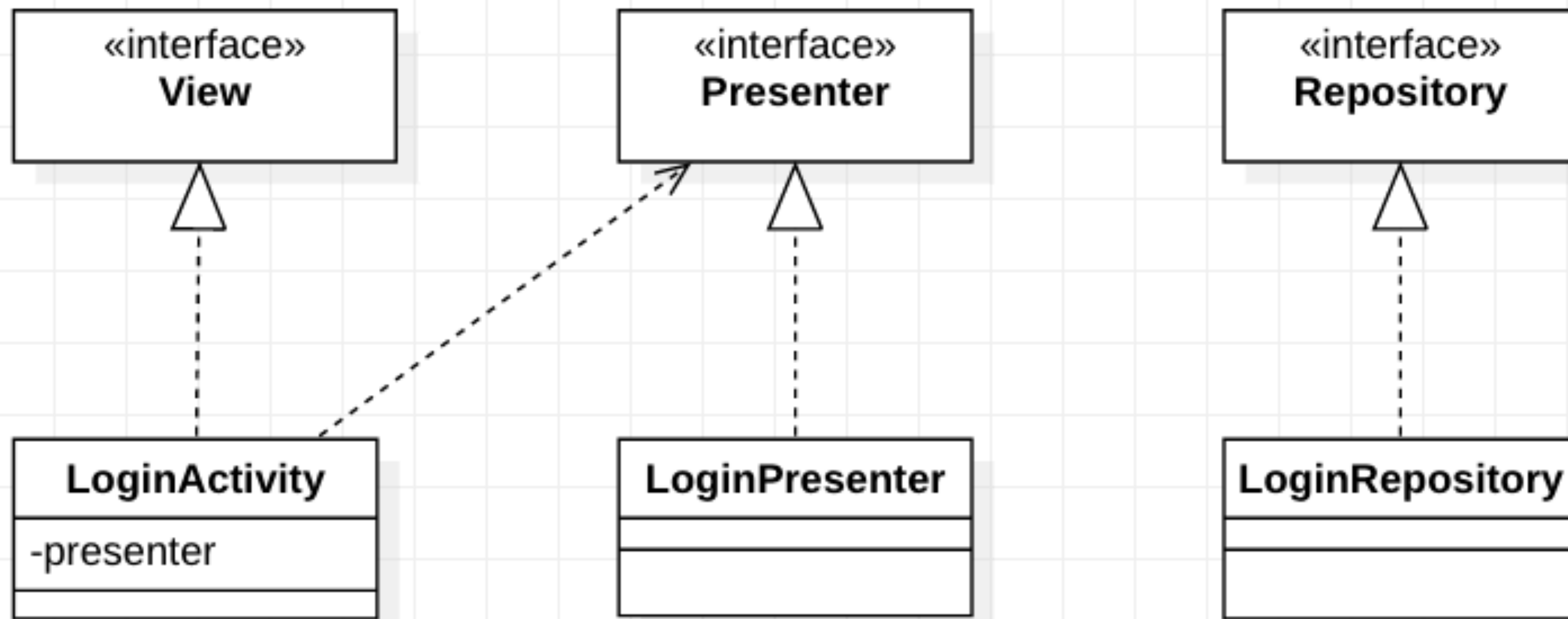
Develop email login with TDD



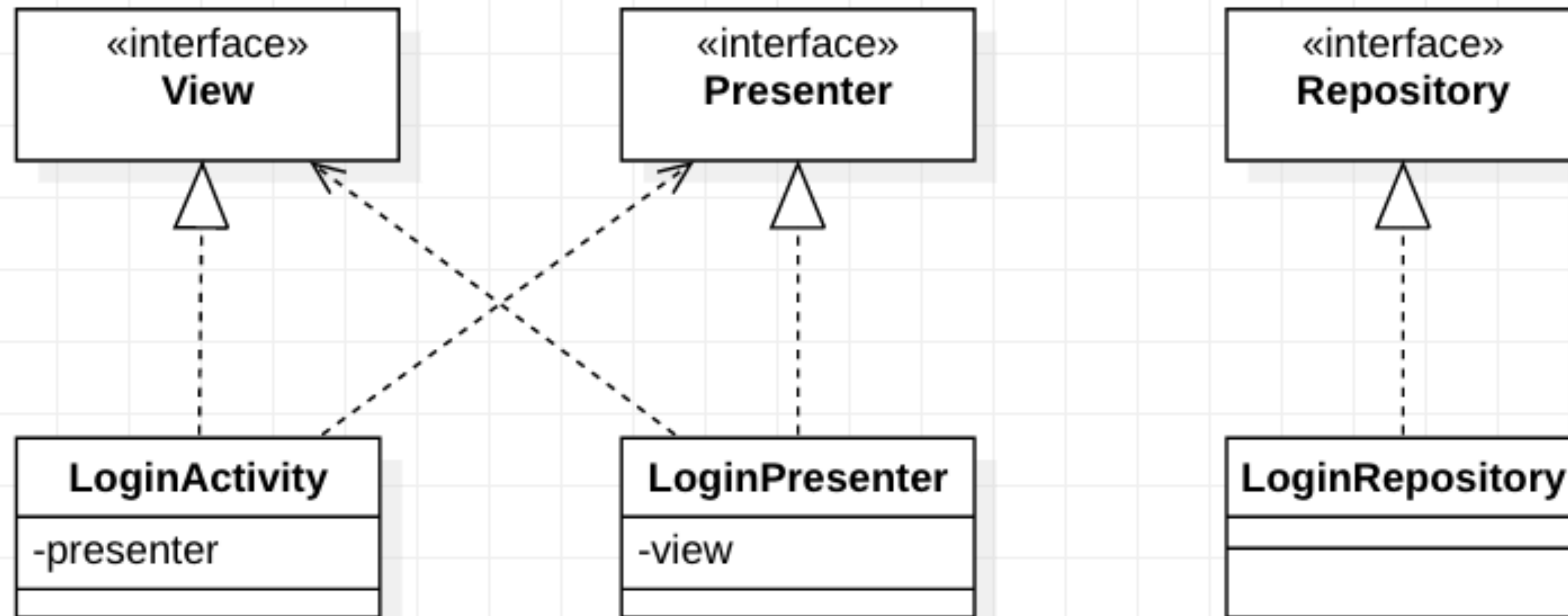
MVP Example Class Diagram



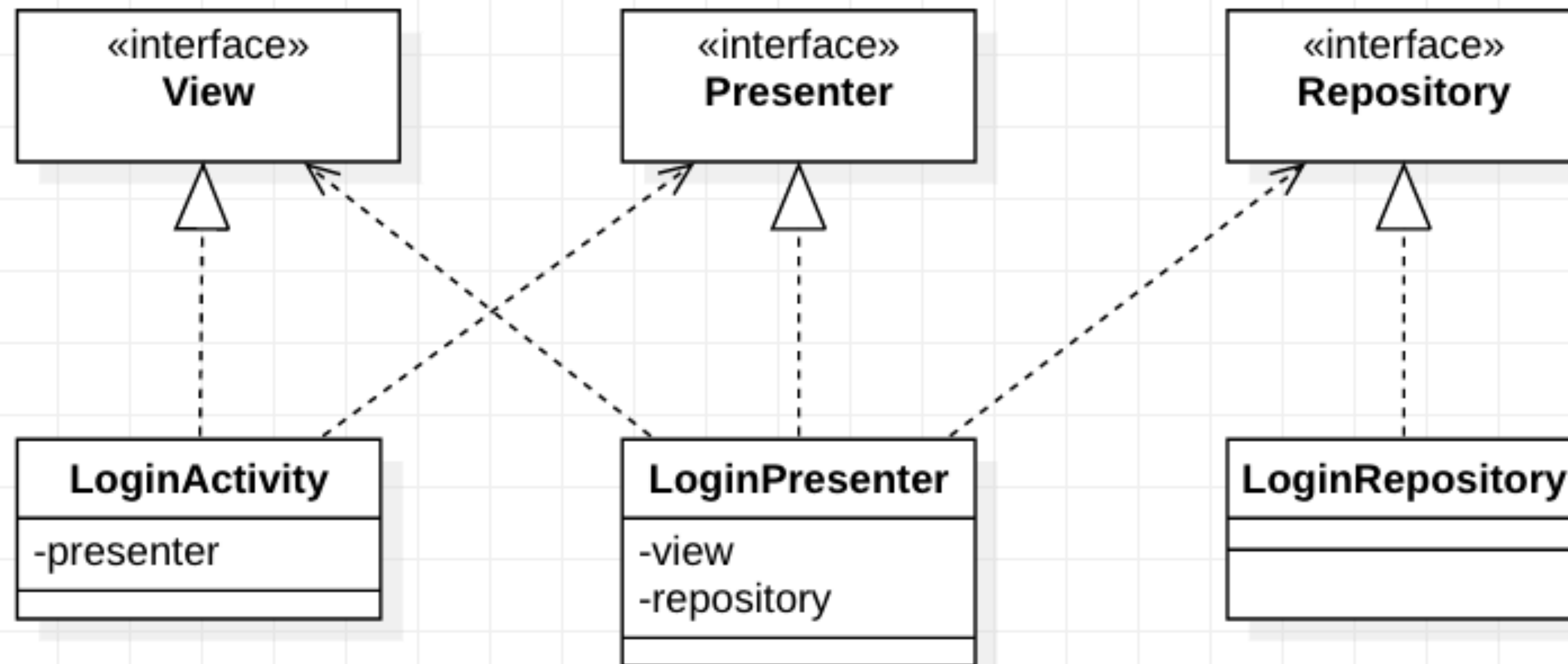
MVP Example Class Diagram



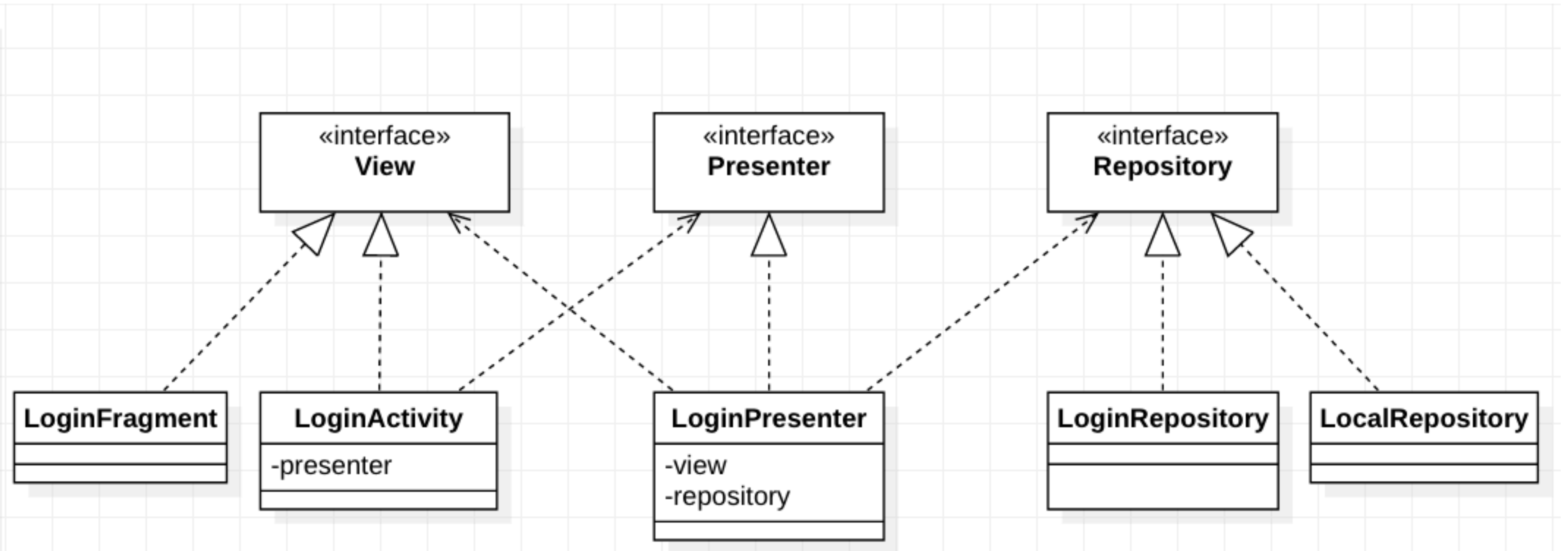
MVP Example Class Diagram



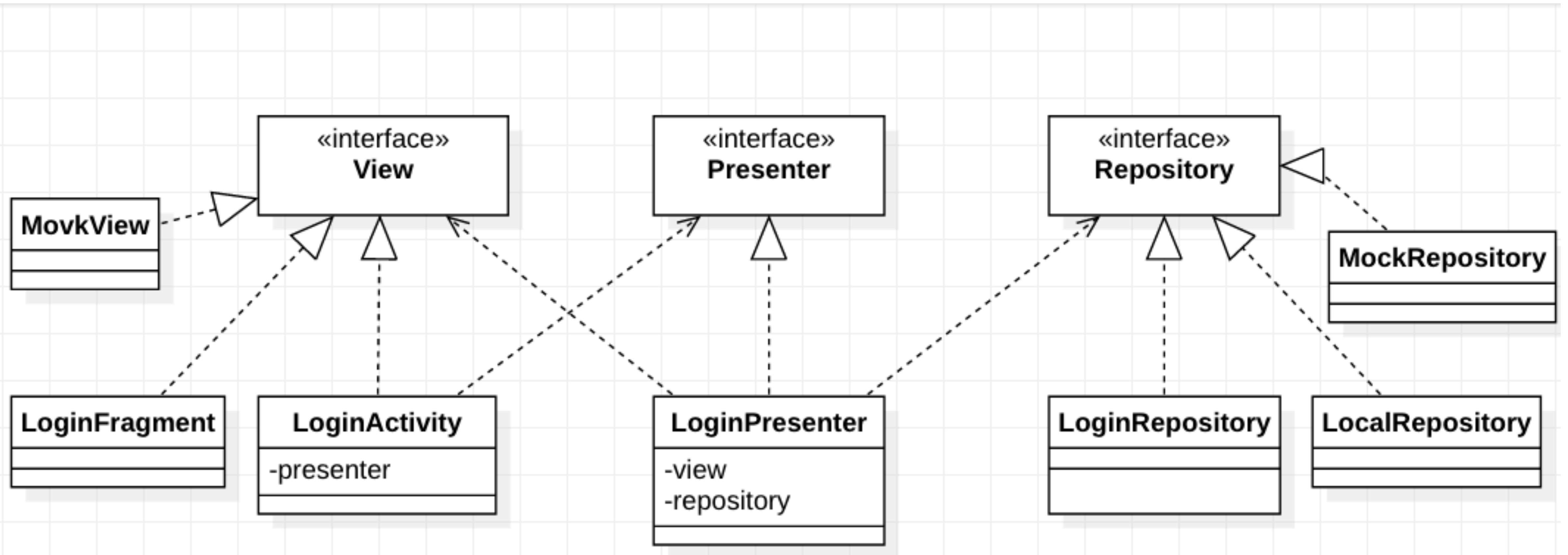
MVP Example **Class Diagram**



MVP Example Class Diagram



MVP Example UML Class Diagram



Contract

```
class LoginContract {  
    interface View {  
    }  
  
    interface Presenter {  
    }  
  
    interface Repository {  
    }  
}
```

Contract

```
class LoginContract {  
    interface View {  
    }  
  
    interface Presenter {  
    }  
  
    interface Repository {  
    }  
}
```

```
class LoginContract {  
    interface View {  
    }  
  
    interface Presenter {  
    }  
}  
  
interface Repository {  
    }  
}
```


Implementation of LoginContract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {
```

```
    private lateinit var presenter: LoginContract.Presenter
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_login)
```

```
        presenter = LoginPresenter(this, Injector.provideLoginRepository())
```

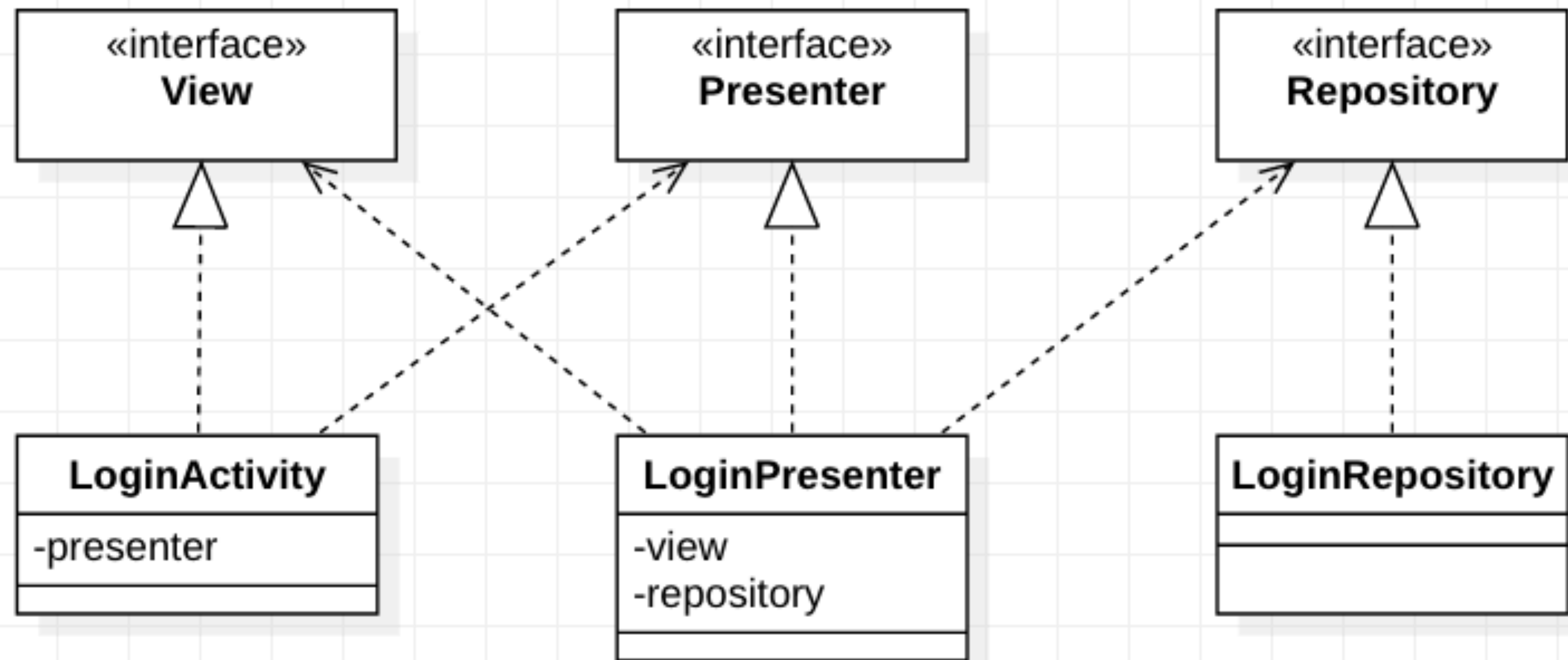
```
    }
```

```
}
```

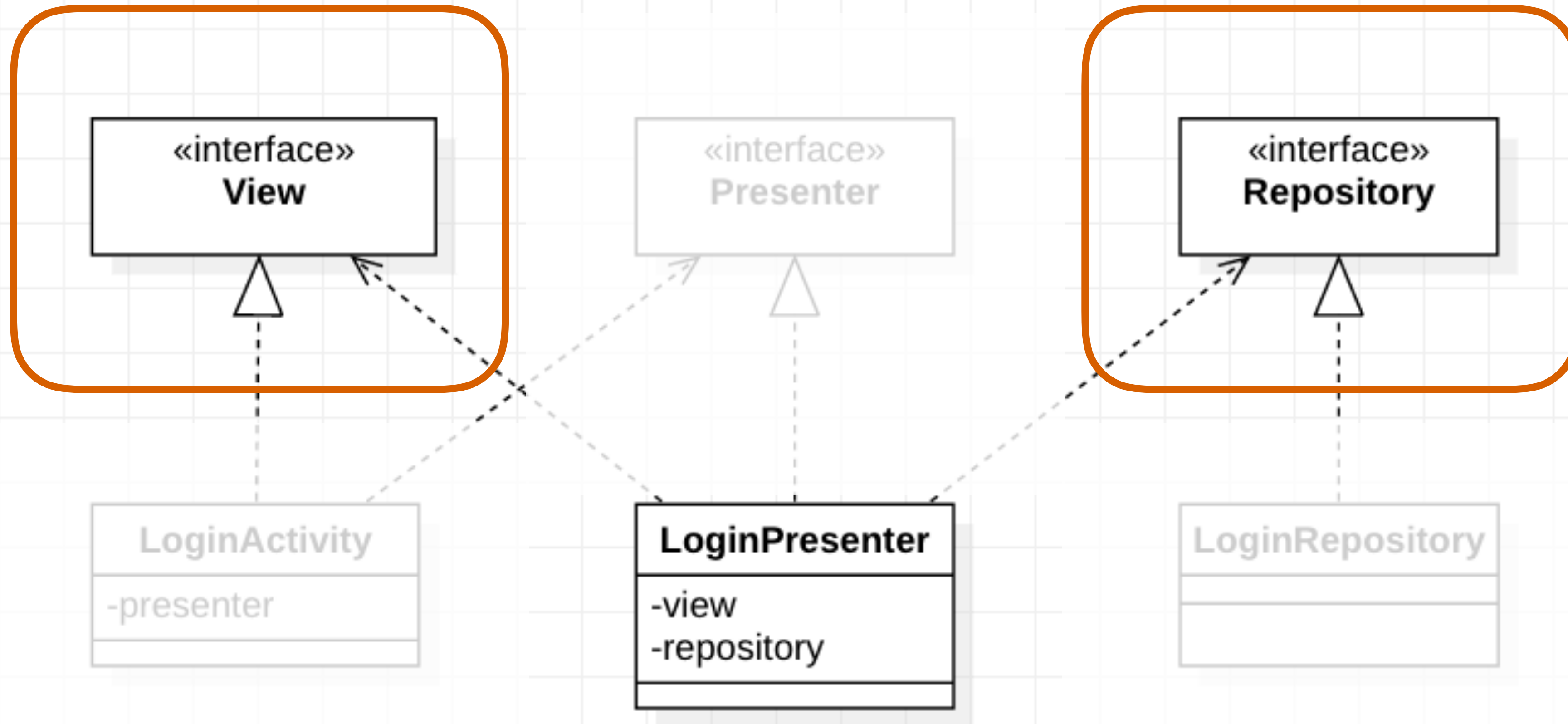
```
class LoginPresenter(val view: LoginContract.View,  
                   val repository: LoginContract.Repository) : LoginContract.Presenter {}
```

```
class LoginRepository : LoginContract.Repository {}
```

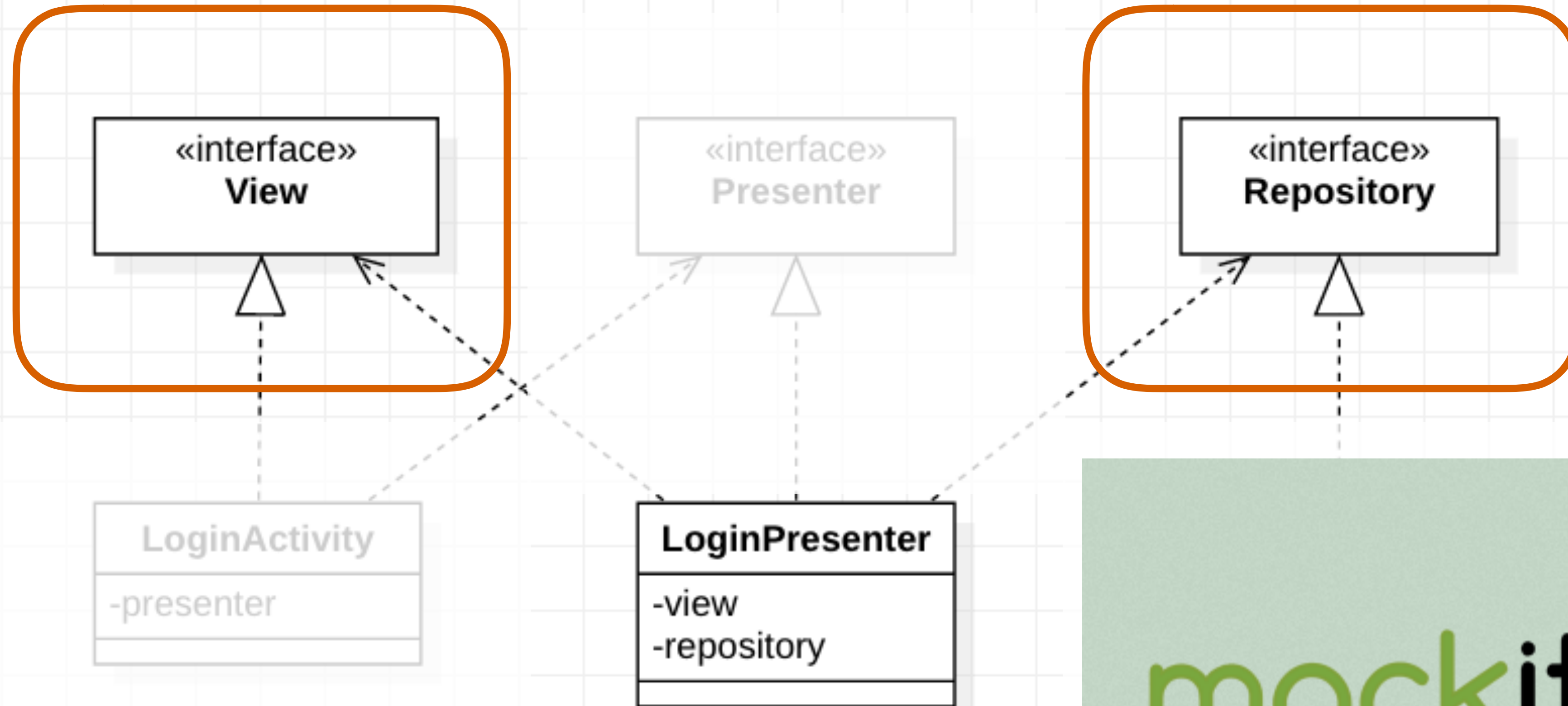
MVP Example Class Diagram



MVP Example Class Diagram



MVP Example Class Diagram



LoginPresenterTest

```
@RunWith(MockitoJUnitRunner::class)
class LoginPresenterTest {

    @Mock
    private lateinit var view: LoginContract.View

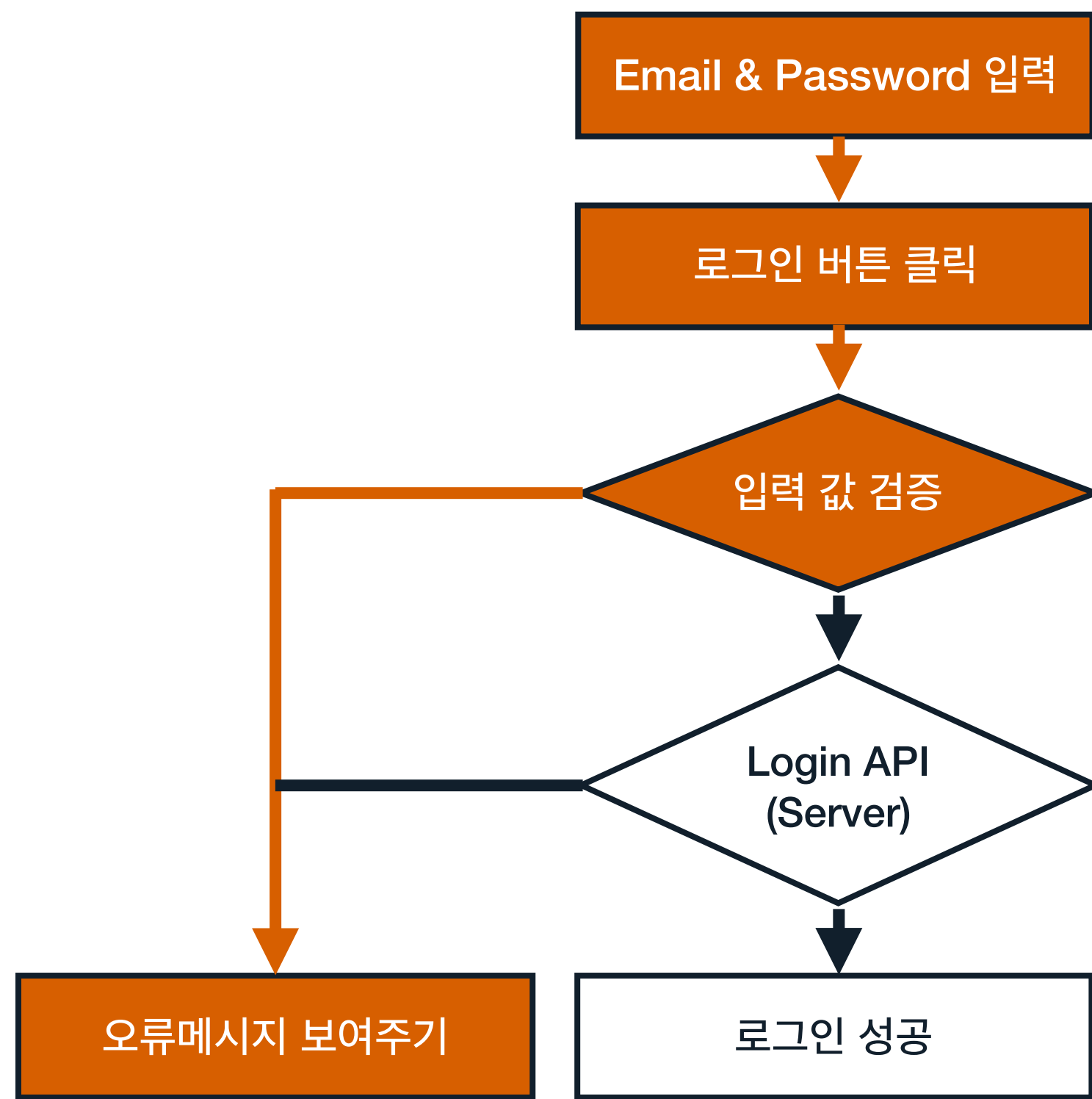
    @Mock
    private lateinit var repository: LoginContract.Repository

    private lateinit var presenter: LoginPresenter

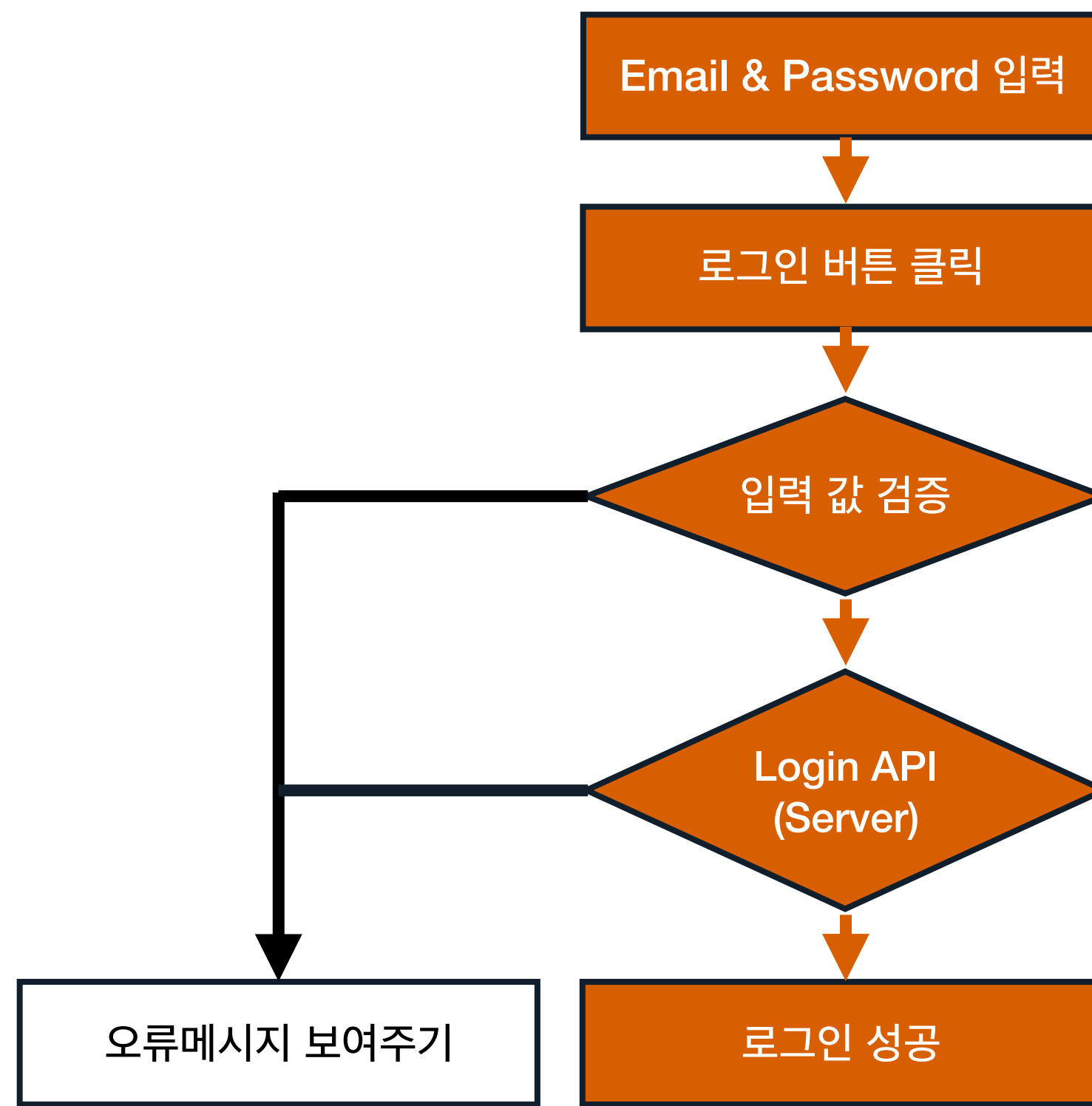
    private lateinit var inOrder: InOrder

    @Before
    fun setup() {
        presenter = LoginPresenter(view, repository)
        inOrder = Mockito.inOrder(view, repository)
    }
}
```

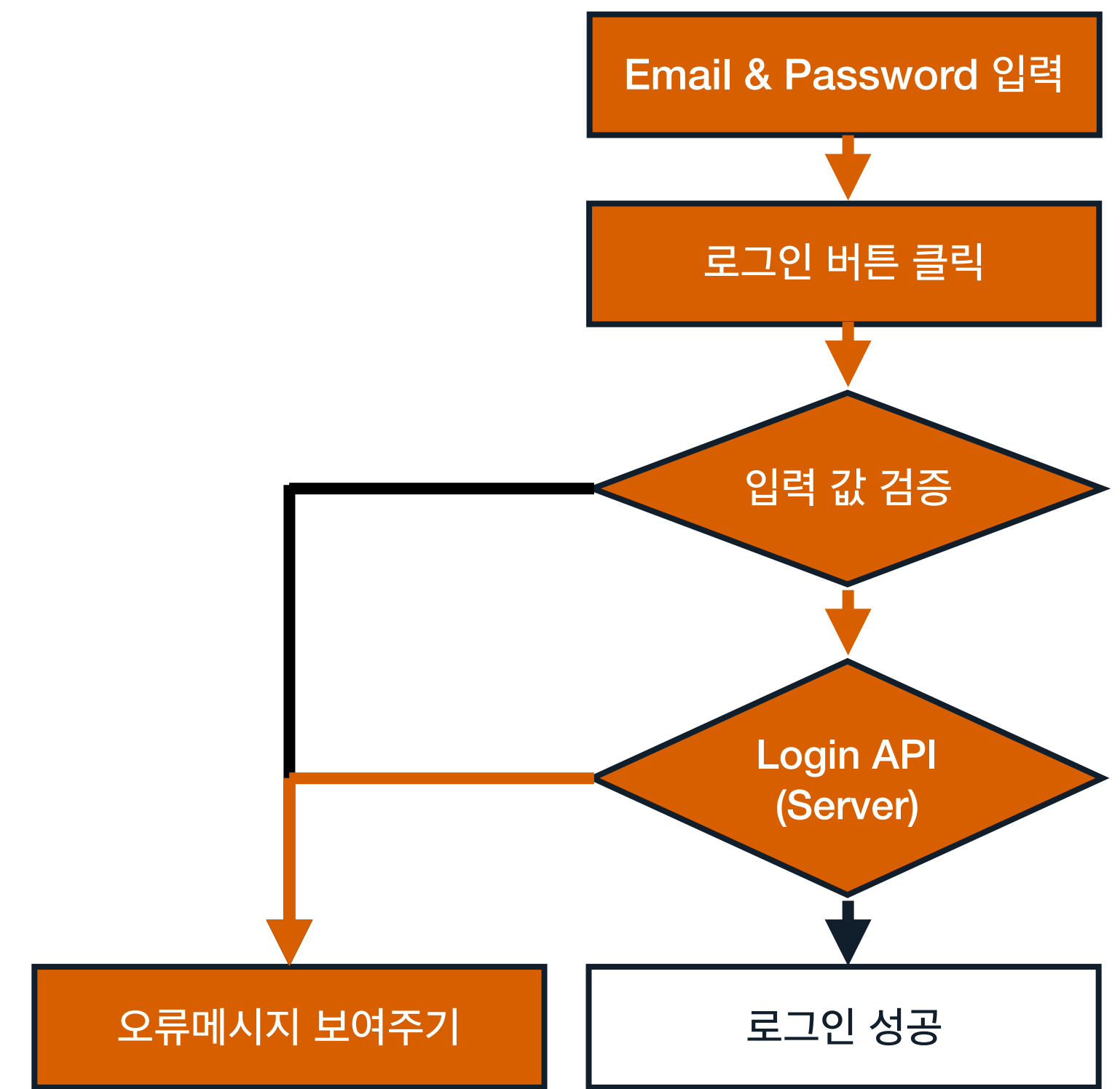
TestCase - Scenarios



[입력값(이메일/패스워드) 검증 실패]



[로그인 성공]



[로그인 실패]

테스트 작성 법칙

Given

When

Then

테스트 작성 법칙

Given - 특정 상황이 주어지고..

When - (테스트하려는) 특정 액션이 발생했을 때..

Then - 변화 된 상태나 수행되는 행동을 검증한다.

테스트 작성 법칙

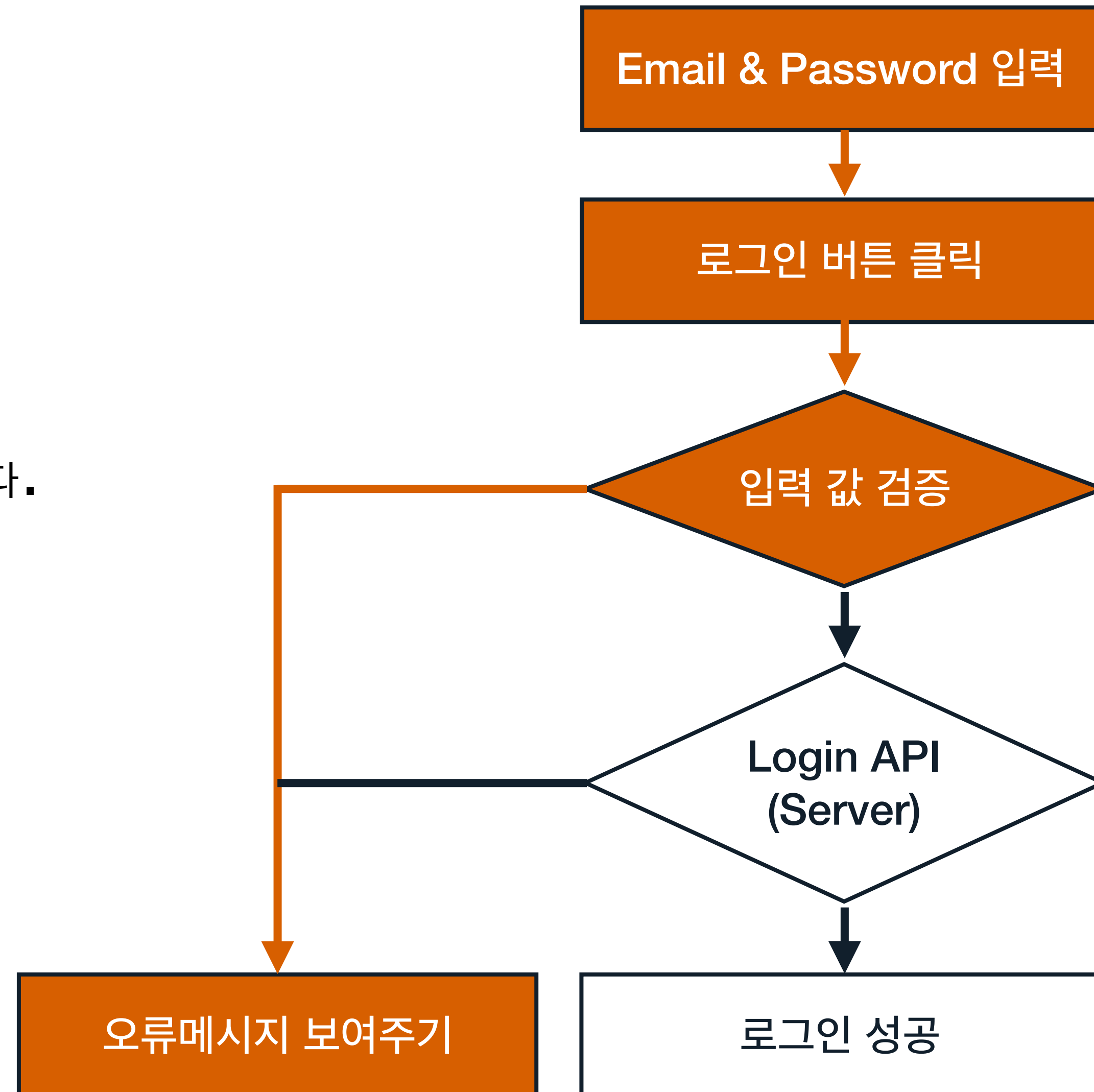
- Given - 특정 상황이 주어지고..
 -> 잘못 된 이메일을 입력한다.
- When - (테스트하려는) 특정 액션이 발생했을 때..
 -> 로그인 버튼을 클릭한다.
- Then - 변화 된 상태나 수행되는 행동을 검증한다.
 -> 이메일 검증 실패 메시지를 보여준다.

TestCase 1 - Scenario - 잘못된 이메일 형식 입력

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```



TestCase 1 - Write test code

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {

}
```

TestCase 1 - Write test code

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {
    when`(view.getInputEmail()).thenReturn("roy.kim@goodoc")
```



```
}
```

TestCase 1 - Write test code

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {
    `when`(view.getInputEmail()).thenReturn("roy.kim@goodoc")

    presenter.onClickEmailLogin()

}
```

TestCase 1 - Write test code

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {
    `when`(view.getInputEmail()).thenReturn("roy.kim@goodoc")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectEmail()
}
```

TestCase 1 - Compile error

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {
    //given
    잘못 된 이메일을 입력한다.

    //when
    이메일 로그인 버튼 클릭한다.

    //then
    유효성 검증 실패 & 이메일 검증 오류 메시지를 보여준다.
}
```

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {
    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectEmail()
}
```

TestCase 1 - Define contract

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectEmail()
}
```

```
class LoginContract {

    interface View {
        fun getInputEmail(): String
        fun showMessageForIncorrectEmail()
    }

    interface Presenter {
        fun onClickEmailLogin()
    }

    interface Repository {
    }
}
```


TestCase 1 - Implement contract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {
    private lateinit var presenter: LoginContract.Presenter

    override fun onCreate(savedInstanceState: Bundle?) {
        (.....)
        emailLoginButton.setOnClickListener {
            presenter.onClickEmailLogin()
        }
    }

    override fun getInputEmail(): String { TODO("not implemented") }

    override fun showMessageForIncorrectEmail() { TODO("not implemented") }
}

class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {
    override fun onClickEmailLogin() {

    }
}
```

TestCase 1 - Run Test - Fail

RED



The screenshot shows the Android Studio test runner interface. At the top, a status bar indicates "Tests failed: 1 of 1 test - 29 ms". Below this, a tree view on the left shows the test hierarchy: "LoginPresenterTest (com.droidknights.tdd)" with a duration of 29 ms, and a sub-item "test_onClickEmailLogin_FailLogin_EnterIncorrectEmail" also with a duration of 29 ms. The main panel on the right displays the failure details in red text:

```
Wanted but not invoked:  
view.showMessageForIncorrectEmail();  
-> at com.droidknights.tdd.LoginPresenterTest.test_onClickEmailLogin_FailLogin_EnterIncorrectEmail()  
Actually, there were zero interactions with this mock.  
  
Wanted but not invoked:  
view.showMessageForIncorrectEmail();
```

Wanted but not invoked:
`view.showMessageForIncorrectEmail();`

TestCase 1 - Run Test - Fail

RED



```
@Test  
fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {  
    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc")  
    presenter.onClickEmailLogin()  
    verify(view).showMessageForIncorrectEmail()  
}
```

TestCase 1 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()

        val patternString = "^[A-Z0-9a-z\\.\\_\\%+\\-]+@[A-Za-z0-9-]+\\.([A-Za-z]{2,})$"
        val pattern = Pattern.compile(patternString)
        val matcher = pattern.matcher(inputEmail)

        if(!matcher.matches()){
            view.showMessageForIncorrectEmail()
            return
        }
    }

    @Test
    fun test_onClickEmailLogin__FailLogin__EnterIncorrectEmail() {

        `when`(view.getInputEmail()).thenReturn("roy.kim@goodoc")

        presenter.onClickEmailLogin()

        verify(view).showMessageForIncorrectEmail()
    }
}
```

TestCase 1 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        val inputEmail = view.getInputEmail()  
  
        val patternString = "[A-Z0-9a-z\\.\\_\\%+\\-]+@[A-Za-z0-9-]+\\.([A-Za-z]{2,})$"
        val pattern = Pattern.compile(patternString)
        val matcher = pattern.matcher(inputEmail)  
  
        if(!matcher.matches()){  
            view.showMessageForIncorrectEmail()  
            return  
        }  
    }  
}
```

TestCase 1 - Run Test - Success



✓

↓^a

↓=

≡

÷

↑

↓

↗

🕒

⚙

✓ Tests passed: 1 of 1 test – 23 ms

▼ ✓ LoginPresenterTest (com.droidknights.tdd)	23 ms	"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home Process finished with exit code 0
✓ test_onClickEmailLogin__FailLogin__EnterIncorrectEmail	23 ms	

TestCase 1 - Refactoring

REFACTOR

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        val inputEmail = view.getInputEmail()  
  
        val patternString = "^[A-Z0-9a-z\\.\\_\\%+\\-]+@[A-Za-z0-9-]+\\.([A-Za-z]{2,})$"   
        val pattern = Pattern.compile(patternString)  
        val matcher = pattern.matcher(inputEmail)  
  
        if(!matcher.matches()){  
            view.showMessageForIncorrectEmail()  
            return  
        }  
    }  
}
```

TestCase 1 - Refactoring

REFACTOR

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()

    }
}
```

```
class Validator {
    companion object {
        fun isValidEmail(email: String): Boolean {
            val patternString = "[A-Z0-9a-z\\.\\_\\%\\+\\-]+@[A-Za-z0-9-]+\\.([A-Za-z]{2,})$"
            val pattern = Pattern.compile(patternString)
            val matcher = pattern.matcher(inputEmail)
            return matcher.matches()
        }
    }
}
```


TestCase 1 - Refactoring

REFACTOR

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()
        if (!Validator.isValidEmail(inputEmail)) {
            view.showMessageForIncorrectEmail()
            return
        }
    }
}
```

```
class Validator {
    companion object {
        fun isValidEmail(email: String): Boolean {
            val patternString = "^[A-Z0-9a-z\\.\\_\\%+-]+@[A-Za-z0-9-]+\\.([A-Za-z]{2,})$"
            val pattern = Pattern.compile(patternString)
            val matcher = pattern.matcher(email)
            return matcher.matches()
        }
    }
}
```

TestCase 1 - Run Test - Success



GREEN

The image shows the Android Studio interface during a test run. At the top, a toolbar contains various icons for development tasks. Below the toolbar, the 'Test Results' pane on the left lists the test suite 'LoginPresenterTest (com.droidknights.tdd)' and its duration '23 ms'. Underneath, a specific test method 'test_onClickEmailLogin__FailLogin__EnterIncorrectEmail' is listed with a duration of '23 ms'. The main 'Output' pane on the right shows the test execution path: `"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home"` and the final status: `Process finished with exit code 0`.

TestCase 2 - Scenario - 잘못된 비밀번호 형식 입력

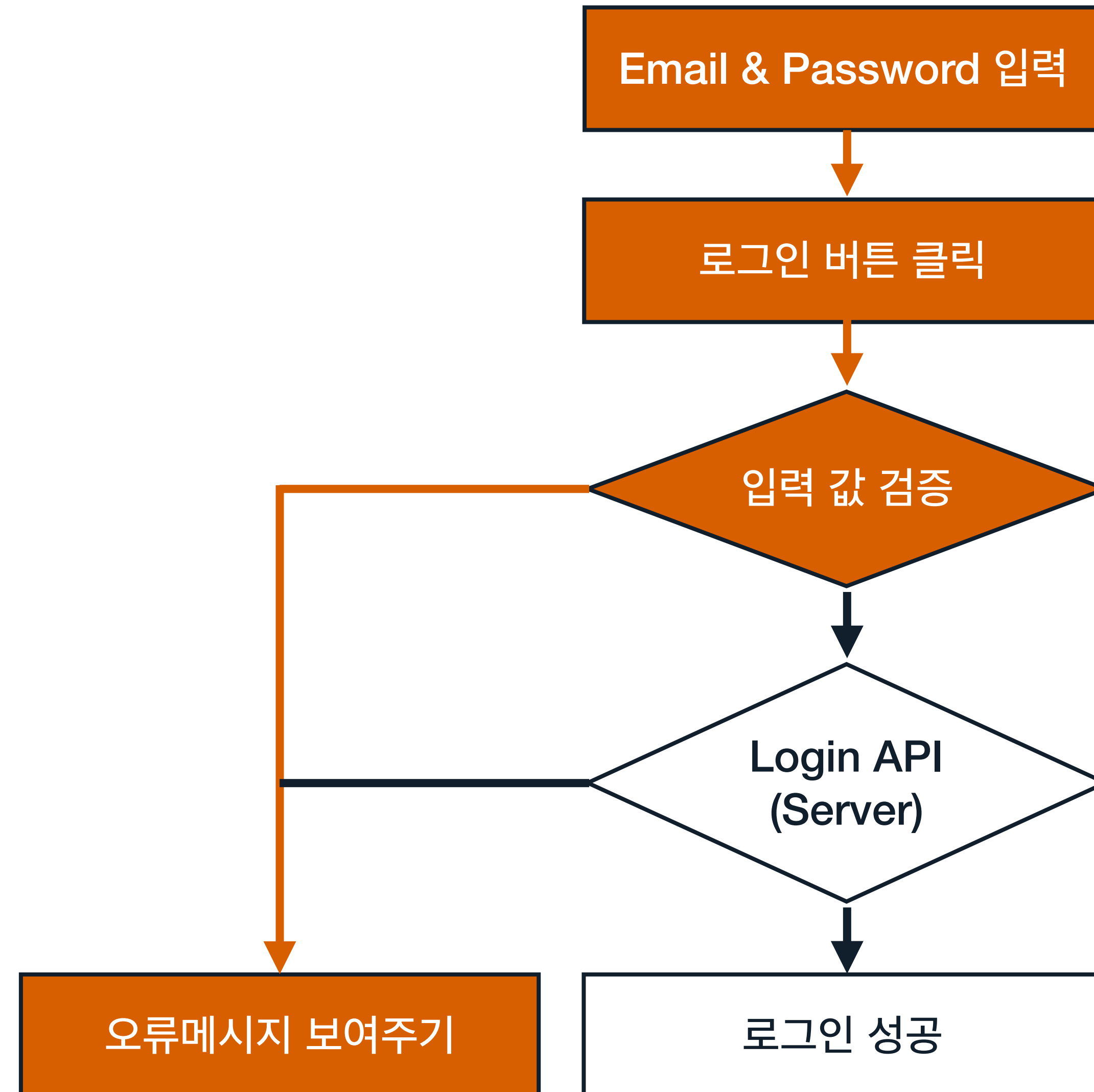
```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {

    올바른 형식의 이메일을 입력한다.
    잘못 된 형식의 비밀번호를 입력한다.

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    비밀번호 검증 오류에 대한 메시지를 보여준다.
}
```



TestCase 2 - Write test code

```
@Test  
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {
```

```
}
```

```
@Test  
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {  
  
    올바른 형식의 이메일을 입력한다.  
    잘못 된 형식의 비밀번호를 입력한다.  
  
    이메일 로그인 버튼 클릭한다.  
  
    입력값을 검증한다.  
  
    비밀번호 검증 오류에 대한 메시지를 보여준다.  
}
```

TestCase 2 - Write test code

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

}
```

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {

    올바른 형식의 이메일을 입력한다.
    잘못 된 형식의 비밀번호를 입력한다.

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    비밀번호 검증 오류에 대한 메시지를 보여준다.
}
```

TestCase 2 - Write test code

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

}
```

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {

    올바른 형식의 이메일을 입력한다.
    잘못 된 형식의 비밀번호를 입력한다.

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    비밀번호 검증 오류에 대한 메시지를 보여준다.

}
```

TestCase 2 - Write test code

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```

```
@Test
fun 이메일로그인_실패케이스_잘못된이메일형식입력() {

    올바른 형식의 이메일을 입력한다.
    잘못 된 형식의 비밀번호를 입력한다.

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    비밀번호 검증 오류에 대한 메시지를 보여준다.
}
```

TestCase 2 - Compile error

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```


TestCase 2 - Define contract

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```

```
class LoginContract {

    interface View {
        fun getInputEmail(): String
        fun showMessageForIncorrectEmail()

    }

    interface Presenter {
        fun onClickEmailLogin()

    }

    interface Repository {

    }

}
```

TestCase 2 - Define contract

```
@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```

```
class LoginContract {

    interface View {
        fun getInputEmail(): String
        fun showMessageForIncorrectEmail()
        fun getInputPassword(): String
        fun showMessageForIncorrectPassword()
    }

    interface Presenter {
        fun onClickEmailLogin()
    }

    interface Repository {
    }
}
```

TestCase 2 - Implement contract

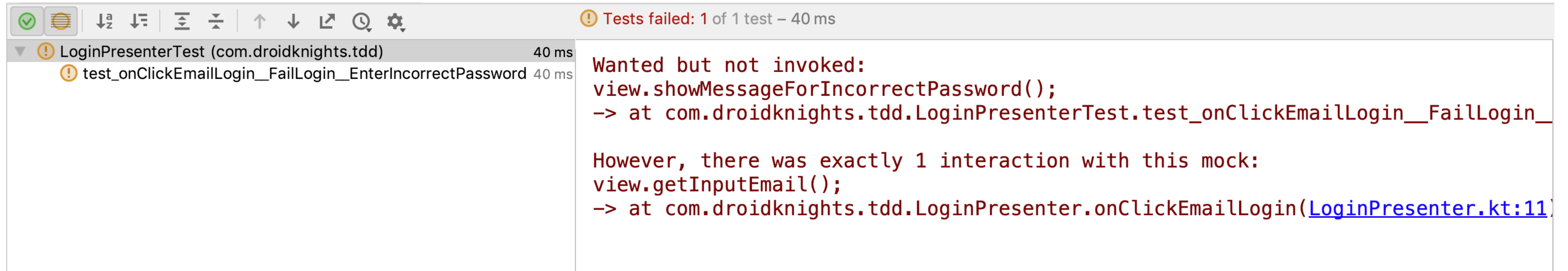
```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    private lateinit var presenter: LoginContract.Presenter  
  
    override fun onCreate(savedInstanceState: Bundle?) {.....}  
  
    override fun getInputEmail(): String {  
        TODO("not implemented")  
    }  
  
    override fun showMessageForIncorrectEmail() {  
        TODO("not implemented")  
    }  
  
}
```

TestCase 2 - Implement contract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    private lateinit var presenter: LoginContract.Presenter  
  
    override fun onCreate(savedInstanceState: Bundle?) {.....}  
  
    override fun getInputEmail(): String {  
        TODO("not implemented")  
    }  
  
    override fun showMessageForIncorrectEmail() {  
        TODO("not implemented")  
    }  
  
    override fun getInputPassword(): String {  
        TODO("not implemented")  
    }  
  
    override fun showMessageForIncorrectPassword() {  
        TODO("not implemented")  
    }  
}
```

TestCase 2 - Run Test - Fail

RED



The screenshot shows the Android Studio test runner interface. At the top, a status bar indicates "Tests failed: 1 of 1 test - 40 ms". Below this, a list of test cases is shown, with the selected test case being "test_onClickEmailLogin__FailLogin__EnterIncorrectPassword" (40 ms). The test result is displayed in a large text area, showing the expected behavior and the actual behavior.

Wanted but not invoked:
view.showMessageForIncorrectPassword();
-> at com.droidknights.tdd.LoginPresenterTest.test_onClickEmailLogin__FailLogin__

However, there was exactly 1 interaction with this mock:
view.getInputEmail();
-> at com.droidknights.tdd.LoginPresenter.onClickEmailLogin(LoginPresenter.kt:11)

Wanted but not invoked:
view.showMessageForIncorrectPassword();

TestCase 2 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()
        if (!Validator.isValidEmail(inputEmail)) {
            view.showMessageForIncorrectEmail()
            return
        }
    }

}

@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when`(view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when`(view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```

TestCase 2 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()
        if (!Validator.isValidEmail(inputEmail)) {
            view.showMessageForIncorrectEmail()
            return
        }

        val inputPassword = view.getInputPassword()
        if (!Validator.isValidPassword(inputPassword)) {
            view.showMessageForIncorrectPassword()
            return
        }
    }
}

@Test
fun test_onClickEmailLogin__FailLogin__EnterIncorrectPassword() {

    `when`(view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when`(view.getInputPassword()).thenReturn("123")

    presenter.onClickEmailLogin()

    verify(view).showMessageForIncorrectPassword()
}
```


TestCase 2 - Run Test - Success



GREEN

The screenshot displays the Android Studio interface during a test run. The top toolbar includes icons for running tests, a list of tests, and various window management tools. The test results pane on the left shows a single test, `LoginPresenterTest (com.droidknights.tdd)`, which passed in 22 ms. Below it, the specific test method `test_onClickEmailLogin_FailLogin_EnterIncorrectPassword` is also shown as passed in 22 ms. The main output pane on the right shows the command `"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" . . .` and the message `Process finished with exit code 0`.

TestCase 2 - Run Test - Success



✓

↓^a

↓_≡

≡

÷

↑

↓

↗

🕒

⚙️

Tests passed: 1 of 1 test – 22 ms

▼

✓ LoginPresenterTest (com.droidknights.tdd) 22 ms

✓ test_onClickEmailLogin_FailLogin_EnterIncorrectPassword 22 ms

"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...

Process finished with exit code 0

✓

↓^a

↓_≡

≡

÷

↑

↓

↗

🕒

⚙️

Tests passed: 2 of 2 tests – 23 ms

▼

✓ LoginPresenterTest (com.droidknights.tdd) 23 ms

✓ test_onClickEmailLogin_FailLogin_EnterIncorrectEmail 23 ms

✓ test_onClickEmailLogin_FailLogin_EnterIncorrectPassword 0 ms

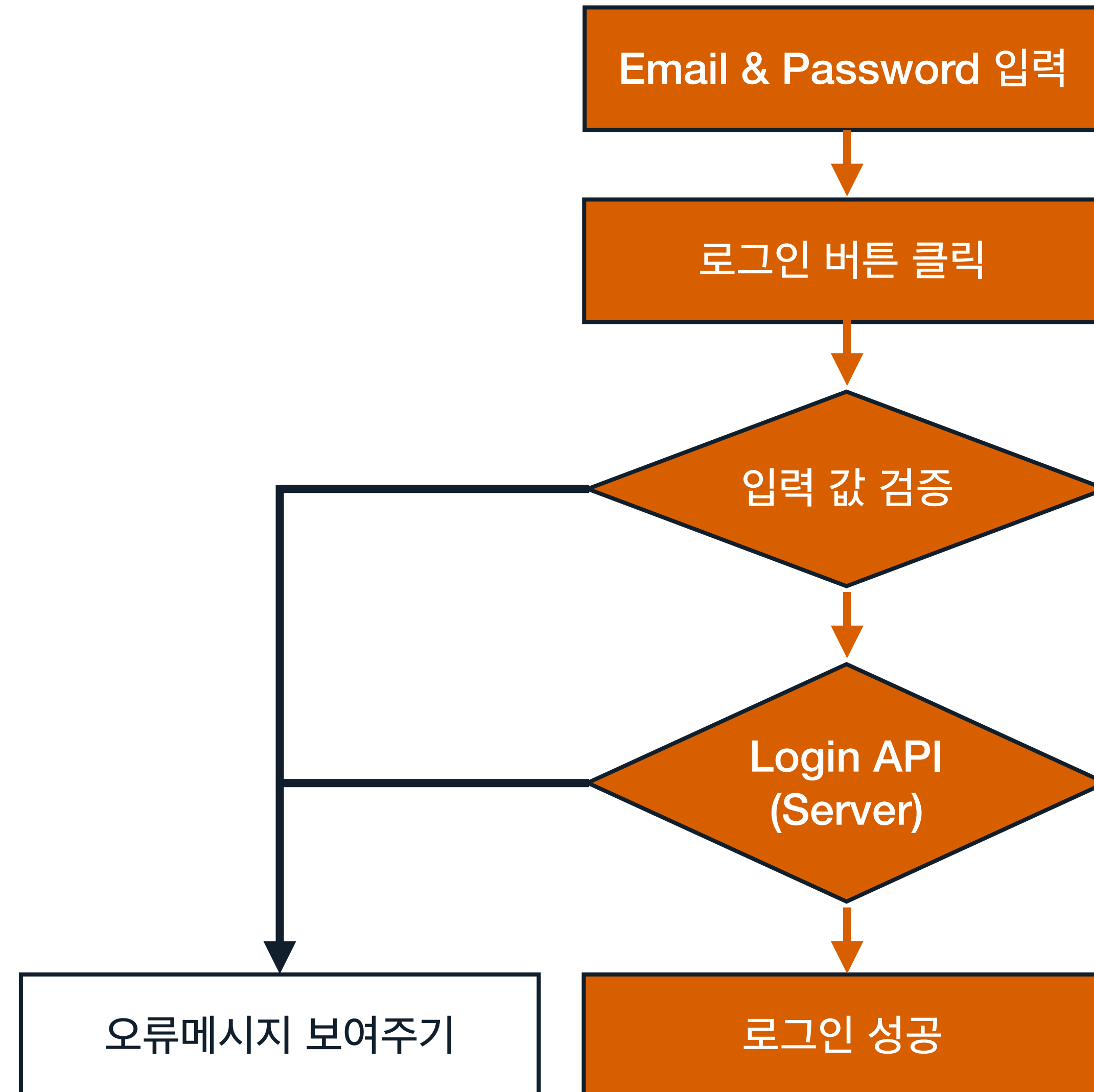
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...

Process finished with exit code 0

TestCase 3 - Scenario - 로그인 성공

@Test

```
fun 이메일로그인_성공() {  
    올바른 형식의 이메일을 입력한다.  
    올바른 형식의 비밀번호를 입력한다.  
  
    이메일 로그인 버튼 클릭한다.  
  
    입력값을 검증한다.  
  
    키보드를 숨긴다.  
    로딩 다이얼로그를 보여준다.  
  
    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.  
    (로그인 성공)  
  
    로딩 다이얼로그를 감춘다.  
    로그인 성공 메시지를 보여준다.  
    로그인 화면을 종료한다.  
}
```



TestCase 3 - Write test code

```
@Test
fun test_onClickEmailLogin__SuccessLogin() {
    올바른 형식의 이메일을 입력한다.
    올바른 형식의 비밀번호를 입력한다.

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    키보드를 숨긴다.
    로딩 다이얼로그를 보여준다.

    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
    (로그인 성공)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```

TestCase 3 - Write test code

```
@Test  
fun test_onClickEmailLogin__SuccessLogin() {
```

```
    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")  
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")
```

이메일 로그인 버튼 클릭한다.

입력값을 검증한다.

키보드를 숨긴다.

로딩 다이얼로그를 보여준다.

서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
(로그인 성공)

로딩 다이얼로그를 감춘다.

로그인 성공 메시지를 보여준다.

로그인 화면을 종료한다.

```
}
```

TestCase 3 - Write test code

```
@Test
fun test_onClickEmailLogin__SuccessLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    이메일 로그인 버튼 클릭한다.

    입력값을 검증한다.

    키보드를 숨긴다.
    로딩 다이얼로그를 보여준다.

    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
    (로그인 성공)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```

TestCase 3 - Write test code

@Test

fun test_onClickEmailLogin__SuccessLogin() {

 `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")

 `when` (view.getInputPassword()).thenReturn("@Abcd1234")

presenter.onClickEmailLogin()

 입력값을 검증한다.

 키보드를 숨긴다.

 로딩 다이얼로그를 보여준다.

 서버에 아이디/비밀번호를 전달해서 로그인 처리한다.

 (로그인 성공)

 로딩 다이얼로그를 감춘다.

 로그인 성공 메시지를 보여준다.

 로그인 화면을 종료한다.

}

TestCase 3 - Write test code

```
@Test
fun test_onClickEmailLogin__SuccessLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    입력값을 검증한다.

    키보드를 숨긴다.
    로딩 다이얼로그를 보여준다.

    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
    (로그인 성공)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```

TestCase 3 - Write test code

```
@Test
fun test_onClickEmailLogin__SuccessLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
    (로그인 성공)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```


TestCase 3 - Write test code

```
@Test
fun test_onClickEmailLogin__SuccessLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.
    (로그인 성공)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```

TestCase 3 - Write test code

```
@Captor
private lateinit var loginResultListener: ArgumentCaptor<LoginResultCallback>

@Test
fun test_onClickEmailLogin__SuccessLogin() {
    val mockUserInfo = mock<UserInfo> { }

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onSuccess(mockUserInfo)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
```

```
}
```

TestCase 3 - Write test code

```
@Captor
private lateinit var loginResultListener: ArgumentCaptor<LoginResultCallback>

@Test
fun test_onClickEmailLogin__SuccessLogin() {
    val mockUserInfo = mock<UserInfo> { }

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onSuccess(mockUserInfo)

    로딩 다이얼로그를 감춘다.
    로그인 성공 메시지를 보여준다.
    로그인 화면을 종료한다.
}
```

TestCase 3 - Write test code

```
@Captor
private lateinit var loginResultListener: ArgumentCaptor<LoginResultCallback>

@Test
fun test_onClickEmailLogin__SuccessLogin() {
    val mockUserInfo = mock<UserInfo> { }

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onSuccess(mockUserInfo)

    inOrder.verify(view).hideLoadingDialog()
    inOrder.verify(view).showMessageForSuccessLogin()
    inOrder.verify(view).finishActivity()
}
```

TestCase 3 - Write test code

```
@Captor
private lateinit var loginResultListener: ArgumentCaptor<LoginResultCallback>

@Test
fun test_onClickEmailLogin__SuccessLogin() {
    val mockUserInfo = mock<UserInfo> { }

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onSuccess(mockUserInfo)

    inOrder.verify(view).hideLoadingDialog()
    inOrder.verify(view).showMessageForSuccessLogin()
    inOrder.verify(view).finishActivity()
}
```

TestCase 3 - Compile error

```
@Captor
private lateinit var loginResultListener: ArgumentCaptor<LoginResultCallback>

@Test
fun test_onClickEmailLogin__SuccessLogin() {
    val mockUserInfo = mock<UserInfo> { }

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")

    presenter.onClickEmailLogin()

    inOrder.verify(view).hideSoftKeyboard()
    inOrder.verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onSuccess(mockUserInfo)

    inOrder.verify(view).hideLoadingDialog()
    inOrder.verify(view).showMessageForSuccessLogin()
    inOrder.verify(view).finishActivity()
}
```

TestCase 3 - Define contract

```
class LoginContract {  
    interface View {  
        fun getInputEmail(): String  
        fun showMessageForIncorrectEmail()  
        fun getInputPassword(): String  
        fun showMessageForIncorrectPassword()  
        fun hideSoftKeyboard()  
        fun showLoadingDialog()  
        fun hideLoadingDialog()  
        fun showMessageForSuccessLogin()  
        fun finishActivity()  
    }  
  
    interface Presenter {  
        fun onClickEmailLogin()  
    }  
  
    interface Repository {  
        fun login(email: String, password: String, callback: LoginResultCallback)  
    }  
}
```

```
interface LoginResultCallback {  
    fun onSuccess(userInfo: UserInfo)  
    fun onFail(code: Int, message: String)  
}
```

TestCase 3 - Implement contract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    (.....)
```

```
}
```


TestCase 3 - Implement contract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    (.....)  
  
    override fun hideSoftKeyboard() {  
        TODO("not implemented")  
    }  
  
    override fun showLoadingDialog() {  
        TODO("not implemented")  
    }  
  
    override fun hideLoadingDialog() {  
        TODO("not implemented")  
    }  
  
    override fun showMessageForSuccessLogin() {  
        TODO("not implemented")  
    }  
  
    override fun finishActivity() {  
        TODO("not implemented")  
    }  
}
```

TestCase 3 - Implement contract

```
class LoginRepository : LoginContract.Repository {
```

}

TestCase 3 - Implement contract

```
class LoginRepository : LoginContract.Repository {  
    override fun login(email: String, password: String, callback: LoginResultCallback) {  
        TODO("not implemented")  
    }  
}
```

TestCase 3 - Run Test - Fail

RED

▼ ⓘ LoginPresenterTest (com.droidknights.tdd)	48 ms	
ⓘ test_onClickEmailLogin__SuccessLogin	48 ms	<p>Wanted but not invoked: view.hideSoftKeyboard(); -> at com.droidknights.tdd.LoginPresenterTest.test_onClickEmailLogin__SuccessLogin(LoginPresenter.kt:11)</p> <p>However, there were exactly 4 interactions with this mock: view.getInputEmail(); -> at com.droidknights.tdd.LoginPresenter.onClickEmailLogin(LoginPresenter.kt:11)</p> <p>view.getInputPassword();</p>

Wanted but not invoked:
view.hideSoftKeyboard();

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        (... 이메일/비밀번호 입력 값 검증...)  
  
    }  
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        (... 이메일/비밀번호 입력 값 검증...)  
  
        inOrder.verify(view).hideSoftKeyboard()  
        inOrder.verify(view).showLoadingDialog()  
  
        val email = view.getInputEmail()  
        val password = view.getInputPassword()  
  
        inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())  
        loginResultListener.firstValue.onSuccess(mockUserInfo)  
  
        inOrder.verify(view).hideLoadingDialog()  
        inOrder.verify(view).showMessageForSuccessLogin()  
        inOrder.verify(view).finishActivity()  
    }  
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        (... 이메일/비밀번호 입력 값 검증...)  
  
        inOrder.verify(view).hideSoftKeyboard()  
        inOrder.verify(view).showLoadingDialog()  
  
        val email = view.getInputEmail()  
        val password = view.getInputPassword()  
  
        inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())  
        loginResultListener.firstValue.onSuccess(mockUserInfo)  
  
        inOrder.verify(view).hideLoadingDialog()  
        inOrder.verify(view).showMessageForSuccessLogin()  
        inOrder.verify(view).finishActivity()  
    }  
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,  
                    val repository: LoginContract.Repository) : LoginContract.Presenter {  
  
    override fun onClickEmailLogin() {  
        (... 이메일/비밀번호 입력 값 검증...)  
  
        view.hideSoftKeyboard()  
        view.showLoadingDialog()  
  
        val email = view.getInputEmail()  
        val password = view.getInputPassword()  
  
        inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())  
        loginResultListener.firstValue.onSuccess(mockUserInfo)  
  
        inOrder.verify(view).hideLoadingDialog()  
        inOrder.verify(view).showMessageForSuccessLogin()  
        inOrder.verify(view).finishActivity()  
    }  
}
```


TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        val email = view.getInputEmail()
        val password = view.getInputPassword()

        inOrder.verify(repository).login(eq(email), eq(password), loginResultListener.capture())
        loginResultListener.firstValue.onSuccess(mockUserInfo)

        inOrder.verify(view).hideLoadingDialog()
        inOrder.verify(view).showMessageForSuccessLogin()
        inOrder.verify(view).finishActivity()
    }
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        val email = view.getInputEmail()
        val password = view.getInputPassword()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                inOrder.verify(view).hideLoadingDialog()
                inOrder.verify(view).showMessageForSuccessLogin()
                inOrder.verify(view).finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                TODO("not implemented")
            }
        })
    }
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        val email = view.getInputEmail()
        val password = view.getInputPassword()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                inOrder.verify(view).hideLoadingDialog()
                inOrder.verify(view).showMessageForSuccessLogin()
                inOrder.verify(view).finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                TODO("not implemented")
            }
        })
    }
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        val email = view.getInputEmail()
        val password = view.getInputPassword()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                view.hideLoadingDialog()
                view.showMessageForSuccessLogin()
                view.finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                TODO("not implemented")
            }
        })
    }
}
```

TestCase 3 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        val email = view.getInputEmail()
        val password = view.getInputPassword()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                view.hideLoadingDialog()
                view.showMessageForSuccessLogin()
                view.finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                TODO("not implemented")
            }
        })
    }
}
```

TestCase 3 - Run Test - Success



✓

↓^a

↓≡

≡

÷

↑

↓

↗

🕒

⚙️

✓ Tests passed: 1 of 1 test – 44 ms

▼ ✓ LoginPresenterTest (com.droidknights.tdd)	44 ms	<pre>"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ... Process finished with exit code 0</pre>
✓ test_onClickEmailLogin__SuccessLogin	44 ms	



TestCase 3 - Run Test - Success



✓

↓^a

↓

≡

÷

↑

↓

↗

🕒

⚙️

✓ Tests passed: 1 of 1 test – 44 ms

▼ ✓ LoginPresenterTest (com.droidknights.tdd)	44 ms	<code>"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...</code> Process finished with exit code 0
✓ test_onClickEmailLogin__SuccessLogin	44 ms	

✓

↓^a

↓

≡

÷

↑

↓

↗

🕒

⚙️

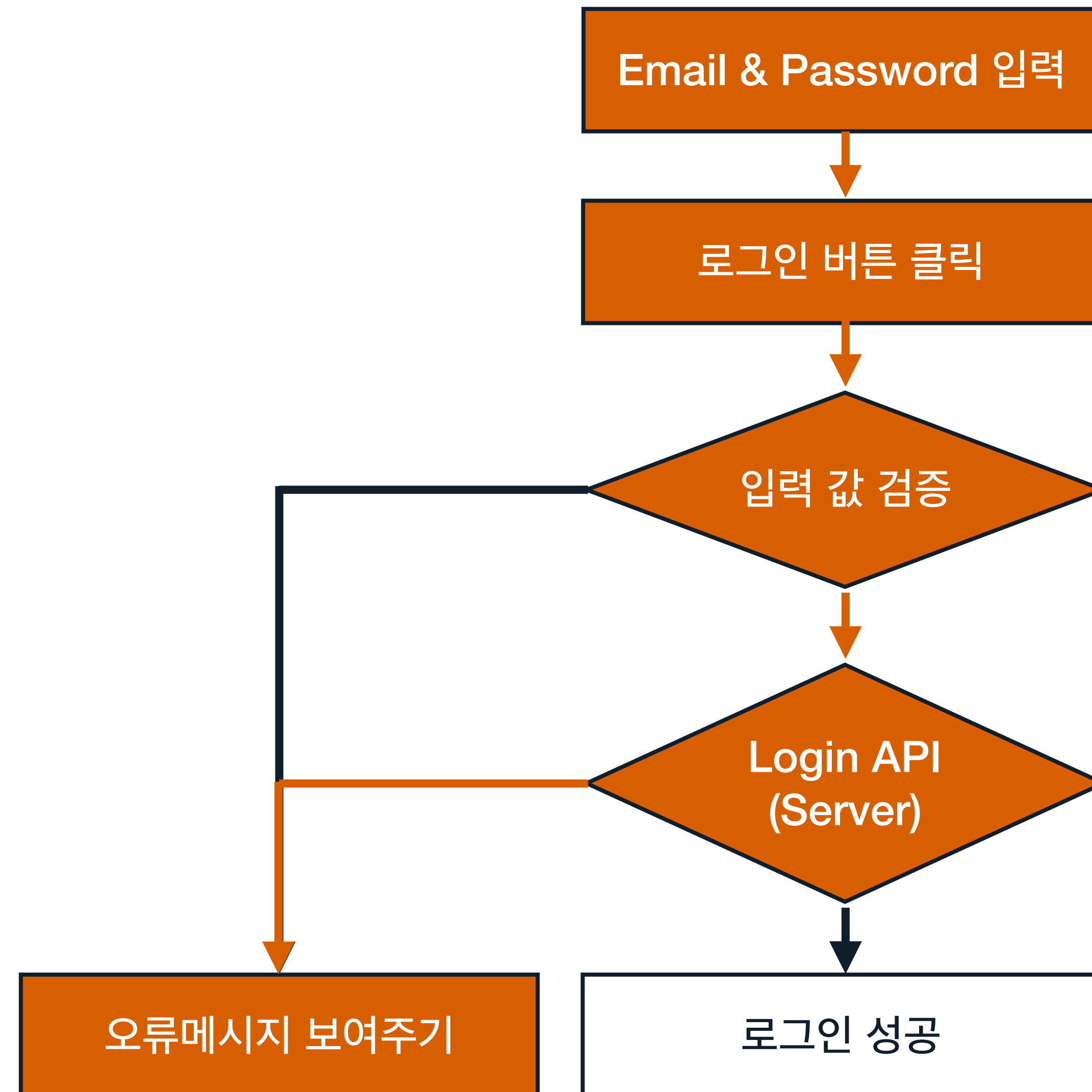
✓ Tests passed: 3 of 3 tests – 45 ms

▼ ✓ LoginPresenterTest (com.droidknights.tdd)	45 ms	<code>"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...</code> Process finished with exit code 0
✓ test_onClickEmailLogin__FailLogin__EnterIncorrectEmail	22 ms	
✓ test_onClickEmailLogin__SuccessLogin	23 ms	
✓ test_onClickEmailLogin__FailLogin__EnterIncorrectPassword	0 ms	

TestCase 4 - Scenario - 로그인 실패

@Test

```
fun 이메일로그인_실패_가입되지않은회원정보() {  
    올바른 형식의 이메일을 입력한다.  
    올바른 형식의 비밀번호를 입력한다.  
  
    이메일 로그인 버튼 클릭한다.  
  
    입력값을 검증한다.  
  
    키보드를 숨긴다.  
    로딩 다이얼로그를 보여준다.  
  
    서버에 아이디/비밀번호를 전달해서 로그인 처리한다.  
    (로그인 실패 - 서버 오류)  
  
    로딩 다이얼로그를 감춘다.  
    로그인 실패 메시지를 보여준다.  
}
```



TestCase 4 - Write test code

```
@Test
fun test_onClickEmailLogin__FailLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")
    val errorCode = 1021
    val errorMsg = "Server error message."

    presenter.onClickEmailLogin()

    verify(view).hideSoftKeyboard()
    verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onFail(errorCode, errorMsg)

    verify(view).hideLoadingDialog()
    verify(view).showMessageForFailLogin(errorMsg)
}
```

TestCase 4 - Compile error

```
@Test
fun test_onClickEmailLogin__FailLogin() {

    `when` (view.getInputEmail()).thenReturn("roy.kim@goodoc.co.kr")
    `when` (view.getInputPassword()).thenReturn("@Abcd1234")
    val errorCode = 1021
    val errorMsg = "Server error message."

    presenter.onClickEmailLogin()

    verify(view).hideSoftKeyboard()
    verify(view).showLoadingDialog()

    val email = view.getInputEmail()
    val password = view.getInputPassword()

    verify(repository).login(eq(email), eq(password), loginResultListener.capture())
    loginResultListener.firstValue.onFail(errorCode, errorMsg)

    verify(view).hideLoadingDialog()
    verify(view).showMessageForFailLogin(errorMsg)
}
```

TestCase 4 - Define contract

```
class LoginContract {  
  
    interface View {  
        fun getInputEmail(): String  
        fun showMessageForIncorrectEmail()  
        fun getInputPassword(): String  
        fun showMessageForIncorrectPassword()  
        fun hideSoftKeyboard()  
        fun showLoadingDialog()  
        fun hideLoadingDialog()  
        fun showMessageForSuccessLogin()  
        fun finishActivity()  
        fun showMessageForFailLogin(errorMsg: String)  
    }  
  
    interface Presenter {  
        fun onClickEmailLogin()  
    }  
  
    interface Repository {  
        fun login(email: String, password: String, callback: LoginResultCallback)  
    }  
}
```

TestCase 4 - Implement contract

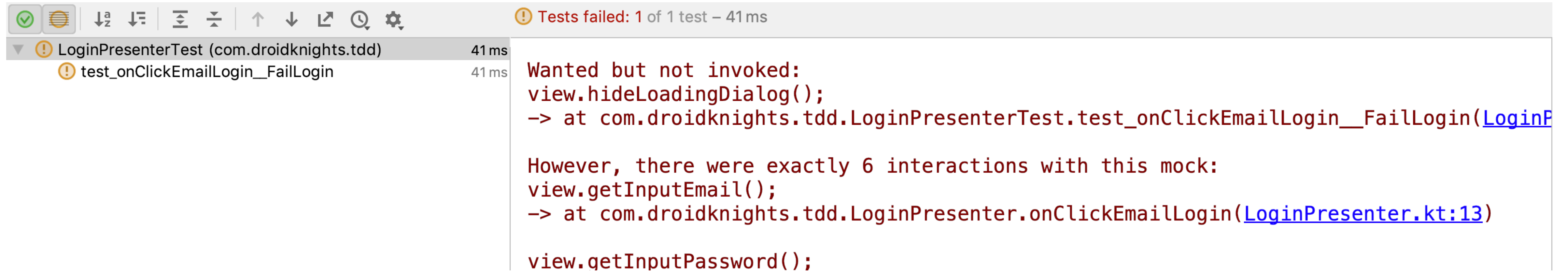
```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    (.....)  
  
}
```

TestCase 4 - Implement contract

```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
    (.....)  
  
    override fun showMessageForFailLogin(errorMsg: String) {  
        TODO("not implemented")  
    }  
}
```

TestCase 4 - Run Test - Fail

RED



The screenshot shows the Android Studio test runner interface. At the top, a status bar indicates "Tests failed: 1 of 1 test – 41 ms". Below this, a list of test cases is shown. The first test case, "LoginPresenterTest (com.droidknights.tdd)", is expanded, revealing a sub-test case "test_onClickEmailLogin__FailLogin" which has failed. The failure message is displayed in a large font, stating: "Wanted but not invoked: view.hideLoadingDialog(); -> at com.droidknights.tdd.LoginPresenterTest.test_onClickEmailLogin__FailLogin(LoginP...". It then provides context: "However, there were exactly 6 interactions with this mock: view.getInputEmail(); -> at com.droidknights.tdd.LoginPresenter.onClickEmailLogin(LoginPresenter.kt:13) view.getInputPassword();".

Tests failed: 1 of 1 test – 41 ms

▼ ! LoginPresenterTest (com.droidknights.tdd) 41 ms

! test_onClickEmailLogin__FailLogin 41 ms

Wanted but not invoked:
view.hideLoadingDialog();
-> at com.droidknights.tdd.LoginPresenterTest.test_onClickEmailLogin__FailLogin(LoginP

However, there were exactly 6 interactions with this mock:
view.getInputEmail();
-> at com.droidknights.tdd.LoginPresenter.onClickEmailLogin(LoginPresenter.kt:13)

view.getInputPassword();

Wanted but not invoked:
view.hideLoadingDialog();

TestCase 4 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {

        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                view.hideLoadingDialog()
                view.showMessageForSuccessLogin()
                view.finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                TODO("not implemented")
            }
        })
    }
}
```

TestCase 4 - Write only enough code

```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {

        (... 이메일/비밀번호 입력 값 검증...)

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                view.hideLoadingDialog()
                view.showMessageForSuccessLogin()
                view.finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                view.hideLoadingDialog()
                view.showMessageForFailLogin(message)
            }
        })
    }
}
```


TestCase 4 - Run Test - Success



✓

☰

↓^a

↓_≡

≡

÷

↑

↓

↗

🕒

⚙

✓ Tests passed: 1 of 1 test – 26 ms

▼ ✓ LoginPresenterTest (com.droidknights.tdd)	26 ms	<code>"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...</code> <code>Process finished with exit code 0</code>
✓ test_onClickEmailLogin__FailLogin	26 ms	

TestCase 4 - Run Test - Success



GREEN

The image shows the Android Studio interface during a test run. At the top, a toolbar includes icons for running, debugging, and testing. The test results pane on the left shows a successful test run for 'LoginPresenterTest (com.droidknights.tdd)' with a duration of 26 ms. Below it, the sub-test 'test_onClickEmailLogin_FailLogin' is also shown with a duration of 26 ms. The main output pane on the right displays the command path: '/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java' followed by 'Process finished with exit code 0'.

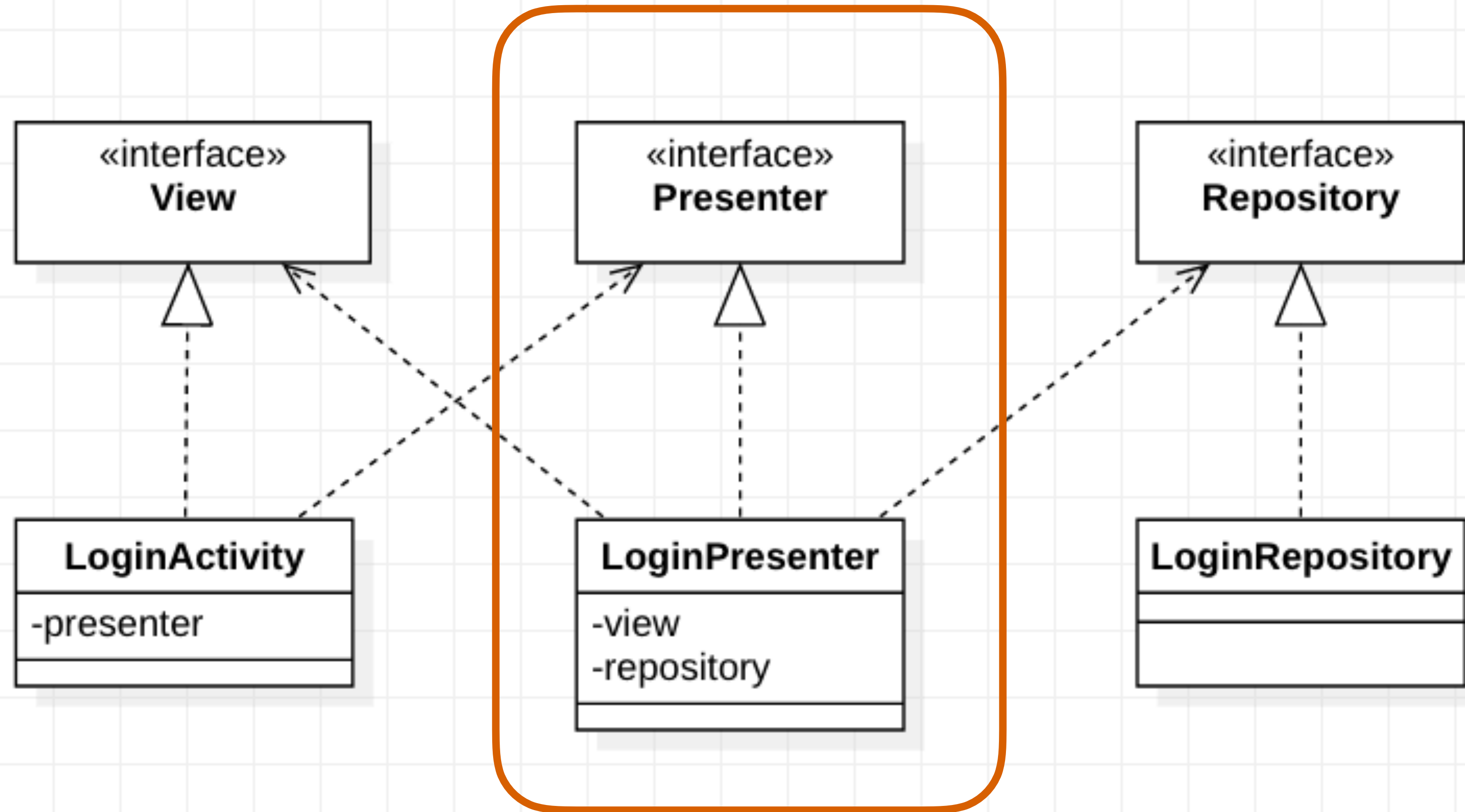
The screenshot displays the Android Studio interface during a test run. The top toolbar includes icons for running, debugging, and other development tools. The top status bar indicates "Tests passed: 4 of 4 tests - 48 ms". The left sidebar shows a tree view of the test results for the package `com.droidknights.tdd`. The test `LoginPresenterTest` is expanded, showing four sub-tests, all of which passed successfully. The right pane shows the output of the test run, which includes the Java command used to execute the tests and a confirmation that the process finished with exit code 0.

Test Name	Duration
<code>LoginPresenterTest (com.droidknights.tdd)</code>	48 ms
<code>test_onClickEmailLogin__FailLogin__EnterIncorrectEmail</code>	23 ms
<code>test_onClickEmailLogin__SuccessLogin</code>	23 ms
<code>test_onClickEmailLogin__FailLogin</code>	1 ms
<code>test_onClickEmailLogin__FailLogin__EnterIncorrectPassword</code>	1 ms

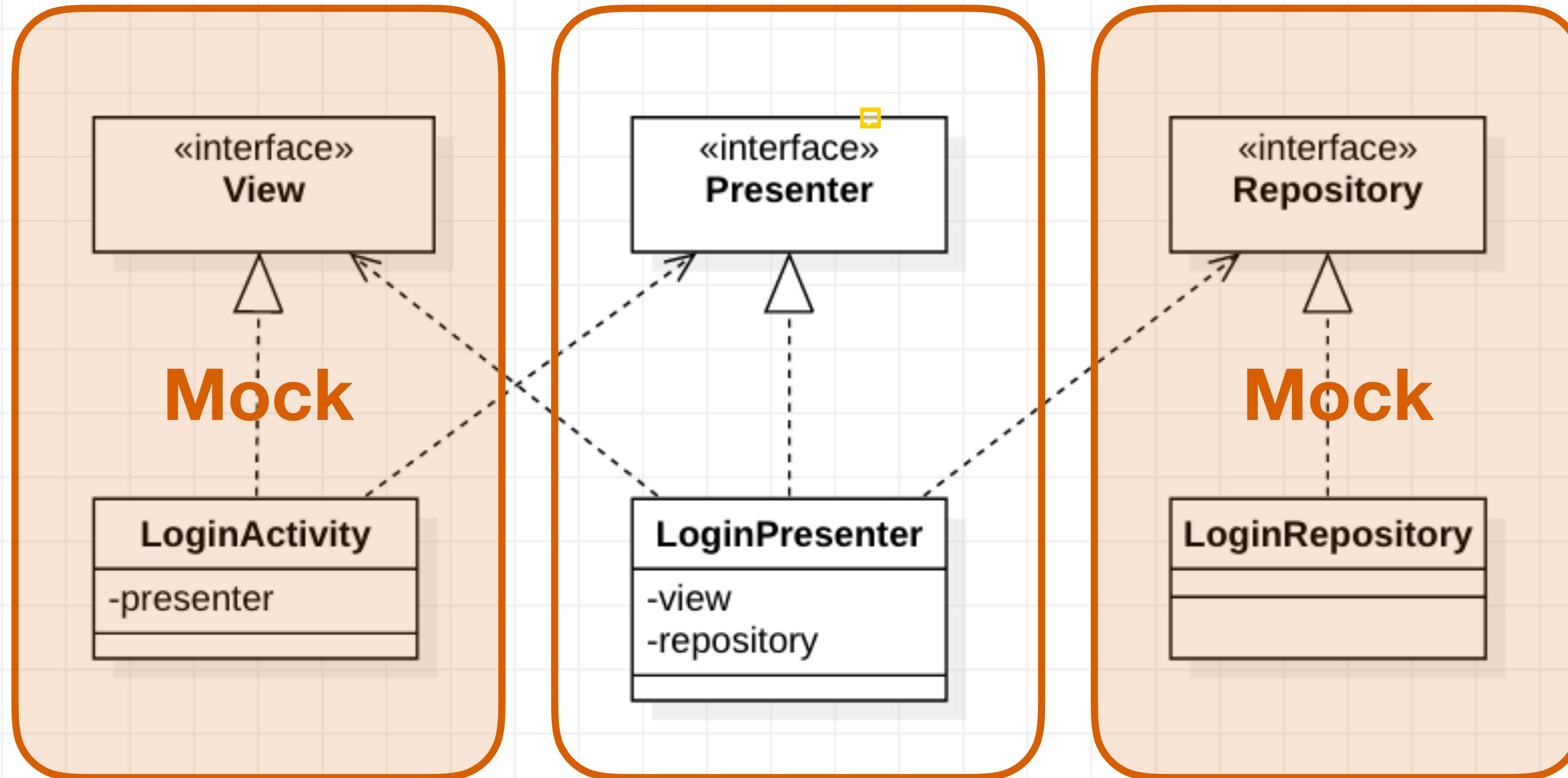
Output:

```
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...  
  
Process finished with exit code 0
```

Complete logic with TDD



Complete logic with TDD



```
class LoginPresenter(val view: LoginContract.View,
                    val repository: LoginContract.Repository) : LoginContract.Presenter {

    override fun onClickEmailLogin() {
        val inputEmail = view.getInputEmail()
        if (!Validator.isValidEmail(inputEmail)) {
            view.showMessageForIncorrectEmail()
            return
        }

        val inputPassword = view.getInputPassword()
        if (!Validator.isValidPassword(inputPassword)) {
            view.showMessageForIncorrectPassword()
            return
        }

        view.hideSoftKeyboard()
        view.showLoadingDialog()

        repository.login(inputEmail, inputPassword, object : LoginResultCallback {
            override fun onSuccess(userInfo: UserInfo) {
                view.hideLoadingDialog()
                view.showMessageForSuccessLogin()
                view.finishActivity()
            }

            override fun onFail(code: Int, message: String) {
                view.hideLoadingDialog()
                view.showMessageForFailLogin(message)
            }
        })
    }
}
```



```
class LoginRepository : LoginContract.Repository {  
    override fun login(email: String, password: String, callback: LoginResultCallback) {  
        TODO("not implemented")  
    }  
}
```

```
class LoginRepository : LoginContract.Repository {  
    override fun login(email: String, password: String, callback: LoginResultCallback) {  
        val client = OkHttpClient()  
        val request = // TODO build POST request...  
        client.newCall(request).execute()  
    }  
}
```




```
class LoginActivity : AppCompatActivity(), LoginContract.View {

    private lateinit var presenter: LoginContract.Presenter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        presenter = LoginPresenter(this, Injector.provideLoginRepository())

        textViewEmailLogin.setOnClickListener {
            presenter.onClickEmailLogin()
        }
    }

    override fun getInputEmail(): String { TODO("not implemented") }

    override fun showMessageForIncorrectEmail() { TODO("not implemented") }

    override fun getInputPassword(): String { TODO("not implemented") }

    override fun showMessageForIncorrectPassword() { TODO("not implemented") }

    override fun hideSoftKeyboard() { TODO("not implemented") }

    override fun showLoadingDialog() { TODO("not implemented") }

    override fun hideLoadingDialog() { TODO("not implemented") }

    override fun showMessageForSuccessLogin() { TODO("not implemented") }

    override fun finishActivity() { TODO("not implemented") }

    override fun showMessageForFailLogin(errorMsg: String) { TODO("not implemented") }
}
```



```
class LoginActivity : AppCompatActivity(), LoginContract.View {

    private lateinit var presenter: LoginContract.Presenter

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

    override fun getInputEmail() = editTextEmail.text.toString()

    override fun showMessageForIncorrectEmail() { showToast(R.string.msg_for_incorrect_email) }

    override fun getInputPassword() = editTextPassword.text.toString()

    override fun showMessageForIncorrectPassword() { showToast(R.string.msg_for_incorrect_password) }

    override fun hideSoftKeyboard() { KeyboardManager.hideKeyboard(editTextEmail) }

    override fun showLoadingDialog() { ProgressDialog(this).show() }

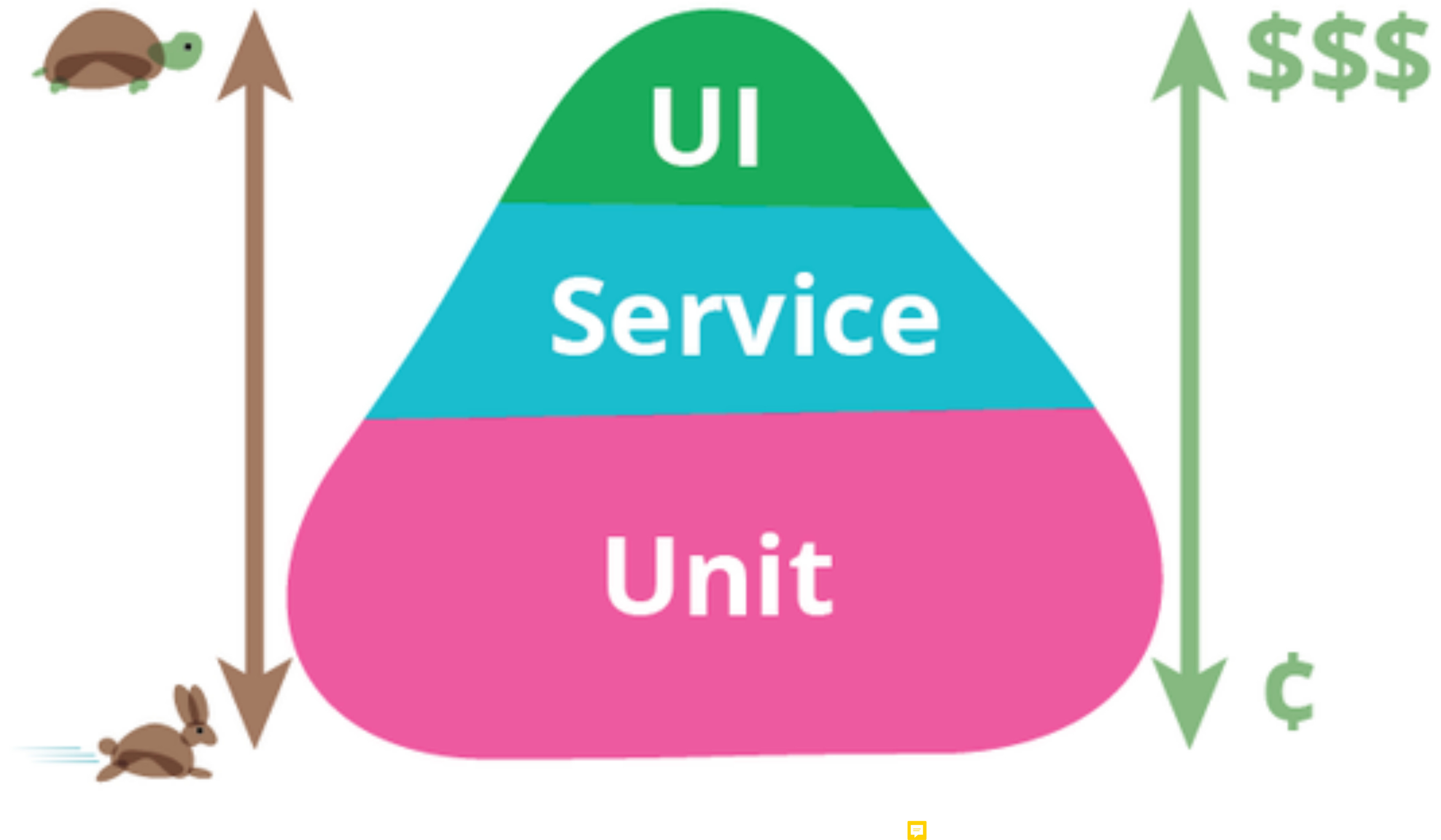
    override fun hideLoadingDialog() { ProgressDialog(this).hide() }

    override fun showMessageForSuccessLogin() { showToast(R.string.msg_for_success_login) }

    override fun finishActivity() { finish() }

    override fun showMessageForFailLogin(errorMsg: String) { showToast(R.string.msg_for_fail_login) }

    private fun showToast(@StringRes resId: Int) {
        Toast.makeText(this, resId, Toast.LENGTH_SHORT).show()
    }
}
```



Write UI Test with Espresso



Espresso

UI test code structure

1. Find a view

onView(Matcher)

2. Perform an action

.perform(ViewAction)

.check(ViewAssertion)

3. Inspect the result

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        화면에 Email 입력창이 보이는지 확인한다.  
        화면에 Password 입력창이 보이는지 확인한다.  
  
        Email 입력창에 회원 이메일을 입력한다.  
        Password 입력창에 회원 비밀번호를 입력한다.  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        화면에 Email 입력창이 보이는지 확인한다.  
        화면에 Password 입력창이 보이는지 확인한다.  
  
        Email 입력창에 회원 이메일을 입력한다.  
        Password 입력창에 회원 비밀번호를 입력한다.  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        Email 입력창에 회원 이메일을 입력한다.  
        Password 입력창에 회원 비밀번호를 입력한다.  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        Email 입력창에 회원 이메일을 입력한다.  
        Password 입력창에 회원 비밀번호를 입력한다.  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```


UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))  
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))  
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))  
  
        로그인 버튼을 클릭한다.  
  
        로그인이 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))  
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))  
  
        onView(withId(R.id.emailLoginButton)).perform(click())  
  
        로그인에 성공했다는 메시지를 보여준다.  
    }  
}
```

UI Test Scenario

```
class LoginActivityTest {  
  
    @get:Rule  
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)  
  
    @Test  
    fun test_SuccessEmailLogin() {  
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))  
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))  
  
        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))  
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))  
  
        onView(withId(R.id.emailLoginButton)).perform(click())  
  
        로그인에 성공했다는 메시지를 보여준다.  
    }  
}
```

Write UI test code

```
class LoginActivityTest {

    @get:Rule
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)

    @Test
    fun test_SuccessEmailLogin() {
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))

        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))

        onView(withId(R.id.emailLoginButton)).perform(click())

        onView(withText(R.string.msg_for_success_login)).inRoot(ToastMatcher())
        .check(matches(isDisplayed()))
    }
}
```

```
class LoginActivityTest {

    @get:Rule
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)

    @Test
    fun test_SuccessEmailLogin() {
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))

        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))

        onView(withId(R.id.emailLoginButton)).perform(click())

        onView(withText(R.string.msg_for_success_login)).inRoot(ToastMatcher()).check(matches(isDisplayed()))
    }

    @Test
    fun test_FailEmailLogin() {
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))

        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1"))

        onView(withId(R.id.emailLoginButton)).perform(click())

        onView(withText(R.string.msg_for_fail_login)).inRoot(ToastMatcher()).check(matches(isDisplayed()))
    }
}
```

```
class LoginActivityTest {

    @get:Rule
    val loginActivityRule = ActivityTestRule(LoginActivity::class.java, true, true)

    @Test
    fun test_SuccessEmailLogin() {
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))

        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1234"))

        onView(withId(R.id.emailLoginButton)).perform(click())

        onView(withText(R.string.msg_for_success_login)).inRoot(ToastMatcher()).check(matches(isDisplayed()))
    }

    @Test
    fun test_FailEmailLogin() {
        onView(withId(R.id.editTextEmail)).check(matches(isDisplayed()))
        onView(withId(R.id.editTextPassword)).check(matches(isDisplayed()))

        onView(withId(R.id.editTextEmail)).perform(typeText("roy.kim@goodoc.co.kr"))
        onView(withId(R.id.editTextPassword)).perform(typeText("@Abcd1"))

        onView(withId(R.id.emailLoginButton)).perform(click())

        onView(withText(R.string.msg_for_fail_login)).inRoot(ToastMatcher()).check(matches(isDisplayed()))
    }
}
```


Testing

Test Complete

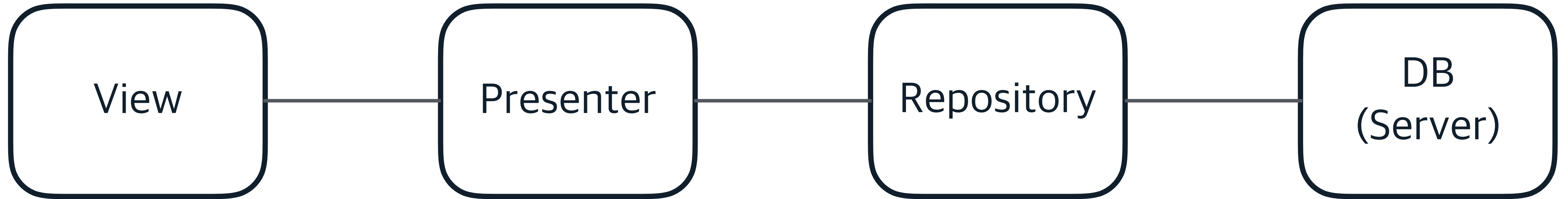
Real Object

View

Presenter

Repository

DB
(Server)



로그인 성공

ID : roy.kim@goodoc.co.kr
PW : @Abcd1234

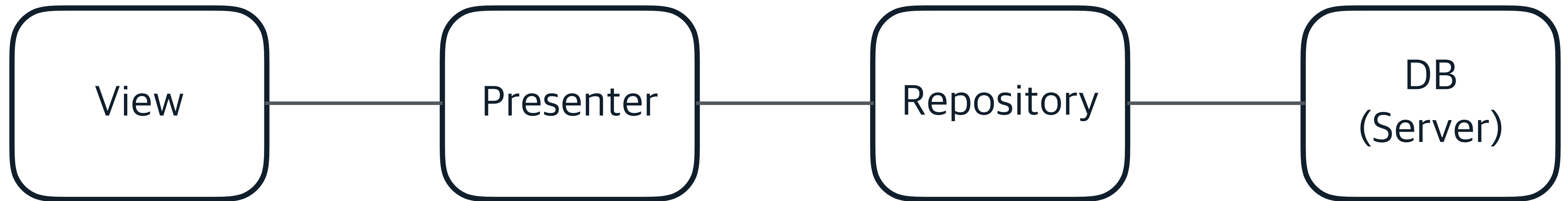
로그인 실패

ID : roy.kim@goodoc.co.kr
PW : @Abcd1

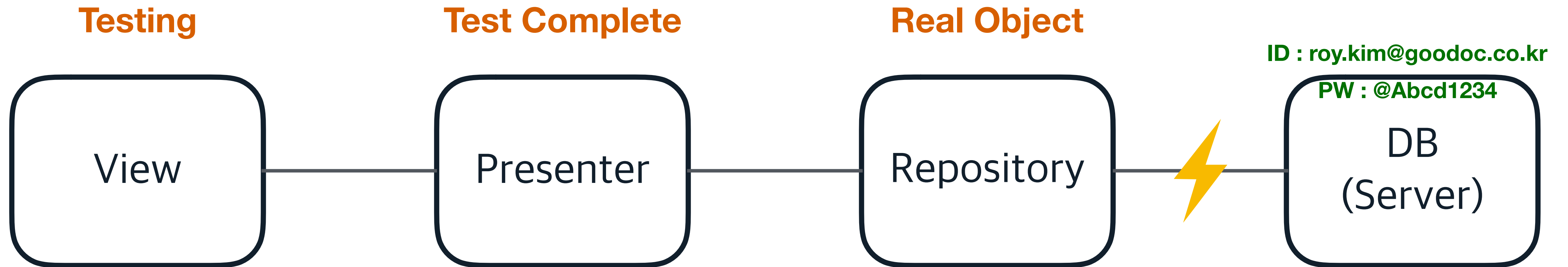
Testing

Test Complete

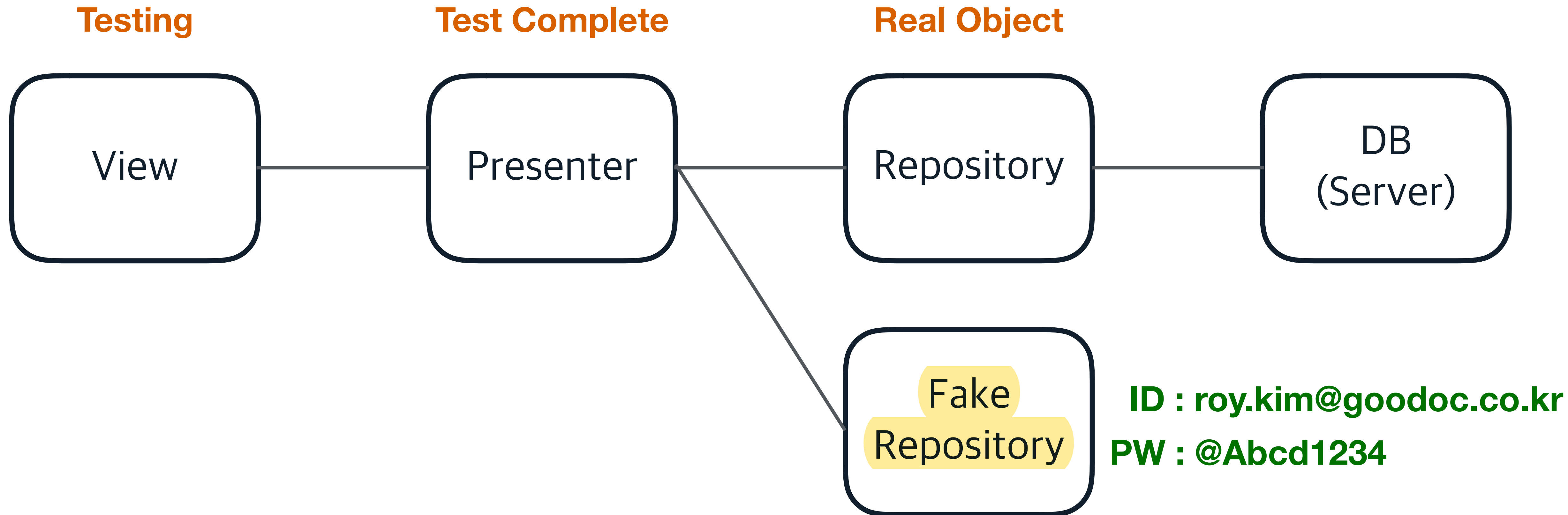
Real Object



같은 입력에 항상 같은 결과를 반환하는 코드



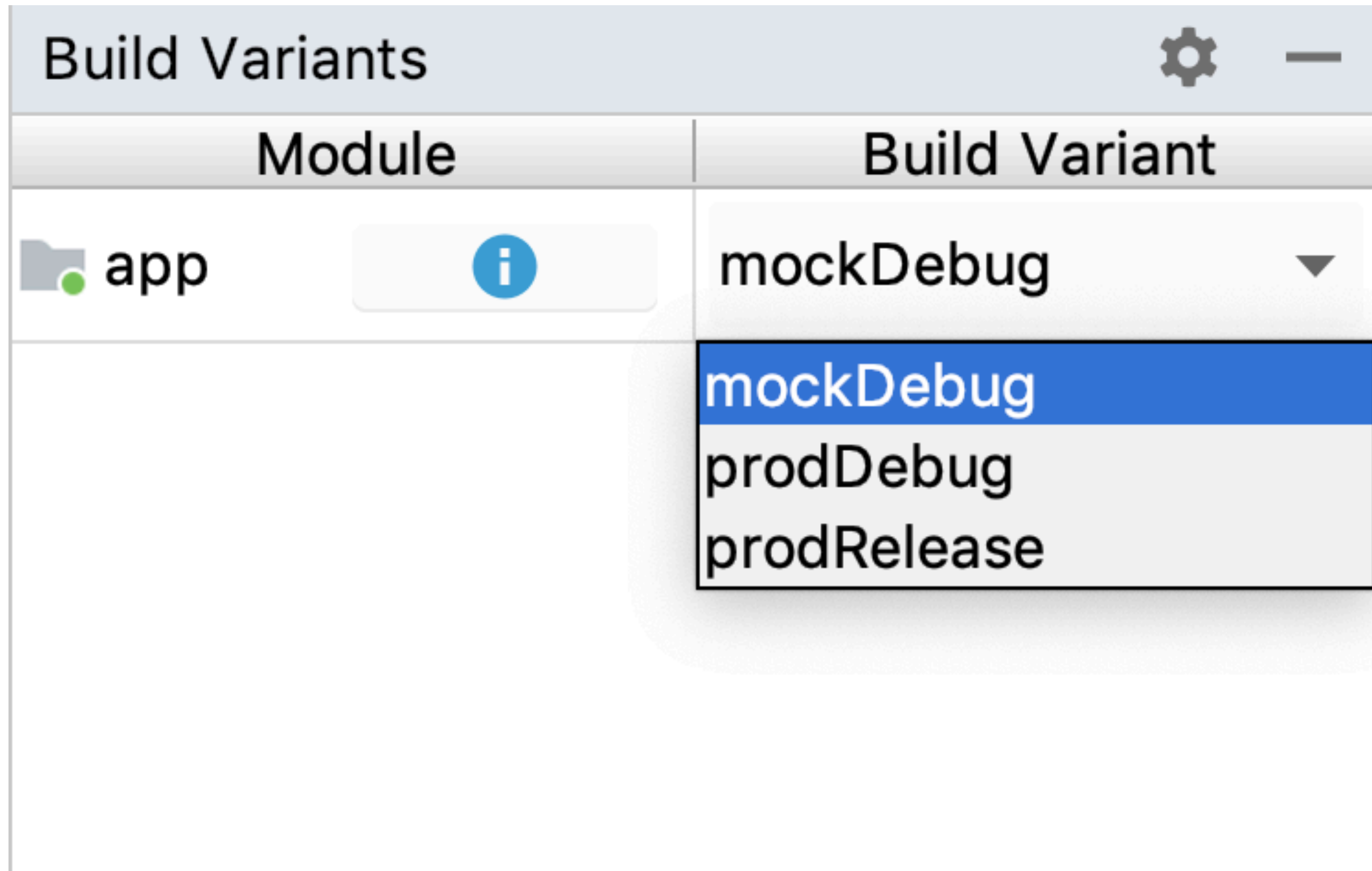
같은 입력에 항상 같은 결과를 반환하는 코드



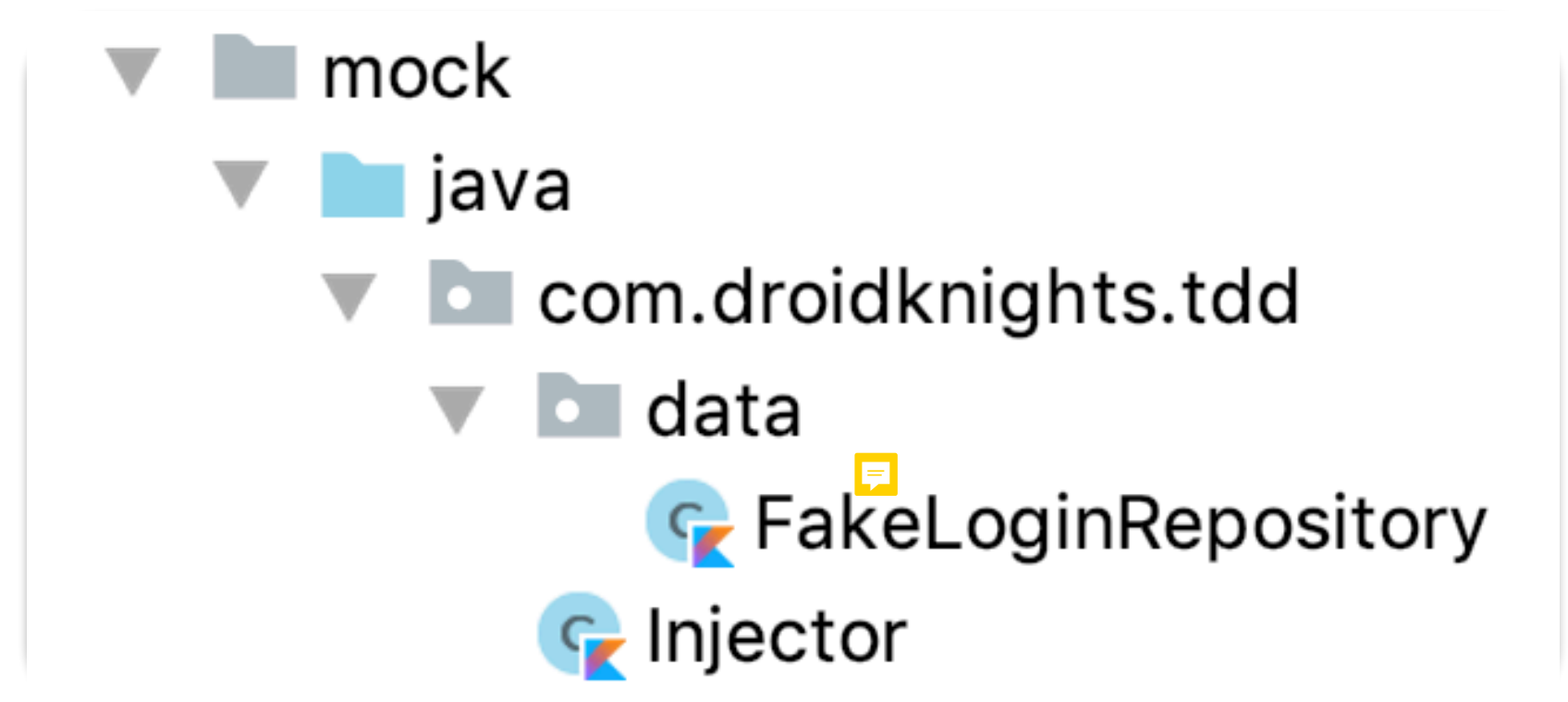
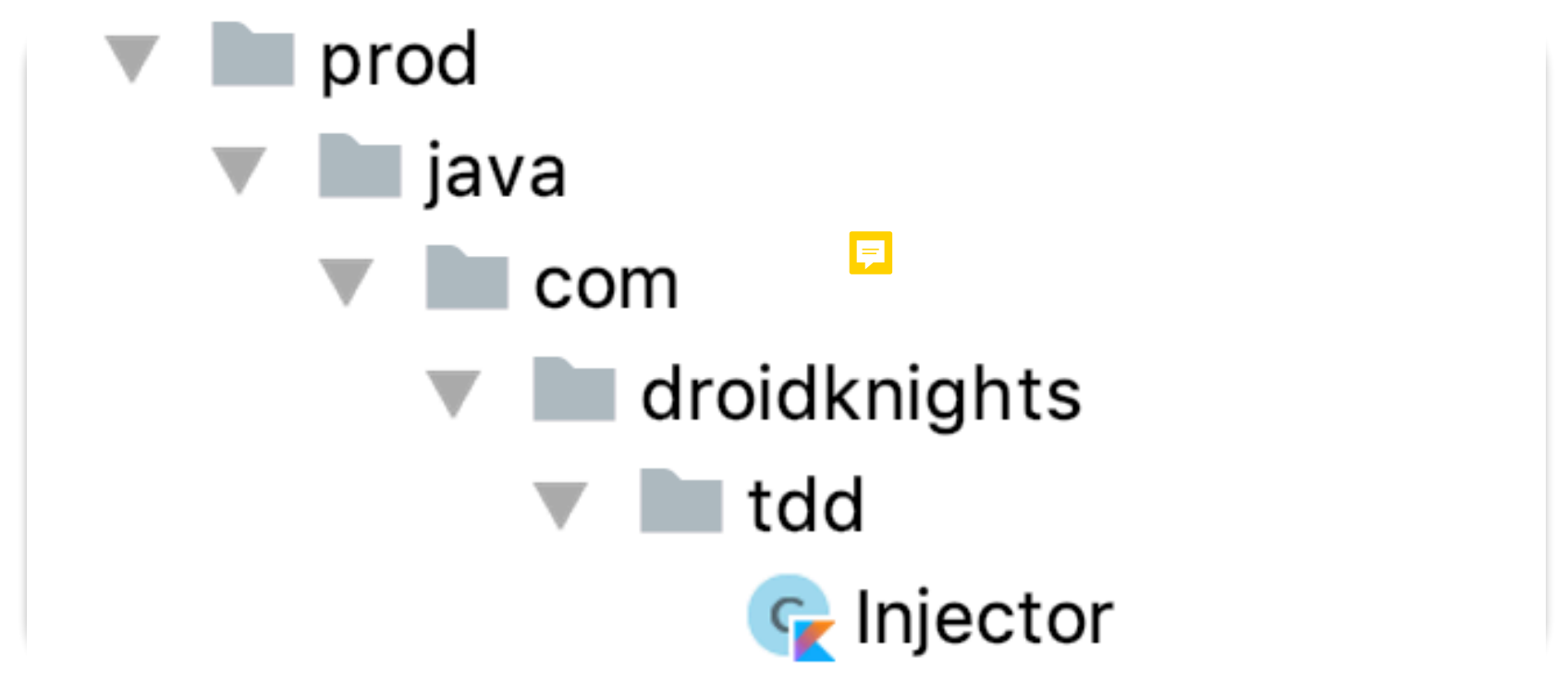
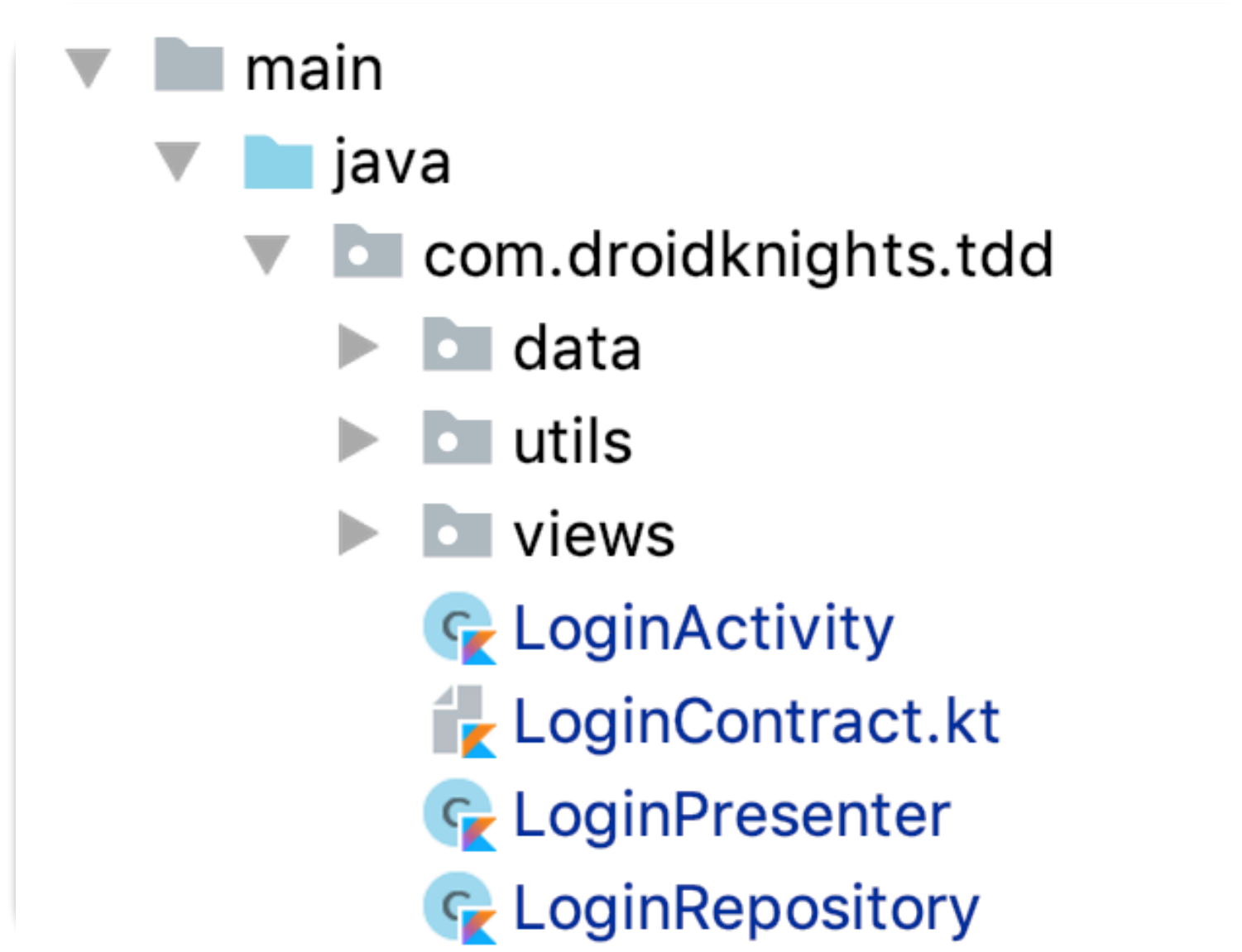
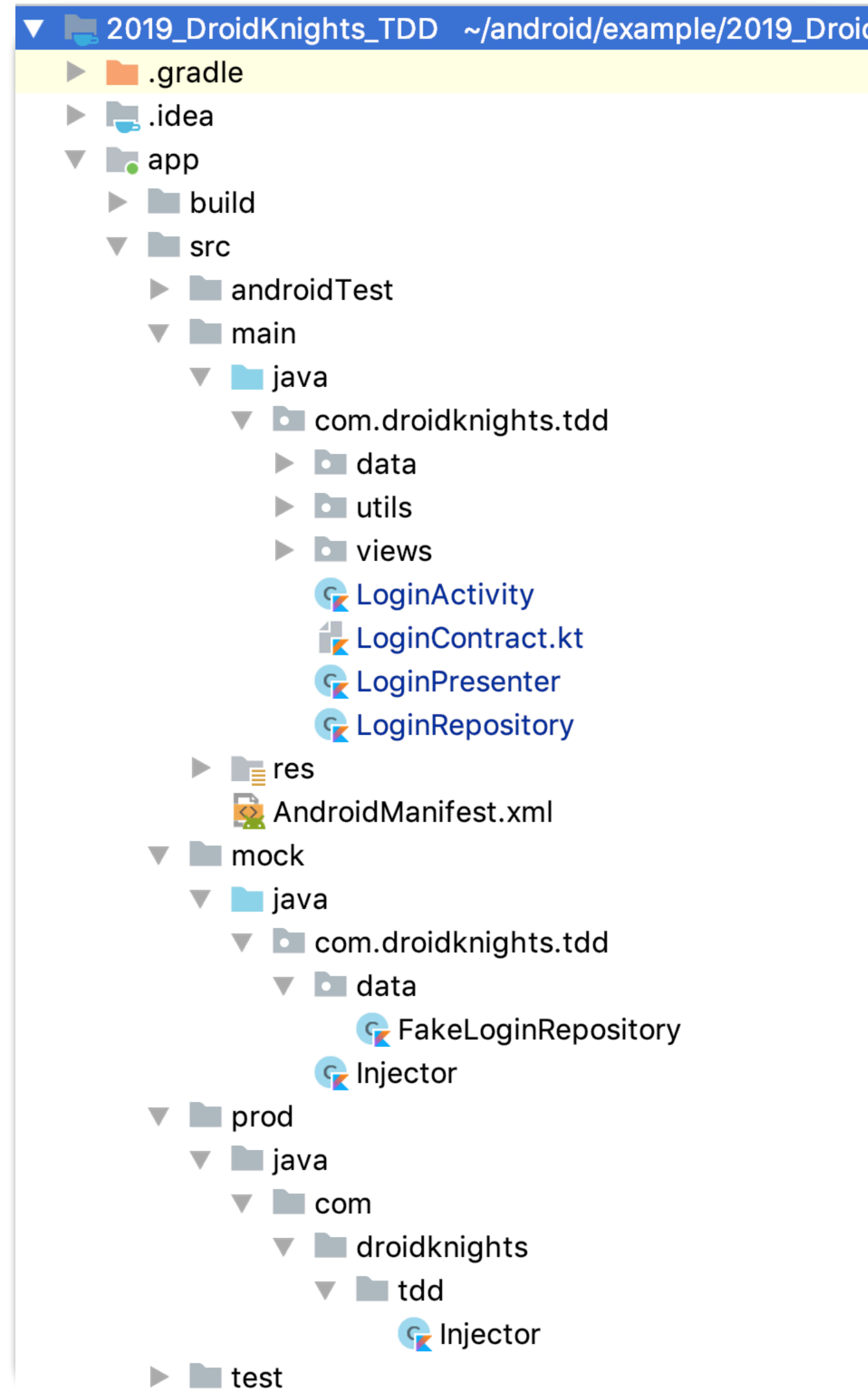
Flavor 와 DI를 이용한 mock Repository 생성

```
class LoginActivity : AppCompatActivity(), LoginContract.View {  
  
    private lateinit var presenter: LoginContract.Presenter  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_login)  
  
        presenter = LoginPresenter(this, Injector.provideLoginRepository())  
  
        emailLoginButton.setOnClickListener {  
            presenter.onClickEmailLogin()  
        }  
    }  
}
```

Flavor 와 DI를 이용한 mock Repository 생성



Flavor 와 DI를 이용한 mock Repository 생성



Flavor 와 DI를 이용한 mock Repository 생성

```
package com.droidknights.tdd
```


```
class Injector {  
    companion object {  
        fun provideLoginRepository(): LoginContract.Repository {  
            return LoginRepository()  
        }  
    }  
}
```


```
package com.droidknights.tdd
```

```
import com.droidknights.tdd.data.FakeLoginRepository
```








```
class Injector {  
    companion object {  
        fun provideLoginRepository(): LoginContract.Repository {  
            return FakeLoginRepository()  
        }  
    }  
}
```

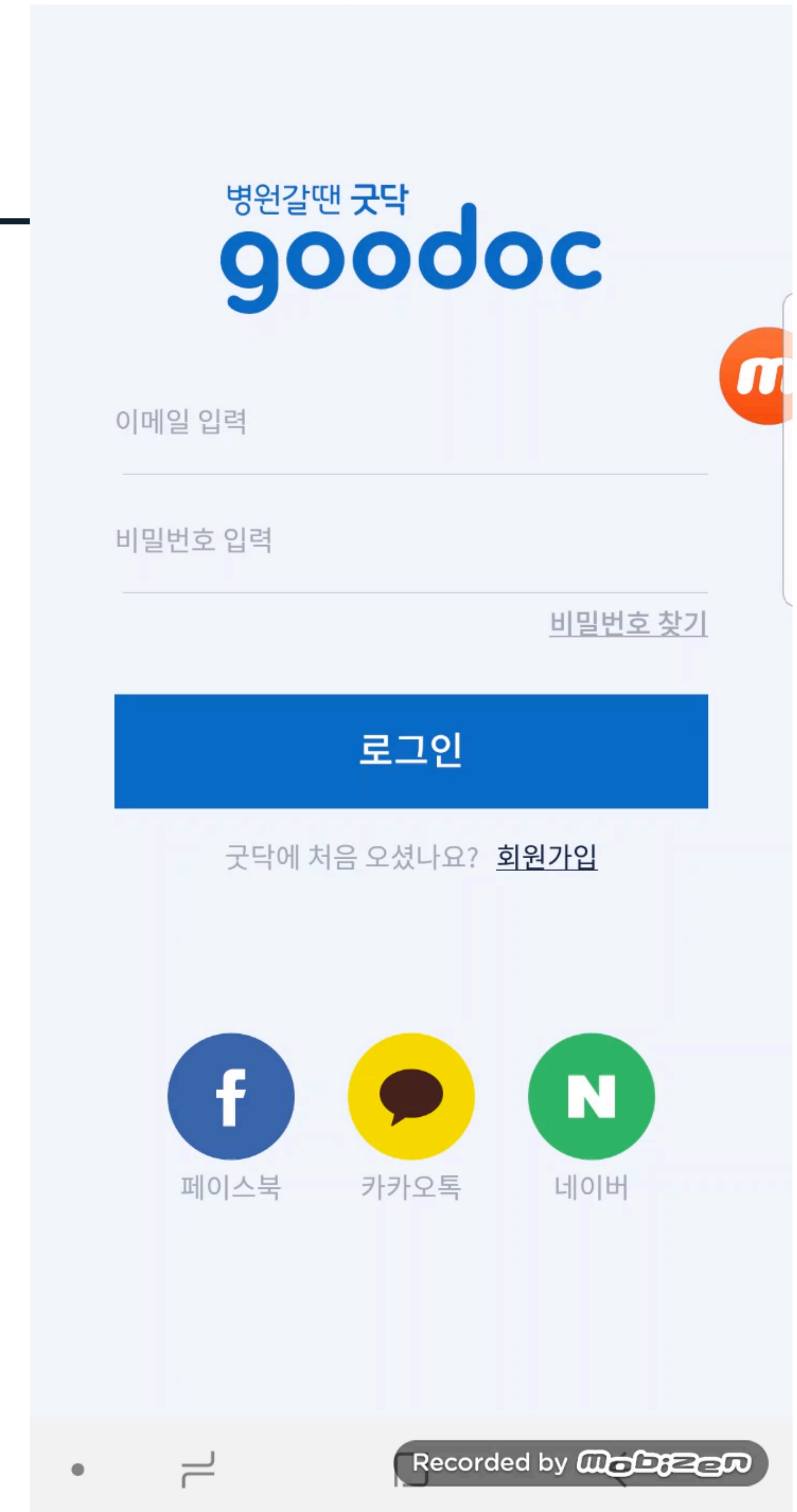
Flavor 와 DI를 이용한 mock Repository 생성

```
class LoginRepository : LoginContract.Repository {  
  
    override fun login(email: String, password: String, callback: LoginResultCallback) {  
        val client = OkHttpClient()  
        val request = // TODO build POST request...  
        client.newCall(request).execute()   
    }  
}
```

```
class FakeLoginRepository : LoginContract.Repository {  
  
    override fun login(email: String, password: String, callback: LoginResultCallback) {  
        if (email == "roy.kim@goodoc.co.kr" && password == "@Abcd1234") {  
            callback.onSuccess(UserInfo("0001"))  
        } else {  
            callback.onFail(1012, "미가입 회원입니다.")   
        }  
    }  
}
```


UI test cases..

- m  test_NotEnterEmailPassword(): Unit
- m  test_OnlyEnterEmail(): Unit
- m  test_OnlyEnterPassword(): Unit
- m  test_EnterInvalidEmail(): Unit
- m  test_Fail_UnregisteredUser(): Unit
- m  test_Fail_IncorrectPassword(): Unit
- m  test_SuccessEmailLogin(): Unit



What are the benefits of TDD?

- **코드를 보는 관점 변경**
 - 코드를 사용하는 입장으로 생각 전환
 - 단일 책임의 원칙
 - 캡슐화
 - 오버 엔지니어링 방지

What are the benefits of TDD?

용기 & 자신감

feat. 리팩토링

What is important to you now?

- **Mind**
- **Test coding skill**
- **Testable Architecture**
- **Diagram & Flowchart**

감사합니다.