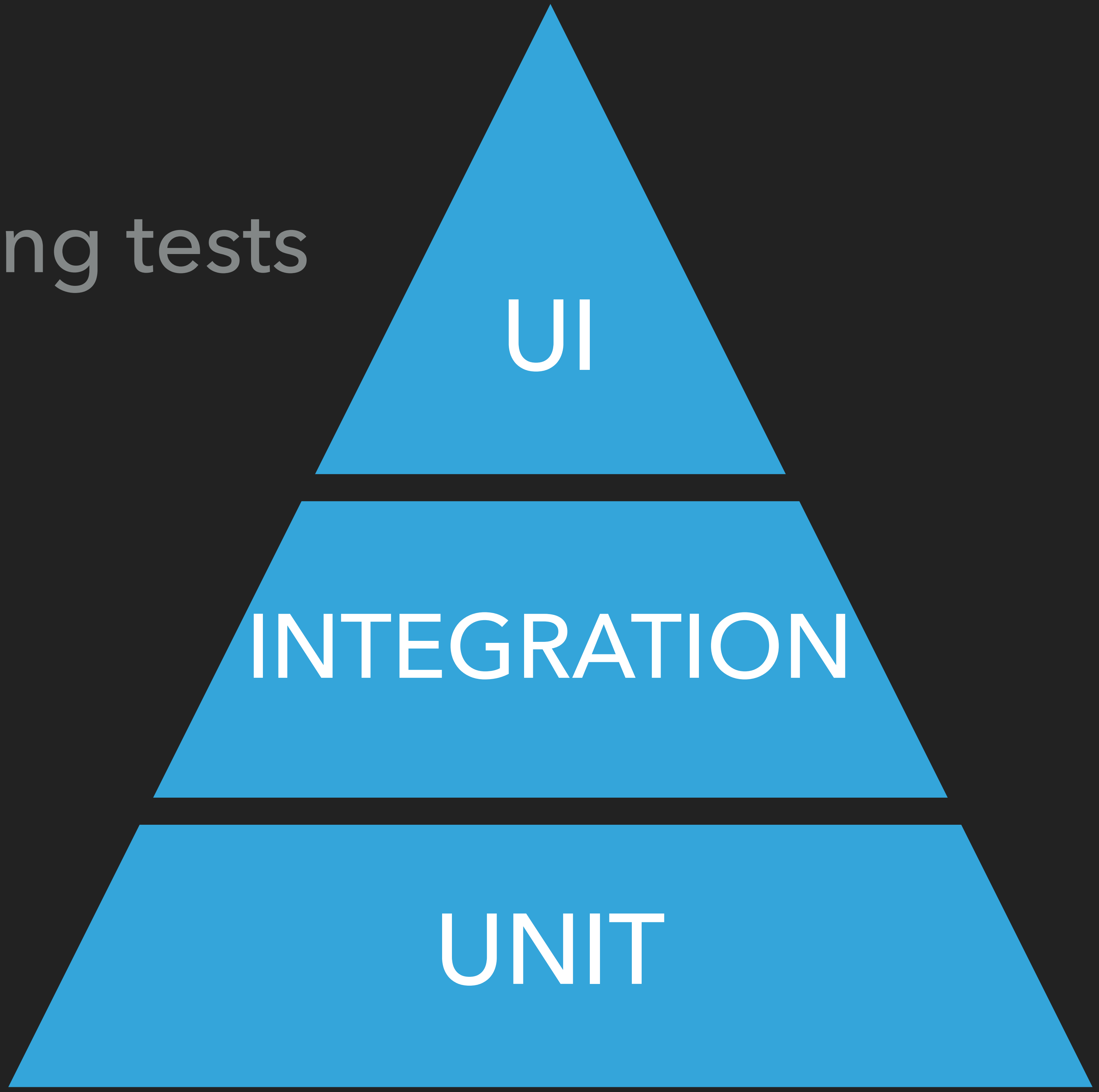
WRITING TEST-DRIVEN APPS WITH

# HTTP4K

DAVID DENTON / IVAN SANCHEZ

- Problems encountered in app development lifecycle:
  - Slow test suites impact delivery speed / MTTR
  - Flakey tests cause build instability
  - Switching technologies impossible without ditching tests
  - Duplication caused by different levels of pyramid

Can we write HTTP tests that won’t get in our way?

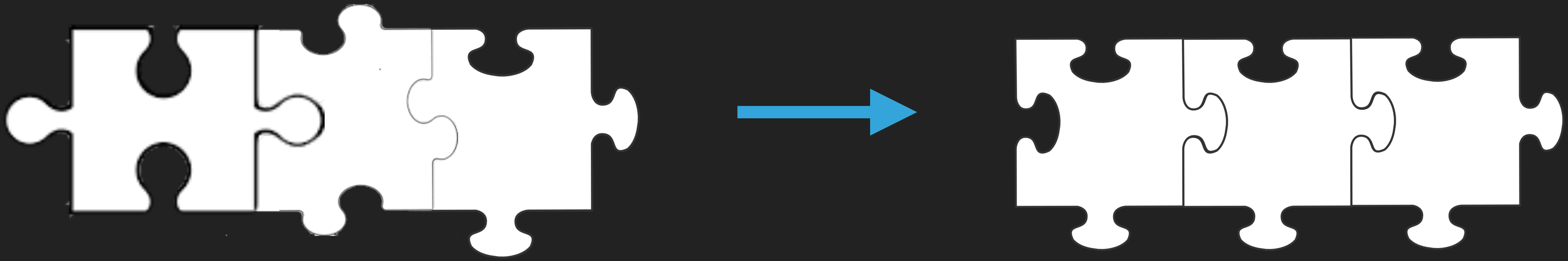


- What makes a good test?
  - Easy to write and maintain
  - Quick & reliable to run
  - Uncoupled to underlying technology
  - Reusable in different contexts

- http4k == Server as a Function in Kotlin

```
typealias HttpHandler = (Request) -> Response
```

- Uniform: HTTP server == HTTP Client



  - HTTP Messages are immutable data classes
  - Designed for Testability

- Writing an HTTP application to Analyse Sentences
- Endpoints:
    - Valid word count in a sentence: POST to /count
    - Record and report app hit count: GET to /calls
    - Analyse the letter makeup of a sentence: POST to /analyse
- Utilise 3rd party Dictionary HTTP service

Search Everywhere Double ⇧

Project View ⌘1

Go to File ⇧⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open

- Decorate HttpHandlers with Filter

```
typealias Filter = (HttpHandler) -> HttpHandler
```

- Use to modify HTTP messages or Security, Logging etc..
- They compose together into “stacks”:

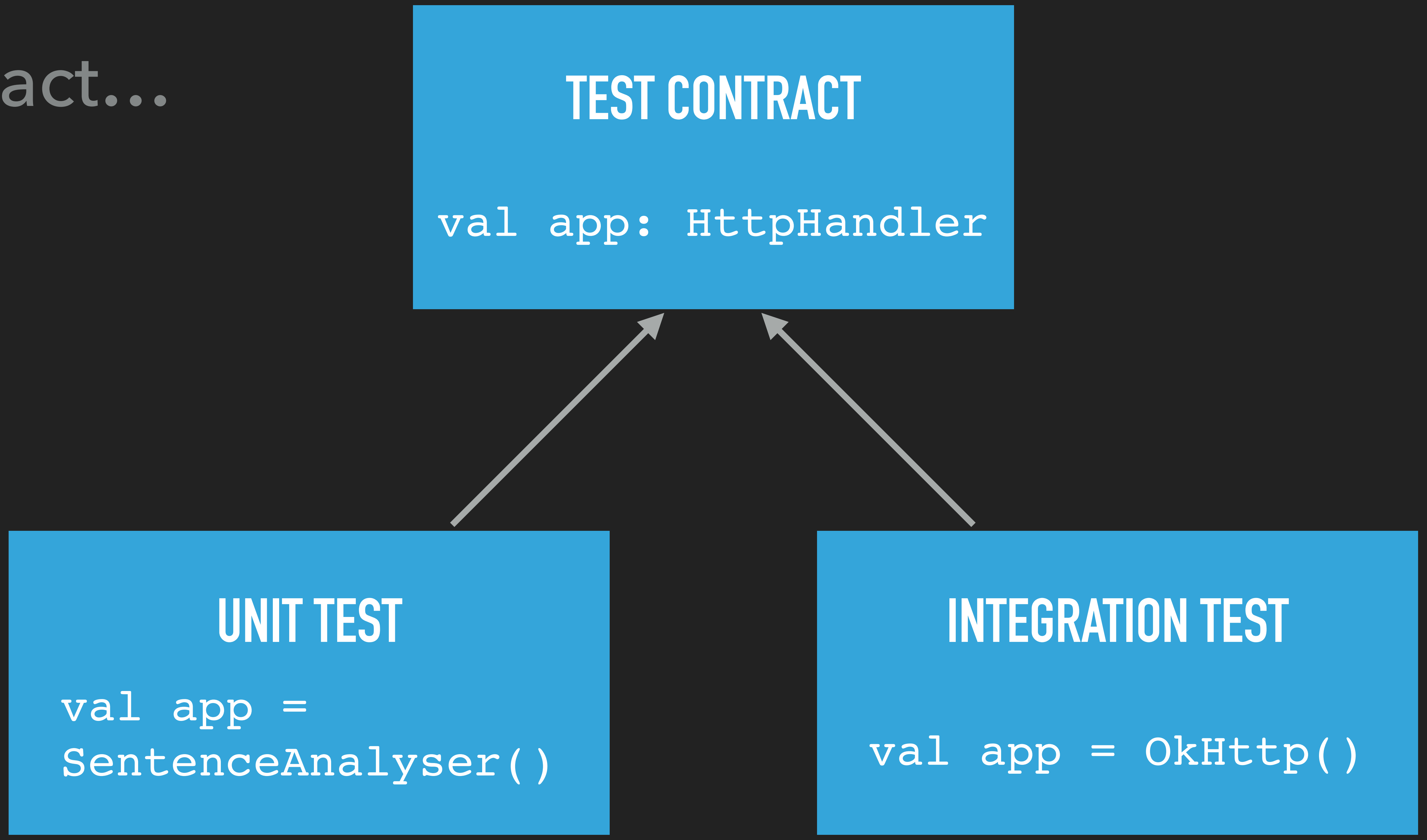
```
val stack: Filter = logging.then(security)

val app: HttpHandler = stack.then(httpHandler)
```

Project
sentence-analyser ~/Projects/s
.gradle
.idea
gradle
out
src
main
test
kotlin
SentenceAnalyserApp.kt
SentenceAnalyserT
resources
.gitignore
build.gradle
gradlew
gradlew.bat
README.md
settings.gradle
WordCounter.can analyse vali
WordCounter.can count valid
External Libraries
Scratches and Consoles

```kotlin
import ...

fun SentenceAnalyserApp(): HttpHandler = routes(
    "/count" bind Method.POST to { request: Request ->
        Response(Status.OK).body(request.bodyString().split(" ").size.toString())
    },
    "/calls" bind Method.GET to { request: Request -> Response(Status.OK).body("not implement
)

fun main() {
    SentenceAnalyserApp().asServer(SunHttp(8080)).start()
}
```

main()

- Service tests should be reusable
  - Unit tests - entirely in memory
  - Integration tests - running a live server
- We can extract a common test contract…



```kotlin
import org.http4k.core.Method
import org.http4k.core.Request
import org.http4k.core.Status
import org.http4k.hamkrest.hasBody
import org.http4k.hamkrest.hasStatus
import org.junit.jupiter.api.Test

class SentenceAnalyserTest {
    private val app = SentenceAnalyserApp()

    @Test
    fun `can count words`() {
        val request = Request(Method.POST, "/count").body("the lazy lazy cat")

        val response = app(request)

        assertThat(response, hasStatus(Status.OK) and hasBody("4"))
    }

    @Test
    fun `keeps track of total of calls`() {
        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("0"))

        app(Request(Method.POST, "/count").body("the lazy lazy cat"))

        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("1"))
    }
}
```

SentenceAnalyserTest › keeps track of total of calls()

- Provide JSON breakdown of character content in a submitted sentence

INPUT

```
POST /analyse

BODY: david ivan
```

OUTPUT

```
{
    "breakdown":{
        “a":2,
        "d":2,
        “i”:2,
        “n”:1,
        "v":2,
    }
}
```

```kotlin
import org.http4k.client.OkHttp
import org.http4k.core.HttpHandler
import org.http4k.core.Uri
import org.http4k.core.then
import org.http4k.filter.ClientFilters
import org.http4k.server.SunHttp
import org.http4k.server.asServer
import org.junit.jupiter.api.AfterEach
import org.junit.jupiter.api.BeforeEach

class SentenceAnalyserRemoteTest : SentenceAnalyserContract() {
    private val server = SentenceAnalyserApp().asServer(SunHttp(0))
    override val app: HttpHandler = ClientFilters.SetBaseUriFrom(Uri.of("http://localhost:${server.port()}")).then(OkHttp())

    @BeforeEach
    fun start() {
        server.start()
    }

    @AfterEach
    fun stop() {
        server.stop()
    }
```

SentenceAnalyserRemoteTest › val app

Run: SentenceAnalyse... × SentenceAnalyse... ×

Tests passed: 4 of 4 tests – 156 ms

```
<default package>            156 ms
  SentenceAnalyserRemote     148 ms
  SentenceAnalyserTest         8 ms
```

```
Process finishe
```

Tests passed: 4

- What is a Lens - it's 2 functions!

```
Inject<X>: (HttpMessage, X) -> HttpMessage
Extract<X>: (HttpMessage) -> X
```

- Represent JSON model objects as Kotlin data classes
- Lens creation:

```
val lens = Body.auto<Analysis>().toLens()
```

```kotlin
import ...

fun SentenceAnalyserApp(): HttpHandler {
    val counter = AtomicInteger()
    return routes(
        "/count" bind Method.POST to CallCounter(counter).then { request: Request ->
            Response(Status.OK).body(request.bodyString().split(" ").size.toString())
        },
        "/calls" bind Method.GET to { Response(Status.OK).body(counter.get().toString()) },
        "/analyse" bind Method.POST to { request: Request -> Response(Status.OK) }
    )
}

fun main() {
    SentenceAnalyserApp().asServer(SunHttp(8080)).start()
}
```

SentenceAnalyserApp() › {...}

Run: SentenceAnalyse... SentenceAnalyse...

Tests failed: 1 of 1 test – 49 ms

Test Results 49 ms
SentenceAnalyserTest 49 ms

```
but had Status that was: 404 Route not found
in: HTTP/1.1 404 Route not found
```

- Use the dictionary service to validates words
  - Lives at [http://api.dictionary.com:10000/](http://api.dictionary.com:10000/)
  - Endpoint GET /{word} - returns 200 (valid) or 404
  - Create a domain client
  - We can write a contract test to prove our usage

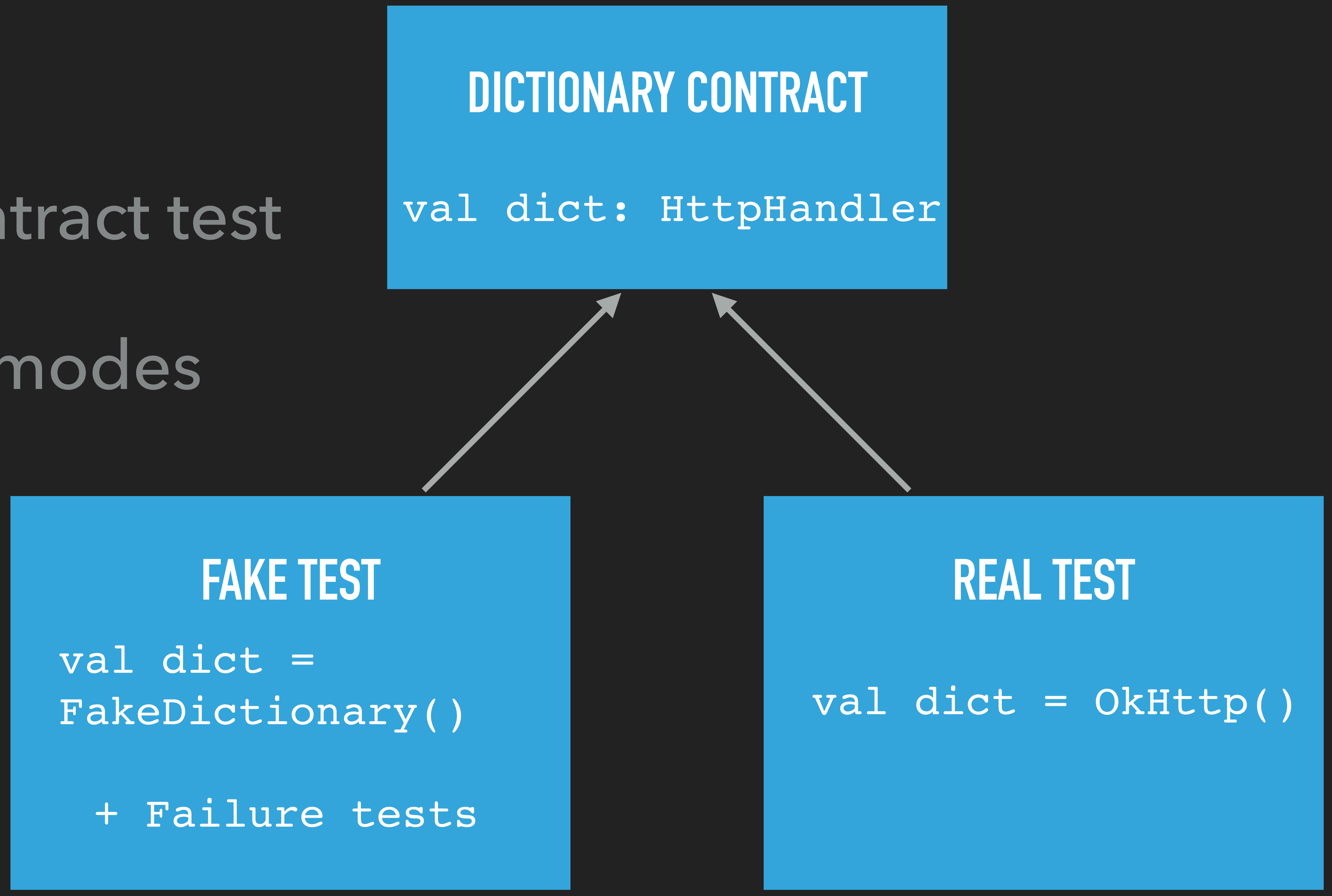
Project
n analyse a sentence.approved
n analyse an empty sentence.approved
se a sentence.approved
se an empty sentence.approved

```
    @Test
    fun `can count words`() {
        val request = Request(Method.POST, "/count").body("the lazy lazy cat")

        val response = app(request)

        assertThat(response, hasStatus(Status.OK) and hasBody("4"))
    }

    @Test
    fun `keeps track of total of calls`() {
        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("0"
        
        app(Request(Method.POST, "/count").body("the lazy lazy cat"))

        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("1"
    }

    @Test
    fun `can analyse an empty sentence`(approver: Approver) {
        approver.assertApproved(app(Request(Method.POST, "/analyse").body("")))
    }
```

SentenceAnalyserContract › can analyse an empty sentence()
Run: SentenceAnalyse... SentenceAnalyse...

```
Tests passed: 8 of 8 tests – 638 ms
<default package>          638 ms
  SentenceAnalyserRemot    622 ms
  SentenceAnalyserTest      16 ms

Process finishe
Tests passed: 8
```

- 3rd party test services are slow and unreliable
  - Create a simple stateful fake
  - Can start this as a server
  - Prove behaviour using reusable contract test
  - Can add test cases to check failure modes



```kotlin
import ...

class Dictionary(private val http: HttpHandler) {
    fun  isValid(word: String) = when(http(Request(Method.GET, "/$word")).status){
        Status.OK -> true
        else -> false
    }

}
```

Dictionary

```kotlin
import org.junit.jupiter.api.Test
import org.junit.jupiter.api.extension.ExtendWith

@ExtendWith(JsonApprovalTest::class)
abstract class SentenceAnalyserContract {
    protected abstract val app: HttpHandler

    @Test
    fun `can count words`() {
        val request = Request(Method.POST, "/count").body("the lazy lazy cat")

        val response = app(request)

        assertThat(response, hasStatus(Status.OK) and hasBody("3"))
    }

    @Test
    fun `keeps track of total of calls`() {
        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("0"))

        app(Request(Method.POST, "/count").body("the lazy lazy cat"))

        assertThat(app(Request(Method.GET, "/calls")), hasStatus(Status.OK) and hasBody("1"))
```

SentenceAnalyserContract › can count words()

Run: SentenceAnalyse... × SentenceAnalyse... ×

Tests failed: 1, passed: 3 of 4 tests – 594 ms

```
✓ can analyse an empty 565 ms
✓ keeps track of total of c 17 ms
✗ can count words()  7 ms
✓ can analyse a sentence( 5 ms
```

```
ected: a value that has Status that is equal to 200 OK and has Body that is equal to "3"
```

- What did we gain?
  - Fast in memory tests - no port required!
  - Reusable test code (no custom infrastructure!)
  - 3rd party dependency problems mitigated:
    - Flakey
    - Can't make them fail

- http4k also supports:
  - Chaos/failure mode testing with the ChaosEngine
  - Service Virtualisation with Servirtium
  - In-memory browser testing with WebDriver

@daviddenton
david@http4k.org

@s4nchez
ivan@http4k.org

# #questions

source code: bit.ly/tdd_http4k_code

@http4k

quickstart: start.http4k.org
web: www.http4k.org
slack: #http4k @ kotlinlang