Ted M. Young
Java Trainer, Coach, & Live Coder

Testable Architecture

Designing for Testable Code

Me: https://**ted.dev**/about
BlueSky: @**ted.dev**
Twitch: https://JitterTed.**Stream**
YouTube: https://JitterTed.**TV**

Source Code? Slides? Go here:
https://ted.dev/talks

Ted M. Young

ted@tedmyoung.com

I Can Help Your Team…

Write more Testable code
with more Effective tests

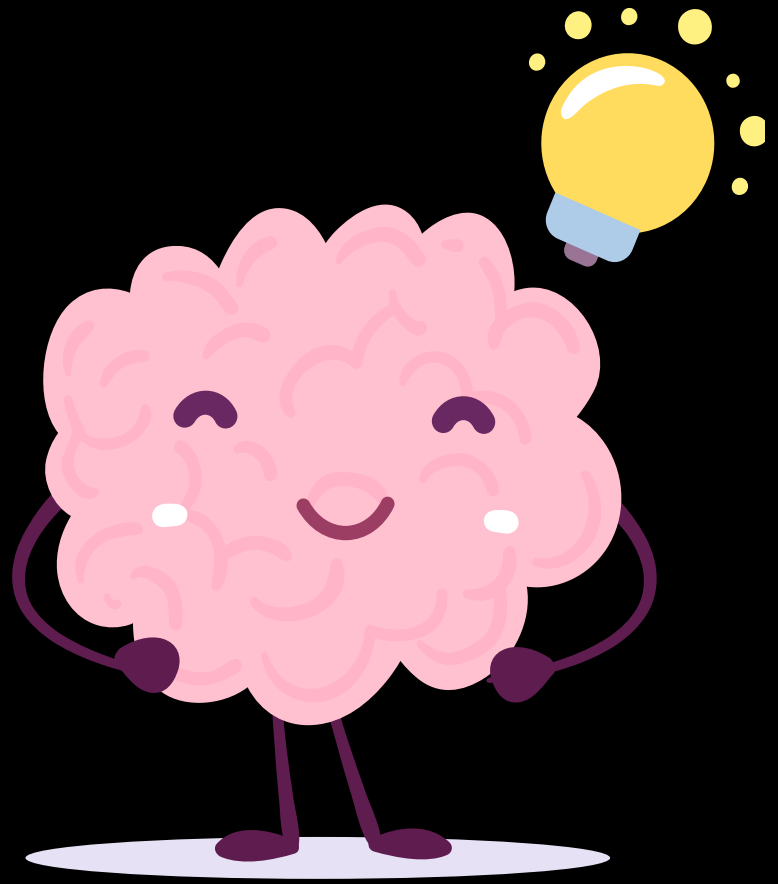
Be more productive in
Java & Spring

Effectively use
TDD

Refactor Messy Code

https://ted.dev/about

Ask Questions As You Need



I may defer the answer if I'm going to cover it later!

https://ted.dev/about

Tests...

What are they for?

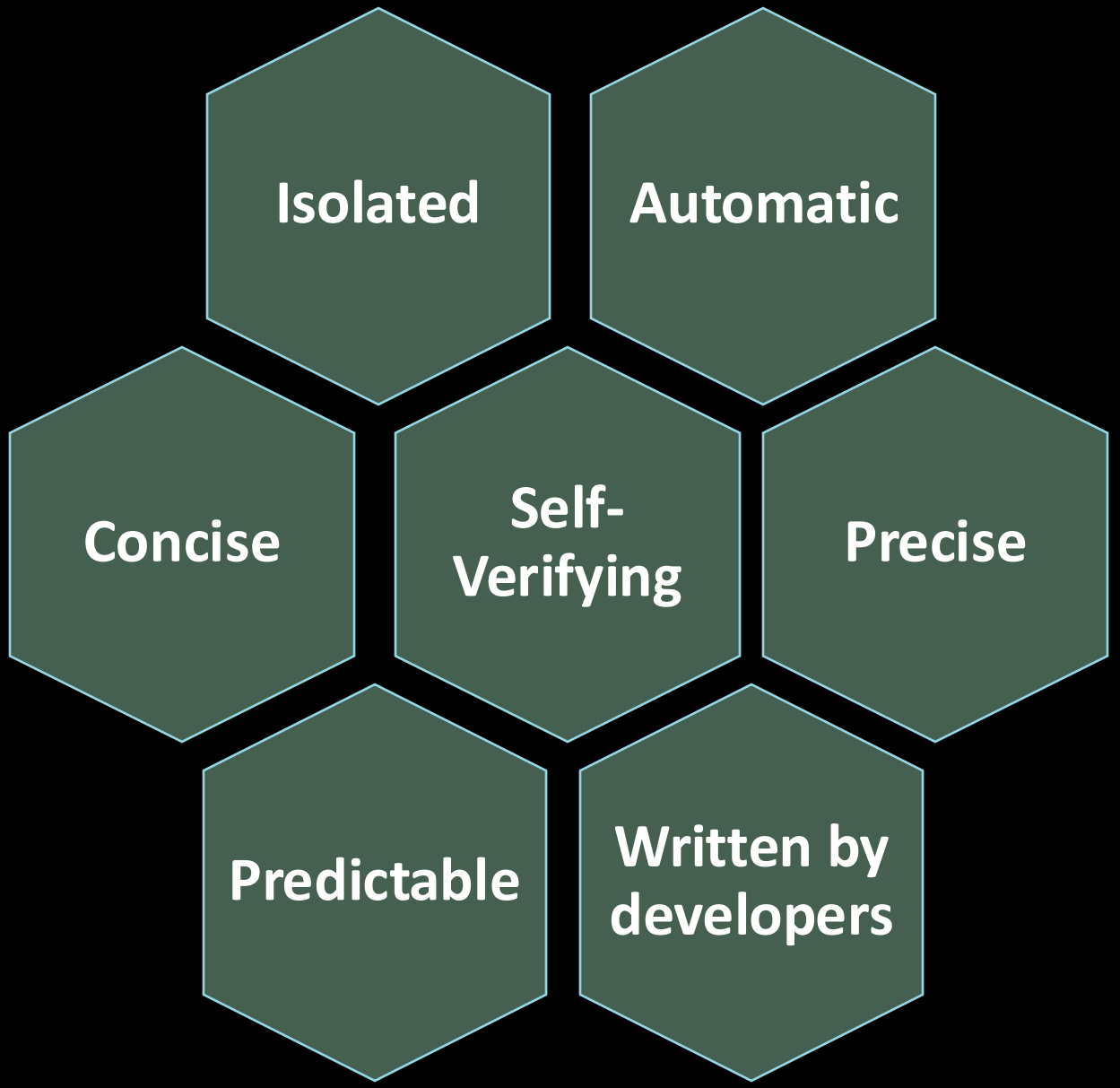
Define Desired Behavior

"Specification"

Verify Behavior is as Desired

aka "Validation"

Tests Are



https://ted.dev/about

Faster Test Feedback

Majority Fast, Some Slow, A Few Slowest

Why is writing tests so hard? 😫

what do you think?

https://ted.dev/about

Writing tests is hard because…

Classes are poorly organized	Classes mix logic and I/O access
Too much state Too many responsibilities	Infrastructure Hardware Framework

I/O is…

Anything outside of the current process

I/O includes…

- Hardware
  - System clock, Random number generator, Local files
- Remote Services
  - via HTTP, Queues, RPC
- Databases
  - Including "in-memory" databases (H2, SQLite, Redis)
- Other Processes
- Frameworks (Spring, Quarkus, etc.)

SCION-T 🤔

Separating
Concerns of
I/O and
Non-I/O for
Testability

Testable Architecture

https://ted.dev/about

Why do tests care about dependencies on I/O?

SPEED, EASE, & PREDICTABILITY


UNIT OR INTEGRATION?
HOW ABOUT NO

Two Kinds of Tests

I/O-Based

I/O-Free*

* These are practically "free" to execute

https://ted.dev/about

Inside the Code

A Simple Example

```java
public class MixedMeal {
    private static final NumberFormat CURRENCY_FORMATTER = NumberFormat.getCurrencyInstance();
    private static final double SALES_TAX_PERCENT = 8.25;
    private static final double SALES_TAX_RATE = SALES_TAX_PERCENT / 100;

    public static void main(String[] args) {
        System.out.println("""
                Choose an item to purchase:
                1) Burger        $11.95
                2) Pizza         $ 8.95
                3) Fried Chicken $ 9.95""");

        Scanner scanner = new Scanner(System.in);
        int choice = scanner.nextInt();

        double price = switch (choice) {
            case 1 -> 11.95;
            case 2 -> 8.95;
            case 3 -> 9.95;
            default -> throw new IllegalArgumentException("Invalid choice: " + choice);
        };

        double taxAmount = price * SALES_TAX_RATE;
        double total = price + taxAmount;

        String receipt = """
                Subtotal: %s
                     Tax: %s
                   Total: %s""".formatted(
                CURRENCY_FORMATTER.format(price),
                CURRENCY_FORMATTER.format(taxAmount),
                CURRENCY_FORMATTER.format(total));
        System.out.println(receipt);
    }
}
```

Mixed Concerns

https://ted.dev/about

Testing Mixed Concerns

```
18  class MixedMealTest {
19
20      ...
44
45      private static void sendChoiceOf(String simulatedInput) {
46          String simulatedInputWithNewline = simulatedInput + "\n";
47          System.setIn(new ByteArrayInputStream(
48                  simulatedInputWithNewline.getBytes(StandardCharsets.UTF_8)));
49      }
50
51      private String outputString() {
52          return testOut.toString(StandardCharsets.UTF_8);
53      }
54
55      @Test
56      void burgerChoiceTaxedAtCorrectRateWithTotals() {
57          sendChoiceOf("1"); // choose Burger via injecting to System.in
58
59          MixedMeal.main(new String[0]);
60
61          String output = outputString(); // get captured System.out
62          assertThat(output)
63                  .as("Receipt should include tax and totals for Burger choice")
64                  .contains("Subtotal: $11.95",
65                            "     Tax: $0.99",
66                            "   Total: $12.94");
67      }
68  }
```

https://ted.dev/about

Refactor Out Calculations

aka Domain Logic

Separated Concerns

```
11  ▶   public static void main(String[] args) {
12          System.out.println("""
13                  Choose an item to purchase:
14                  1) Burger        $11.95
15                  2) Pizza         $ 8.95
16                  3) Fried Chicken $ 9.95""");
17
18          Scanner scanner = new Scanner(System.in);
19          int choice = scanner.nextInt();
20
21          String receipt = computeReceiptFor(choice);
22
23          System.out.println(receipt);
24      }
```

```
26  @   public static String computeReceiptFor(int choice) {
27          double price = switch (choice) {
28              case 1 -> 11.95;
29              case 2 -> 8.95;
30              case 3 -> 9.95;
31              default -> throw new IllegalArgumentException(
32                              "Invalid choice: " + choice);
33          };
34
35          double taxAmount = price * SALES_TAX_RATE;
36          double total = price + taxAmount;
37
38          return """
39                  Subtotal: %s
40                       Tax: %s
41                     Total: %s""".formatted(
42                  CURRENCY_FORMATTER.format(price),
43                  CURRENCY_FORMATTER.format(taxAmount),
44                  CURRENCY_FORMATTER.format(total));
45      }
46  }
```

https://ted.dev/about

Testing Separated Concerns

```
7   class SeparatedMealTest {
8
9       @Test
10      void burgerChoiceTaxedAtCorrectRateWithTotals() {
11          String output = SeparatedMeal.computeReceiptFor(1); // choose Burger
12
13          assertThat(output)
14                  .as("Receipt should include tax and totals for Burger choice")
15                  .contains("Subtotal: $11.95",
16                            "     Tax: $0.99",
17                            "   Total: $12.94");
18      }
19  }
```

https://ted.dev/about

Using Separated Concerns

```
8   @RestController
9   public class ReceiptController {
10
11      @GetMapping("/receipt")
12      public ResponseEntity<String> receipt(@RequestParam("choice") String choice) {
13          try {
14
15              int choiceInt = Integer.parseInt(choice);
16              String receipt = SeparatedMeal.computeReceiptFor(choiceInt);
17              return ResponseEntity.ok(receipt);
18
19          } catch (NumberFormatException nfe) {
20              return ResponseEntity.badRequest().body("choice must be an integer");
21          } catch (IllegalArgumentException iae) {
22              return ResponseEntity.badRequest().body(iae.getMessage());
23          }
24      }
25  }
```

What is Architecture?

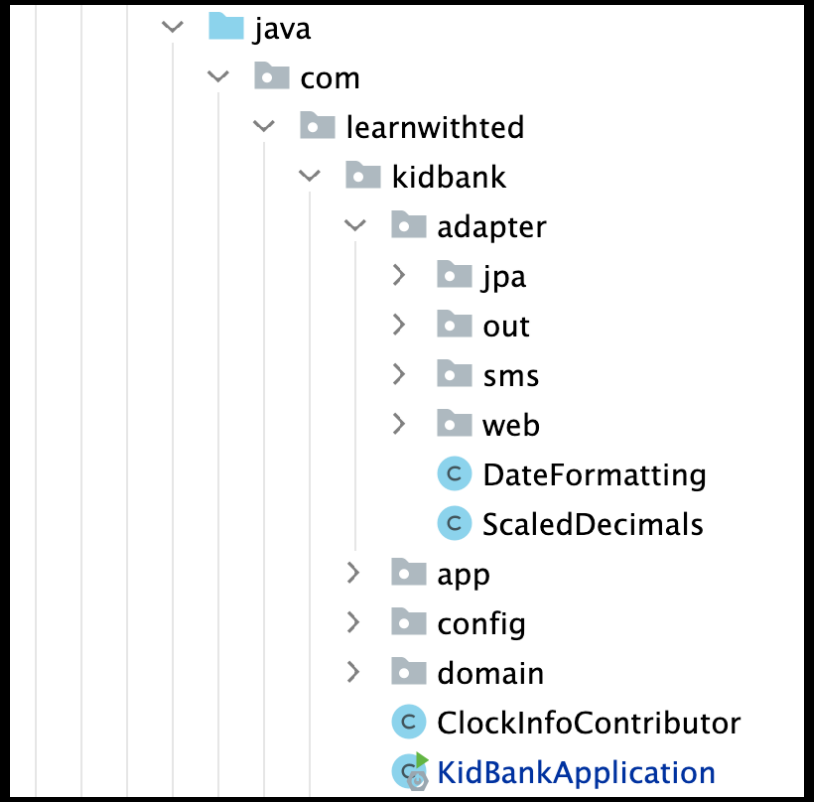
Some Folks Say Architecture…

- Is a way of organizing thoughts
- Provides the structure of a system
- Defines interactions between Components
- Is the vision of the application
- Marks the boundaries of different parts
- Defines scopes of responsibilities and technical deployment
- Are the decisions that are expensive to change

https://ted.dev/about

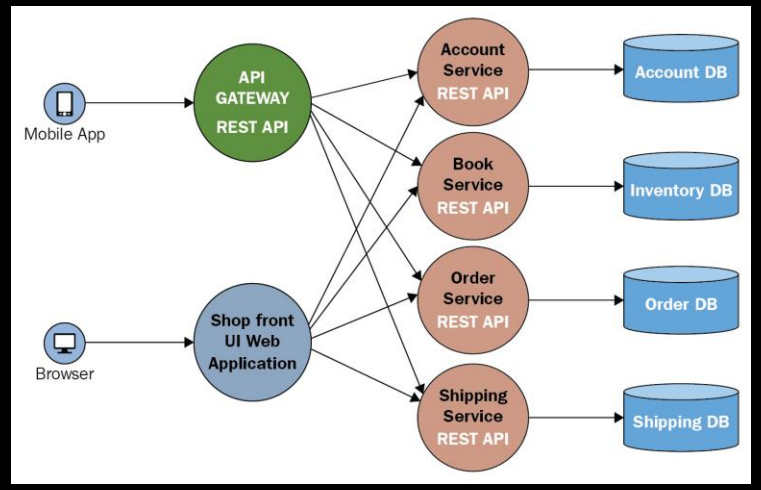
Architecture is…

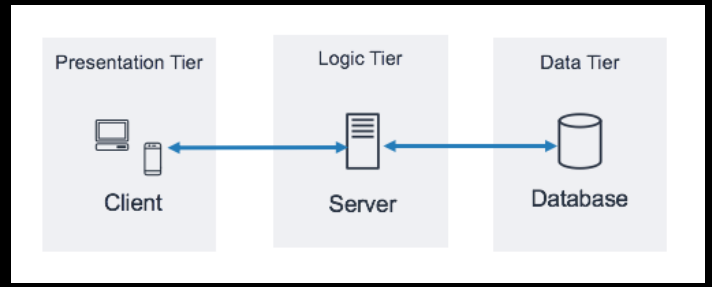
Organization of Code



layers

Deployment/Operational

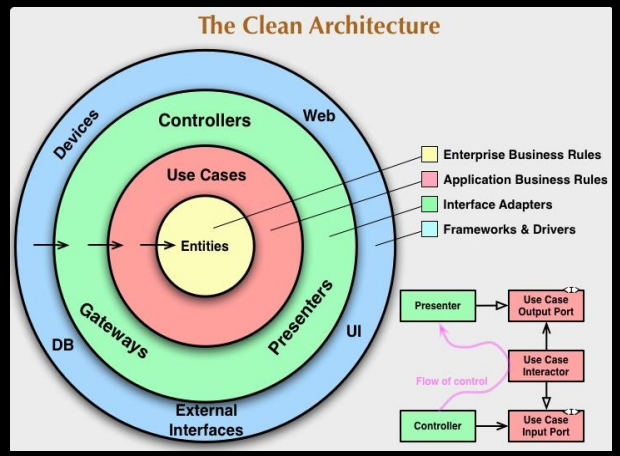


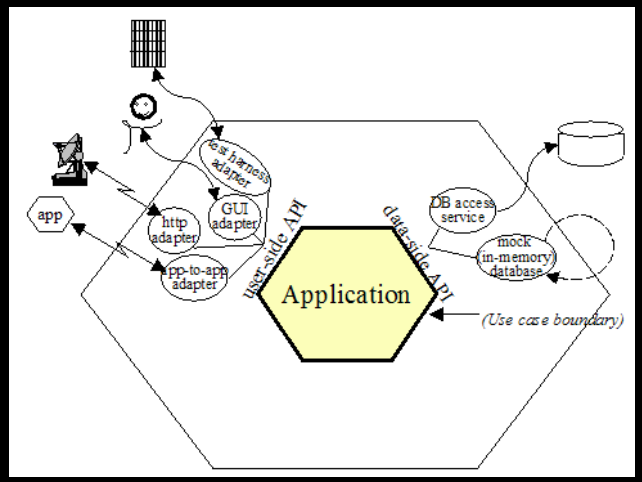


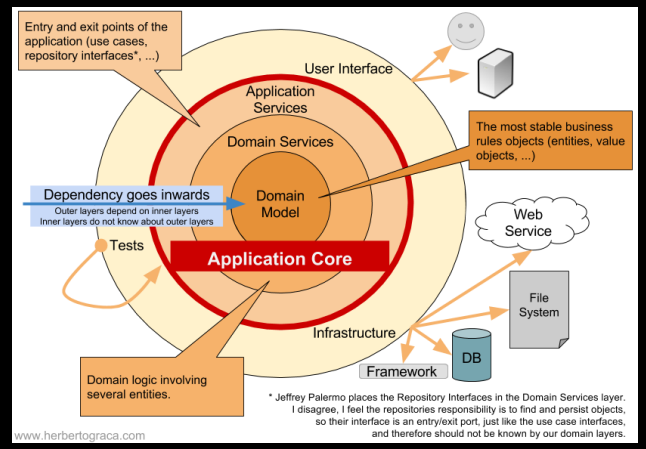
tiers

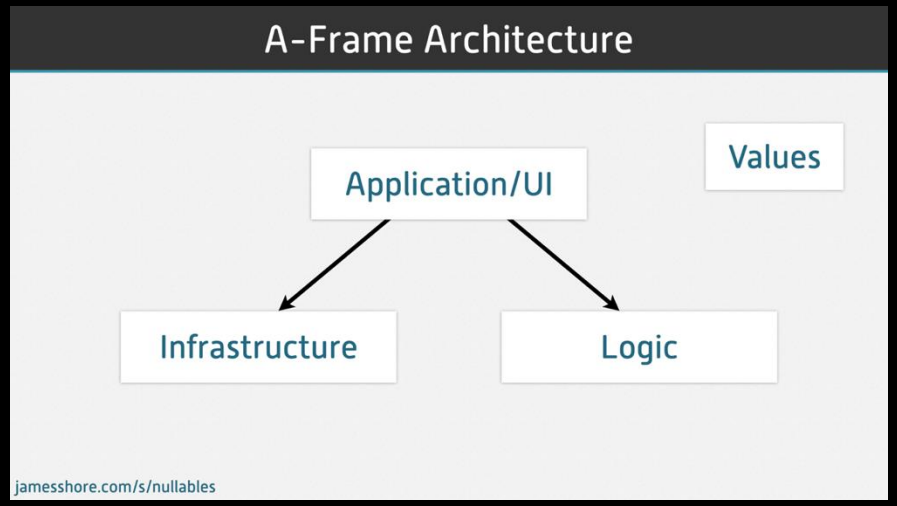
https://ted.dev/about

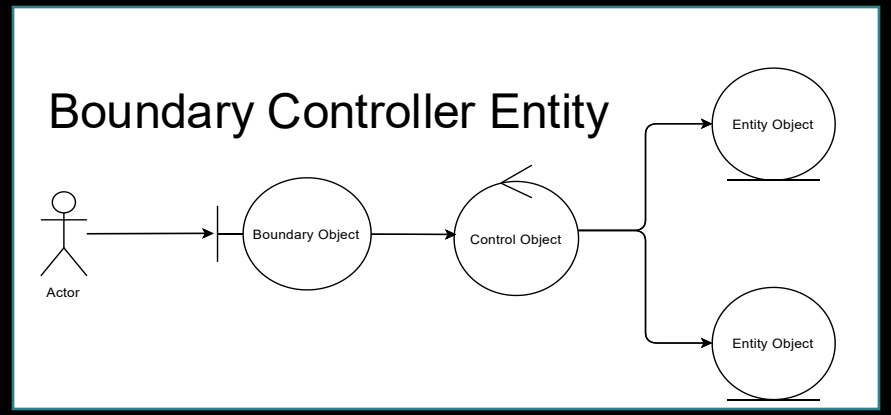
SCION-T(Testable) Architectures

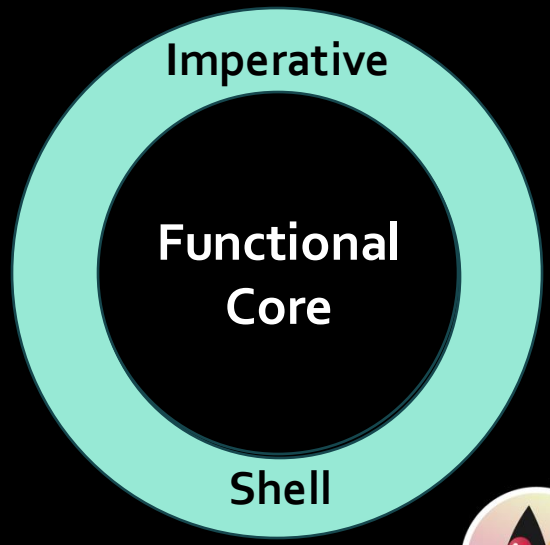








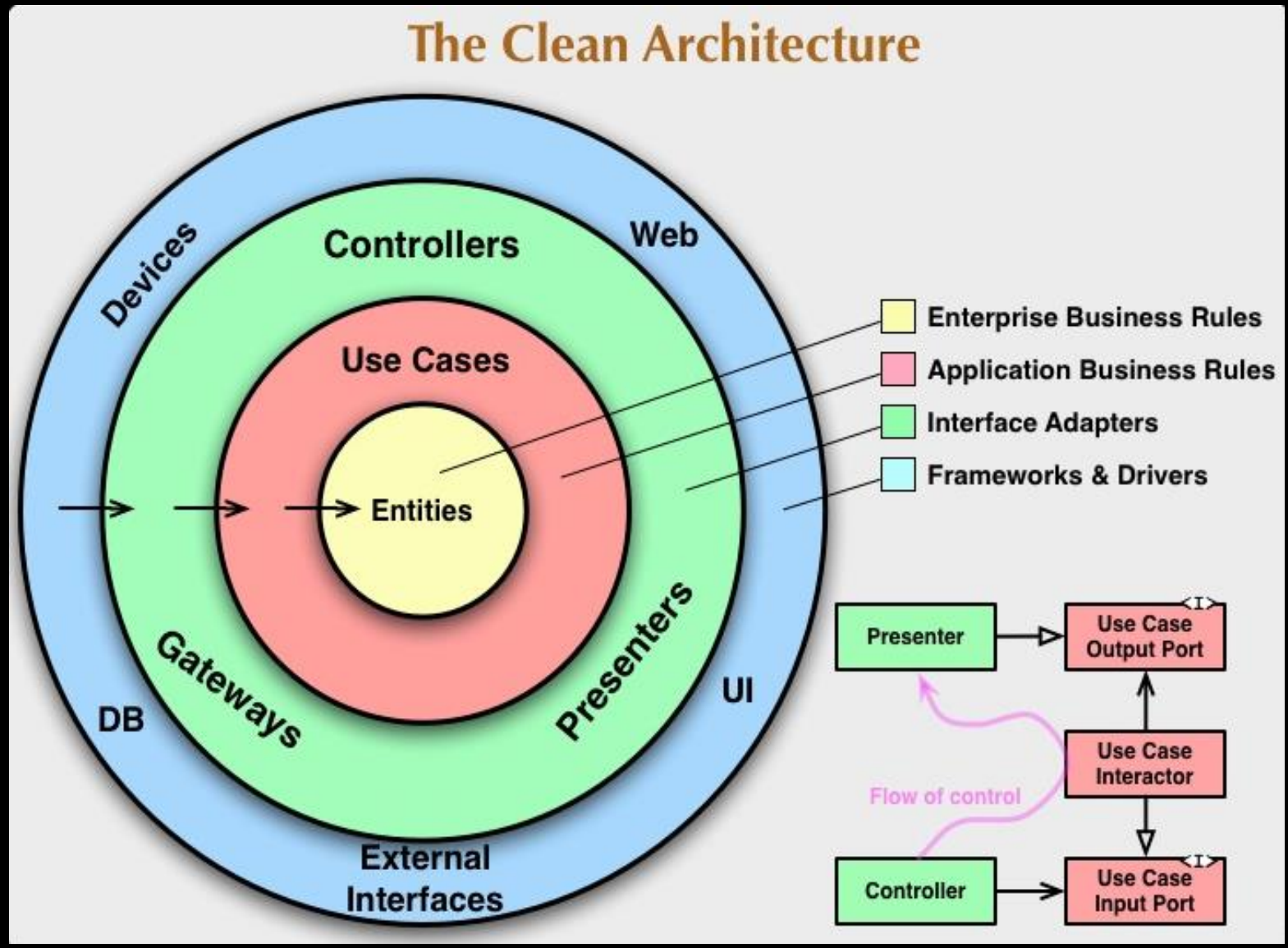




https://ted.dev/about

What About Clean Architecture?

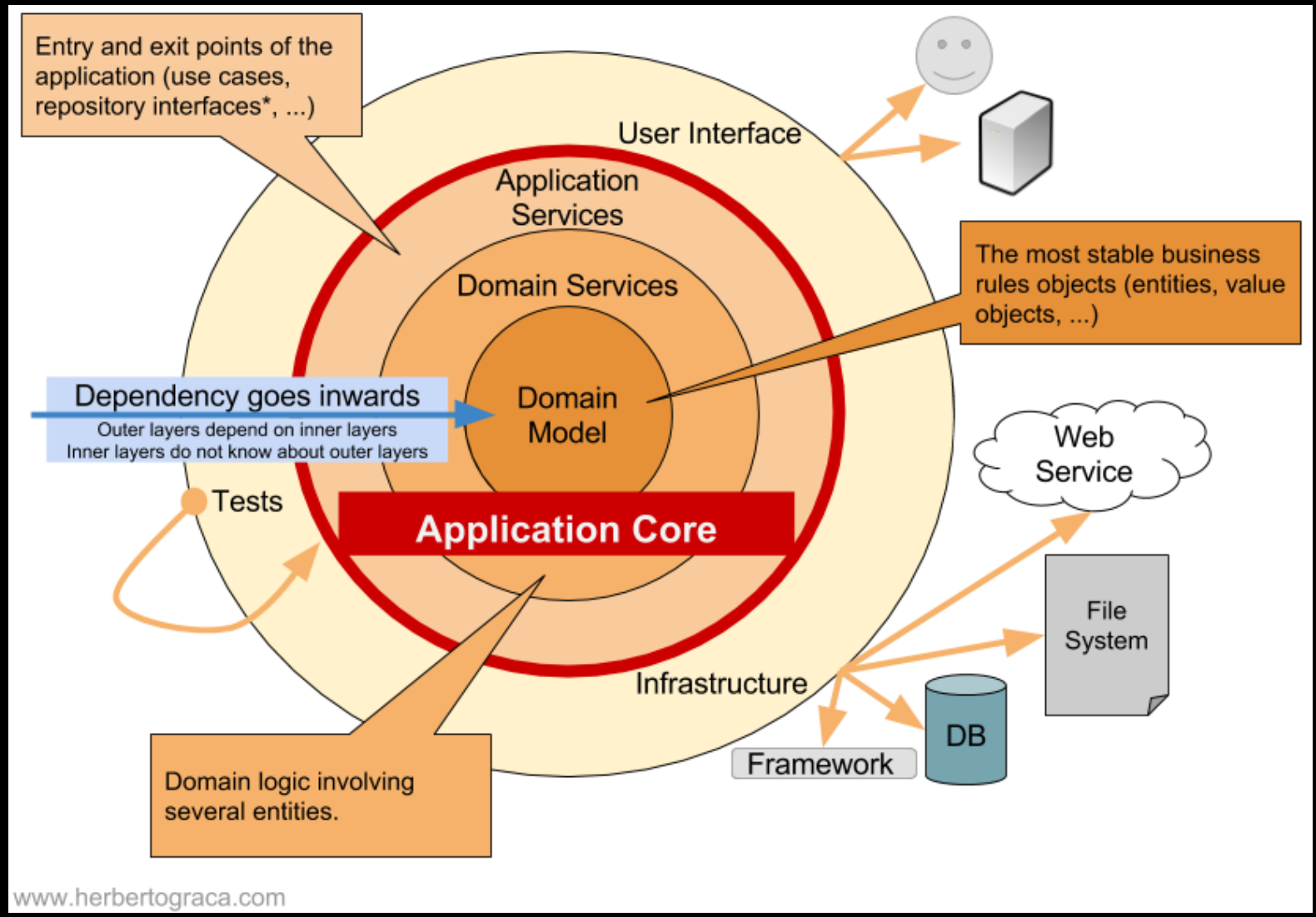
- ☑ Dependency direction
- ☑ Layer separation
- ✕ No left-to-right asymmetry
- ✕ No separation of adapters/drivers



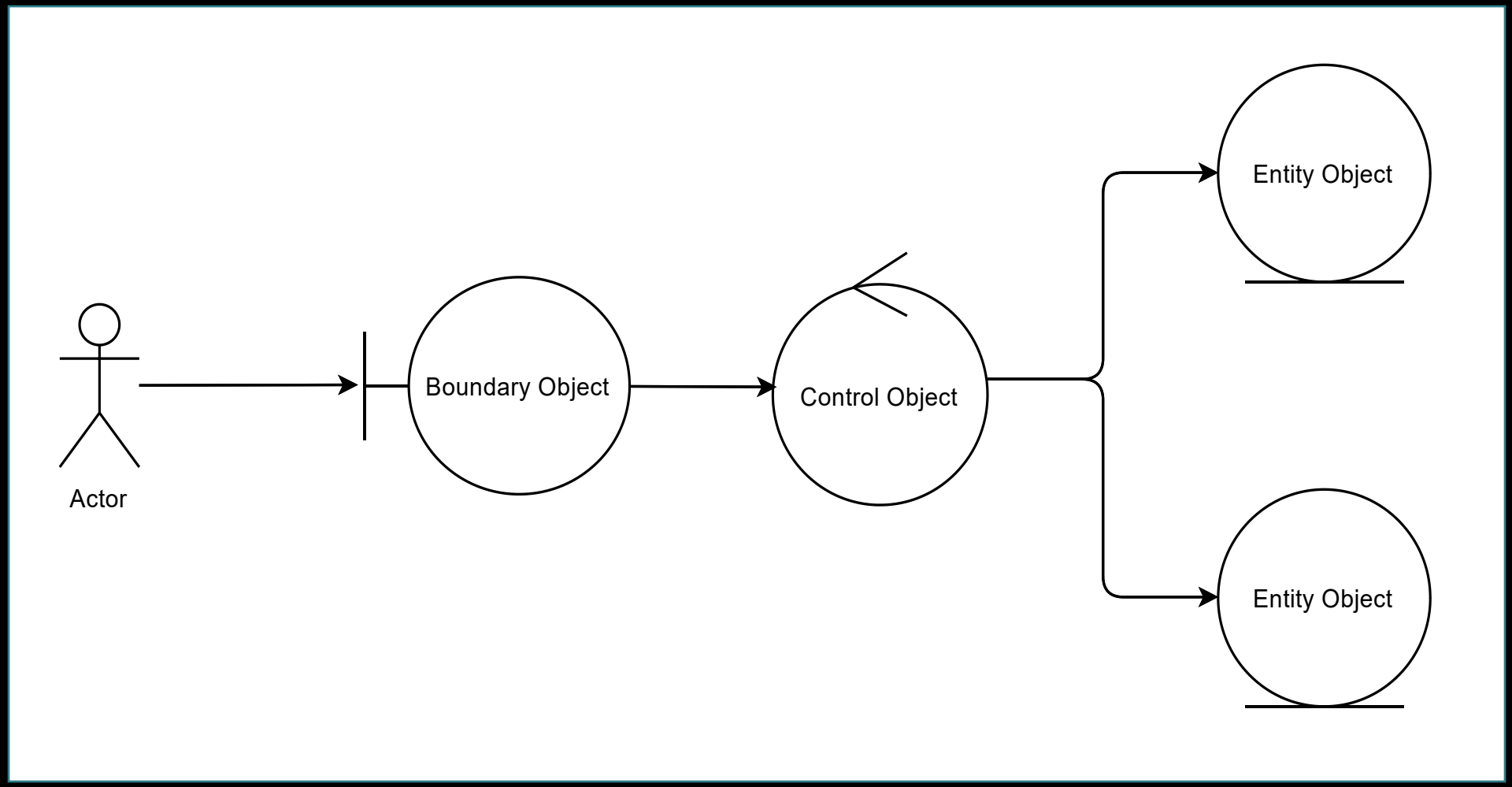
What About Onion Architecture?

- ✓ Dependency direction
- ✓ Layer separation
- ✓ Any outer layer can directly call any inner layer

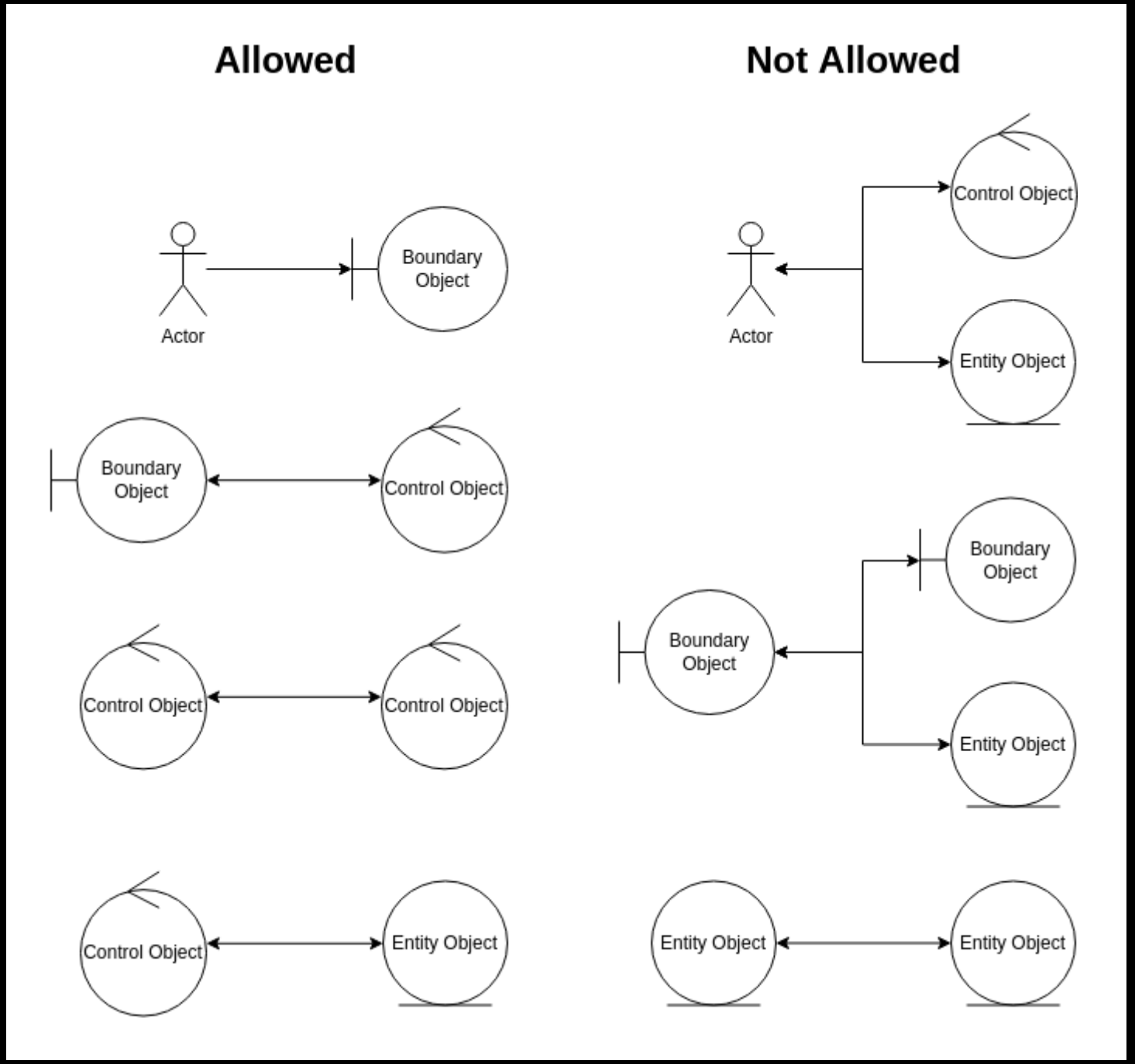
- ✗ No left-to-right asymmetry
- ✗ No separation of infrastructure



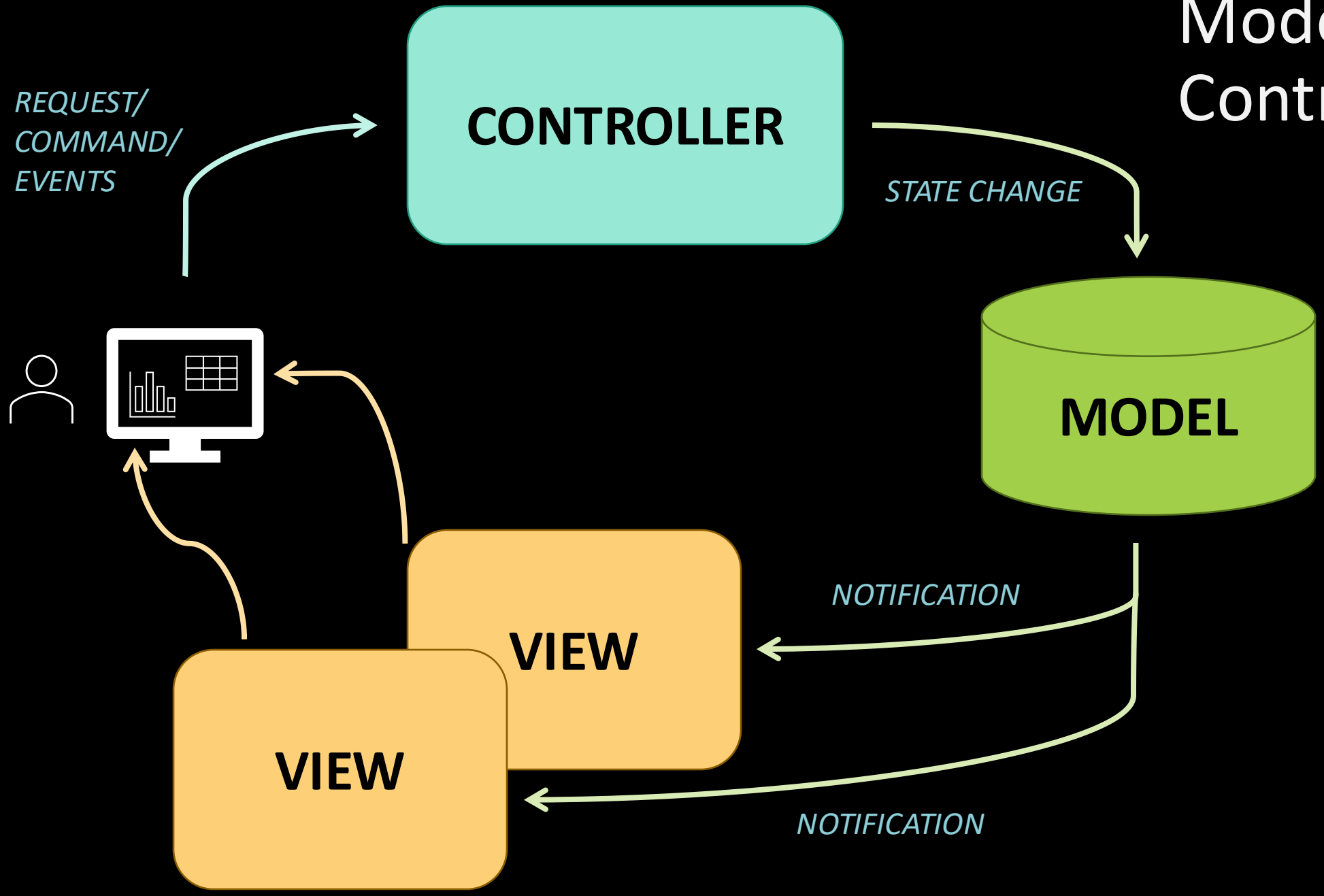
BCE has Similar Flavor



Robustness Diagram: Rules

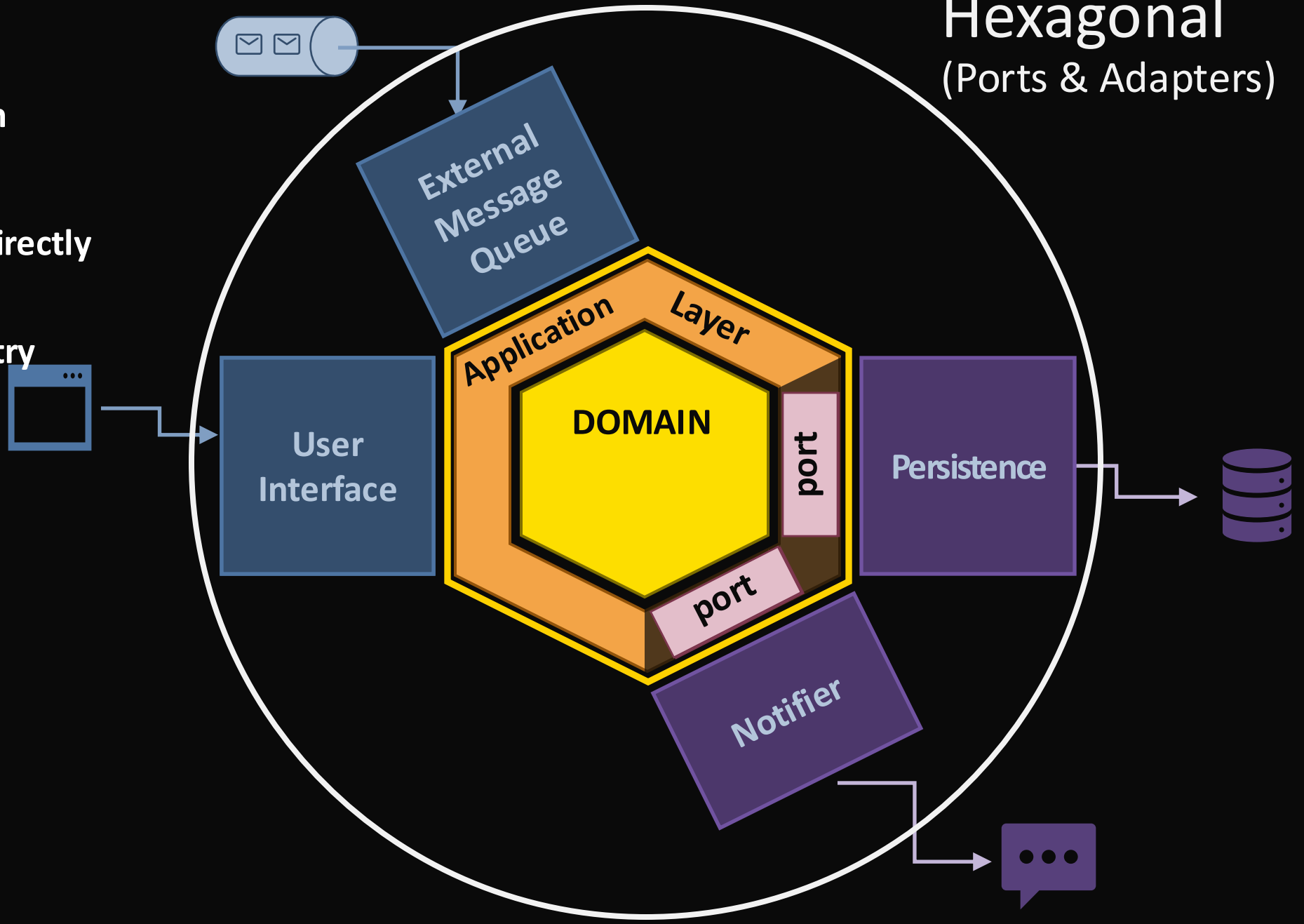


Model View Controller



Hexagonal
(Ports & Adapters)

- ✓ Dependency direction
- ✓ Layer separation
- ✓ Any outer layer can directly call any inner layer
- ✓ Left-to-right asymmetry
- ✓ Separation of infrastructure with dedicated Adapters



Hexagonal + DDD = Testable Architecture

also known as PORTS & ADAPTERS

Disclaimer…

Hexagonal Architecture is not a Specification…

…it's a Pattern

“Allow an application to equally be driven by users, programs, [and] automated tests... and to be developed and tested in isolation from its eventual run-time devices and databases.”

Ports & Adapters Pattern
Alistair Cockburn

“Allow an application to equally be driven by users, programs, [and] automated tests... and to be developed and tested in isolation from its eventual run-time devices and databases.”

Ports & Adapters Pattern
Alistair Cockburn

“Allow an application to equally be driven by users, programs, [and] automated tests... and to be developed and tested in isolation from its eventual run-time devices and databases.”

Ports & Adapters Pattern
Alistair Cockburn

OUTSIDE

OUTSIDE

OUTSIDE

OUTSIDE

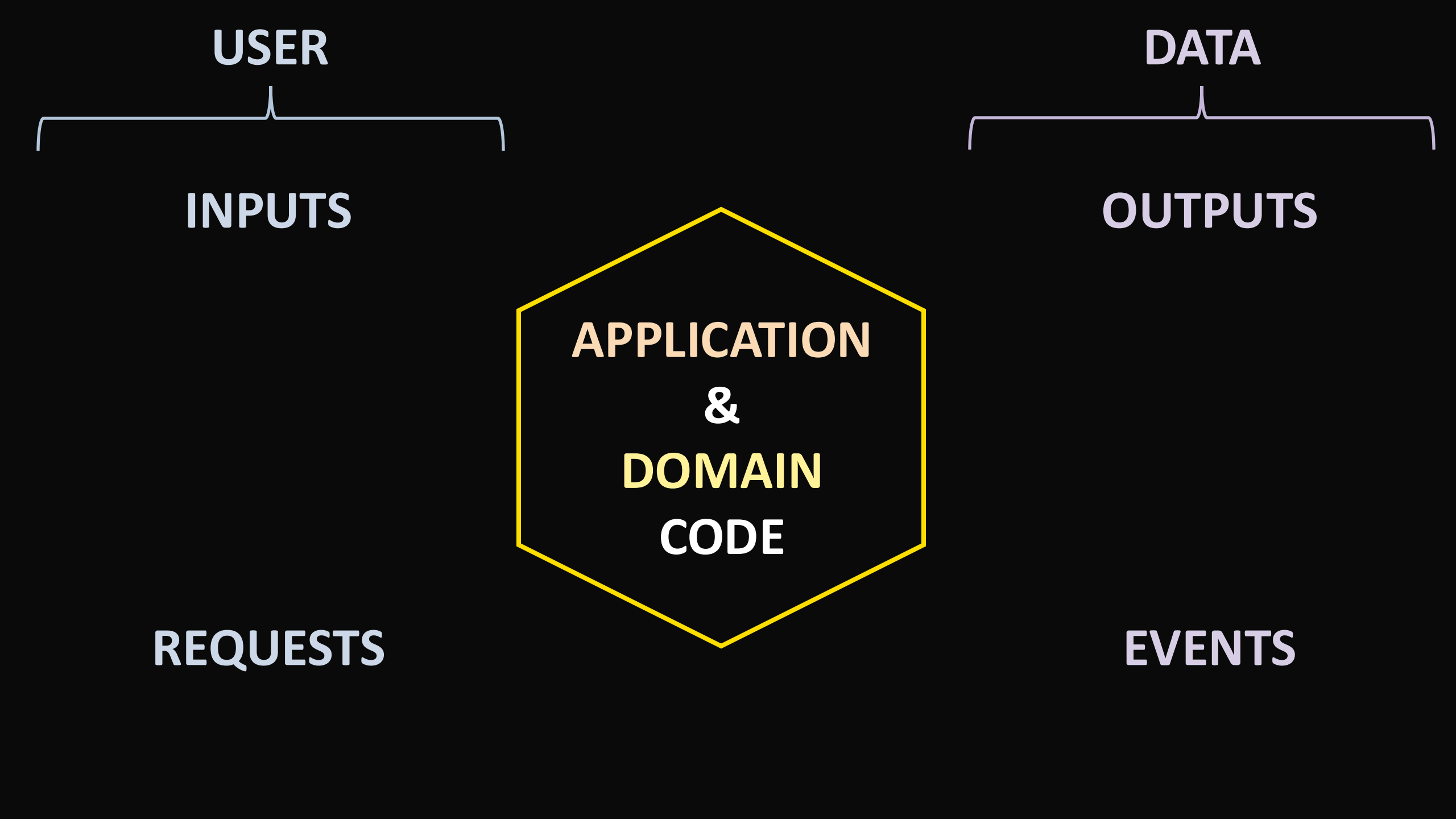

OUTSIDE
OUTSIDE
APPLICATION
&
DOMAIN
CODE
OUTSIDE
OUTSIDE

REST

OF

THE

WORLD


USER
DATA
INPUTS
OUTPUTS
APPLICATION
&
DOMAIN
CODE
REQUESTS
EVENTS

INPUTS

OUTPUTS

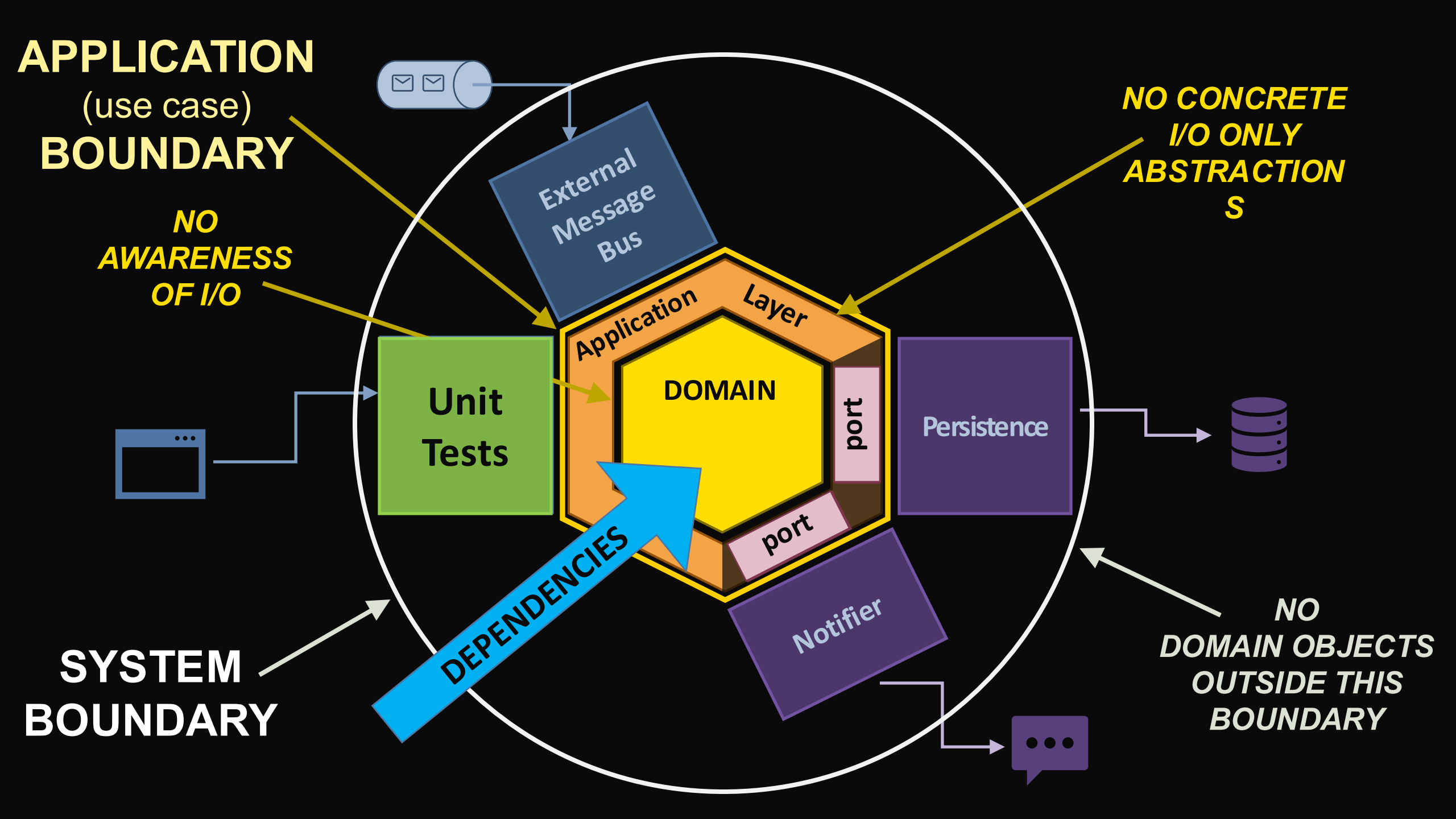
INBOUND

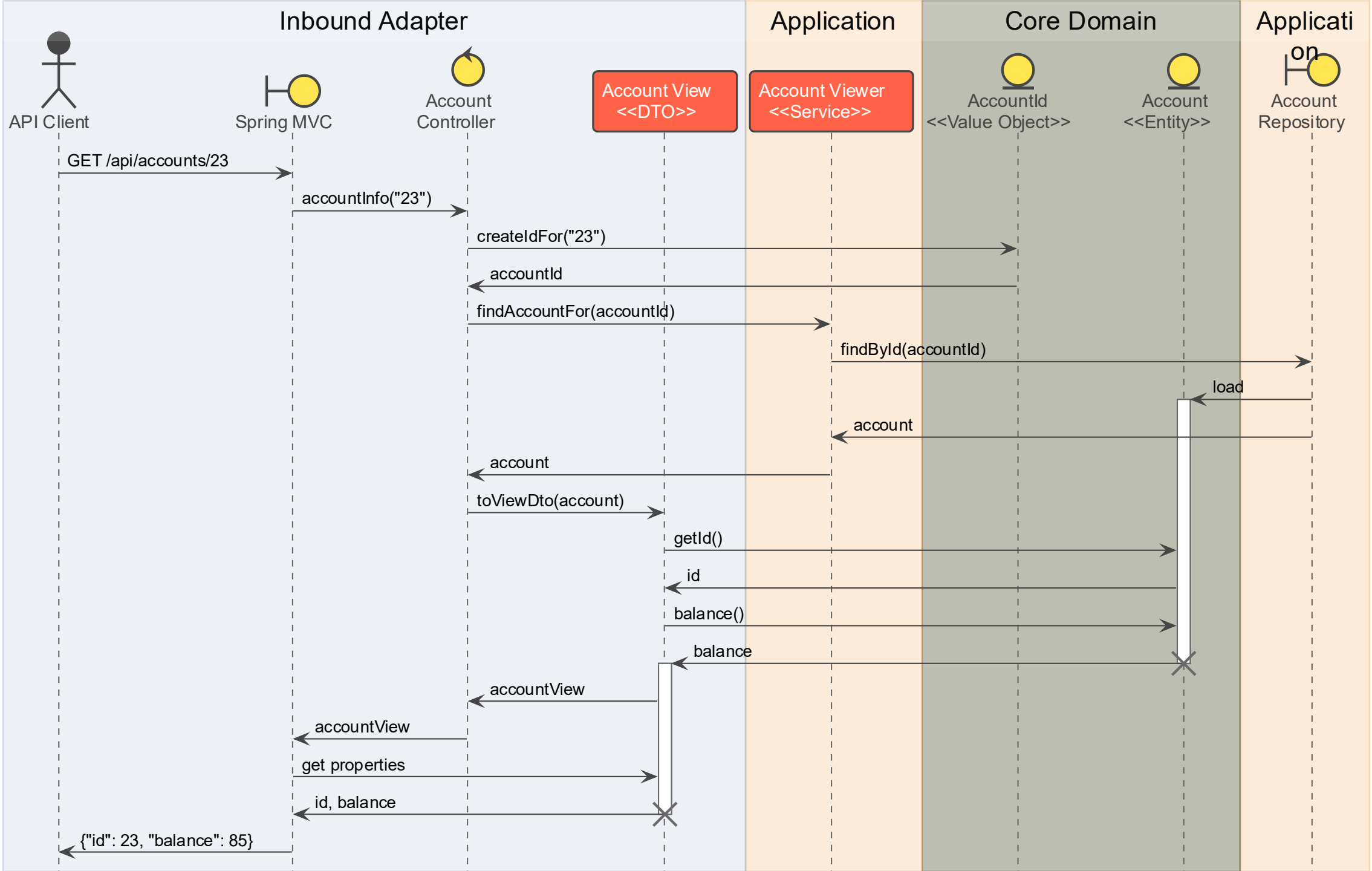
APPLICATION
&
DOMAIN
CODE

OUTBOUND

REQUESTS

EVENTS

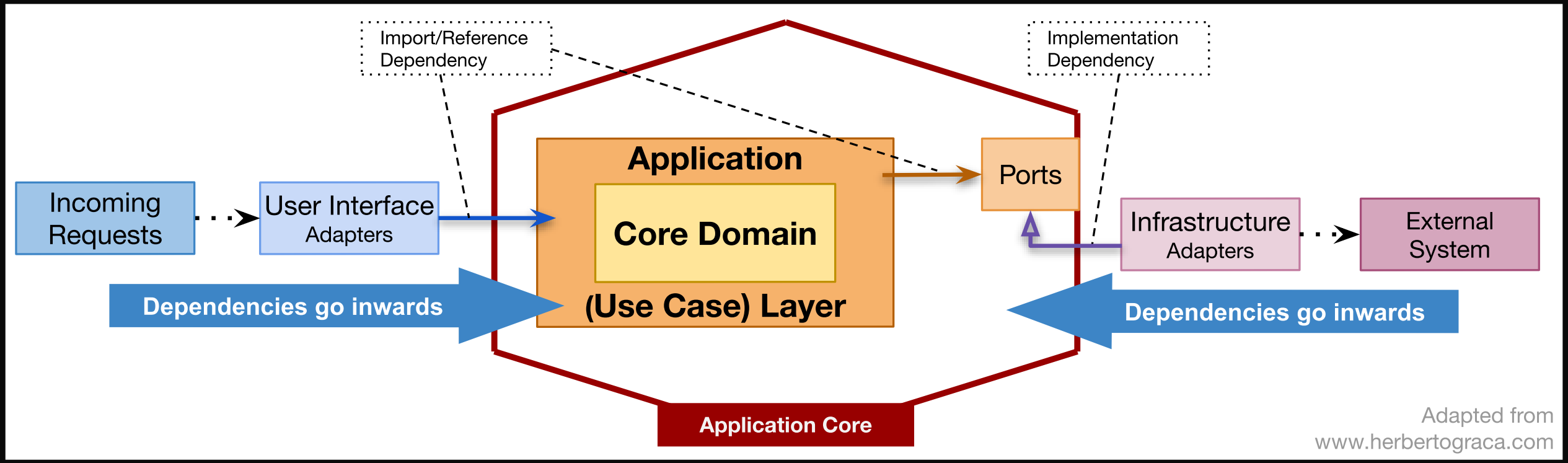

APPLICATION
(use case)
BOUNDARY
NO
AWARENESS
OF I/O
SYSTEM
BOUNDARY
External
Message
Bus
Application
Layer
DOMAIN
port
port
Unit
Tests
Persistence
Notifier
DEPENDENCIES
NO CONCRETE
I/O ONLY
ABSTRACTION
S
NO
DOMAIN OBJECTS
OUTSIDE THIS
BOUNDARY


Inbound Adapter
Application
Core Domain
Applicati
on
API Client
Spring MVC
Account
Controller
Account View
<<DTO>>
Account Viewer
<<Service>>
AccountId
<<Value Object>>
Account
<<Entity>>
Account
Repository
GET /api/accounts/23
accountInfo("23")
createIdFor("23")
accountId
findAccountFor(accountId)
findById(accountId)
load
account
account
toViewDto(account)
getId()
id
balance()
balance
accountView
accountView
get properties
id, balance
{"id": 23, "balance": 85}

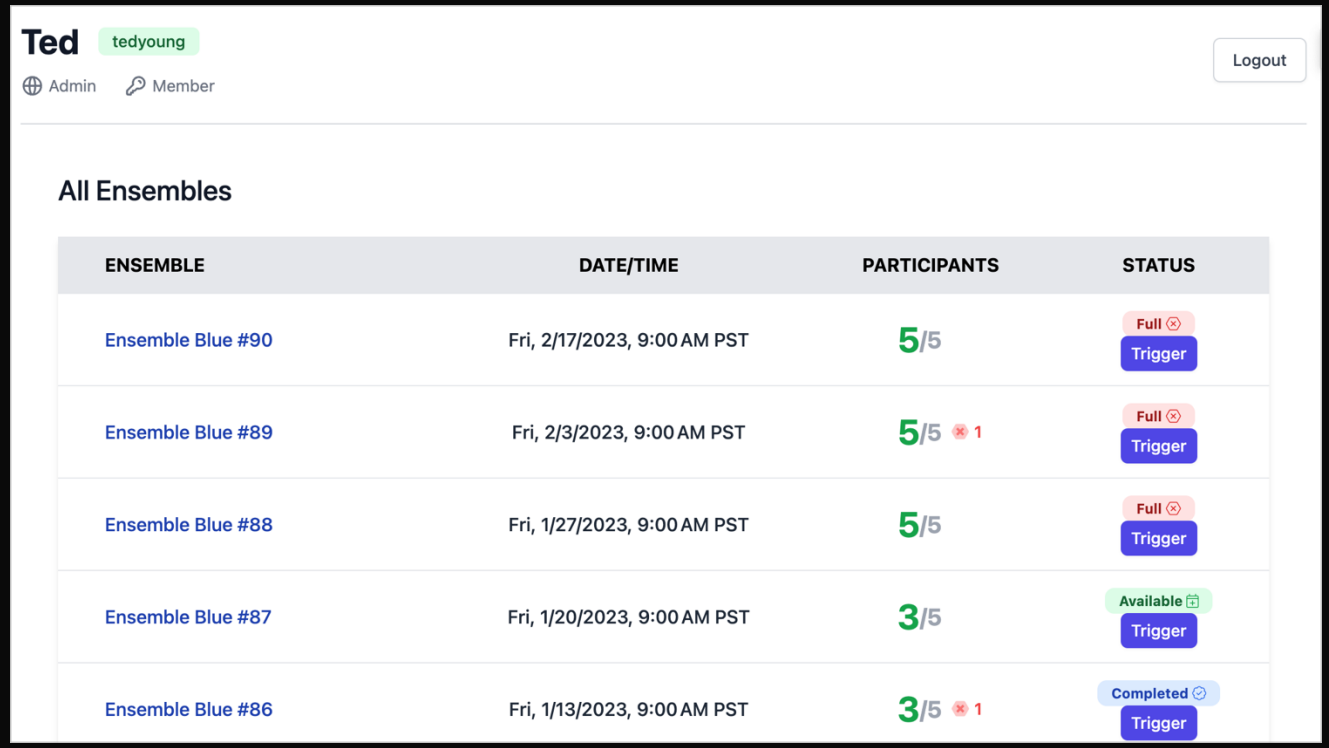
Dependency Rule

Dependencies Point Inwards

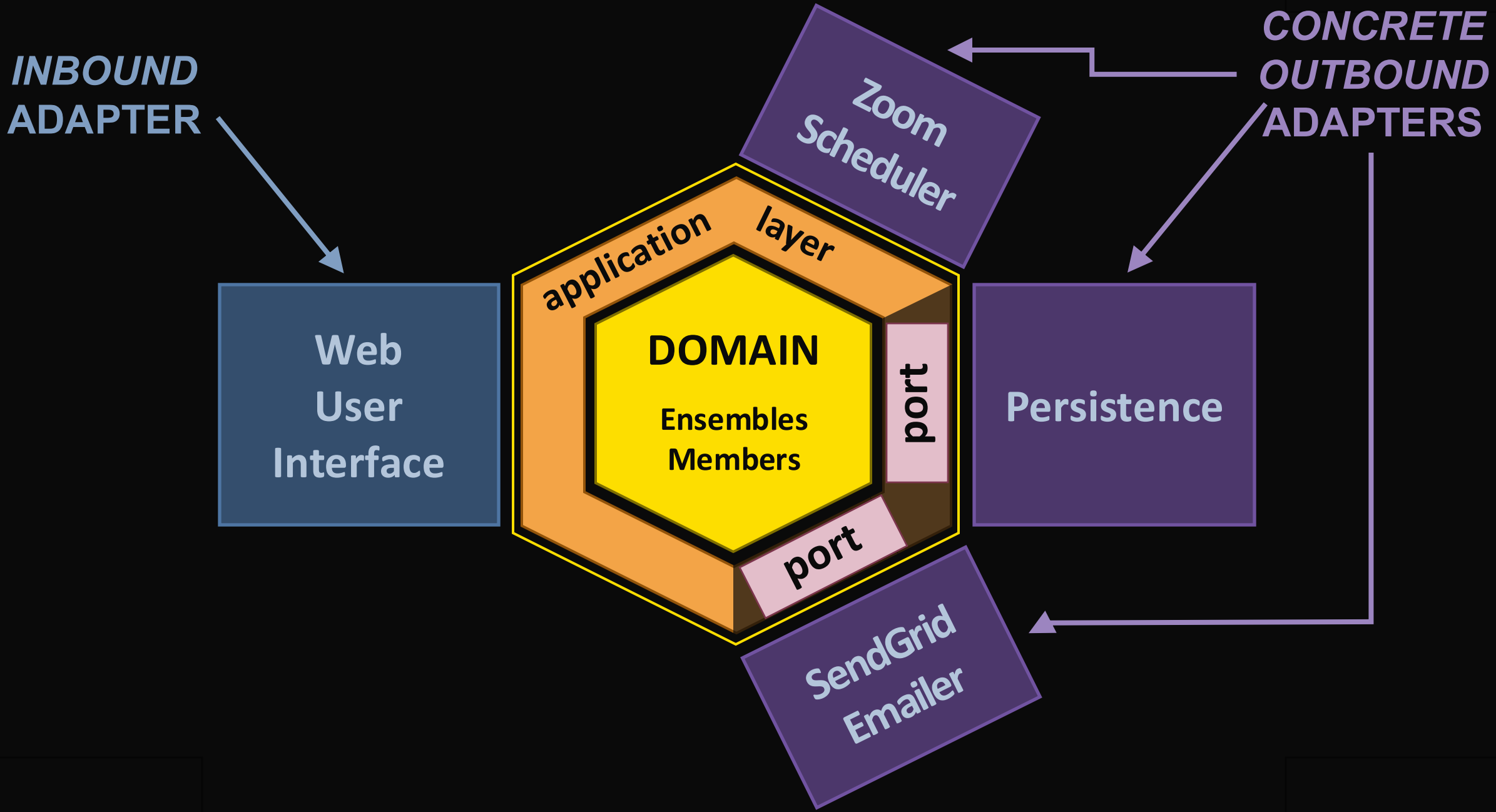
Dependencies Point Inward



ENSEMBLER
managing ensemble events

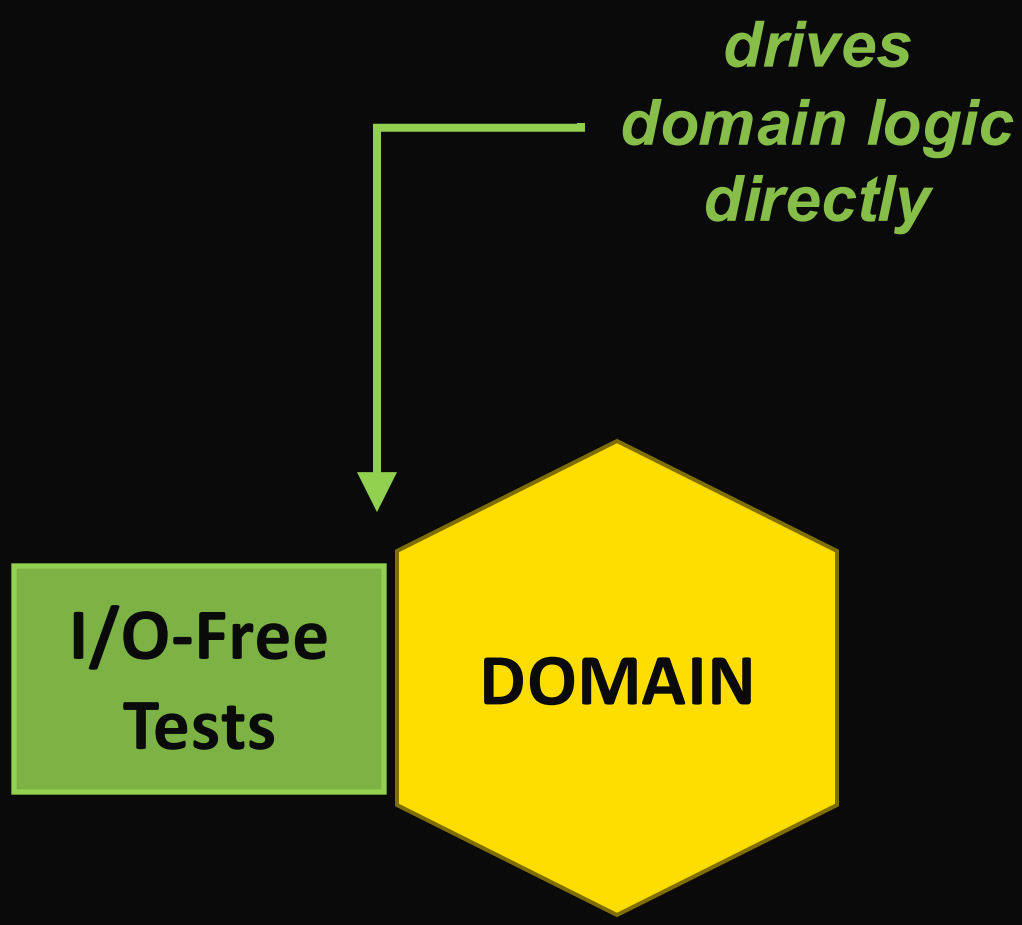


Source Code: https://github.com/jitterted/ensembler


INBOUND
ADAPTER
CONCRETE
OUTBOUND
ADAPTERS
Zoom
Scheduler
application
layer
DOMAIN
Ensembles
Members
port
port
Web
User
Interface
Persistence
SendGrid
Emailer

Types of Tests

Hexagonal+Domain-Driven Design (HexADDD)


drives
domain logic
directly
I/O-Free
Tests
DOMAIN

```java
@Test
void newEnsembleIsNotCompleted() throws Exception {
    Ensemble ensemble = new Ensemble("not completed", ZonedDateTime.now());

    assertThat(ensemble.isCompleted())
            .isFalse();
}

@Test
void whenCompletingEnsembleThenEnsembleIsCompleted() throws Exception {
    Ensemble ensemble = new Ensemble("completed", ZonedDateTime.now());

    ensemble.complete();

    assertThat(ensemble.isCompleted())
            .isTrue();
}

@Test
void whenCancelingScheduledEnsembleThenEnsembleIsCanceled() throws Exception {
    Ensemble ensemble = EnsembleFactory.withStartTimeNow();

    ensemble.cancel();

    assertThat(ensemble.isCanceled())
            .isTrue();
}
```

Testing Domain Directly

https://ted.dev/about

Test Doubles

aka "Mocks"

Collaborators

Code Under Test

Dependencies (aka *Collaborators*)

Direct vs. Indirect Inputs

Predictable Tests

Direct

Passed in Directly as Arguments

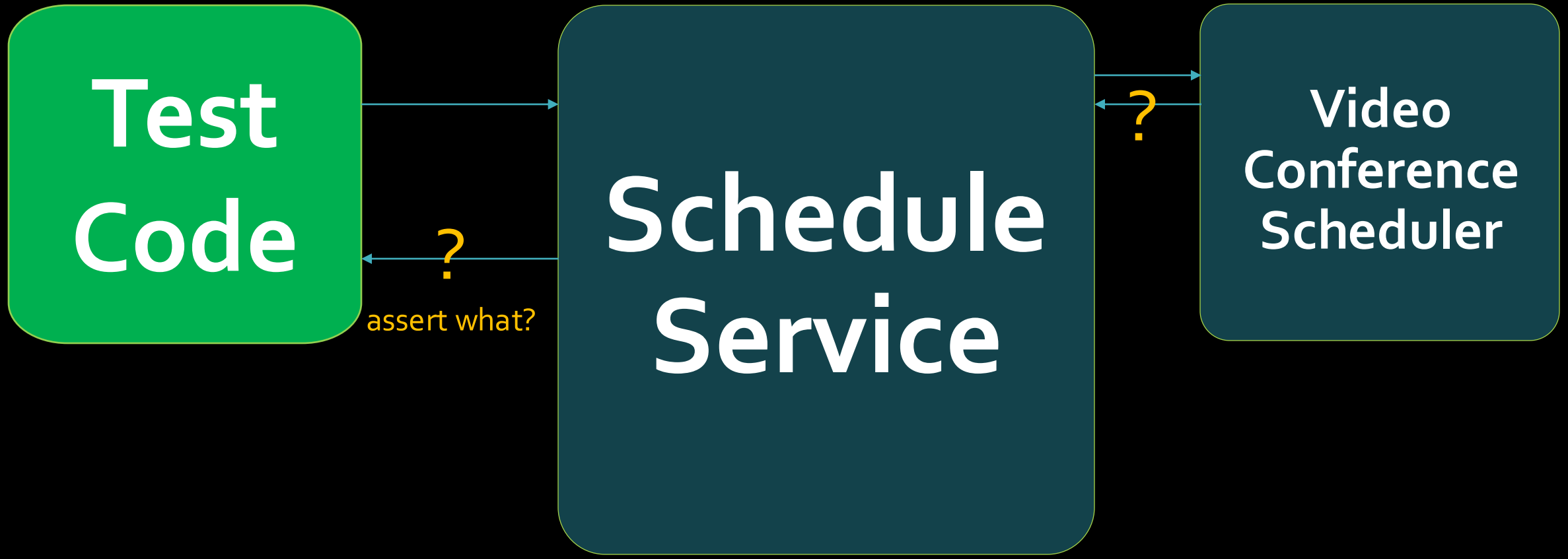
Direct vs. Indirect Inputs

Test Code

Code Under Test

https://ted.dev/about

Indirect Input Provider



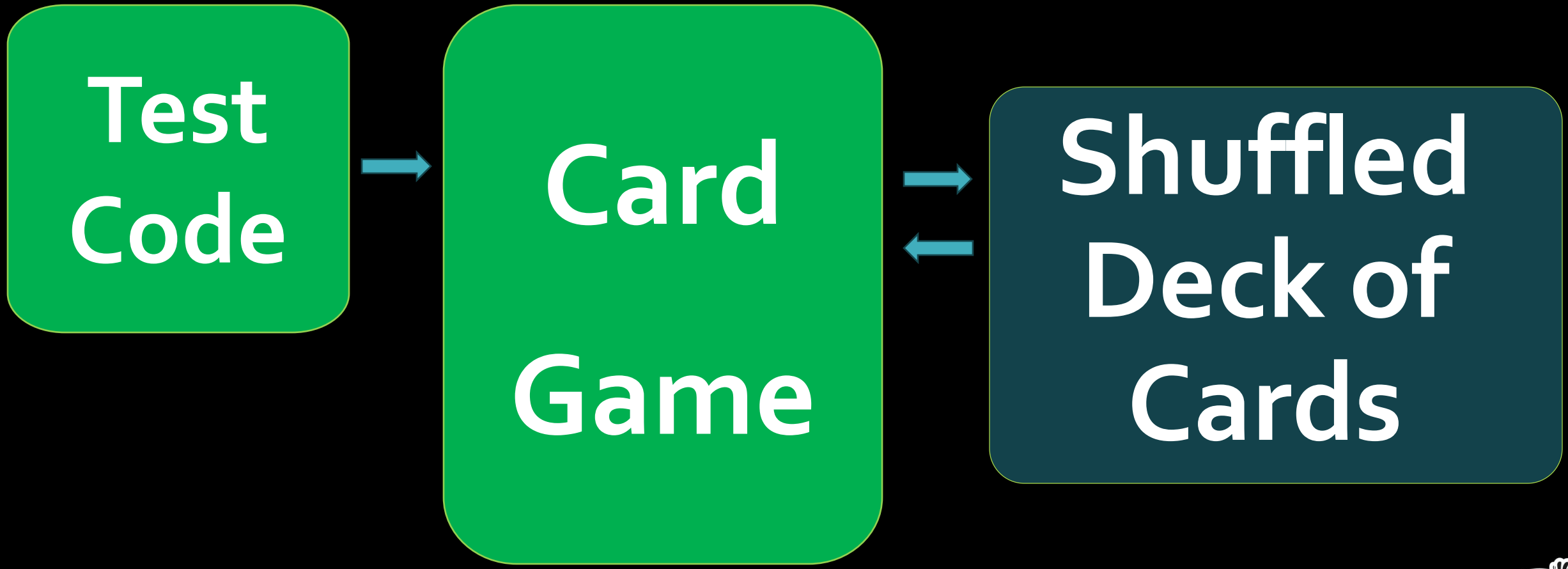
Collaborators

Card Game

Shuffled Deck of Cards

https://ted.dev/about

Unpredictable and Hidden Indirect Input Provider



https://ted.dev/about

Predictable, but Hidden Indirect Input Provider

Test Code

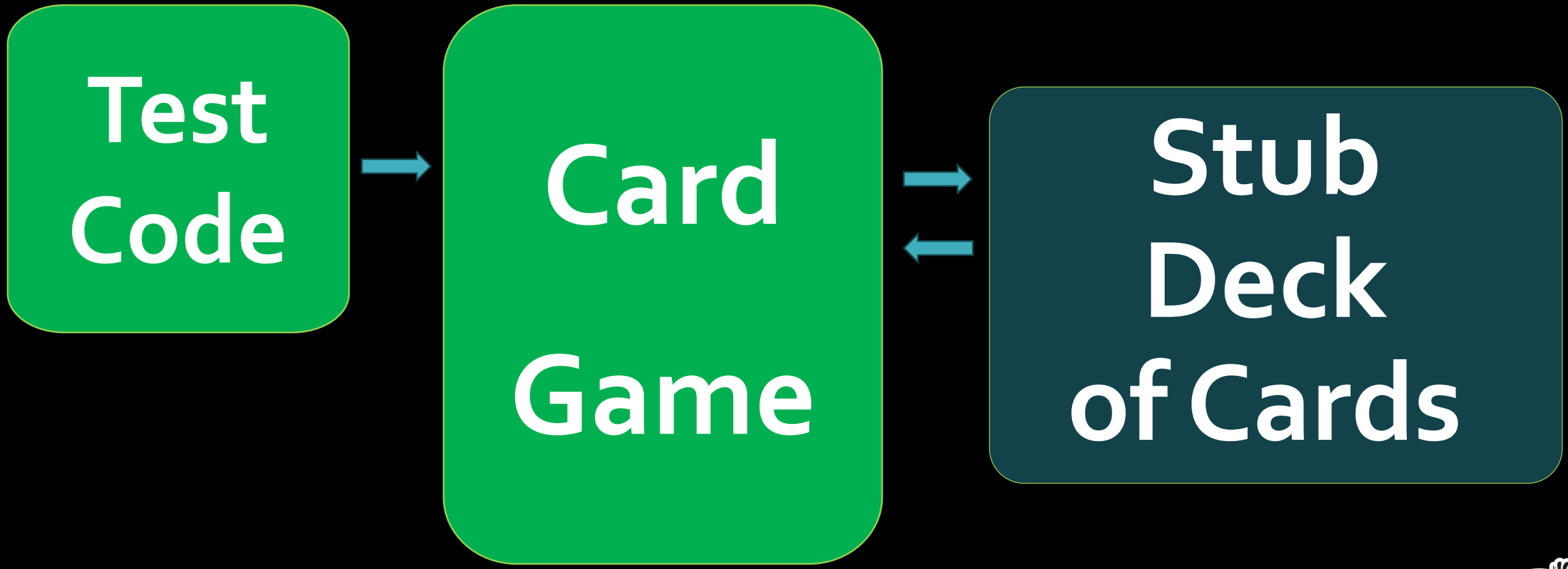
→

calculate total shipping weight (including box, and other packing materials like ice packs)

⇄

Weight of Shipping Boxes and Ice Packs (hidden) *configuration*

Predictable Indirect Input Provider



https://ted.dev/about

Predictable Outcomes

- Can't control real collaborators?
  - They are external to your application
  - They are *Unreliable* or *Unpredictable*
- Need "stand-in" to control:
- Test Doubles
  - From "Stunt Doubles" in movies

Two Types of Test Doubles

Verify State

Dummy

Stub

Fake

Verify Behavior

Spy

Mock

https://ted.dev/about

Dummy

Unused Placeholder

State: not **Read** from or **Written** to

Dummy

naming: **Dummy**Suit
DummyMemberRepository

Testing App Layer Using a Dummy

```java
@Test
void ensembleIdIsTranslatedFromDomainIntoView() {
    MemberService memberService = new DefaultMemberService(new DummyMemberRepository());
    Ensemble ensemble = new Ensemble("test", ZonedDateTime.now());
    ensemble.setId(EnsembleId.of(23));
    EnsembleDetailView ensembleDetailView = EnsembleDetailView.from(ensemble, memberService);

    assertThat(ensembleDetailView.id())
            .isEqualTo(23);
}
```

```java
public class DummyMemberRepository implements MemberRepository {
    @Override
    public Member save(Member member) {
        return null;
    }

    @Override
    public Optional<Member> findByGithubUsername(String githubUsername) {
        return Optional.empty();
    }

    @Override
    public Optional<Member> findById(MemberId memberId) {
        return Optional.empty();
    }

    @Override
    public List<Member> findAll() {
        return Collections.emptyList();
    }
}
```

Stub

Hard-coded Return Values

State: Only **Read** from

Programmable Stub

Configure Return Values in Test

State: Only **Read** from

Stub

naming:

- **Stub**Deck
- Card**Provider**
- Date**Supplier**

Using a Programmable Stub

```java
@Test
void ensembleScheduledThenZoomLinkFetchedFromApiHasConferenceDetails() {
    ConferenceDetails expectedConferenceDetails =
            new ConferenceDetails("123",
                                  URI.create("https://zoom.us/startUrl"),
                                  URI.create("https://zoom.us/joinUrl"));
    VideoConferenceScheduler stubScheduler =
            new StubConferenceScheduler(expectedConferenceDetails);
    EnsembleService ensembleService = EnsembleServiceFactory.with(stubScheduler);

    ensembleService.scheduleEnsembleWithVideoConference(
            "With Zoom", ZonedDateTime.now());

    assertThat(ensembleService.allEnsembles())
            .extracting(Ensemble::conferenceDetails)
            .containsOnly(expectedConferenceDetails);
}
```

```java
private static class StubConferenceScheduler implements VideoConferenceScheduler {
    private final ConferenceDetails stubConferenceDetails;

    public StubConferenceScheduler(ConferenceDetails stubConferenceDetails) {
        this.stubConferenceDetails = stubConferenceDetails;
    }

    @Override
    public ConferenceDetails createMeeting(Ensemble ensemble) {
        return stubConferenceDetails;
    }

    @Override
    public boolean deleteMeeting(ConferenceDetails conferenceDetails) {
        throw new UnsupportedOperationException();
    }
}
```

Fake

A Behavioral Mimic

State: **Read** from & **Written** to

Fake

naming:

FakeGameRepository

InMemoryMemberRepository

```java
public class InMemoryEnsembleRepository implements EnsembleRepository {
    private final Map<EnsembleId, Ensemble> ensembles = new ConcurrentHashMap<>();
    private final AtomicLong sequence = new AtomicLong(0);

    @Override
    public Ensemble save(Ensemble ensemble) {
        if (ensemble.getId() == null) {
            ensemble.setId(EnsembleId.of(sequence.getAndIncrement()));
        }
        ensembles.put(ensemble.getId(), ensemble);
        return ensemble;
    }

    @Override
    public List<Ensemble> findAll() {
        return List.copyOf(ensembles.values());
    }

    @Override
    public Optional<Ensemble> findById(EnsembleId ensembleId) {
        return Optional.ofNullable(ensembles.get(ensembleId));
    }

}
```

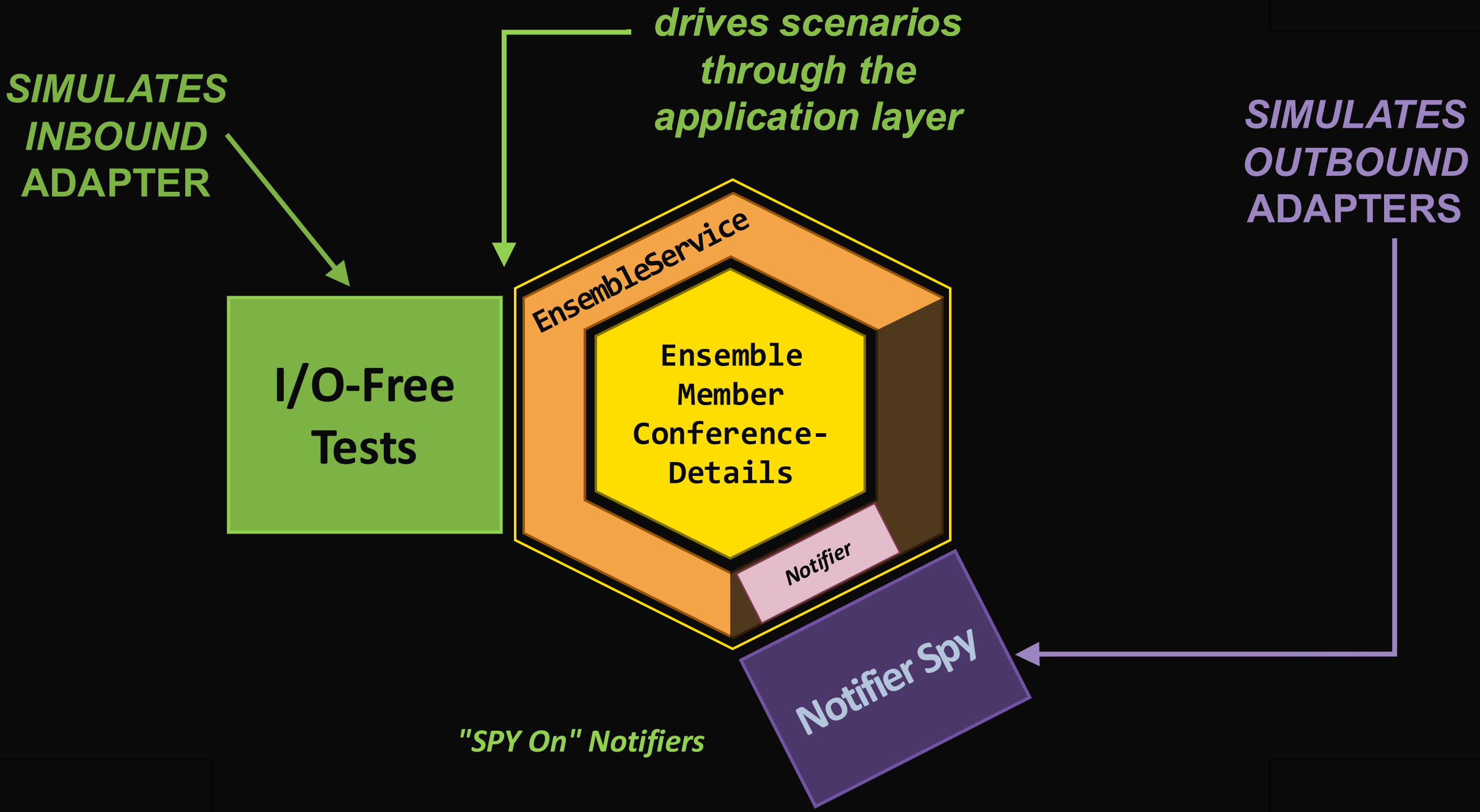
Using a Fake

https://ted.dev/about

Spy

Records Method Calls

Behavior: **Reports** what happened


drives scenarios through the application layer
SIMULATES INBOUND ADAPTER
SIMULATES OUTBOUND ADAPTERS
I/O-Free Tests
EnsembleService
Ensemble Member Conference-Details
Notifier
Notifier Spy
"SPY On" Notifiers

```java
    @Test
    void whenEnsembleScheduledEnsembleOpenNotificationIsSent() throws Exception {
        MockEnsembleScheduledNotifier mockEnsembleScheduledNotifier =
                new MockEnsembleScheduledNotifier();
        EnsembleService ensembleService = new EnsembleService(
                new InMemoryEnsembleRepository(),
                new InMemoryMemberRepository(),
                mockEnsembleScheduledNotifier,
                new DummyVideoConferenceScheduler());

        ensembleService.scheduleEnsemble("Notifying Ensemble",
                ZonedDateTime.of(2021, 11, 10, 17, 0, 0, 0, ZoneOffset.UTC));

        mockEnsembleScheduledNotifier.verify();
    }
```

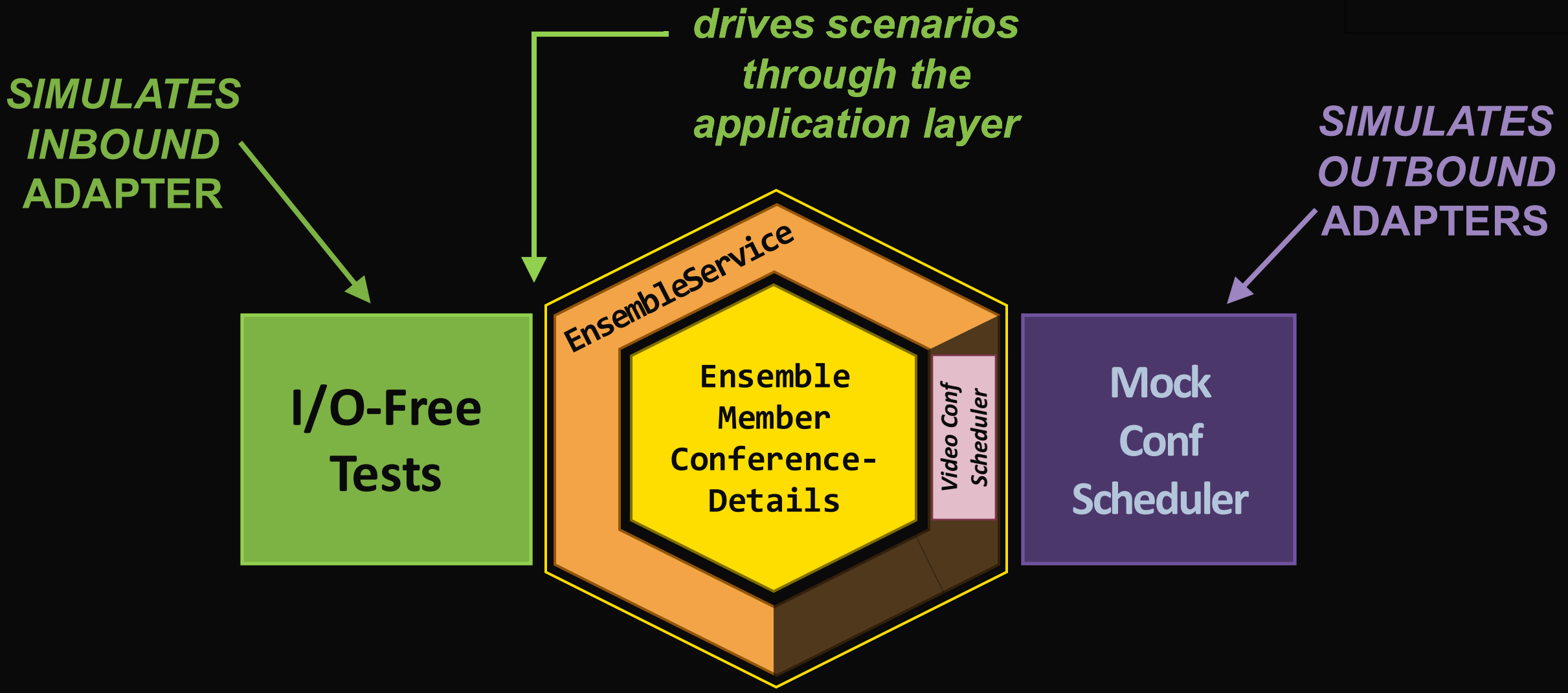
Testing App Layer: Notifier

https://ted.dev/about

Mock

A *Spy* That Verifies the Behavior

Behavior: **Asserts** What Happened


SIMULATES INBOUND ADAPTER
drives scenarios through the application layer
SIMULATES OUTBOUND ADAPTERS
I/O-Free Tests
EnsembleService
Ensemble Member Conference-Details
Video Conf Scheduler
Mock Conf Scheduler

```java
@Test
void canceledEnsembleDeletesVideoConferenceMeeting() throws Exception {
    Ensemble ensemble = new EnsembleBuilder()
            .withConferenceDetails("meetingId", "https://start.link", "https://join.link")
            .build();
    VideoConferenceScheduler succeedsIfMeetingIdMatchesEnsemble =
            new ZoomConferenceSchedulerDeletesExpectedMeetingId("meetingId");
    TestEnsembleServiceBuilder ensembleServiceBuilder =
            new TestEnsembleServiceBuilder()
                    .withVideoConferenceScheduler(succeedsIfMeetingIdMatchesEnsemble)
                    .saveEnsemble(ensemble);
    EnsembleId ensembleId = ensembleServiceBuilder.lastSavedEnsembleId();
    EnsembleService ensembleService = ensembleServiceBuilder.build();

    ensembleService.cancel(ensembleId);

    assertThat(ensemble.conferenceDetails())
            .isEqualTo(ConferenceDetails.DELETED);
}
```

Testing App Layer: Conference Scheduler

https://ted.dev/about

When You Need Test Doubles

Testing Domain Code?

Never!

Testing With Infra?

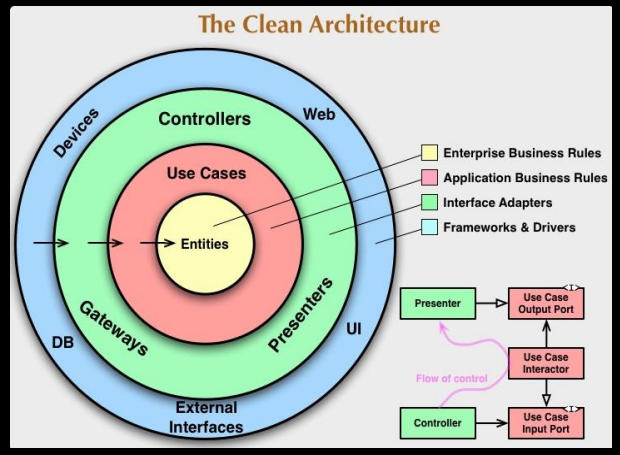
Yes.

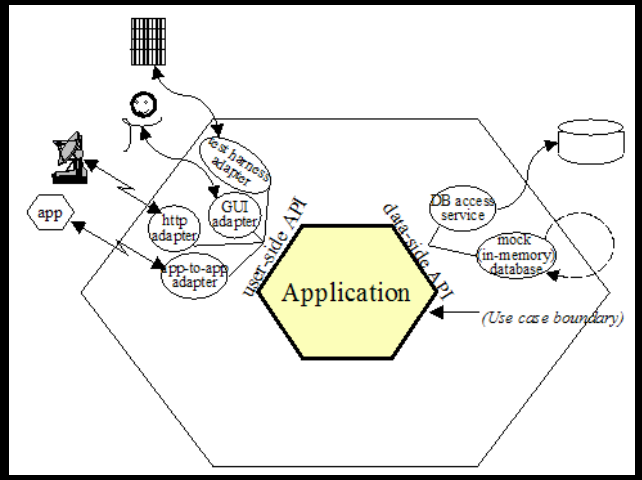
Also test
the REAL things

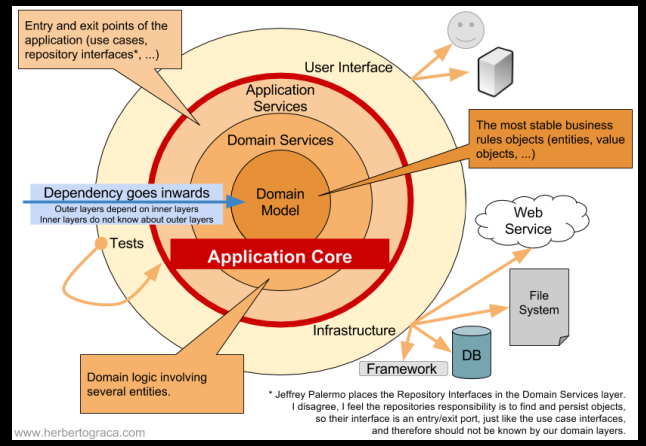
It's All Trade-offs

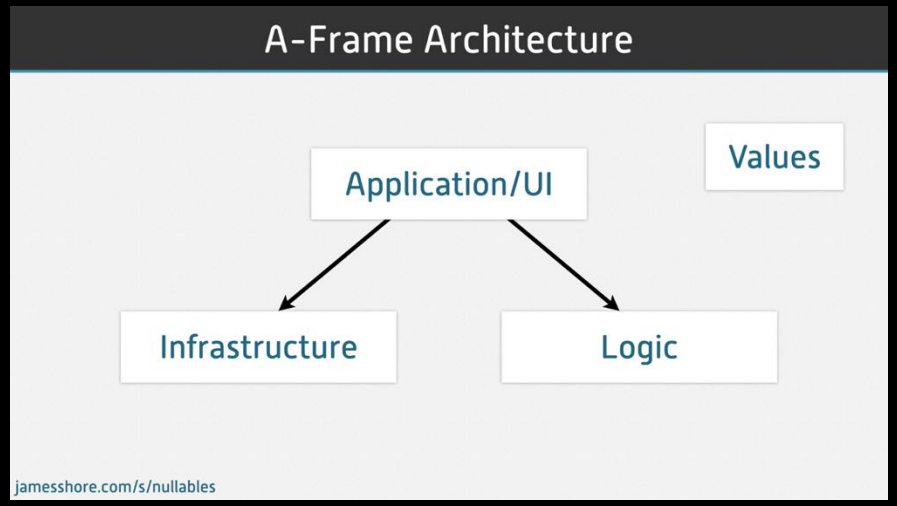
Increase Testability → Lose Simplicity

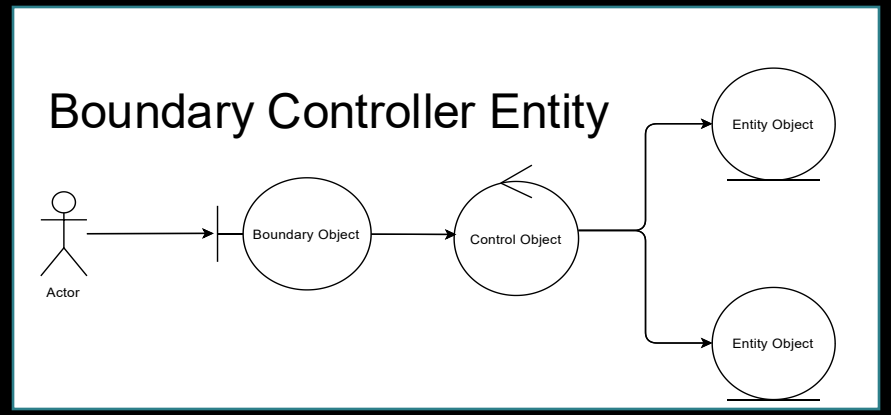
Use a Testable Architecture…

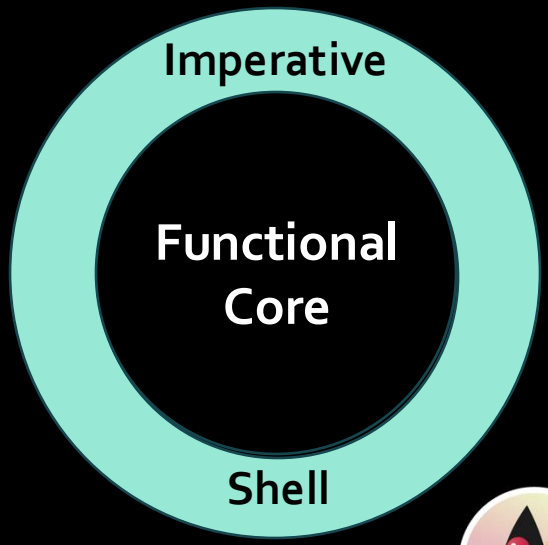












https://ted.dev/about

Ted M. Young
Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com
About me: https://ted.dev/about

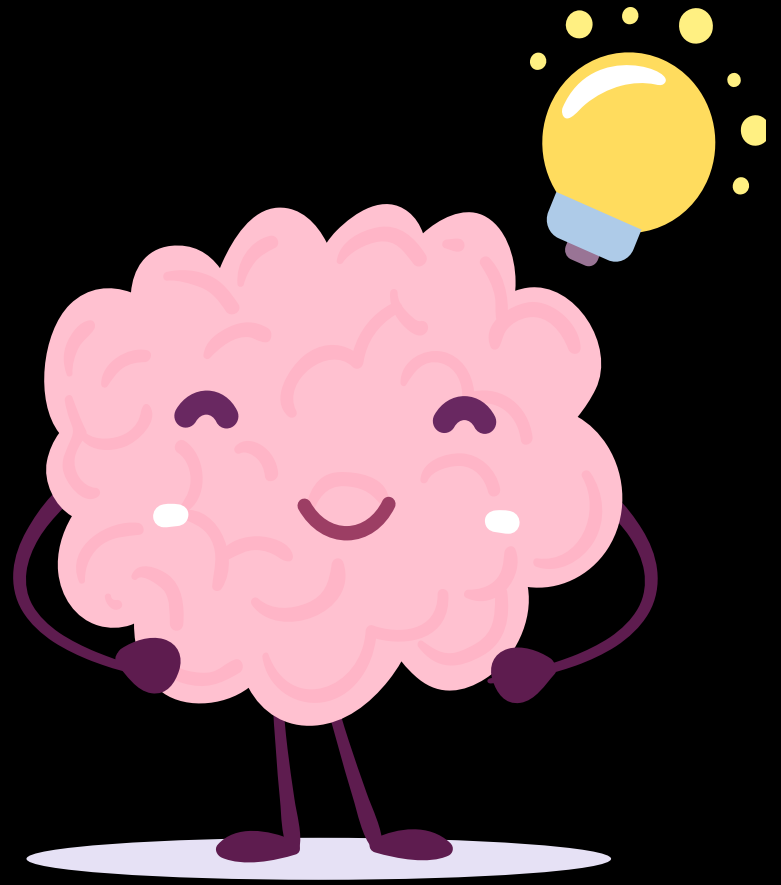
Want Your Code to be Easier to Test?

Keep 'em [I/O] Separated

Ted M. Young
Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com
About me: https://ted.dev/about

What other questions do you have?



Ted M. Young
Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com
Twitter: @JitterTed
Twitch: https://JitterTed.Stream
YouTube: https://JitterTed.TV
About me: https://ted.dev/about

Thank You...

Source Code? Slides? **https://ted.dev/talks/**