

# One Architecture to Rule them All

## Why Hexagons are so Awesome



**TINY  
CLARIFICATIONS**

TINYCLARIFICATIONS.COM



**TINY  
VALIDATIONS**

TINYVALIDATIONS.COM



**Ted M. Young**  
Java Trainer/Coach

**Spiral Learning, LLC**  
ted@SpiralLearningLLC.com

@JitterTed (Twitter & Twitch)  
<https://ted.dev> (Web)

# Ted M. Young

ted@SpiralLearningLLC.com

I Can Help Your Team...

Write more Testable code  
with more Effective unit tests

Become better  
developers

Be more productive in  
**Java & Spring**

Effectively use  
**TDD**

Apply  
**eXtreme**  
**Programming**

# Make Your Code More Testable

**with the Hexagonal Architecture**

**Ted M. Young**  
Java Trainer/Coach

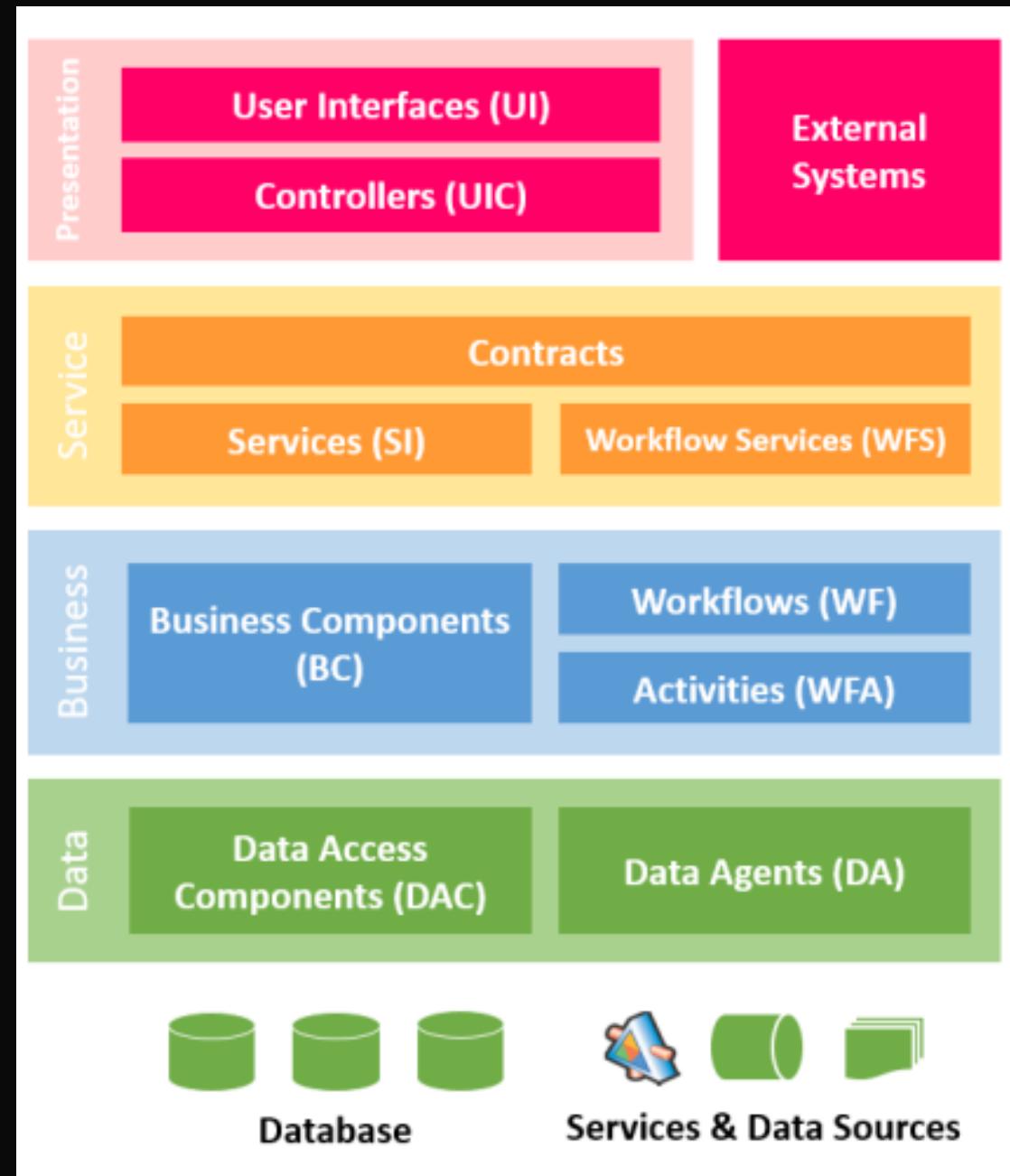
**Spiral Learning, LLC**  
[ted@SpiralLearningLLC.com](mailto:ted@SpiralLearningLLC.com)

**@JitterTed** (Twitter & Twitch)  
<https://ted.dev> (Web)

# Vertically- Layered Architecture

*aka  
Cake Architecture*

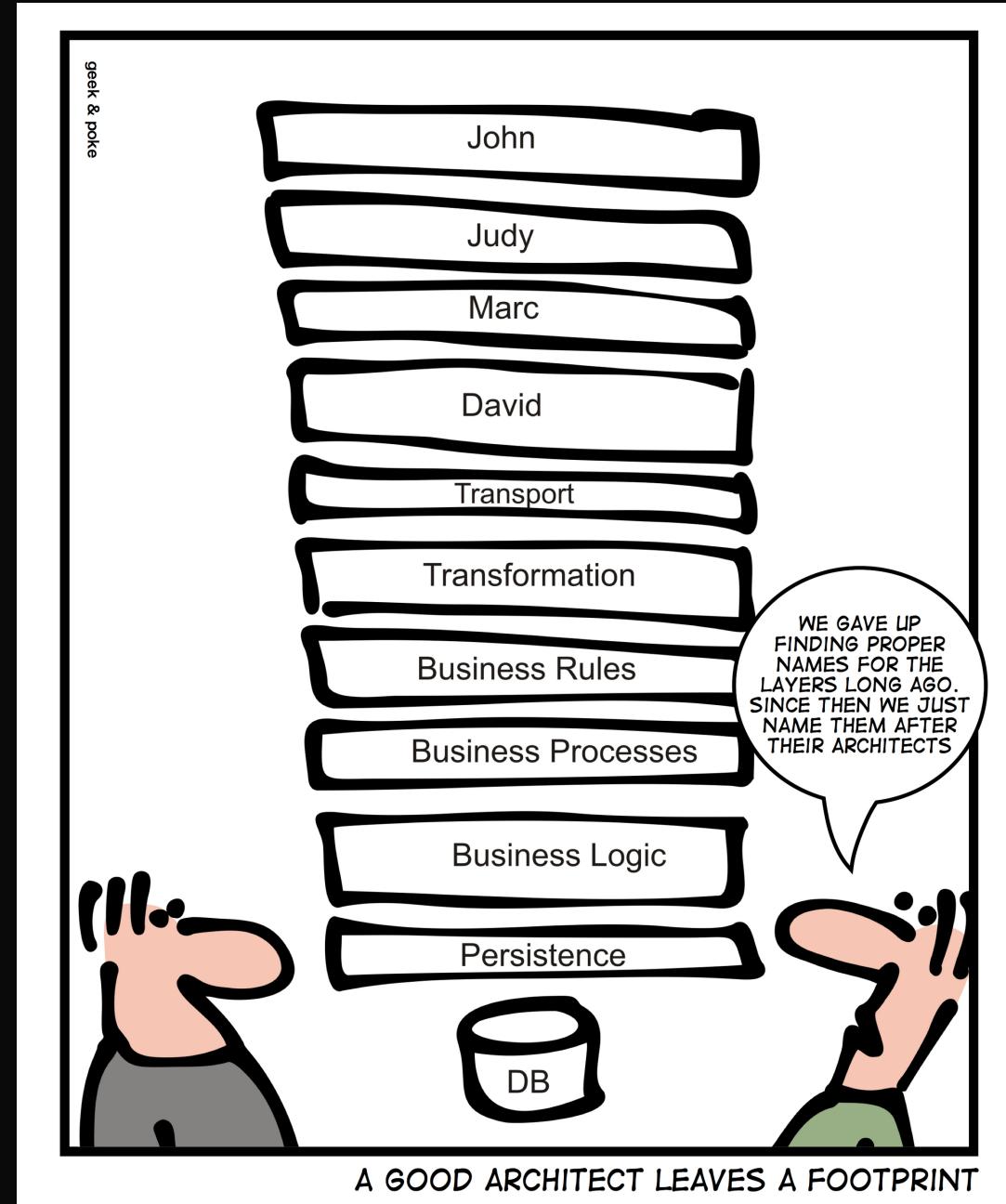
Adapted from  
<https://blogs.msdn.microsoft.com/malaysia/2013/08/02/layered-architecture-for-net/>



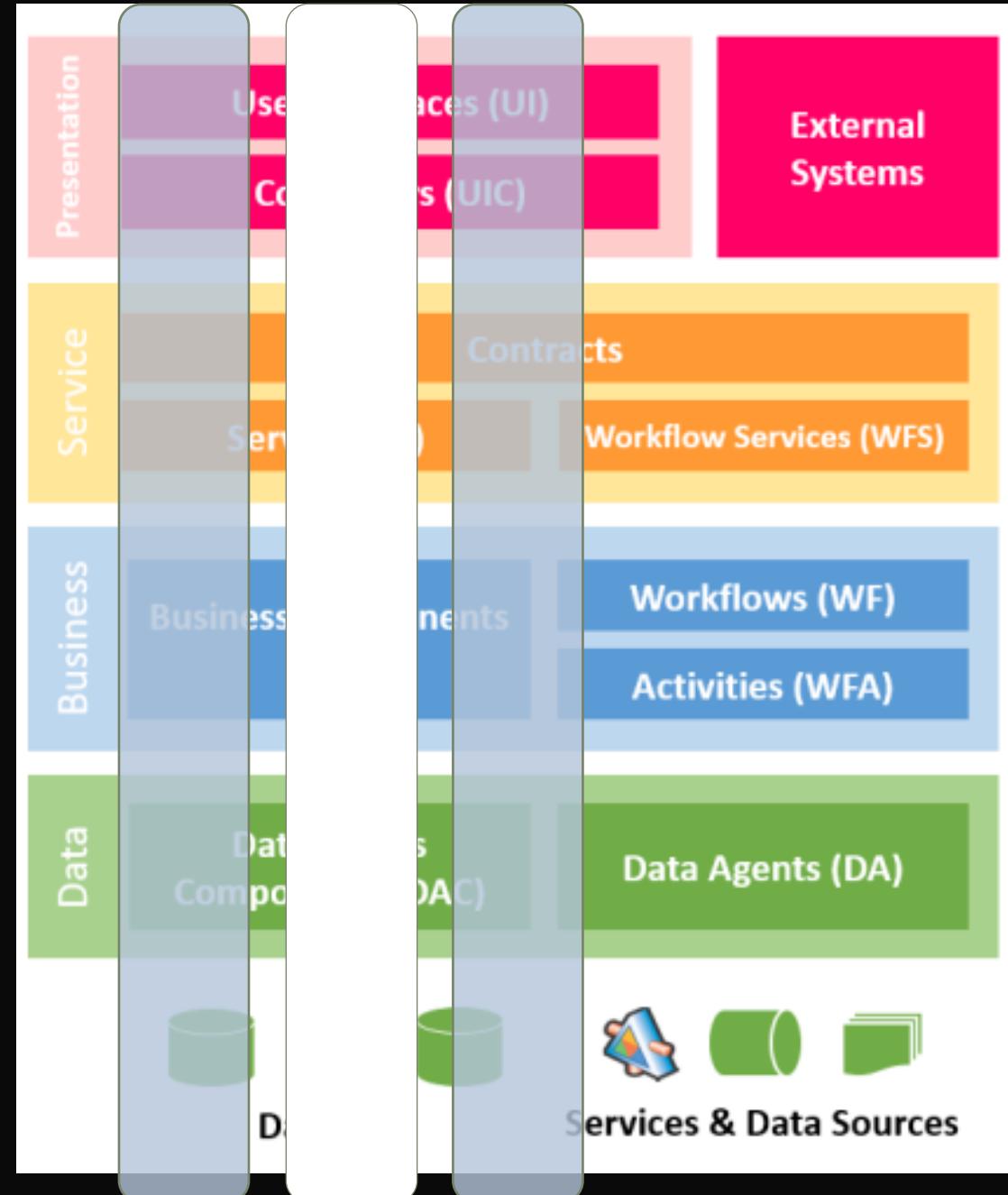
# Database-Centric instead of Business-Centric

# Vertically-Layered Architecture

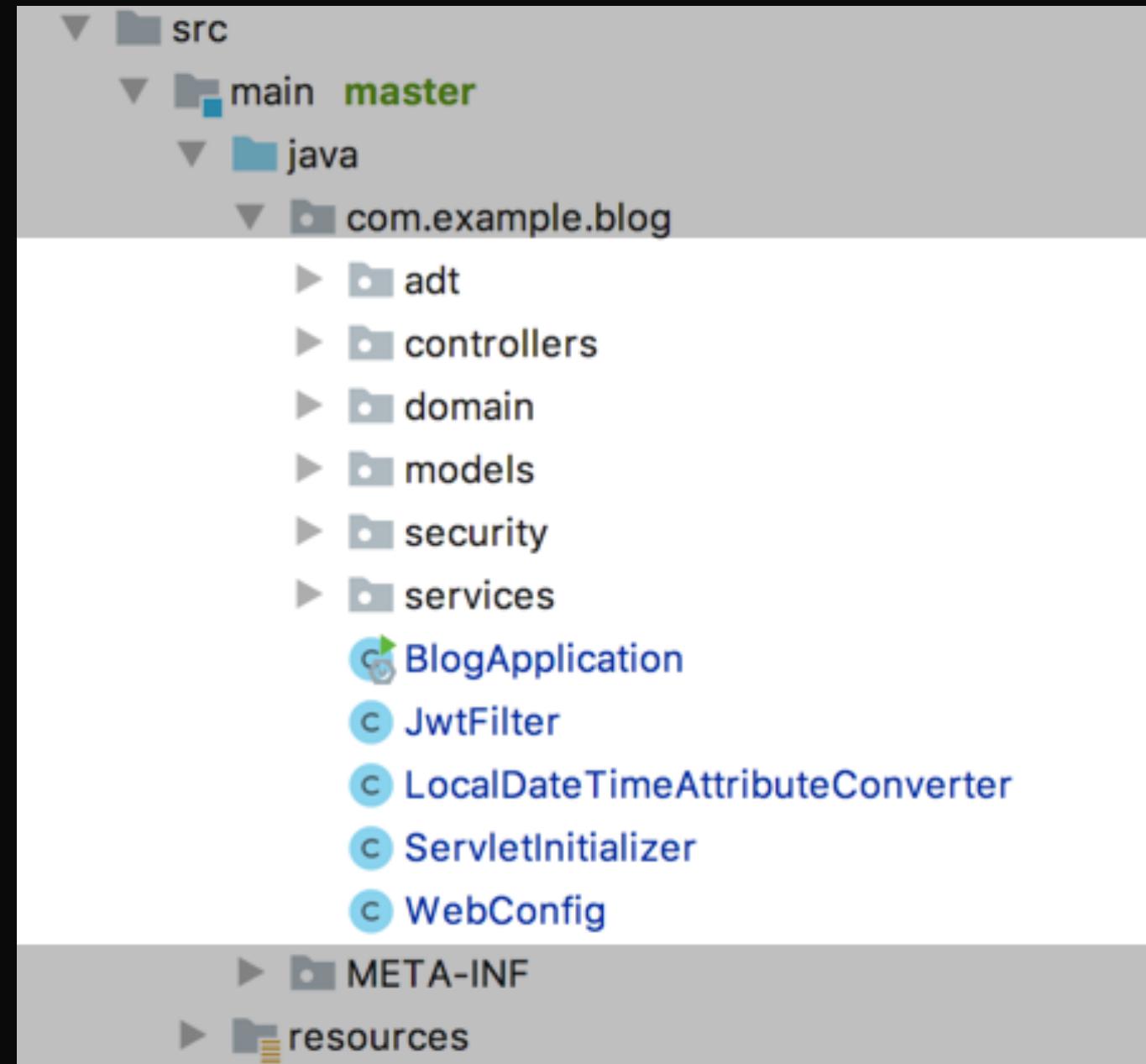
"Footprints" from Geek & Poke  
[http://geek-and-poke.com/  
geekandpoke/2013/7/13/foodprints](http://geek-and-poke.com/geekandpoke/2013/7/13/foodprints)  
Licensed CC-BY 2.0



# Thin Vertical Slices

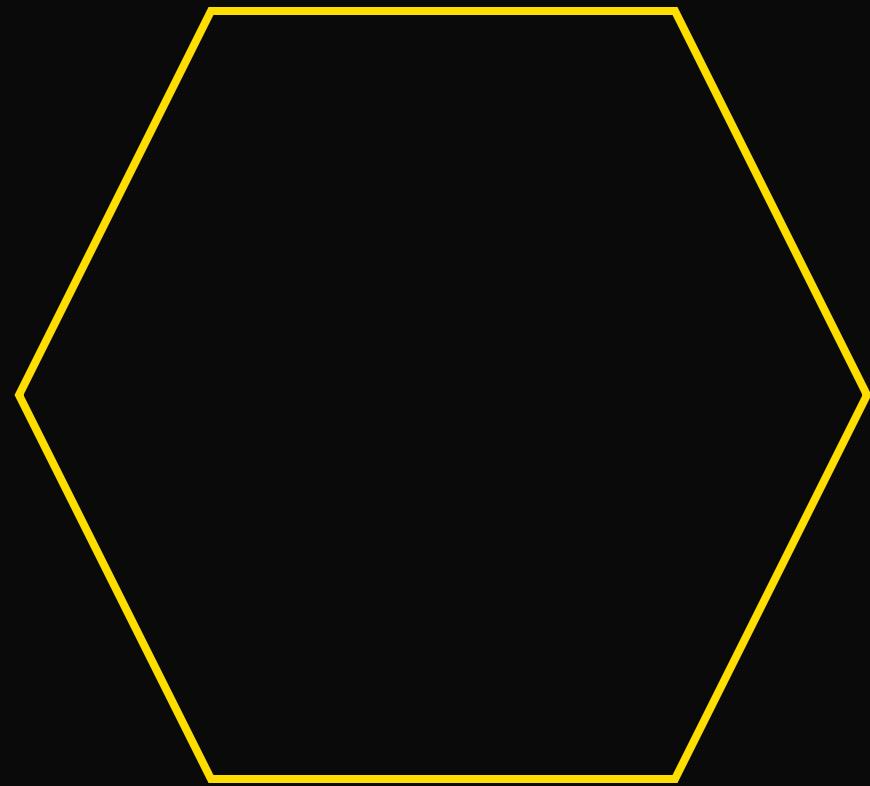


# Vertical Package Structure



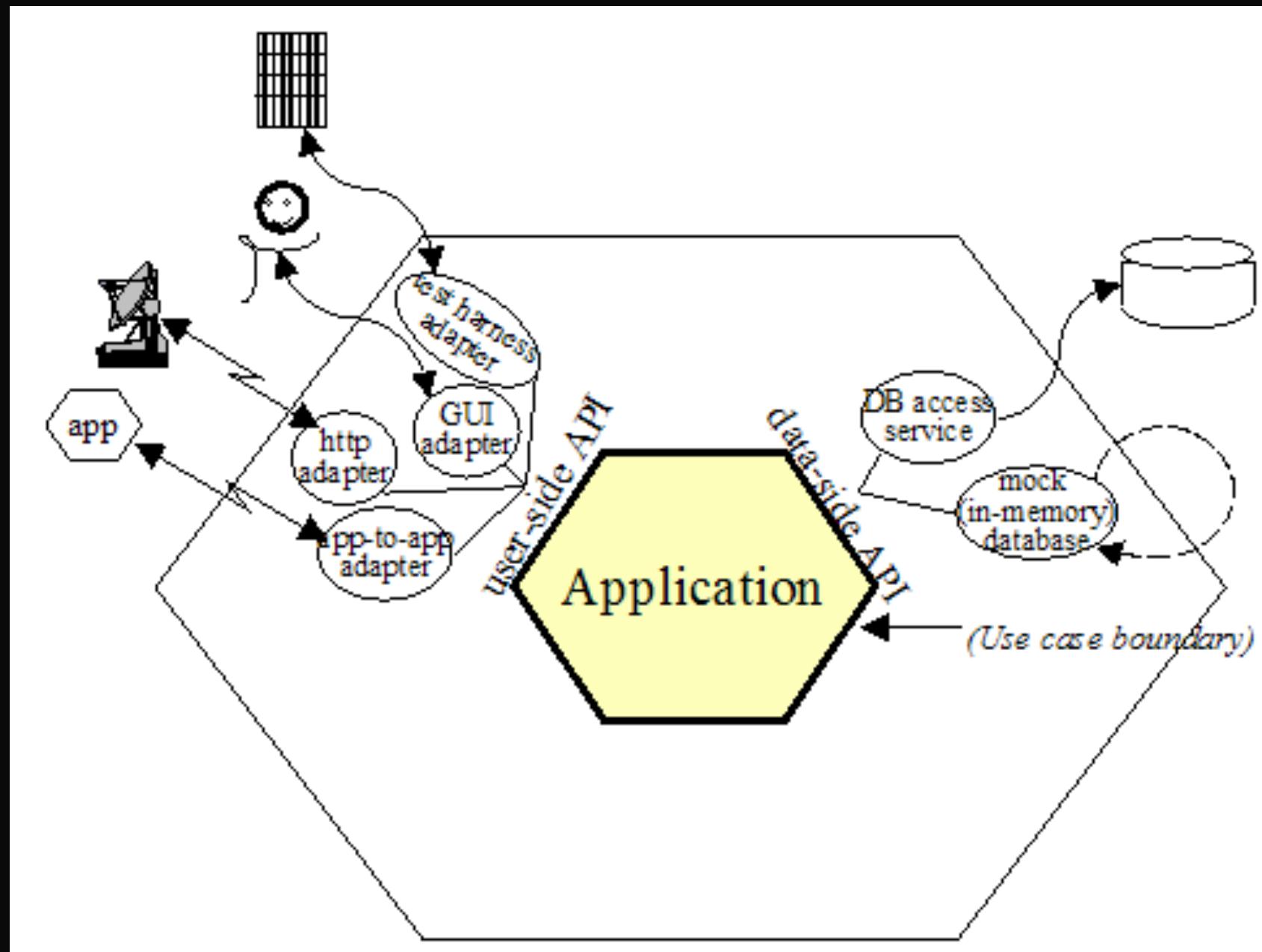


# Hexagonal Architecture



**“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

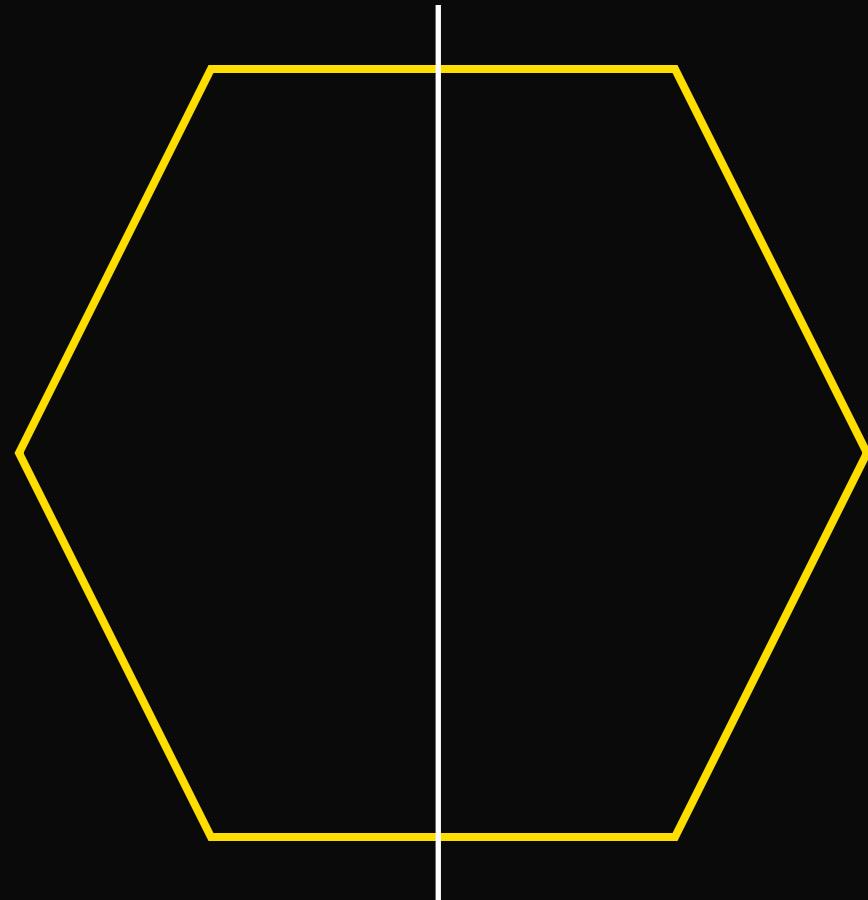
OUTSIDE

OUTSIDE

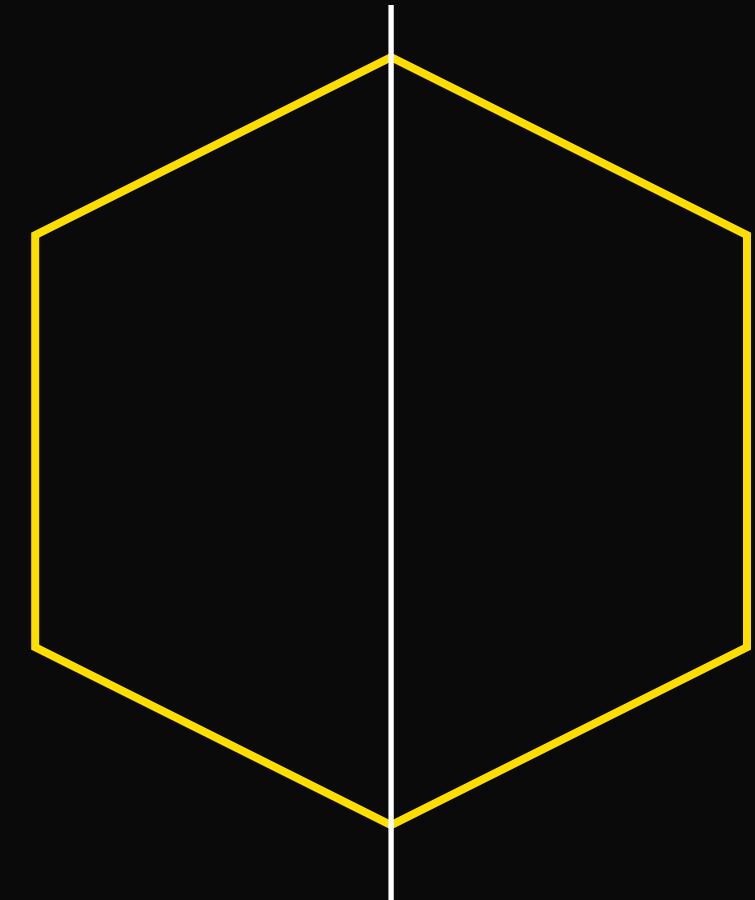
INSIDE

OUTSIDE

**FLAT-TOPPED**



**POINTY-TOPPED**



OUTSIDE

OUTSIDE

OUTSIDE

INSIDE

OUTSIDE

TRIGGERS

OUTCOMES

DRIVING

INSIDE

DRIVEN

MESSAGES

EVENTS



**Live Coding:** <https://Twitch.tv/jitterted>

**Season 1 Videos:** <https://ted.dev/youtube>

**Source Code:** <https://github.com/tedyoung/kid-bank>

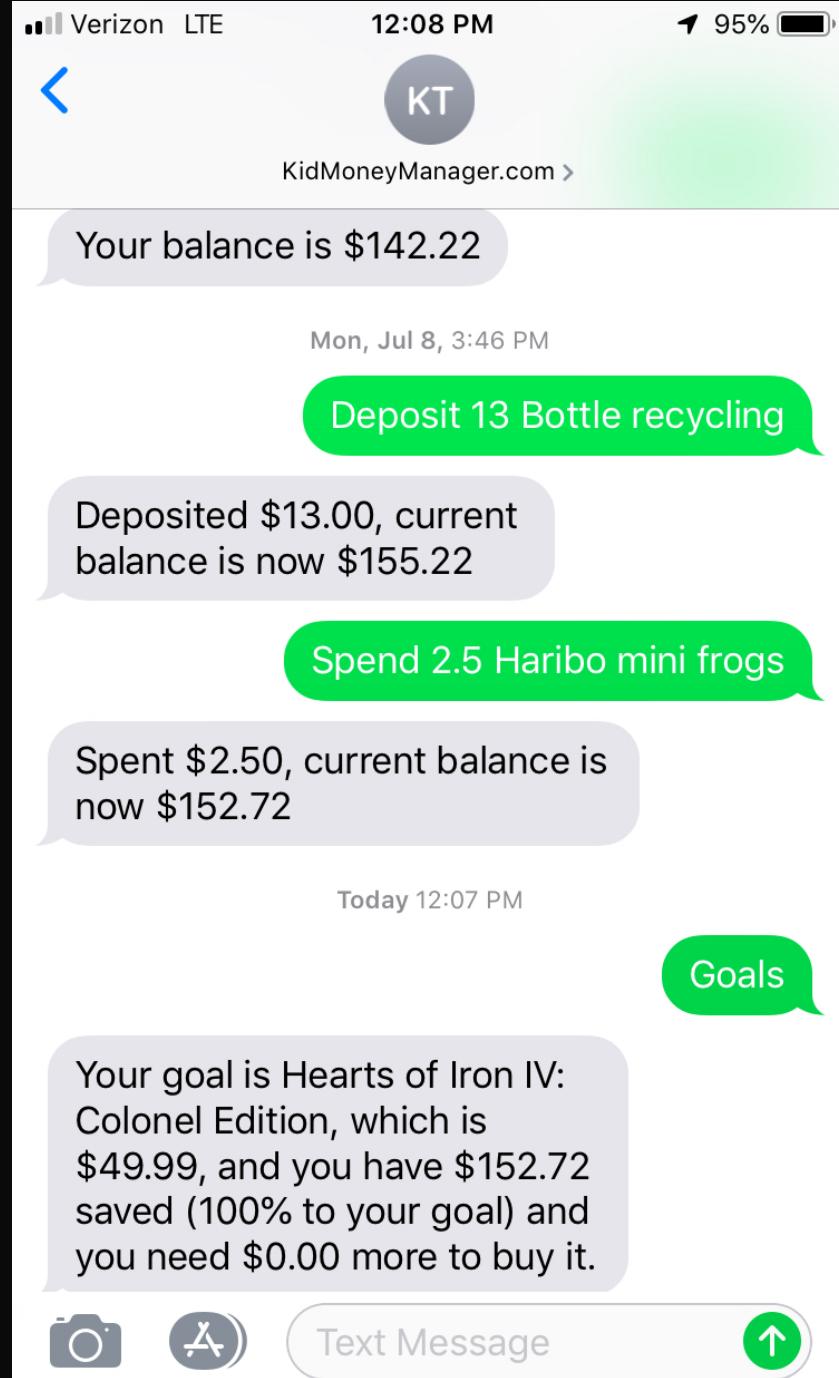
KidBank Balance

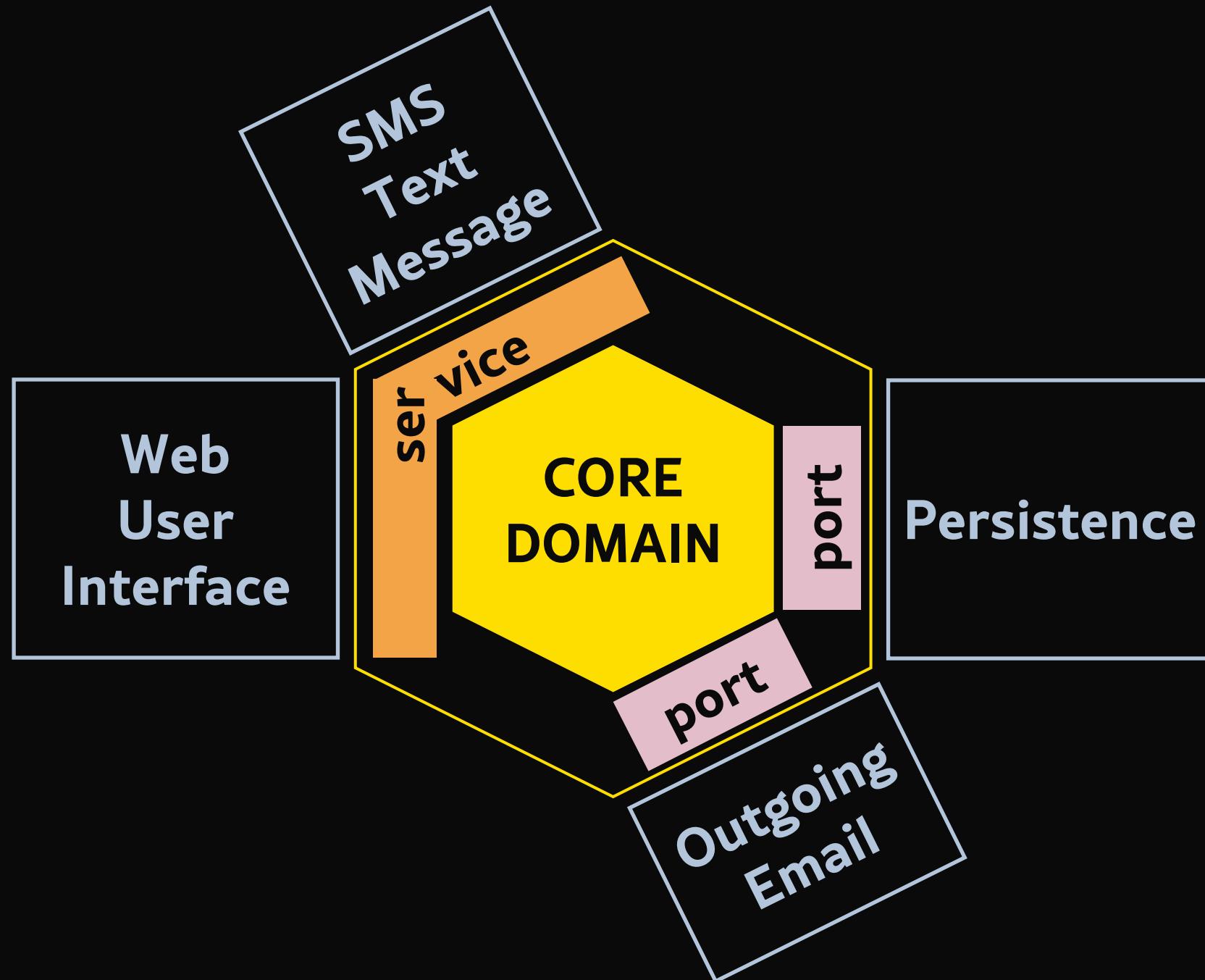
<https://kidmoneymanager.com>

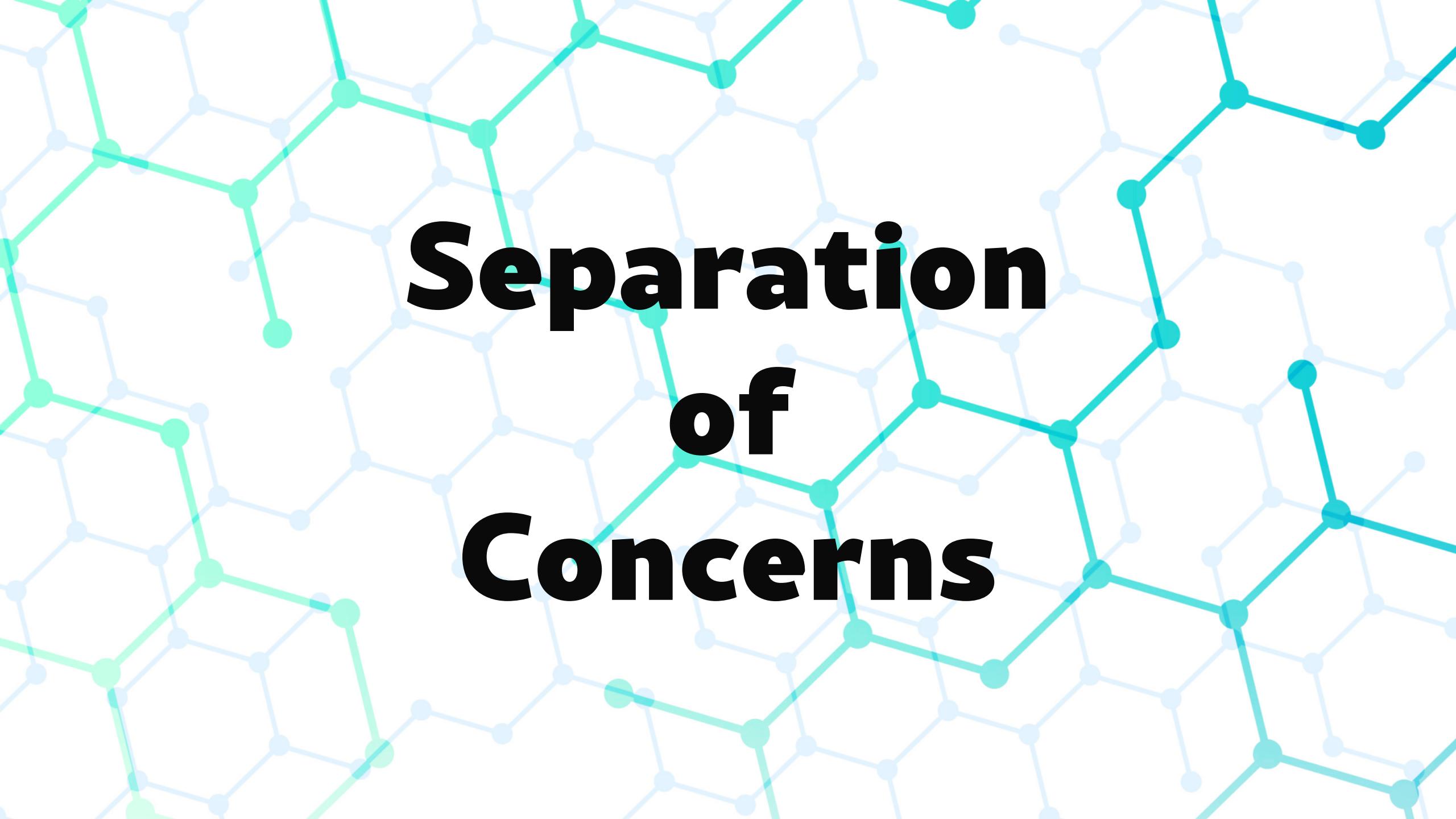
ACCOUNT BALANCE  
**\$153.04**

**DEPOSIT MONEY**   **SPEND MONEY**   **IMPORT CSV**   [Logout](#)

| Date       | Action          | Description                 | Amount   |
|------------|-----------------|-----------------------------|----------|
| 07/08/2019 | Spend           | Haribo mini frogs           | \$2.50   |
| 07/08/2019 | Deposit         | Bottle recycling            | \$13.00  |
| 07/06/2019 | Deposit         | SMS message                 | \$100.00 |
| 07/06/2019 | Spend           | SMS message                 | \$5.00   |
| 07/01/2019 | Interest Credit | Interest Credit             | \$0.10   |
| 06/01/2019 | Interest Credit | Interest Credit             | \$0.10   |
| 05/21/2019 | Deposit         | Traded for Starbucks card   | \$10.00  |
| 05/01/2019 | Interest Credit | Interest Credit             | \$0.08   |
| 04/11/2019 | Spend           | SMS message                 | \$9.99   |
| 04/01/2019 | Interest Credit | Interest Credit             | \$0.10   |
| 03/24/2019 | Spend           | Cookie                      | \$2.50   |
| 03/01/2019 | Interest Credit | Interest Credit             | \$0.10   |
| 02/01/2019 | Interest Credit | Interest Credit             | \$0.10   |
| 01/15/2019 | Deposit         | Unknown                     | \$38.00  |
| 01/05/2019 | Spend           | MTG Cards                   | \$13.00  |
| 01/04/2019 | Spend           | Share of Corsair headphones | \$30.00  |







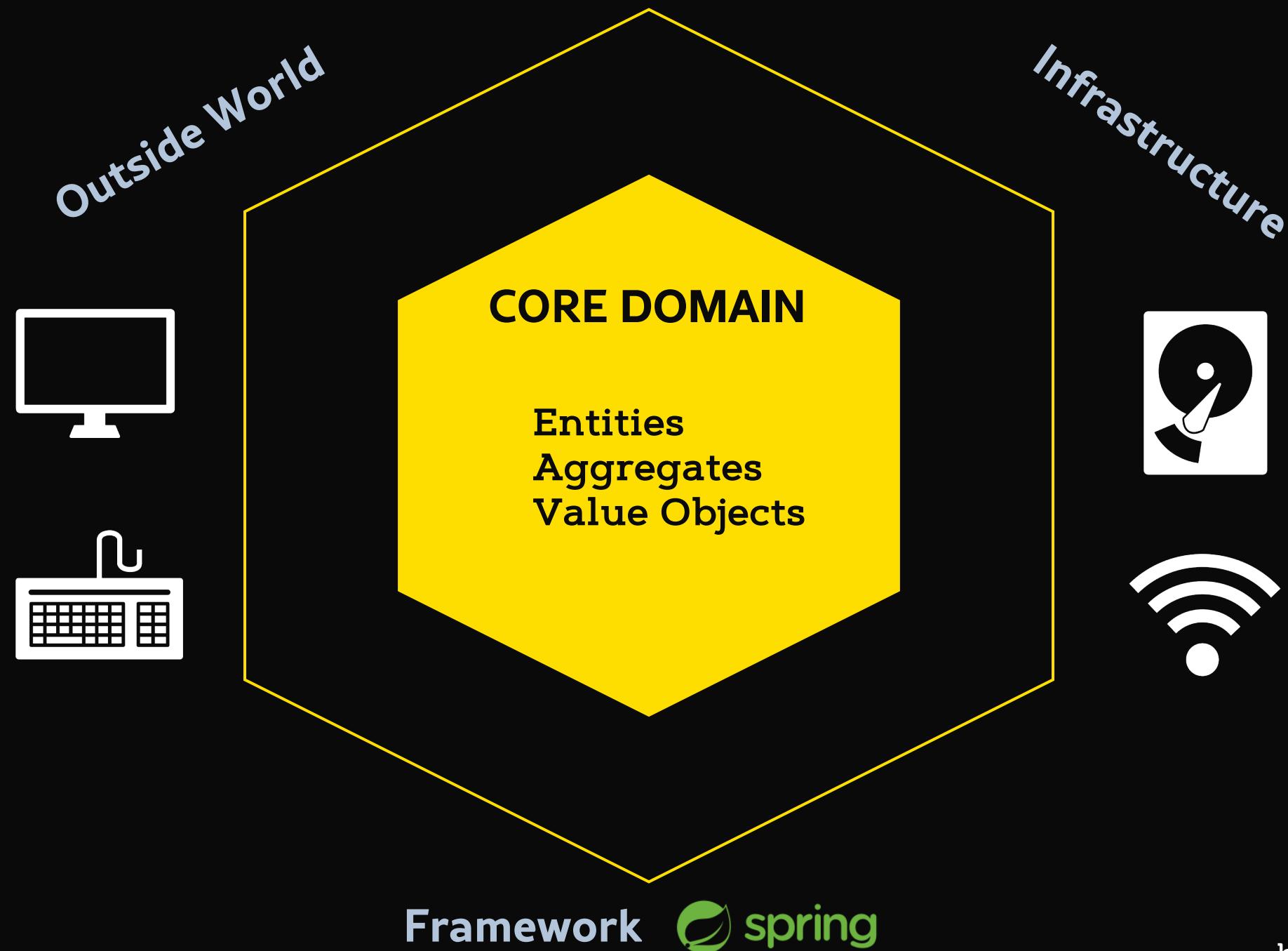
# **Separation of Concerns**

**“[Separation of Concerns]  
is a matter about which a  
great deal is said and very  
little done.”**

**– Charles Dudley Warner**

## Principle

**Separate Infrastructure and  
the Outside World from  
Business Logic**



# Talking to the Infrastructure

## Database Network Files

Listening to the Outside World

Terminal/Console

HTTP

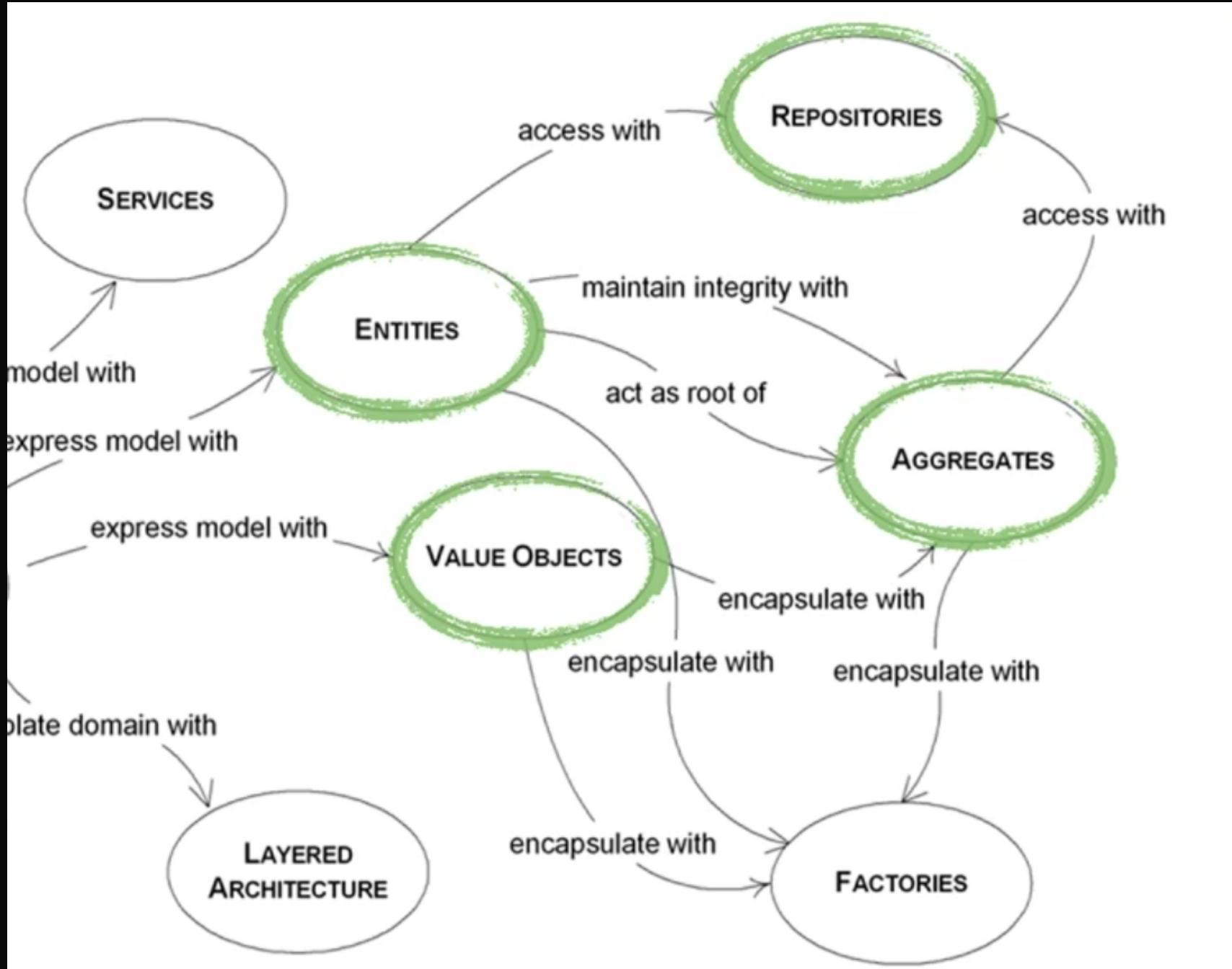
Web

# Language Frameworks

## Spring Micronaut

# Domain-Driven

**Domain is focus of the application**



# Entity

**Has Identity, History, and Attributes**

# Value Object

**Has Only Attributes**

# Aggregate

## Entity that Enforces Consistency

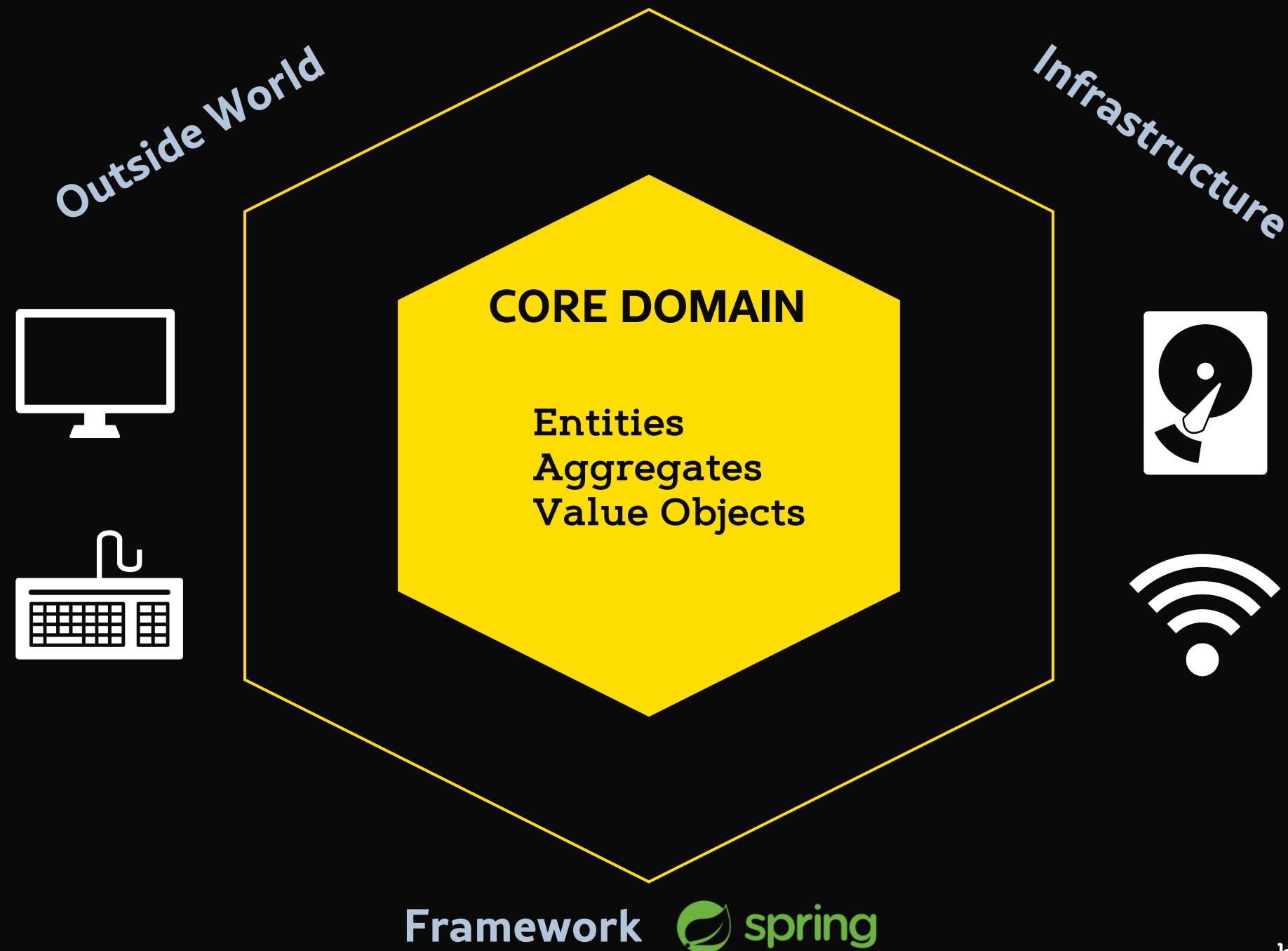
# Repository

**Use for Storing and Retrieving Aggregates**

# Core Domain

Account  
SavingsGoal  
PhoneNumber  
Role

Transaction  
UserProfile



# NOT a Unit Test

- Talks to the database
- Communicates across a network
- Touches the file system

Michael Feathers  
"A Set of Unit Testing Rules"  
(2005)

Principle

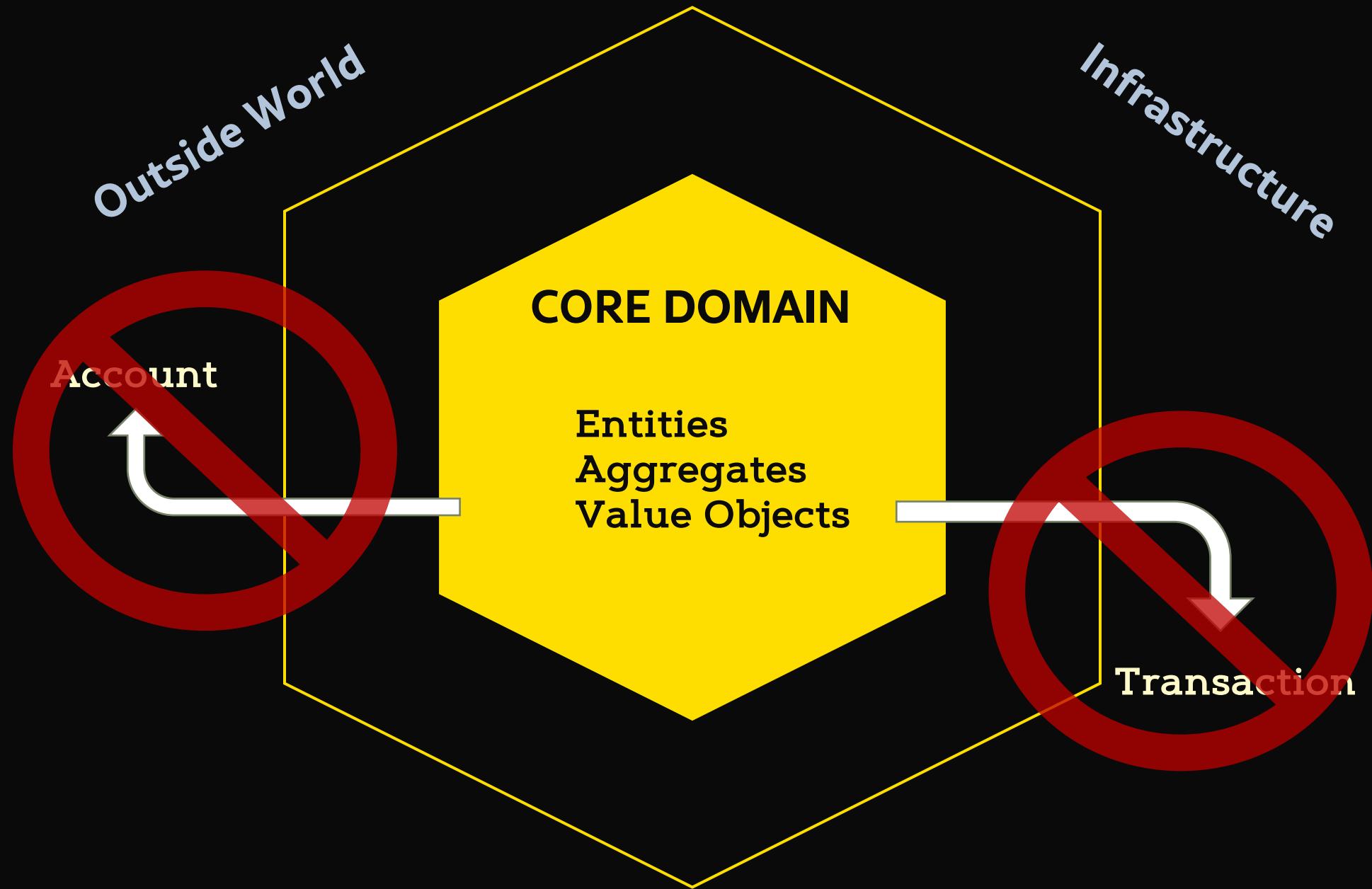
**Tests Must  
Be Isolated**

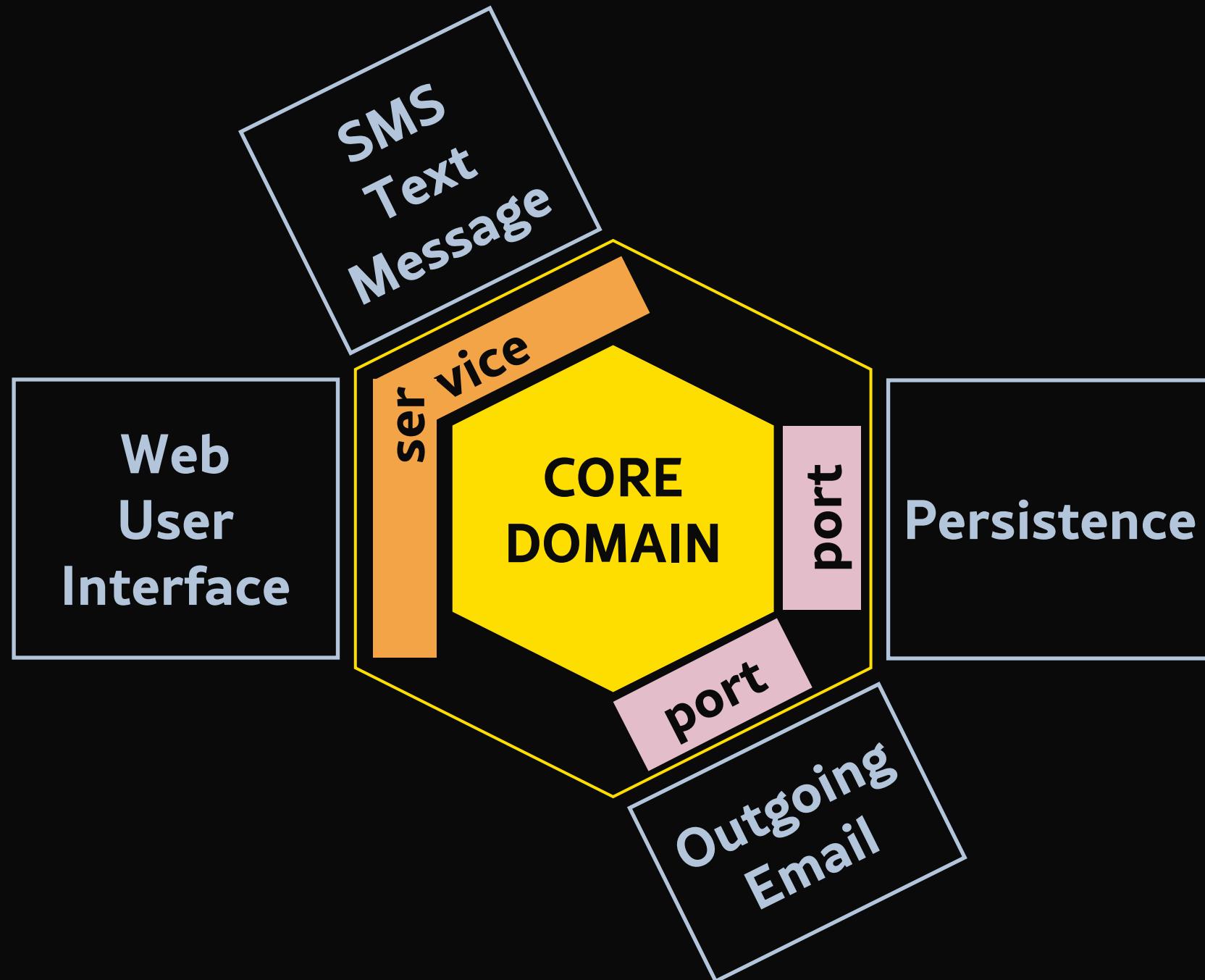
Goal

# Reduce Coupling

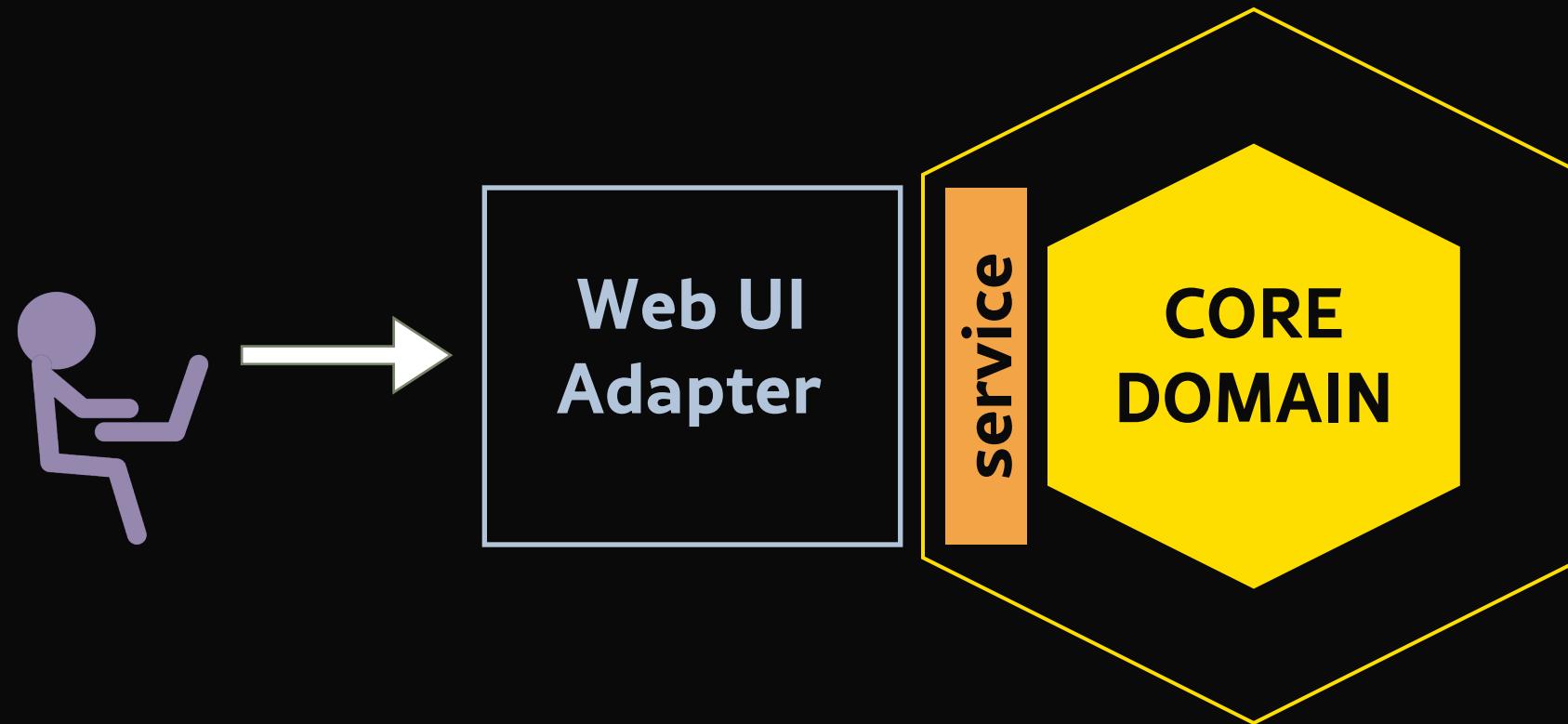
Rule

**Domain Objects  
Never Leave**



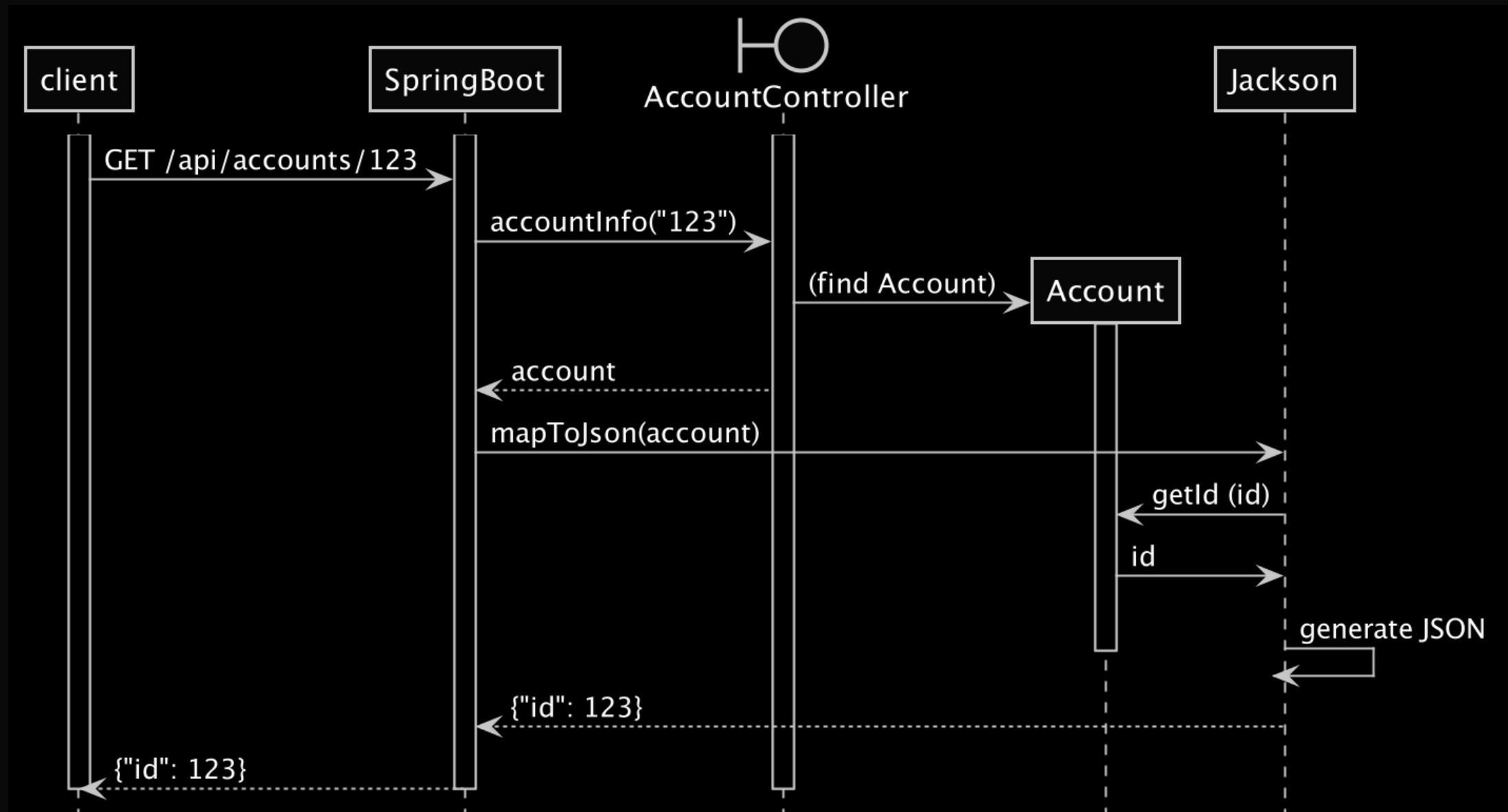


# [Outside World] User/Event-Driven



Adapter Role

Package of Classes



# Domain Object (POJO)

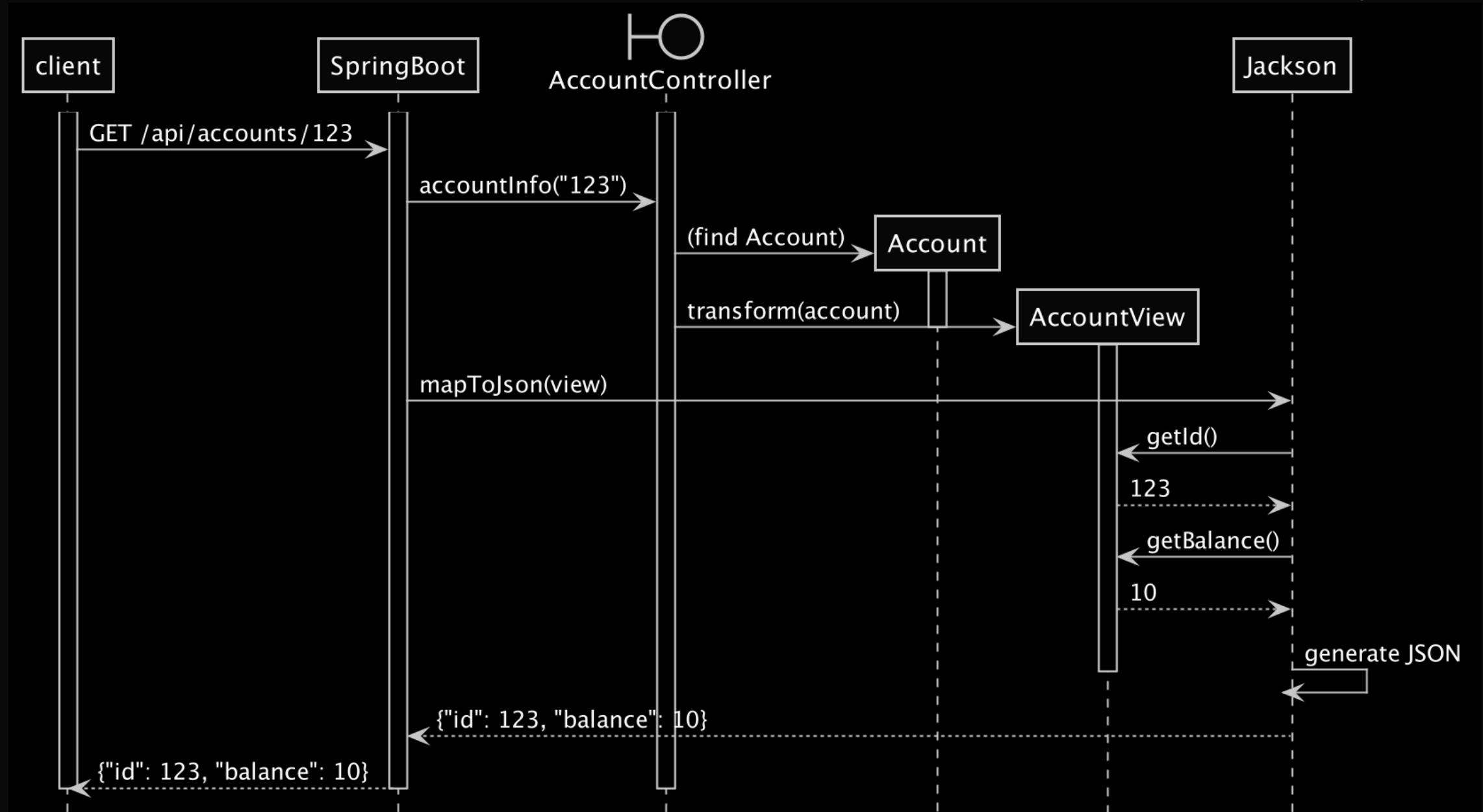
```
public class Account {  
  
    public void deposit(LocalDateTime txnDate, int amount, String descr){...}  
  
    public void spend(LocalDateTime txnDate, int amount, String descr) {...}  
  
    public int balance() {...}  
  
}
```

# Data Transfer Object (DTO)

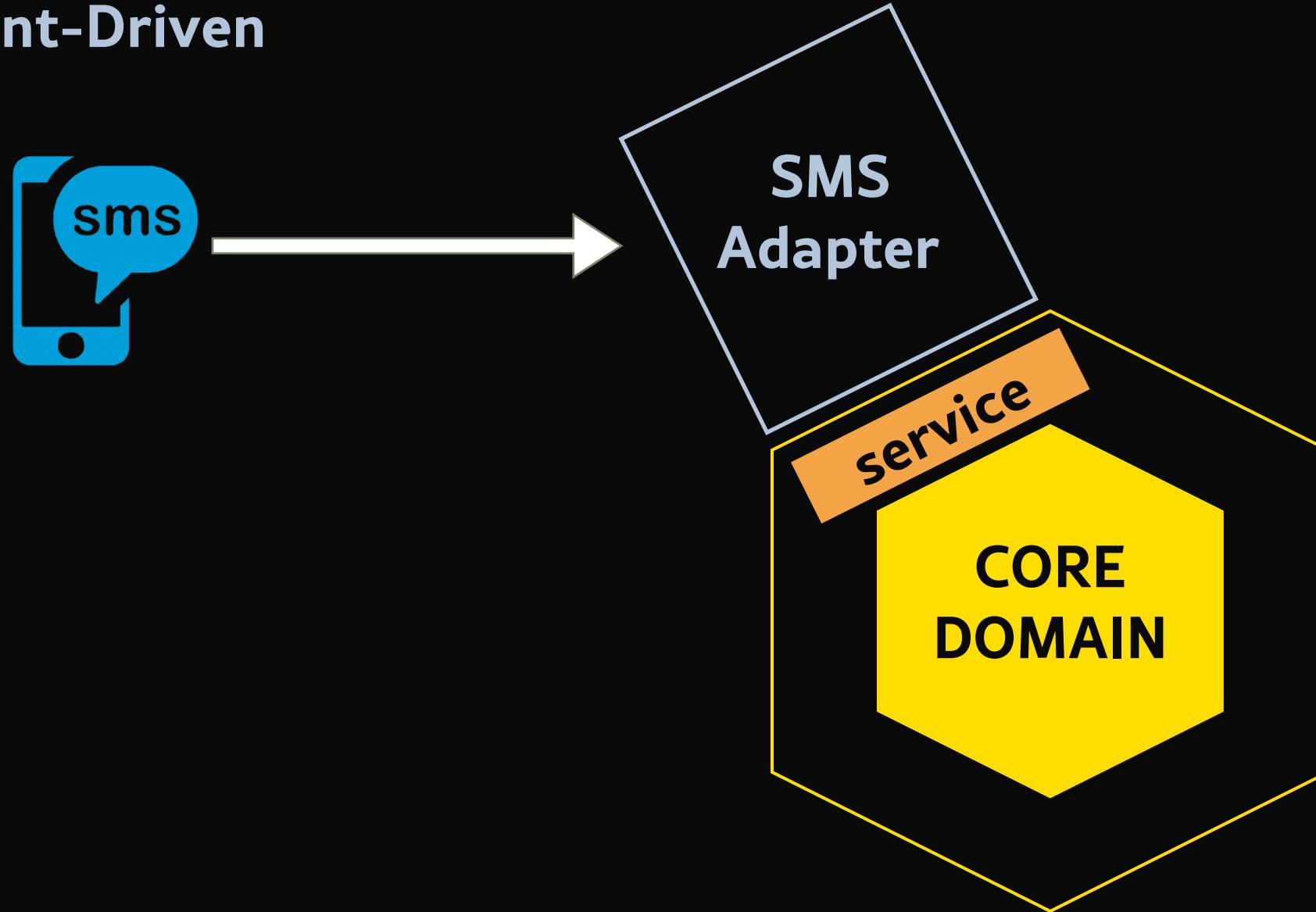
```
public class AccountView {  
    private long id;  
    private int balance;  
  
    /*  
     * setters & getters here  
     */  
}
```

# Adapter Responsibilities

1. Validate input from outside-world
2. Convert to Domain Object
3. Send to Service to do work (*optional*)
4. Domain Object → DTO
5. Return DTO to Outside-World



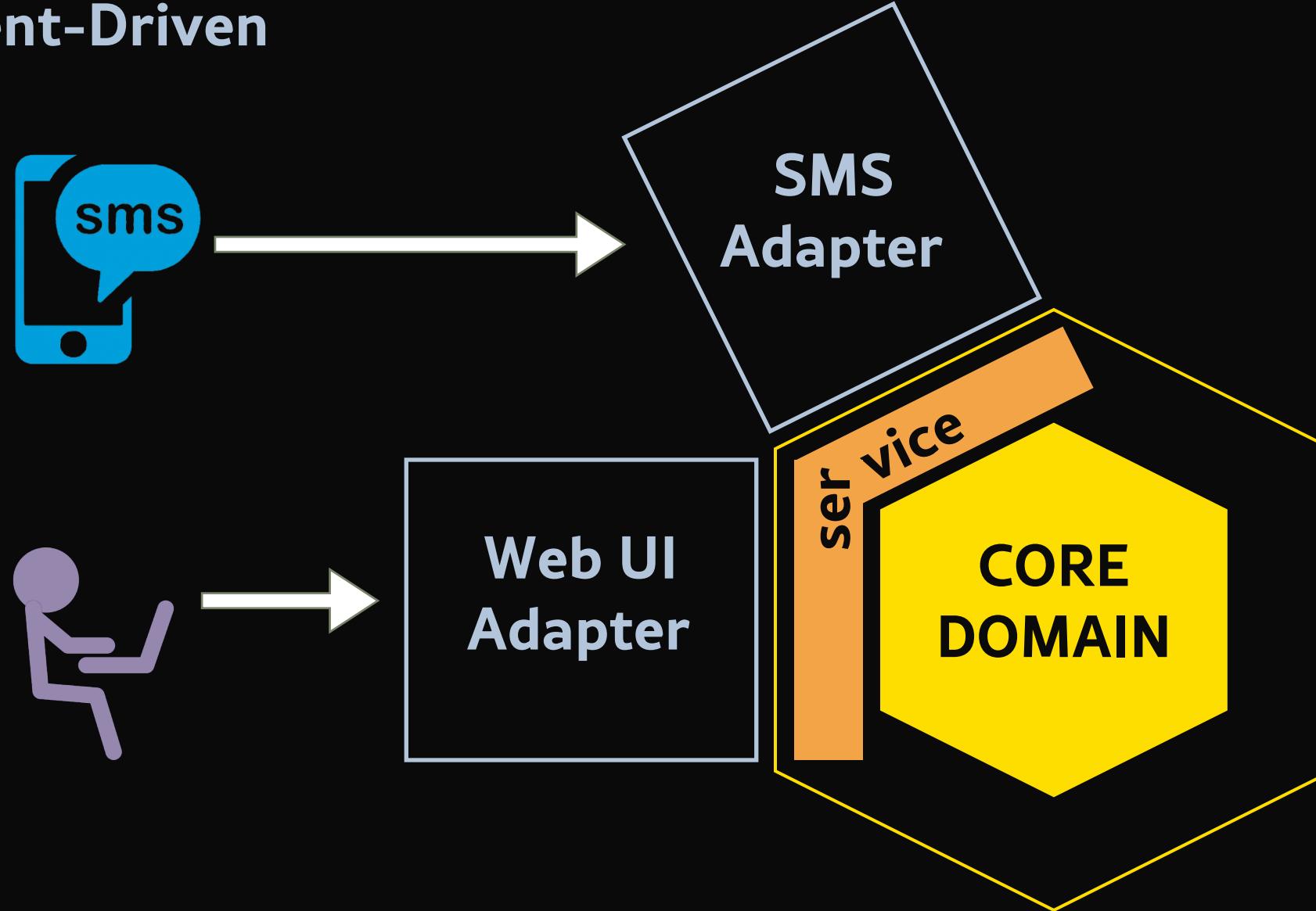
# [Outside World] User/Event-Driven



Rule

# Adapter Per Communication Mechanism

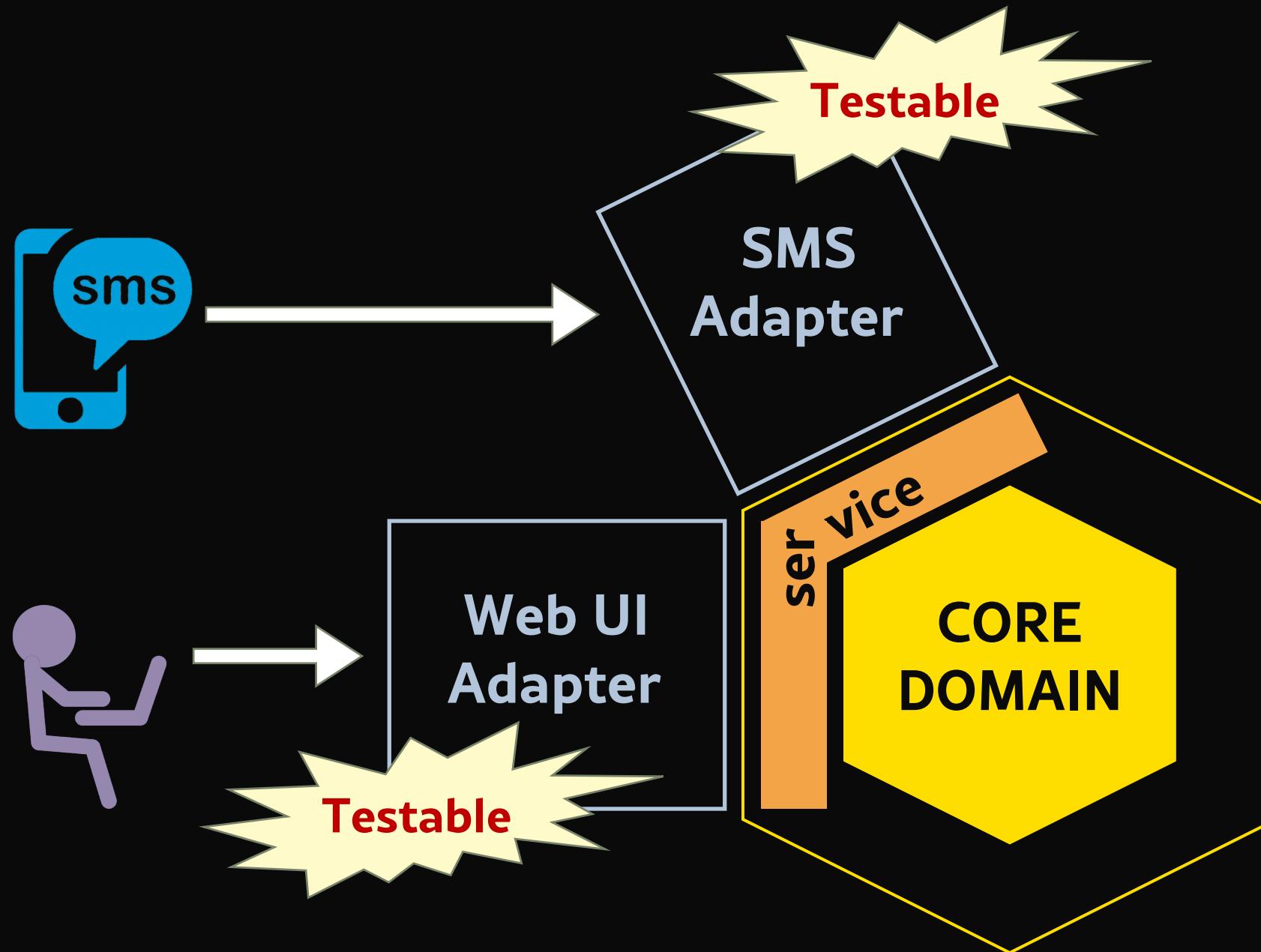
# [Outside World] User/Event-Driven

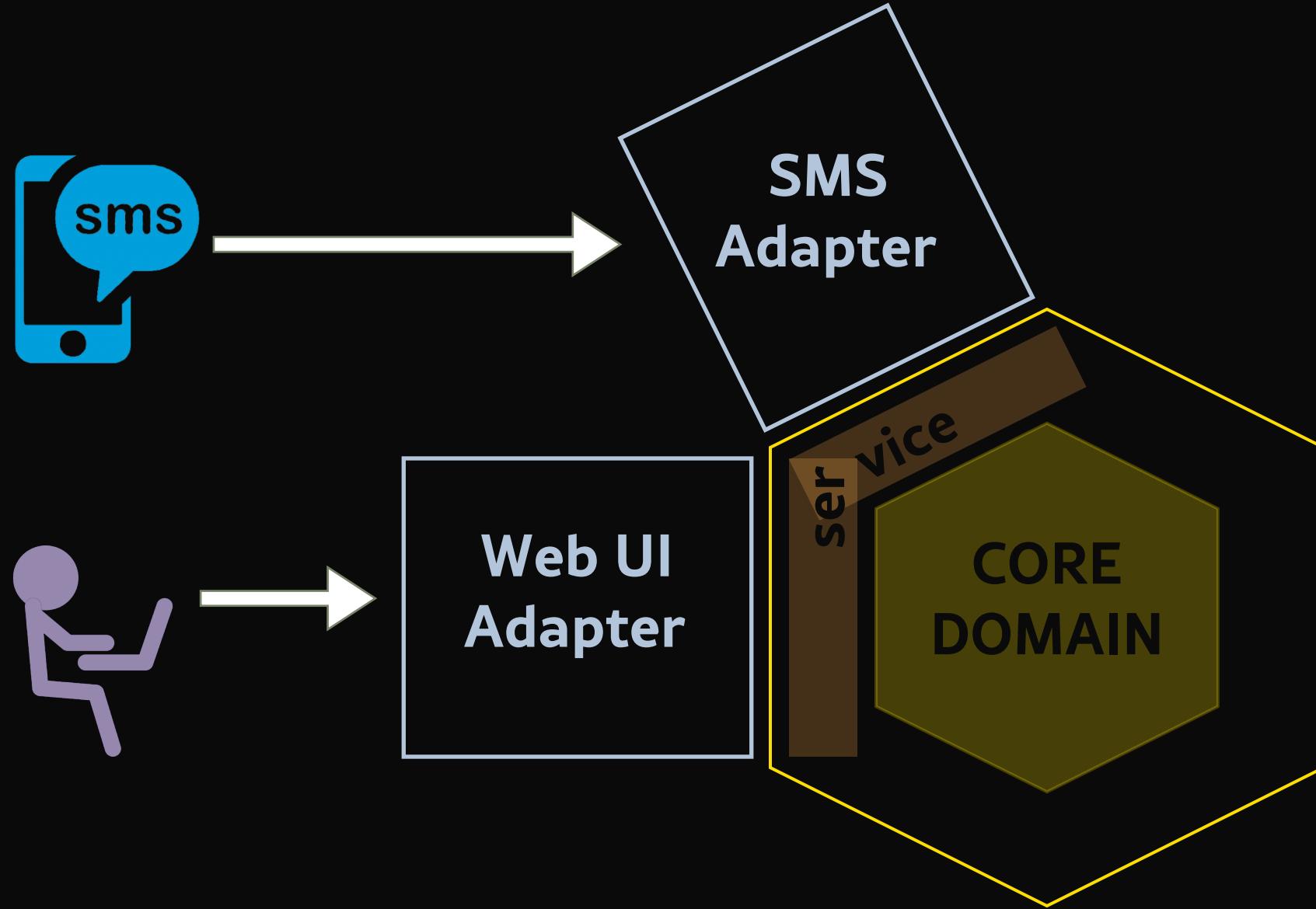


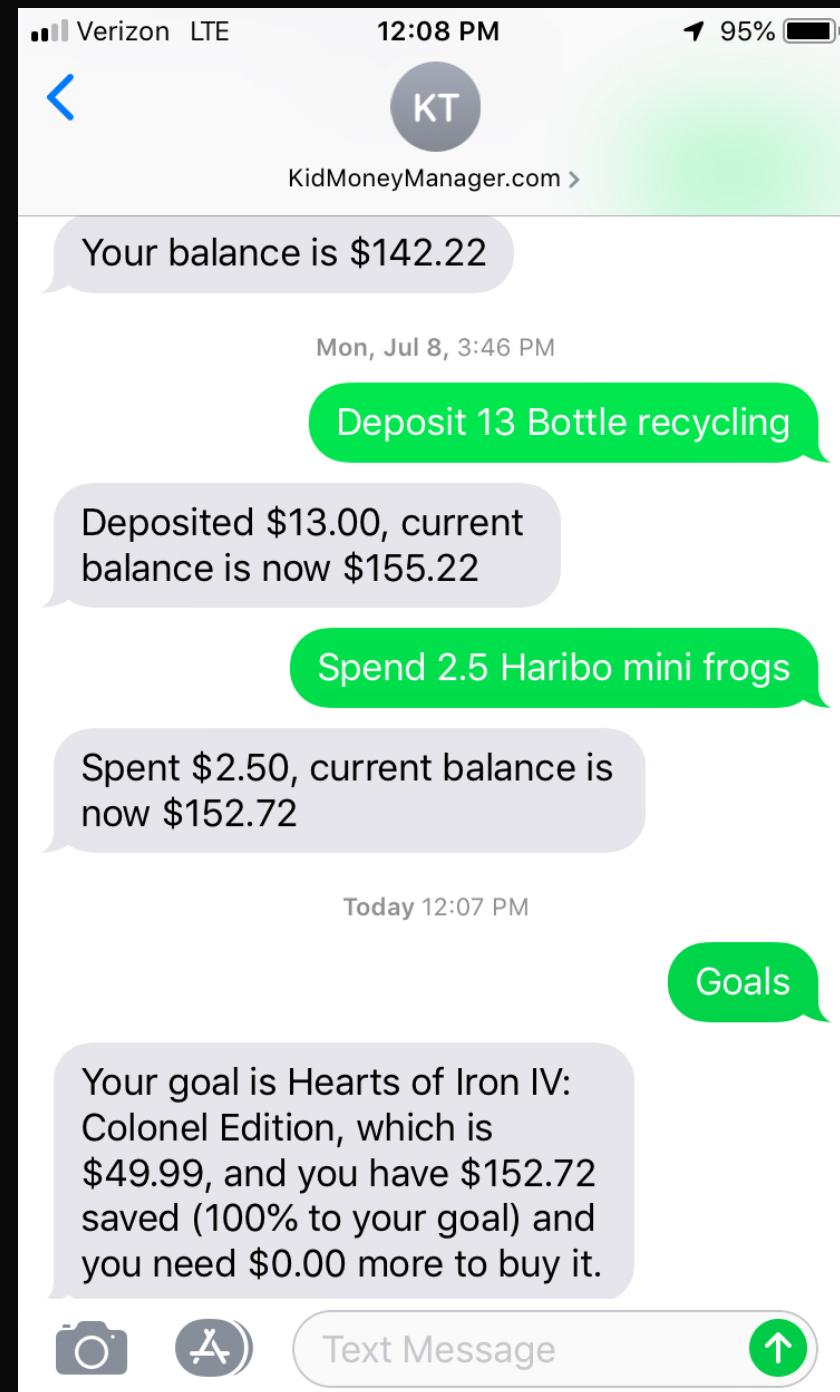
Dependency Rule

**Adapters Don't  
Depend on Adapters**

# Testing Adapters





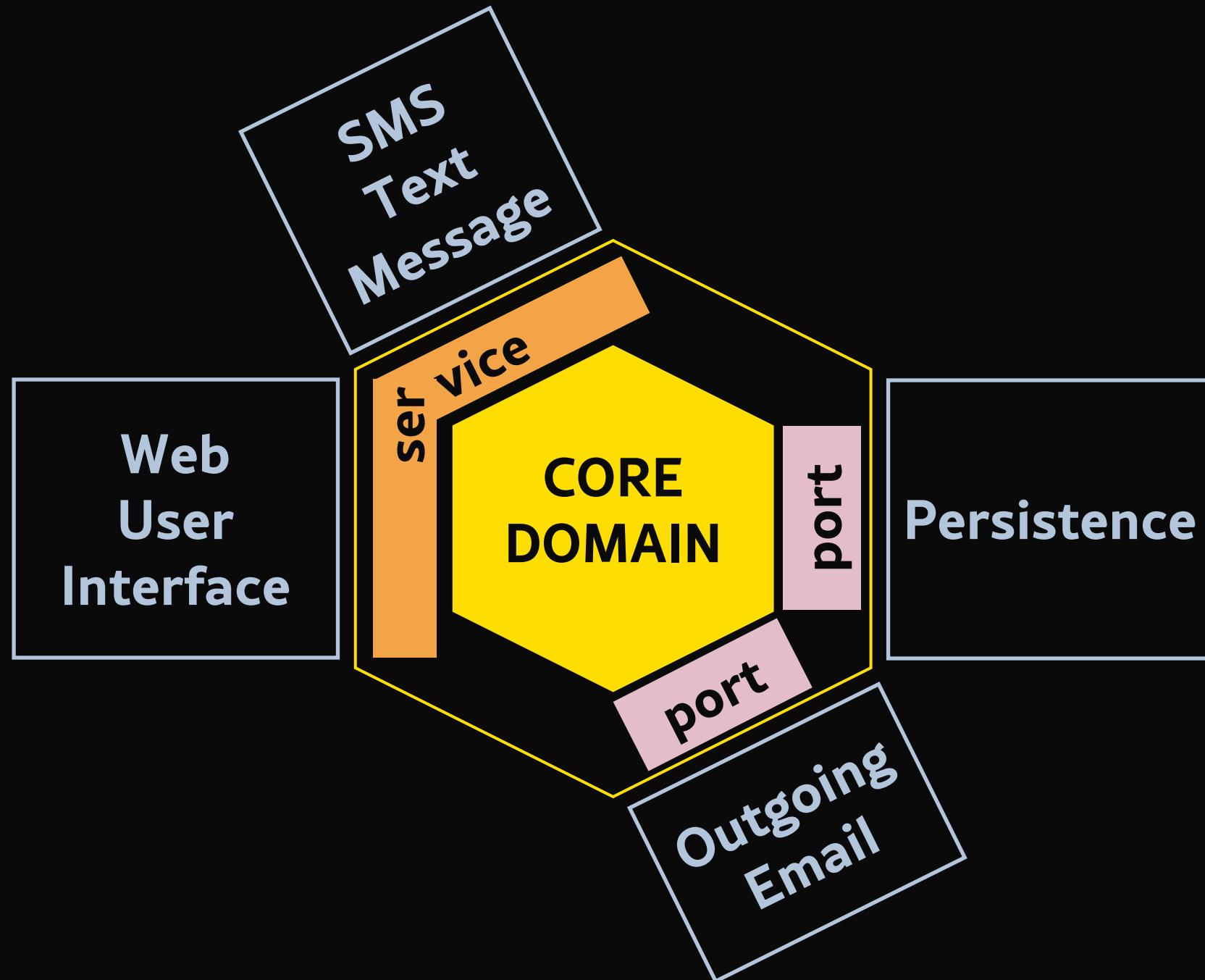


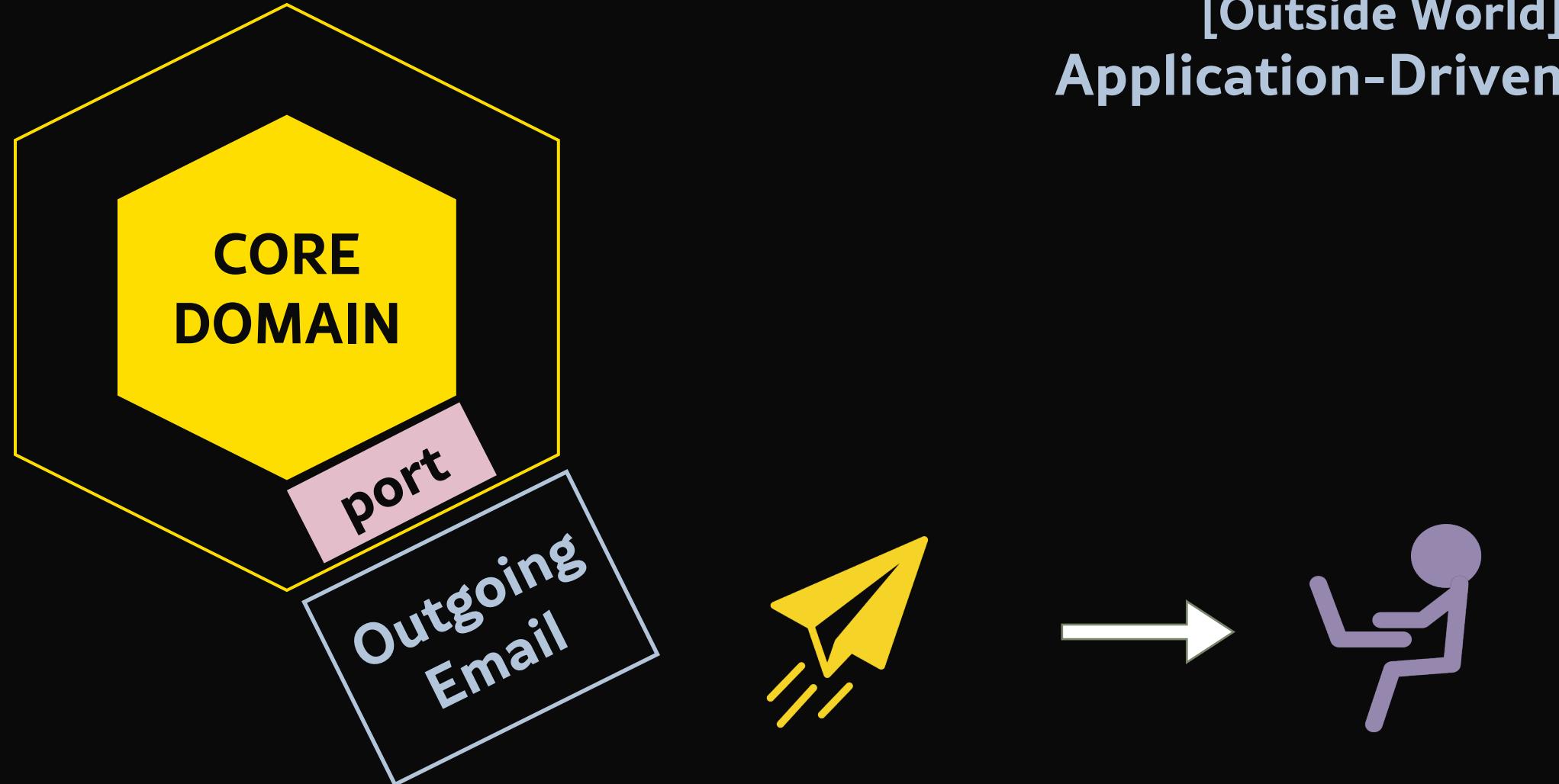
```
public class CommandParserSpendCommandTest {  
  
    @Test  
    public void spendWithSingleWordDescriptionCreatesSpendCommandWithDescription() throws Exception {  
        CoreAccount account = null;  
        CommandParser commandParser = new CommandParser(account);  
  
        TransactionCommand command = commandParser.parse("spend 37.95 Cards");  
  
        assertThat(command)  
            .isEqualTo(new SpendCommand(account, 37_95, "Cards"));  
    }  
  
    @Test  
    public void spendOnlyWithValidAmountShouldBeSpendCommand() throws Exception {  
        CoreAccount account = null;  
        CommandParser commandParser = new CommandParser(account);  
  
        TransactionCommand command = commandParser.parse("spend 55");  
  
        SpendCommand expectedSpendCommand = new SpendCommand(account, 55_00);  
  
        assertThat(command)  
            .isEqualTo(expectedSpendCommand);  
    }  
}
```

```
public class DepositViewIntegrationTest {  
    @Autowired  
    private MockMvc mockMvc;  
  
    @Test  
    public void getAgainstDepositUriShouldReturnDepositFormPage() throws Exception {  
        mockMvc.perform(get("/deposit"))  
            .andExpect(status().isOk())  
            .andExpect(view().name("deposit"))  
            .andExpect(model().attributeExists("balance", "depositCommand"));  
    }  
  
    @Test  
    public void submitDepositFormRedirectsToHomePage() throws Exception {  
        mockMvc.perform(post("/deposit")  
                    .param("amount", "12.45")  
                    .param("date", "2000-01-02")  
                    .param("description", "the source of money"))  
            .andExpect(redirectedUrl(AccountController.ACOUNT_URL));  
  
        mockMvc.perform(get(AccountController.ACOUNT_URL))  
            .andExpect(status().isOk());  
    }  
}
```

# Ports

**Abstractions and Concrete Implementations**





**Step 1: Define Port**

**Ideal Interface**

```
public interface BalanceChangedNotifier {  
    void balanceChanged(int amount, int balance);  
}
```

**Step 2**

# Use Spy to Test

```
@Test
public void depositShouldCauseNotificationOfBalanceChange() throws Exception {
    BalanceChangedNotifier mockNotifier = mock(BalanceChangedNotifier.class);

    // Given an account
    Account account = TestAccountBuilder.builder()
        .notifier(mockNotifier)
        .buildAsCore();

    // When we deposit money
    account.deposit(now(), 25_00, "test");

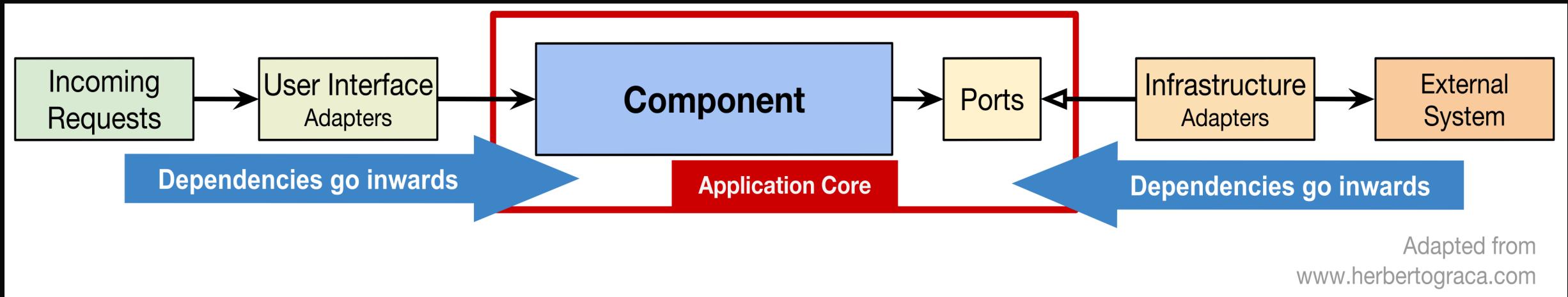
    // Then we expect the notification to be sent
    verify(mockNotifier).balanceChanged(25_00, 25_00);
}
```

**Step 3: Create Implementation**

**Concrete Adapter**

Dependency Rule

Dependencies  
Point Inwards



Rule

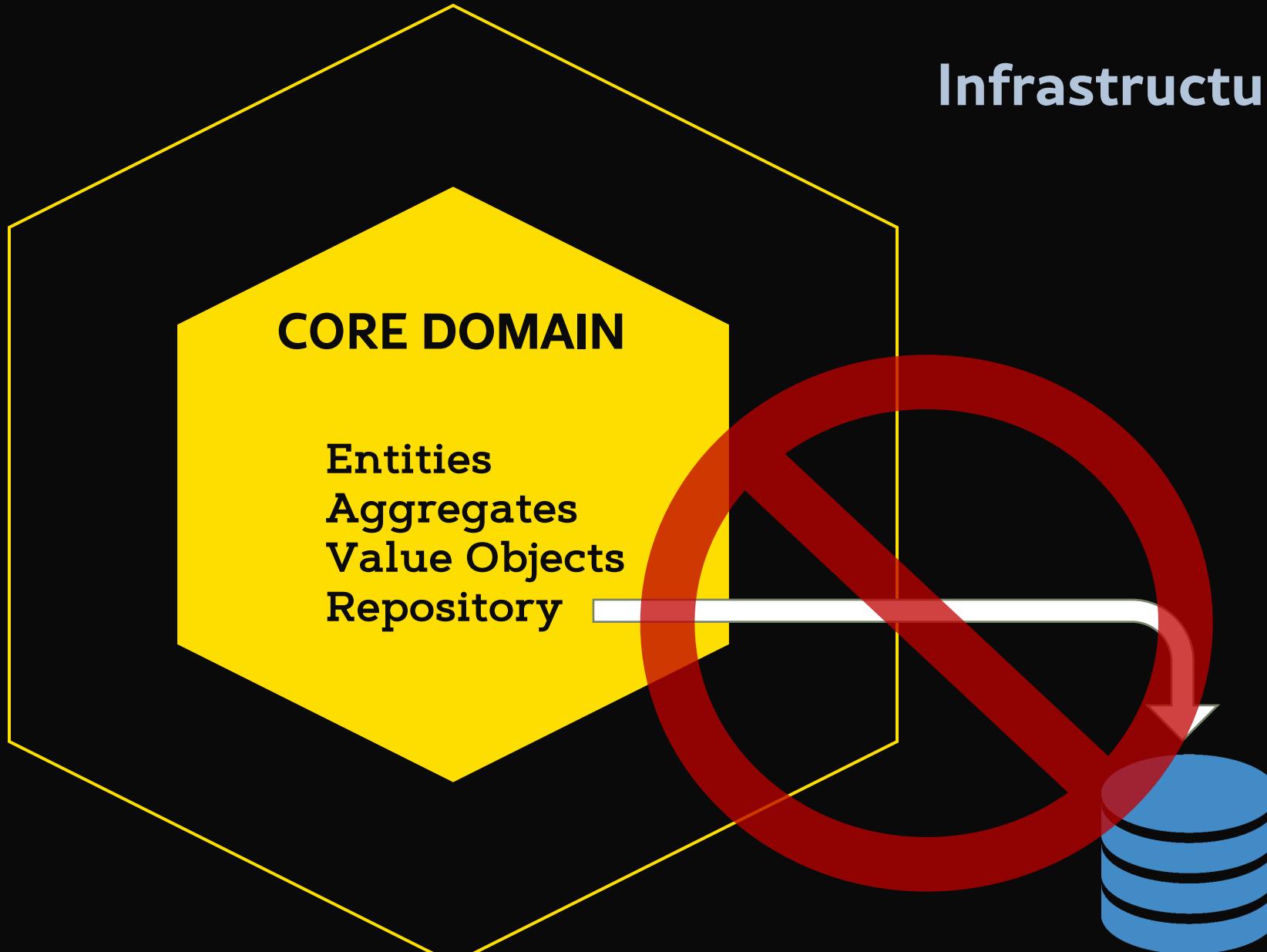
# Persistence Ignorance

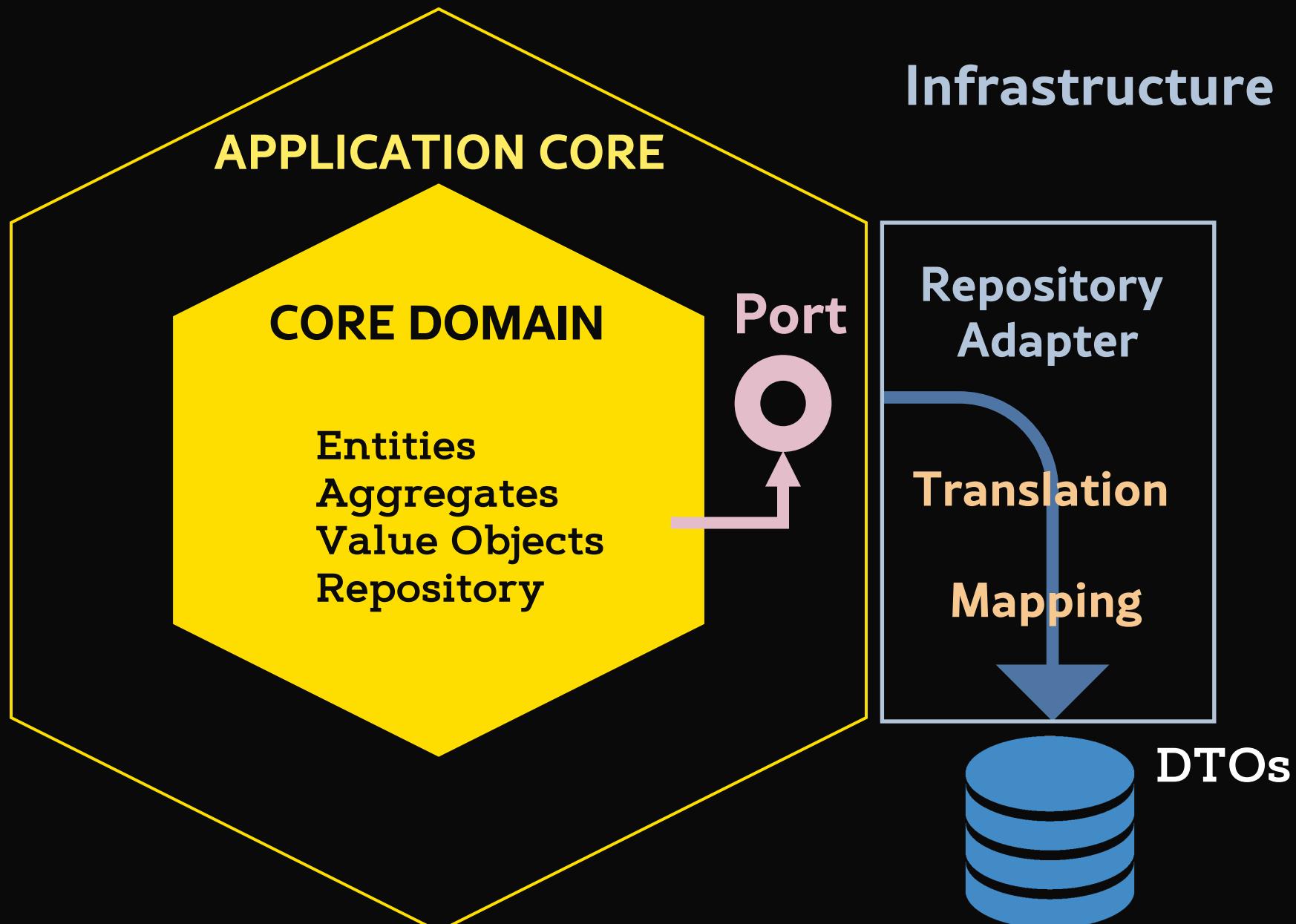
# Core Domain

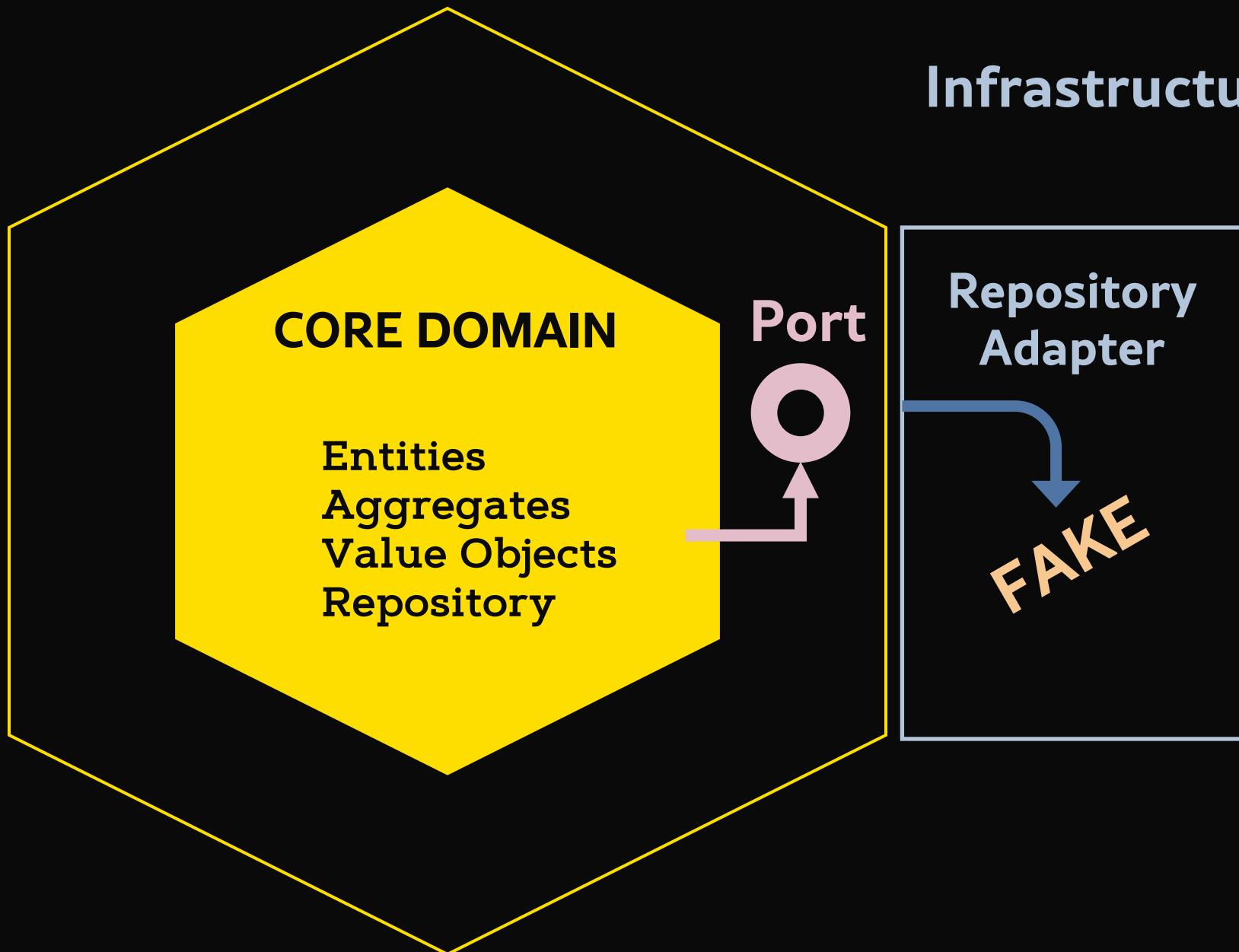
Account  
SavingsGoal  
PhoneNumber  
Role  
Transaction  
Transaction

UserProfile  
GoalRepository  
UserRepository

# Infrastructure







The ID Rule

**IDs are for Persistence  
Only**

# IDs Owned by Persistence

```
public class UserProfile {  
  
    // EXTRINSIC property used by Repository  
    @Getter  
    @Setter  
    private Long id;  
  
    private final String name;  
    private final PhoneNumber phoneNumber; // SMS authorization  
    private final String email; // web-based/OAuth authorization  
    private final Role role;  
  
    public UserProfile(String name, PhoneNumber phoneNumber, String email, Role role) {  
        this.name = name;  
        this.phoneNumber = phoneNumber;  
        this.email = email;  
        this.role = role;  
    }  
}
```

**Problem**

**How do I  
Find Things?**

# Information Foraging Theory

No, really, this is a thing!

# The Sensemaking Process

**Looking and Understanding**

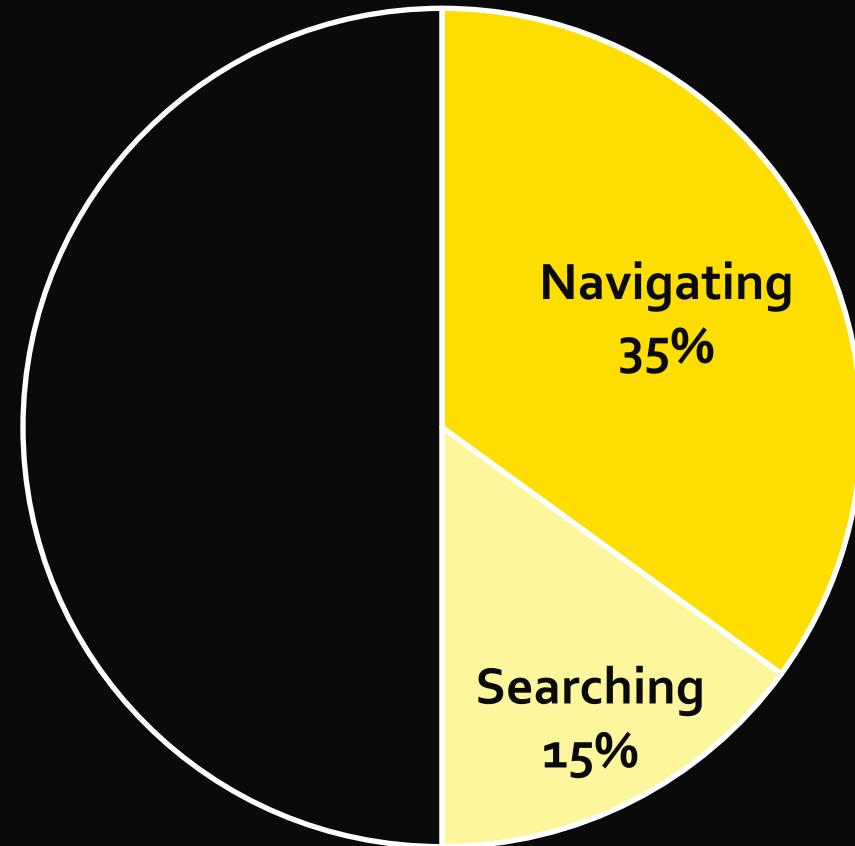
# Time Spent Searching

**The whats and hows of programmers' foraging diets**

<https://dl.acm.org/citation.cfm?id=2466418>

**Foraging and navigations, fundamentally:  
developers' predictions of value and cost**

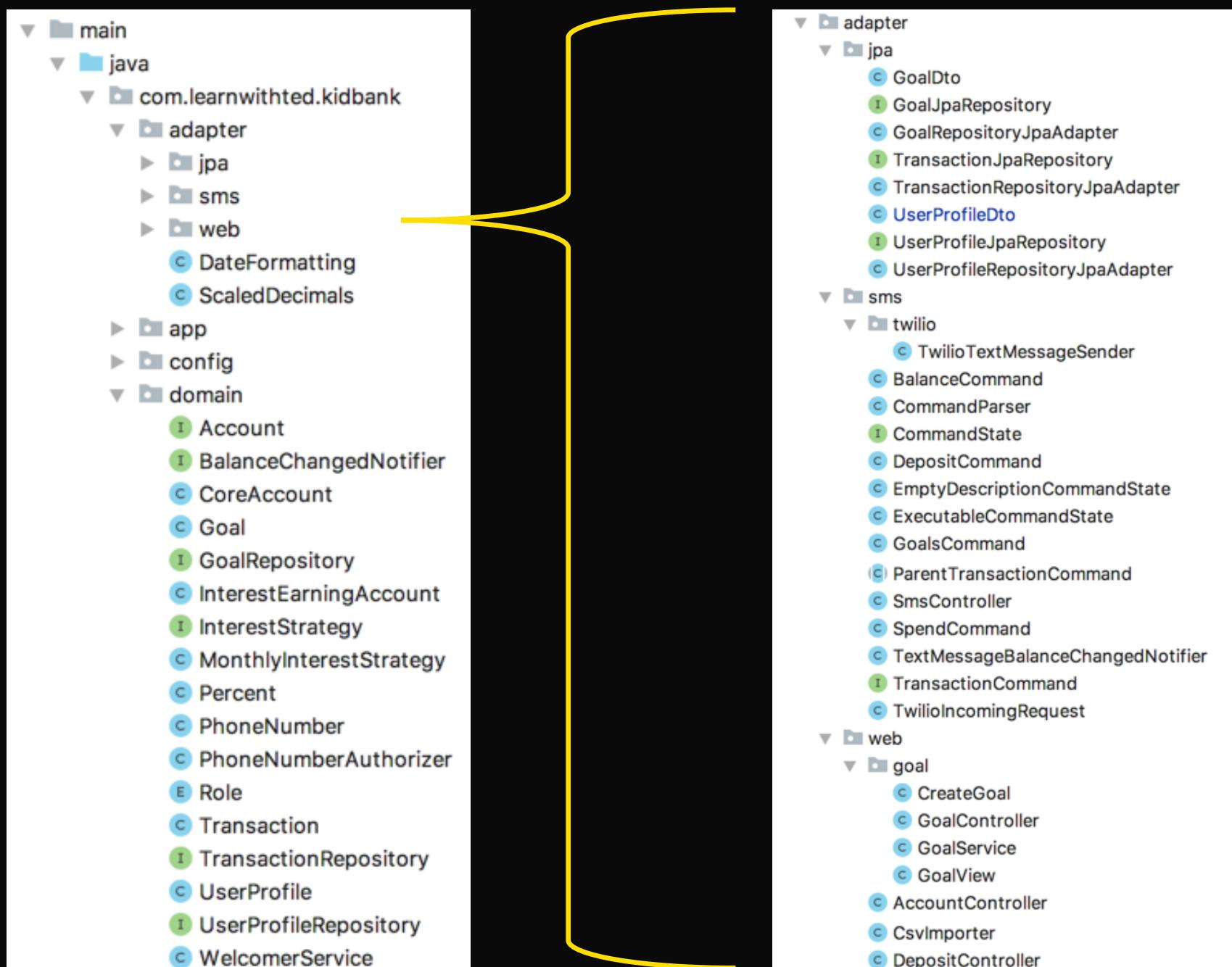
<https://dl.acm.org/citation.cfm?id=2950302>



Don't make your  
future code easier to  
**change**, make it  
easier to **find** what  
needs to be changed.

Rule

# Package Based on Architecture



# Don't Use Hexagonal

Basic CRUD

Simple or non-existent domain

Transformation/Integration-only

# Hexagonal Architecture

Focus on Core Domain

Domain Objects Never Leave

Adapters per Trigger Type

Ports for Ideal Interface to Outside

Dependencies point inwards

Package Based on Architecture

# Thank You!



**Ted M. Young**  
Java Trainer/Coach



**Spiral Learning, LLC**  
[ted@SpiralLearningLLC.com](mailto:ted@SpiralLearningLLC.com)  
Live Coding: <https://twitch.tv/jitterted>

@JitterTed  
<https://ted.dev>  
[ted.dev/youtube](https://ted.dev/youtube)

