One Architecture to Rule them All

Why Hexagons are so Awesome

Ted M. Young
Java Trainer/Coach

Spiral Learning, LLC
ted@SpiralLearningLLC.com

@JitterTed (Twitter & Twitch)
https://ted.dev (Web)
Ted M. Young

I Can Help Your Team...

- Write more Testable code with more Effective unit tests
- Be more productive in Java & Spring
- Effectively use TDD
- Become better developers
- Apply eXtreme Programming

Ted M. Young
ted@SpiralLearningLLC.com
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
Licensed CC-BY 2.0

“A good architect leaves a footprint”
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
FLAT-TOPPED

POINTY-TOPPED
OUTSIDE

INSIDE

OUTSIDE

OUTSIDE

OUTSIDE

OUTSIDE

Ted M. Young

https://ted.dev
TRIGGERS → MESSAGES → INSIDE → EVENTS → OUTCOMES

DRIVING → INSIDE → DRIVEN
KID MONEY MANAGER
SEASON 2

Live Coding: https://Twitch.tv/jitterted
Season 1: https://ted.dev/youtube
### KidBank Balance

**ACCOUNT BALANCE**

$153.04

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

**Current Balance**

Mon, Jul 8, 3:46 PM

Your balance is $142.22

Deposited $13.00, current balance is now $155.22

Spend 2.5 Haribo mini frogs

Spent $2.50, current balance is now $152.72

Today 12:07 PM

Your goal is Hearts of Iron IV: Colonel Edition, which is $49.99, and you have $152.72 saved (100% to your goal) and you need $0.00 more to buy it.
Web User Interface

Web User Interface

SMS Text Message

Outgoing Email

Persistence

CORE DOMAIN

Persistence

Outgoing Email

SMS Text Message

Web User Interface

CORE DOMAIN

Persistence

Outgoing Email

SMS Text Message

Web User Interface
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  Transaction
SavingsGoal  UserProfile
PhoneNumber  Role
CORE DOMAIN

Entities
Aggregates
Value Objects

Outside World
Infrastructure

Framework

Ted M. Young
https://ted.dev
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
CORE DOMAIN

Entities
Aggregates
Value Objects

Outside World

Infrastructure

Account

Transaction
[Outside World]
User/Event-Driven

- Web UI Adapter
- CORE DOMAIN

Ted M. Young
https://ted.dev
Adapter Role

Package of Classes
public class Account {

    public void deposit(LocalDateTime txnDate, int amount, String descr){...}

    public void spend(LocalDateTime txnDate, int amount, String descr) {...}

    public int balance() {...}

}
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 $\rightarrow$ DTO
5. Return DTO to Outside-World
GET /api/accounts/123

accountInfo("123")

(find Account) Account

transform(account)

AccountView

generate JSON

generate JSON

mapToJson(view)

{"id": 123, "balance": 10}
[Outside World]
User/Event-Driven

- SMS

SMS Adapter

service

CORE DOMAIN

Ted M. Young
https://ted.dev
Rule

Adapter Per Communication Mechanism
[Outside World]
User/Event-Driven

Web UI Adapter

SMS Adapter

CORE DOMAIN

User/Event-Driven
Dependency Rule

Adapters Don't Depend on Adapters
Testing Adapters
Subject: Web UI Adapter vs. SMS Adapter

- **Web UI Adapter**
  - Testable
  - Connects to a user interface

- **SMS Adapter**
  - Testable
  - Connects to SMS

---

**Core Domain**

- Service

---

Ted M. Young

https://ted.dev
Your balance is $142.22

Mon, Jul 8, 3:46 PM

Deposit 13 Bottle recycling

Deposited $13.00, current balance is now $155.22

Spend 2.5 Haribo mini frogs

Spent $2.50, current balance is now $152.72

Today 12:07 PM

Goals

Your goal is Hearts of Iron IV: Colonel Edition, which is $49.99, and you have $152.72 saved (100% to your goal) and you need $0.00 more to buy it.
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.ACCOUNT_URL));

        mockMvc.perform(get(AccountController.ACCOUNT_URL))
            .andExpect(status().isOk());
    }
}
Ports
Abstractions and Concrete Implementations
[Outside World] Application-Driven

CORE DOMAIN

Outgoing Email

port

Outgoing Email

Ted M. Young

https://ted.dev
Step 1: Define Port

Ideal Interface
public interface BalanceChangedNotifier {
    void balanceChanged(int amount, int balance);
}
Step 2

Use Spy to Test
@Test

class BankTest {

    @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
Incoming Requests → User Interface Adapters → Component → Ports → Infrastructure Adapters → External System

Dependencies go inwards

Application Core
Rule

Persistence

Ignorance
Core Domain

- Account
- SavingsGoal
- PhoneNumber
- Role
- Transaction

User Profile
- GoalRepository
- UserRepository
CORE DOMAIN

Entities
Aggregates
Value Objects
Repository

Infrastructure
Entities
Aggregates
Value Objects
Repository

Infrastructure
Repository Adapter
Translation
Mapping

DTOs

TEd M. Young
https://ted.dev
The ID Rule

IDs are for Persistence Only
IDs Owned by Persistence

```java
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
Live Coding: https://twitch.tv/jitterted

@JitterTed
https://ted.dev
ted.dev/youtube