Software Design Principles & Patterns

A hands-on workshop. During these two days, we are going to build a backend system in Java. We'll start with the naive approach and will keep applying new design patterns and principles as needed.

Requirements: 5+ years in software development
Target: mid & senior engineers, architects
Duration: 2 days

Agenda

1. We'll start with an Introduction to the example application. We will start with a simplified version of the Big Picture Event Storming session to uncover the requirements.
2. Next, we will run Example Mapping to work on concrete examples. Then, we are going to translate these examples into tests. We'll discuss two flavors of TDD: London School and Chicago School.
3. At the same time of writing tests, we'll be also writing production code. We'll start the naive way (without using any special design patterns).
4. During the refactor step we will address design issues of the naive implementation. That's the moment when I'm going to introduce some theory behind design patterns. I'll also present a selection of the most useful design patterns.
5. Next, I'll present a detailed overview of best practices regarding using design patterns. I'll try to give the best answer to the following questions:
    a. When should we use design patterns?
    b. How should we use design patterns?
6. Now I'll show how to identify anti-patterns and isolate them from the rest of the system.
7. We're going to see how we can use Ports and Adapters (AKA Hexagonal) architecture to build application and infrastructure layers around the core domain.
8. I'll show some practices of integration testing.
9. After that, we'll cover the basics of reactive systems. We'll try to adapt our application to the reactive requirements.
10. We'll continue with the development of the application.