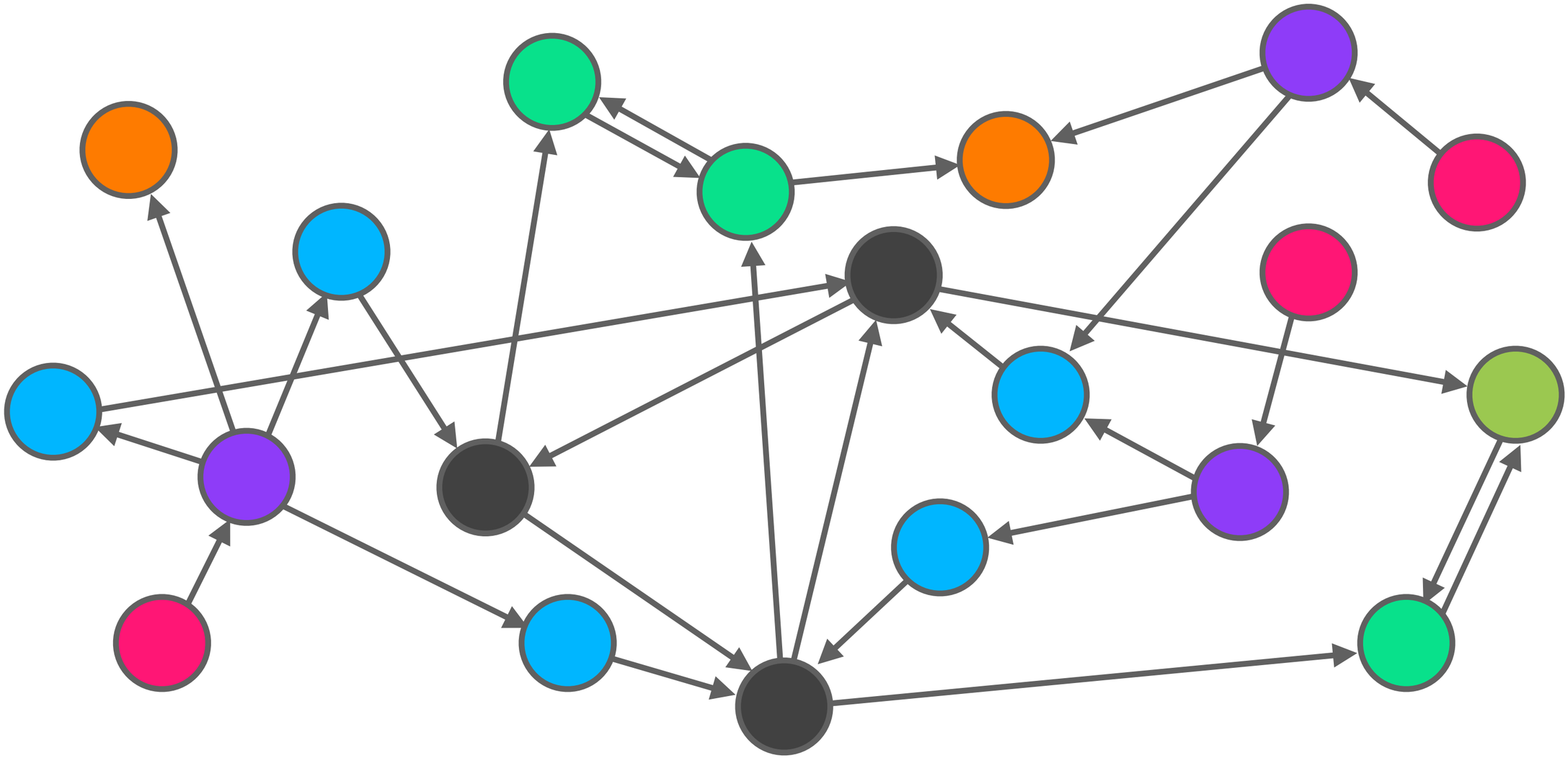
Designing with Abstraction and Encapsulation

Paolo Perrotta

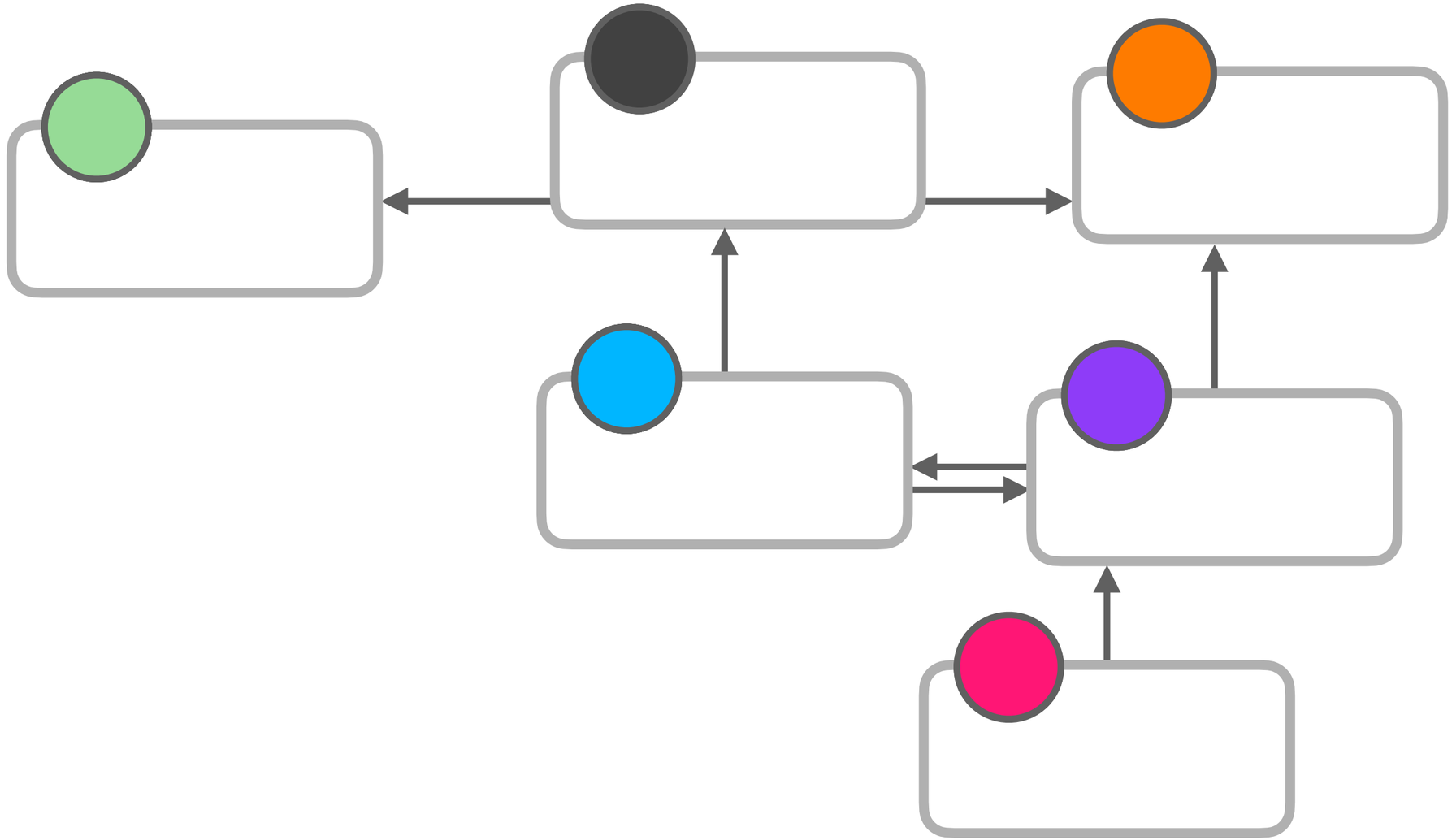
Developer, Author

@nusco | www.paoloperrotta.com

Java at Runtime

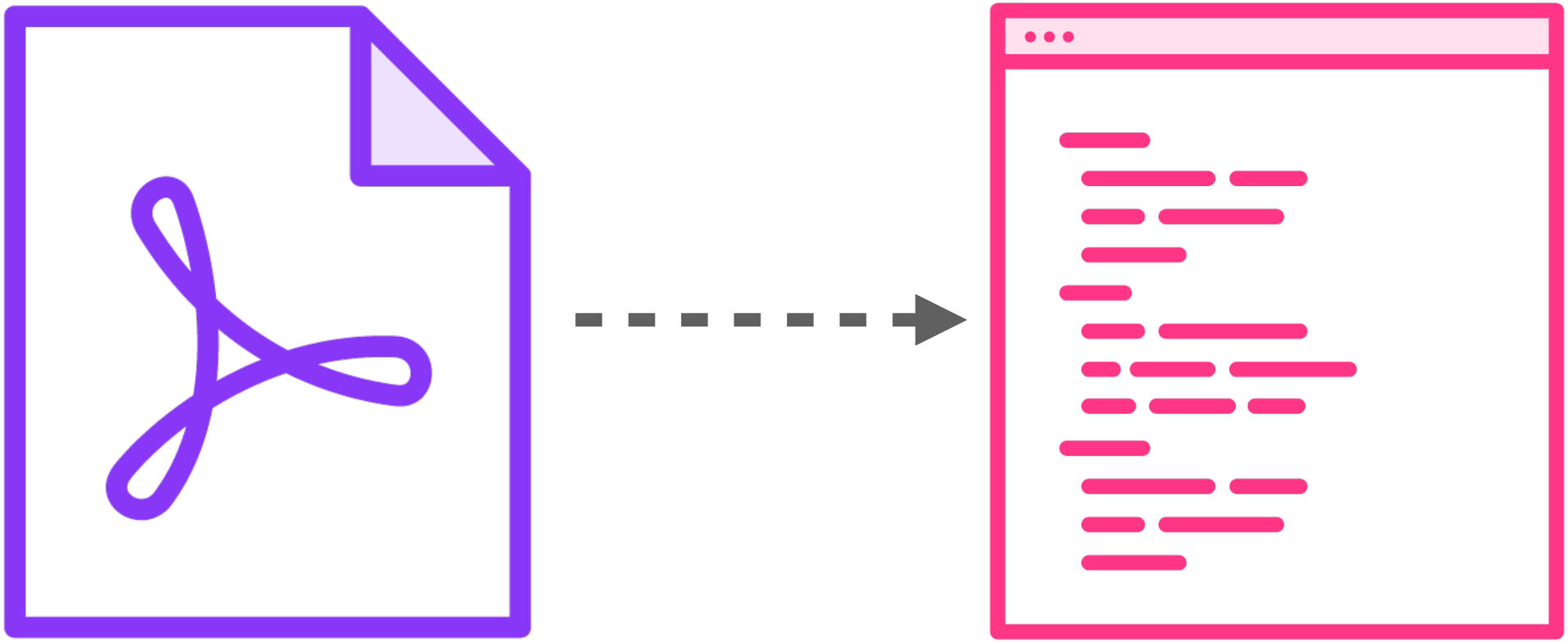


Java at Design Time



“Design” is about deciding what classes you have and how they interact.

From Specs to Code



Problem

Solution

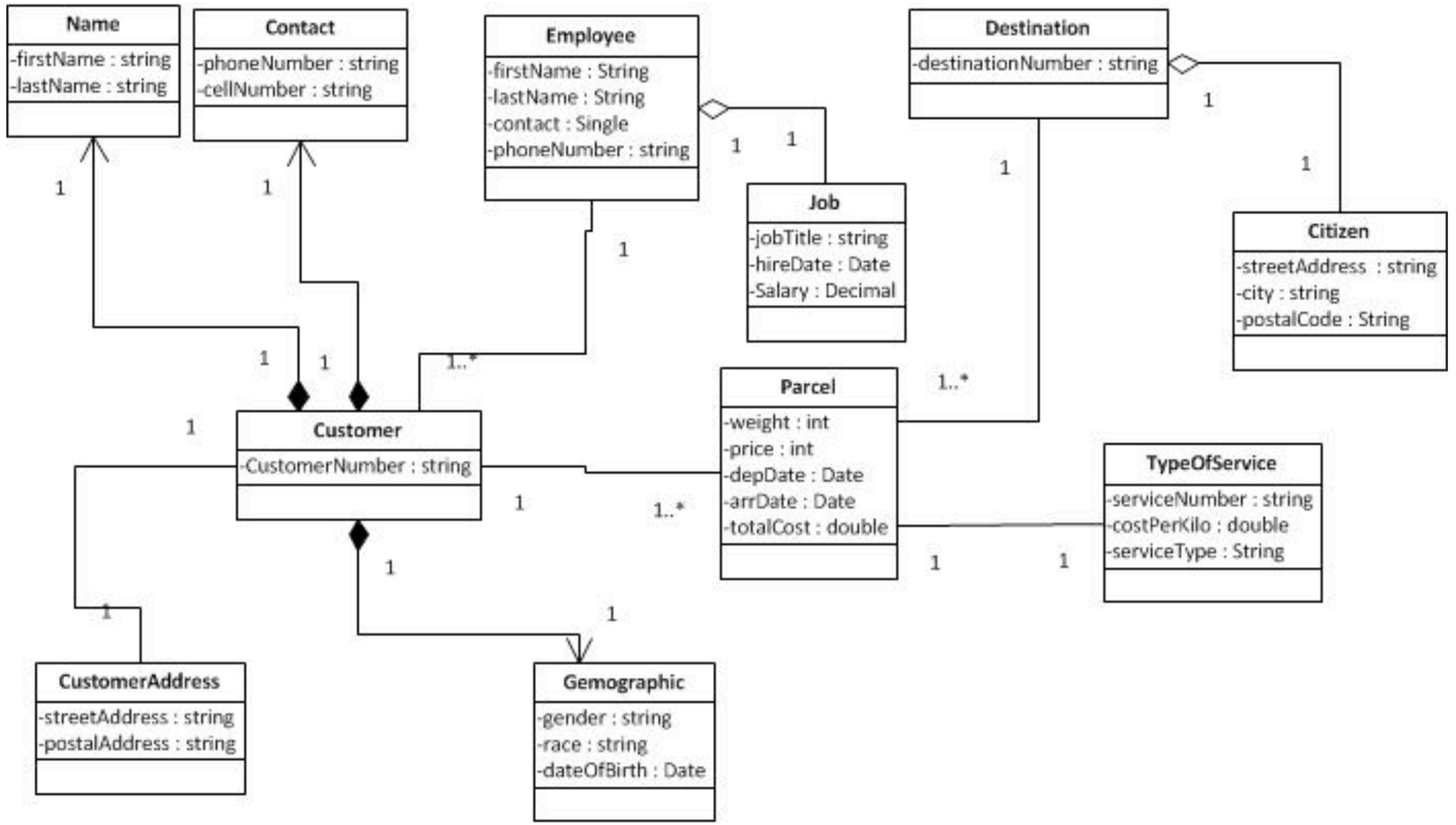
From Specs to Code

The supervisor can add alarms to a dashboard and manually activate deactivate and snooze them. When an alarm activates, it notifies the supervisor and also sends a message to the administrator.

Configuration

MailServer

Class Diagrams



https://commons.wikimedia.org/wiki/File:UML_delivery_system2.jpg

Java Design Trainings on Pluralsight

- Dan Geabunea, Refactoring to SOLID Java
- Bryan Hansen, Java Creational Design Patterns
- Andrejs Doronins, Java Best Practices
- ...

The *Alarm* Class

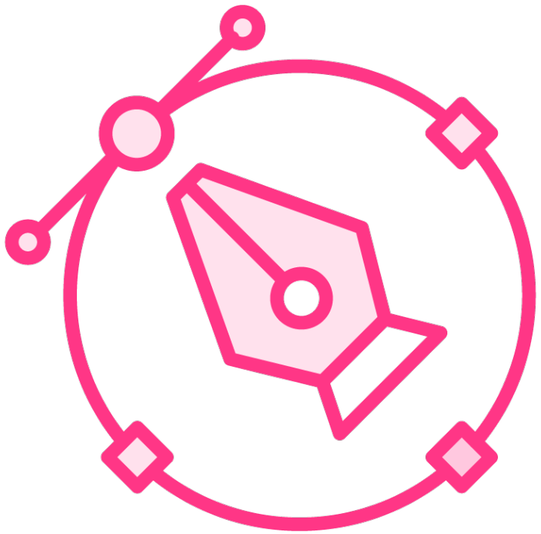
```
public class Alarm {
    private boolean active;
    private final String message;
    private LocalDateTime snoozeUntil;

    public Alarm(String message) {
        this.message = message;
        stopSnoozing();
    }

    public void snooze() {
        if (active)
            snoozeUntil = LocalDateTime.now().plusMinutes(5);
    }

    public LocalDateTime getSnoozeUntil() {
```

These are just guidelines!



More private is better than less private

- If something isn't needed elsewhere, keep it local
- Keep interfaces small

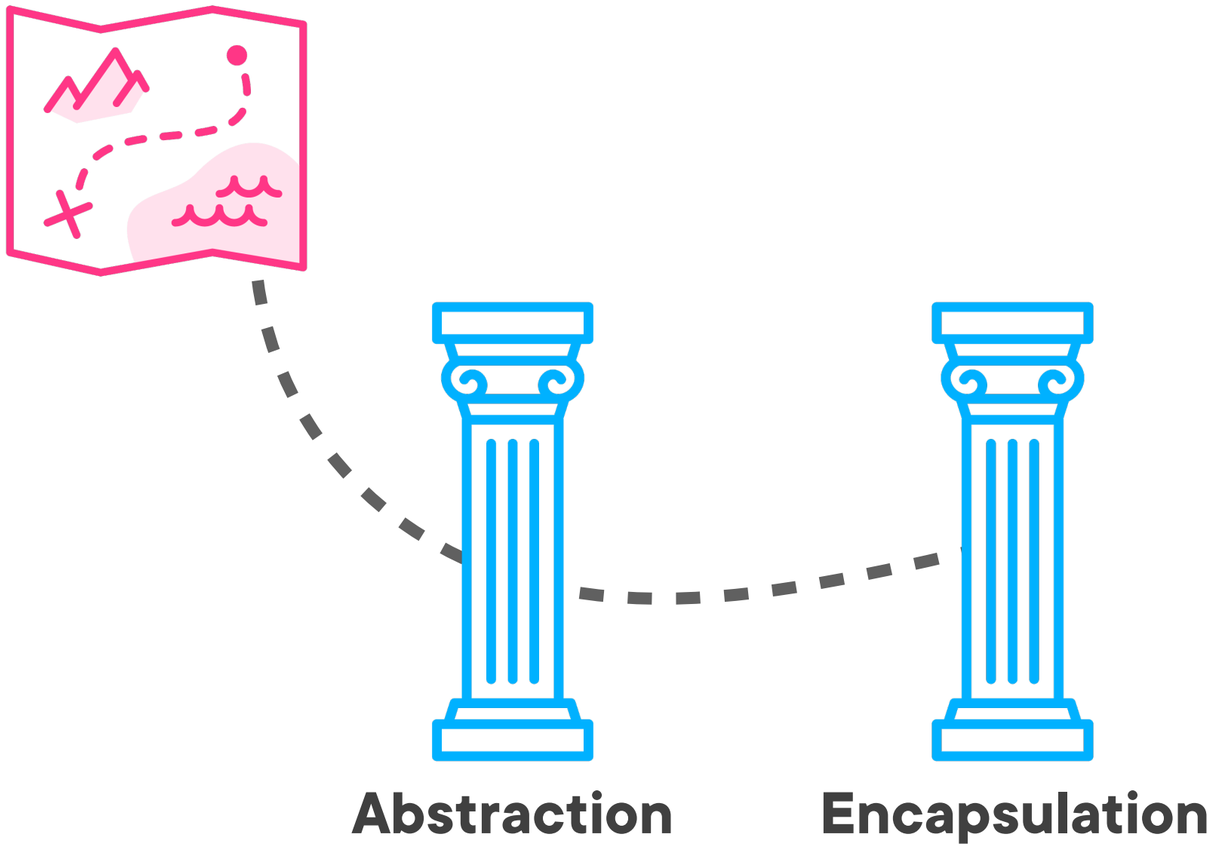
Encapsulated fields are good

- Make fields private
- Only write the setters you need

Final fields are also good

- If a field doesn't need to change, make it final
- Does the field contain an immutable object? Great!

The First Two Pillars



Up Next:

Inheriting from Another Class