Responsibility-Driven Design with the GRASP Patterns
Introduction

- Now what happens?
- What object should receive this message?
- What objects should interact to fulfill the request, and how?

Presentation

Application Logic

Clerk

Video Store

Video ID

Record Rental

Now what happens?

appLogicRequest()
Responsibility-Driven Design (RDD)

- Detailed object design is usually done from the point of view of the metaphor of:
  - Objects have responsibilities
  - Objects collaborate
  - Similar to how we conceive of people

- In RDD we do object design such that we will ask questions such as:
  - What are the responsibilities of this object?
  - Who does it collaborate with?
Responsibilities

- Responsibilities are an abstraction.
  - The responsibility for persistence.
    Large-grained responsibility.
  - The responsibility for the sales tax calculation.
    More fine-grained responsibility.

- They are implemented with methods in objects.
  - 1 method in 1 object
  - 5 methods in 1 object
  - 50 methods across 10 objects
Fundamental Principles of Object Design—GRASP Patterns

- What guiding principles to help us assign responsibilities?

- These principles are captured in the GRASP patterns.
  - General Responsibility Assignment Software Patterns.
  - Very very fundamental, simple, basic principles of object design.
The 9 GRASP Patterns

1. Expert
2. Creator
3. Controller
4. Low Coupling
5. High Cohesion
6. Polymorphism
7. Pure Fabrication
8. Indirection
9. Don’t Talk to Strangers

Memorizing and applying these is the most important, useful objective of this seminar.
(Information) Expert

- What is most basic, general principle of responsibility assignment?

- Assign a responsibility to the object that has the information necessary to fulfill it.
  - “That which has the information, does the work.”
  - E.g., What software object calculates sales tax?
    1. What information is needed to do this?
    2. What object or objects has the majority of this information.
What object creates an X?
- Ignores special-case patterns such as *Factory*.

Choose an object C, such that:
- C contains or aggregates X
- C closely uses X
- C has the initializing data for X

The more, the better.
What object in the domain (or application coordination layer) receives requests for work from the UI layer?

What object should this be?
Controller

- **Candidates:**
  - An object whose name reflects the use case.
    e.g., RentingVideosUCHandler
    part of the Application Coordination layer.
  - An object whose name reflects the overall server, business, or large-scale entity.
    A kind of “façade” object.
    e.g., Store, RentingServer
Exercise Time

- OK, before learning more patterns, we will practice.
  - We will learn the remaining patterns in this module, at a later time.

- We will do some detailed object design of the domain layer
  - Illustrating our ideas with collaboration diagrams
  - and making our design decisions with the GRASP patterns.

- For each message in the collaboration diagram, we will comment it with the choice of GRASP pattern.
  - Goal: To consciously, methodically apply these.
Low Coupling

- Assign responsibilities so that coupling remains low.

- What does low coupling mean?
High Cohesion

- Assign responsibilities so that cohesion remains high?

- What does high cohesion mean?
Polymorphism

How to design for varying, similar cases?

Assign a polymorphic operation to the family of classes for which the cases vary.
  - Don’t use case logic.

E.g., draw()
  - Square, Circle, Triangle
Pure Fabrication

- Where to assign a responsibility, when the usual options based on Expert lead to problems with coupling and cohesion, or are otherwise undesirable.

- Make up an “artificial” class, whose name is not necessarily inspired by the domain vocabulary.

  E.g., database persistence in
  - Video?
  - DatabaseFacade? (a Pure Fabrication)
Indirection

- A common mechanism to reduce coupling?

- Assign a responsibility to an intermediate object to decouple collaboration from 2 other objects.
Don’t Talk to Strangers

- How to design to reduce coupling to knowledge of the structural connections of objects?

- Don’t traverse a network of object connections in order to invoke an operation.

- Rather, promote that operation to a “familiar” of the client.