Software Specification and Design - Week5

By Keeratipong Ukachoke

Today topics

- GRASP
- Example with POS
- A few design patterns

What do we have so far?

- Requirements
- Use cases
- Domain models
- What next?

We need actual classes

- We can borrow some concepts from domain models
- We still have very important questions.
    - Which methods belong to which classes?
    - How all the classes should interact with each other?

Responsibility Driven design

- Think of software objects in term of
    - responsibilities
    - roles
    - collaborations

Two type of responsibilities

- Doing
  - Doing something itself (Creating something or doing some calculation)
  - Initiating action of other objects
  - Controlling & Coordinating activities of other objects

Two type of responsibilities

- Knowing
    - Knowing about private encapsulate data
    - Knowing about related objects
    - Knowing about things it can derived or calculate

Responsibilities are not methods

- Big responsibilities may involve several hundred methods.
- Small responsibilities can be express with one method

GRASP

- General Responsibility Assignment Software Patterns (or Principals)
    - Information Expert
    - Creator
    - Controller
    - Low Coupling
    - High Cohesion
    - Polymorphism
    - Pure Fabrication
    - Indirection
    - Protected Variations

What are patterns?

A pattern is a named and well-known problem/solution pair that can be applied in new contexts, with advice on how to apply it in novel situations and discussion of its trade-offs, implementations, variations, and so forth.

New Pattern?

GRASP Example

- Let’s pick an extremely easy example to demonstrate how we can apply GRASP to the design process.

Mono poly example

- See the diagram on the board together

Who creates the Square?

- Any object can create squares, right?
- How would great engineers choose?
- Should we have another class called ‘Dog’ and use it create squares?

1- The Creator Pattern

Name: Creator

Problem: Who creates an A?

Advice: Assign class B the responsibility to create and instance of class A if one of these is true (The more the better.)

- B contains or aggregates A
- B records A
- B closely uses A
- B has initialising data for A

Who is the best candidate?

- Let’s draw a (partial) sequence diagram
- Let’s draw a (class) diagram

Who can give us square by a key?

- What if we need to check on particular square and see how many pieces are on it?
- Should should be responsible for handling this?

2- Information Expert

Name: Information Expert

Problem: What is a basic principle by which to assign responsibilities to objects?

Advice: Assign a responsibility to the class that has the information needed to fulfill it.

Who is the best candidate?

- Let’s draw a (partial) sequence diagram
- Let’s draw a (class) diagram

Why?

- Why don’t we create a new class to get square by name?
- Let’s see class diagram and sequence diagram.
- See the problem?

3- Low Coupling

Name: Low Coupling

Problem: How to reduce the impact of change?

Advice: Assign responsibilities so that (unnecessary) coupling remains low. Use this principle to evaluate alternatives.

How about UI of the game

- Should each object represents a UI element?
- When a player makes an action, which class should be responsible for doing the work?

4- Controller

Name: Controller

Problem: What first object beyond the UI layer receives and coordinates a system operation?

Advice: Assign responsibilities to an object representing one of these choices

- Represent the overall system or root

- Represent a use case scenario which the operation occurs

The UI and controller

- Let’s see the system sequence diagram

Cohesion of the system

- Loog at these 2 diagrams, which one is better and why?

Cohesion of the system

- Big class with 100 methods and 200 LOCs vs small class with 10 methods and 200 LOCs?
- Not focus
- Hard to change
- Low coupling

5 - High Cohesion

Name: Heigh Cohesion

Problem: How to keep objects focused, understandable, and manageable?

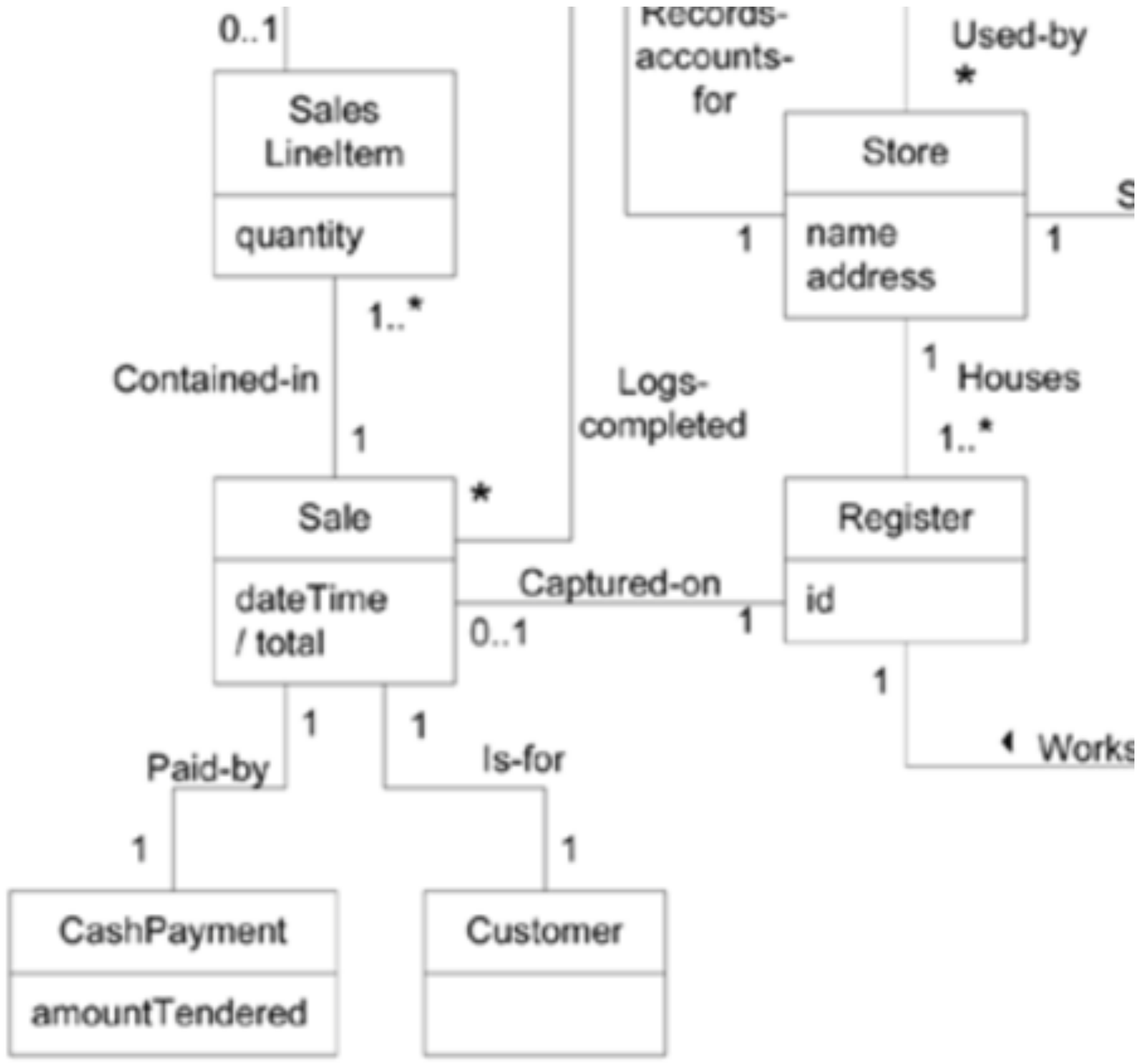
Advice: Assign responsibilities so that cohesion remains high. Use this to evaluate alternatives.

GRASP

- General Responsibility Assignment Software Patterns (or Principals)
    - Information Expert
    - Creator
    - Controller
    - Low Coupling
    - High Cohesion
    - Polymorphism
    - Pure Fabrication
    - Indirection
    - Protected Variations

Example of 5 principles in POS

Who creates a SalesLineItem?



The Creator Pattern

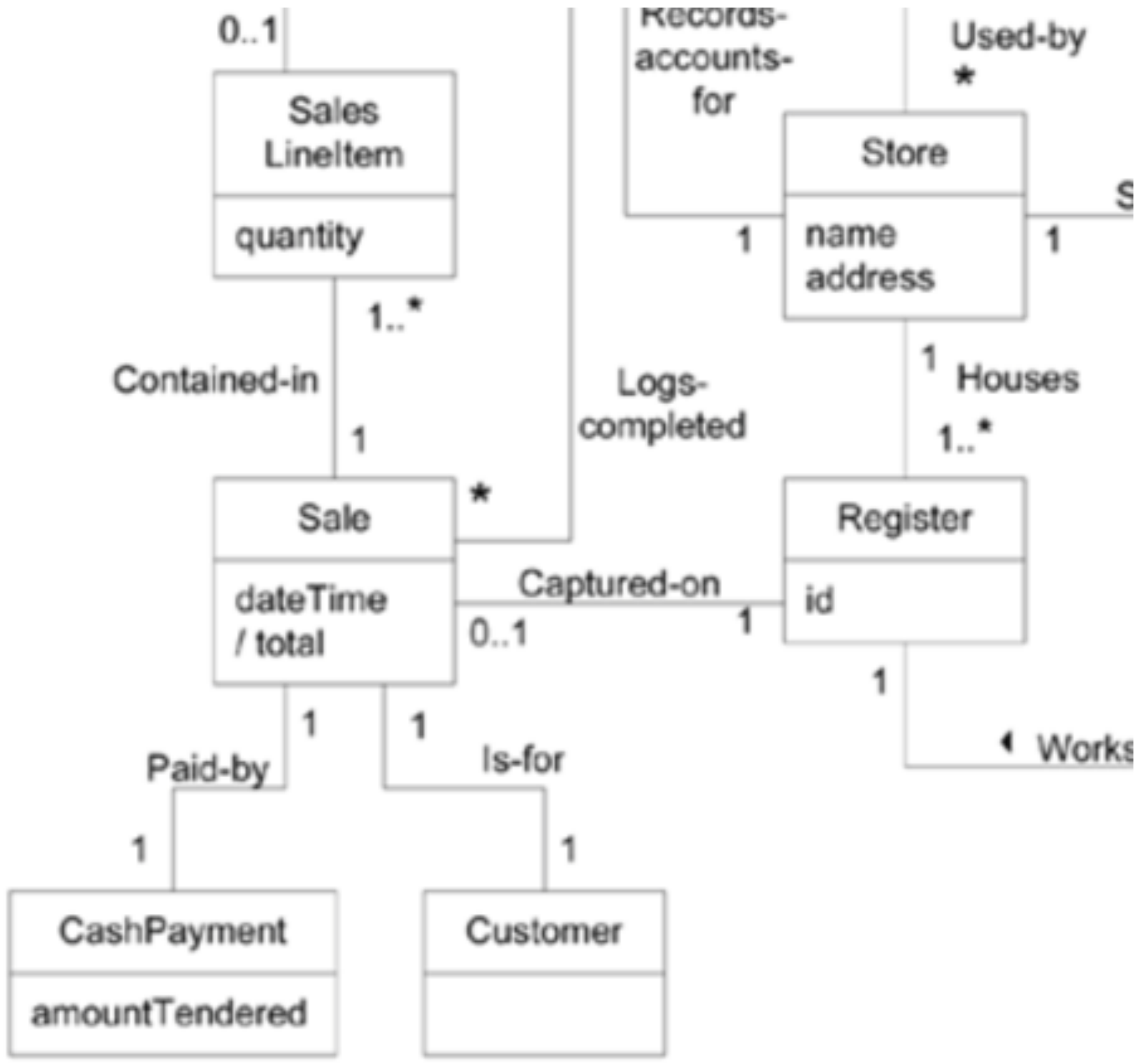
Name: Creator

Problem: Who creates an A?

Advice: Assign class B the responsibility to create and instance of class A if one of these is true (The more the better.)

- B contains or aggregates A
- B records A
- B closely uses A
- B has initialising data for A

Who should know the grand total of a sale?



Information Expert

Name: Information Expert

Problem: What is a basic principle by which to assign responsibilities to objects?

Advice: Assign a responsibility to the class that has the information needed to fulfill it.

Who should create and Link payment?

- Register records a payment
- Let’s see diagram

Low Coupling

Name: Low Coupling

Problem: How to reduce the impact of change?

Advice: Assign responsibilities so that (unnecessary) coupling remains low. Use this principle to evaluate alternatives.

What should be the first class below the UI

- See the diagrams
- Let’s talk about Delegation Pattern
- Let’s talk about Command Pattern

Controller

Name: Controller

Problem: What first object beyond the UI layer receives and coordinates a system operation?

Advice: Assign responsibilities to an object representing the overall system or root

Delegation Pattern

- Delegate tasks to a helper object
- Your boss want to get a coffee, he tells you to get a coffee for him.

Command Pattern

- A class is used to represent a function need to be performed
- Let’s see example.

Assignment

Let’s do some work