Developing Domain Model

Daum Corp.
백명석

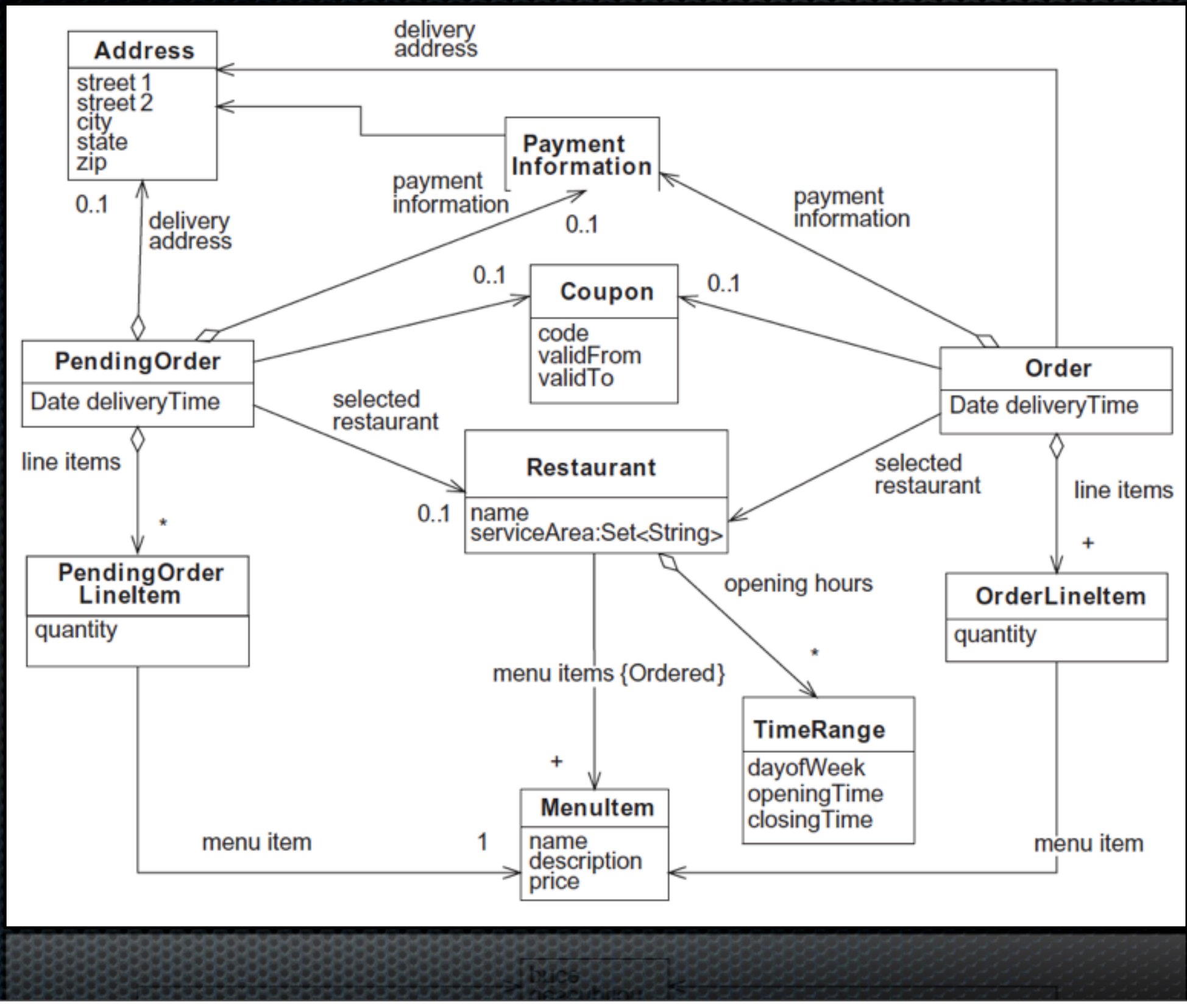
발표목차

- 1. 클래스, 속성, 관계 식별
- 2. 도메인 모델에 행위 추가하기
  - 2.1 요구 사항 식별하기
  - 2.2 메소드 식별하기
  - 2.3 TDD로 메소드 구현하기

1. 클래스/속성/관계 식별

- 도메인 영역의 주요 개념(명사)를 식별
- “Applying UML and Patterms” 참조

1. 초기 도메인 모델



초기 도메인 모델은

- 클래스, 클래스의 속성, 클래스들 간의 관계를 갖는다.

2. 도메인 모델에 행위 추가하기

- 도메인 모델에 행위를 추가함으로써 도메인 모델에 생명력을 부여
- 도메인 모델의 행위를 결정하기 위해 도메인 모델의 책임(Responsibility)과 상호작용(Collaboration)을 식별
- 클래스의 책임
  - 클래스가 아는 것(속성, 관계), 하는 것, 결정 하는 것 등
- 클래스의 상호작용
  - 책임 수행을 위해 호출하는 다른 클래스들

2. 도메인 모델에 행위 추가하기

- 책임과 상호작용을 식별하는 절차
  - 요구 사항(유스케이스, 유저스토리, UI 디자인) 분석을 통해 어플리케이션이 처리해야 하는 요구사항 식별
  - 도메인 모델의 클라이언트(프리젠테이션티어 등)에게 도메인 모델을 노출하기 위한 도메인 모델의 인터페이스(타입, 메소드) 결정
  - 해당 인터페이스를 각각의 요구사항을 고려하여 TDD 접근법으로 구현

2.1 요구 사항 식별하기

- 처리해야 할 요구사항 식별/어떻게 응답할 지 결정
- UI 디자인, 유스케이스, 유저스토리 등을 분석해서
- 요구 사항은 2가지 부분으로 구성
  - 사용자 행위
  - 사용자 행위 요청에 대한 어플리케이션의 응답(책임)
- 어플리케이션의 책임은 2가지로 그룹핑
  - 사용자 입력 검증, 값 계산, 데이터베이스 갱신
  - 값 출력

사용자 행위 기술: “고객이 배송 정보를 입력한다” 등과 같은…
어플리케이션의 응답: 배송 시간 검증, 배송 정보 갱신 등과 같은…

어플리케이션의 책임은 2가지로 그룹핑
사용자 입력 검증, 값 계산, 데이터베이스 갱신 ← 서비스에 의해 구현됨
값 출력 ← 리파지토리에 의해 구현됨(정확히 말하면 값 출력은 뷰가 하고 값 출력을 위한 데이터 제공을 리파지토리가 함)

2.2 메소드 식별하기

- 각 요청에 대해 2가지 메소드들이 존재
  - 서비스 메소드
    - 사용자 요청 검증
    - 계산 수행
    - 데이터베이스 갱신
  - 리파지토리 메소드
    - 출력을 위한 데이터 반환
- 도메인 모델의 클라이언트는 도메인 티어를 2번 호출
  - 서비스 메소드 + 리파지토리 메소드

서비스 메소드가 화면에 출력할 데이터를 반환하도록 하면 도메인 티어가 프리젠테이션 티어에 종속적이게 된다(tightly coupled). – CQS 위반
프리젠테이션 티어로 부터 서비스 객체를 분리하고 프리젠테이션 티어가 자신이 필요한 데이터를 얻기 위해 한번 더 도메인 티어를 호출하도록 하는 것이 좋다.

2.3 TDD로 메소드 구현하기

- 대상 서비스 메소드에 대해 하나 이상의 테스트케이스를 작성하는 것으로 시작
  - 각 테스트케이스는 서로 다른 상황을 재현하기 위해 다른 인자로 구성된다
- mock 객체를 이용해서
  - service methods → repository methods 순으로 top-down 방식으로 구현
  - 구현을 하다가 발견되는 collaborator를 구현하기 위해 머릿속에서 context-switching이 일어날 필요가 없어서 집중하면서 구현

Reference

- POJOs in Action,
  - http://www.manning.com/crichardson
  - 3. Using the Domain Model pattern