

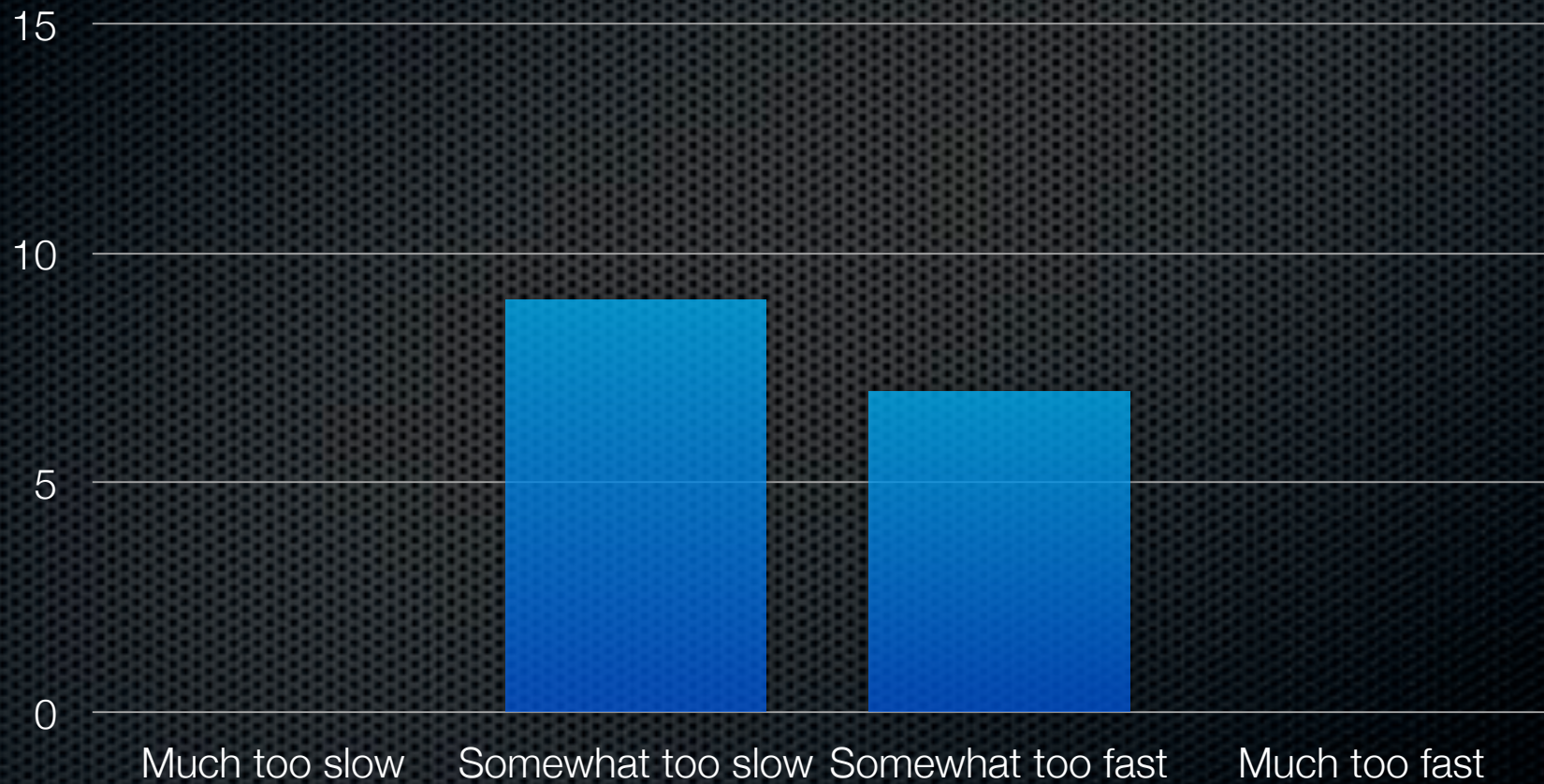
Iteration 3 Kick Off, Domain Model Refinement

Curt Clifton

Rose-Hulman Institute of Technology

2/3 Course Evaluation Results

Lecture Pace



Lecture +/∂

+ Quizzes (5)

+ In-class examples (4)

+ Slides (3)

+ Style/enthusiasm (3)

+ Group work (3)

∂ Detailed example of everything (2)

∂ PDFs of slides sometimes jumbled (1)

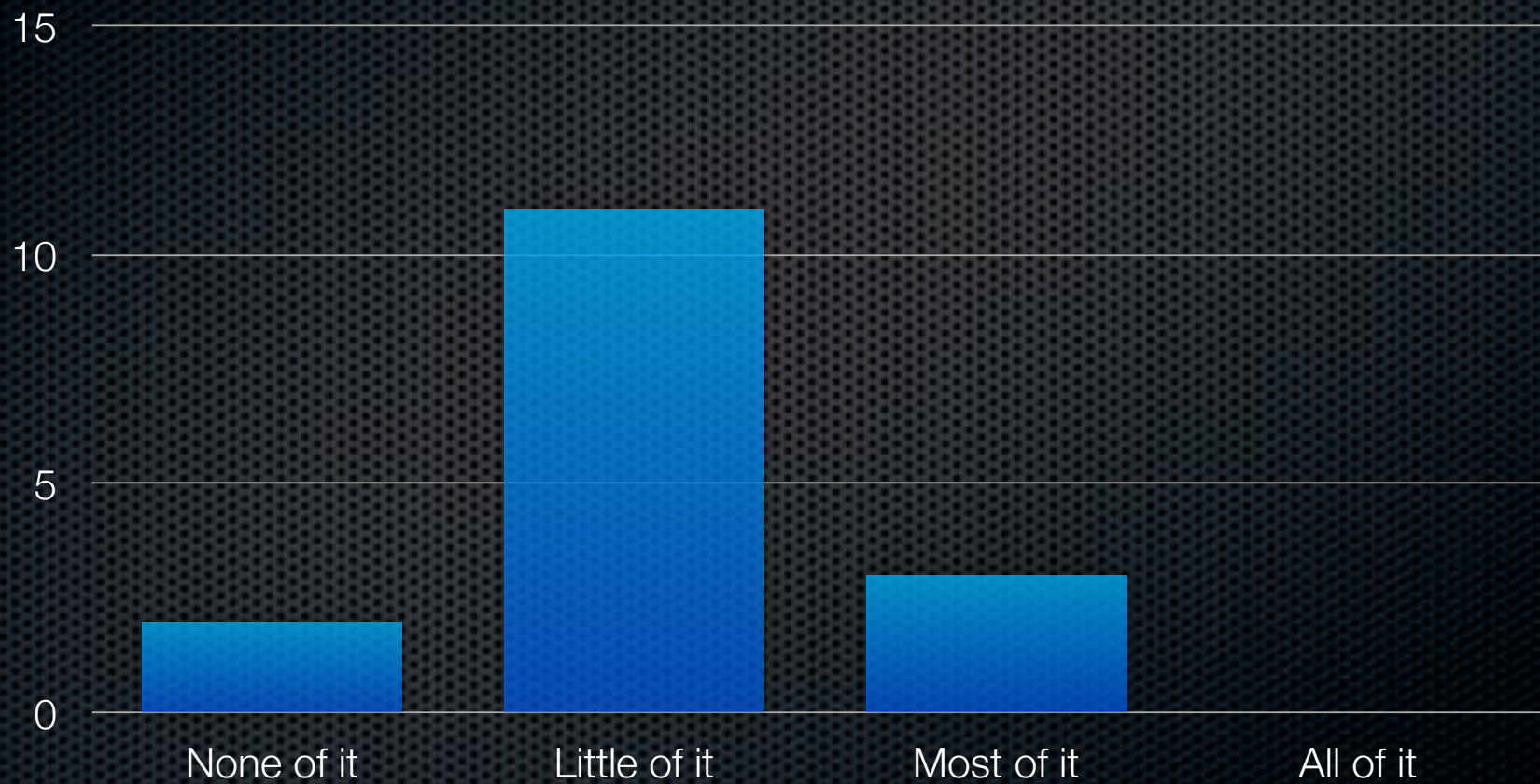
Daily Quizzes Helpfulness



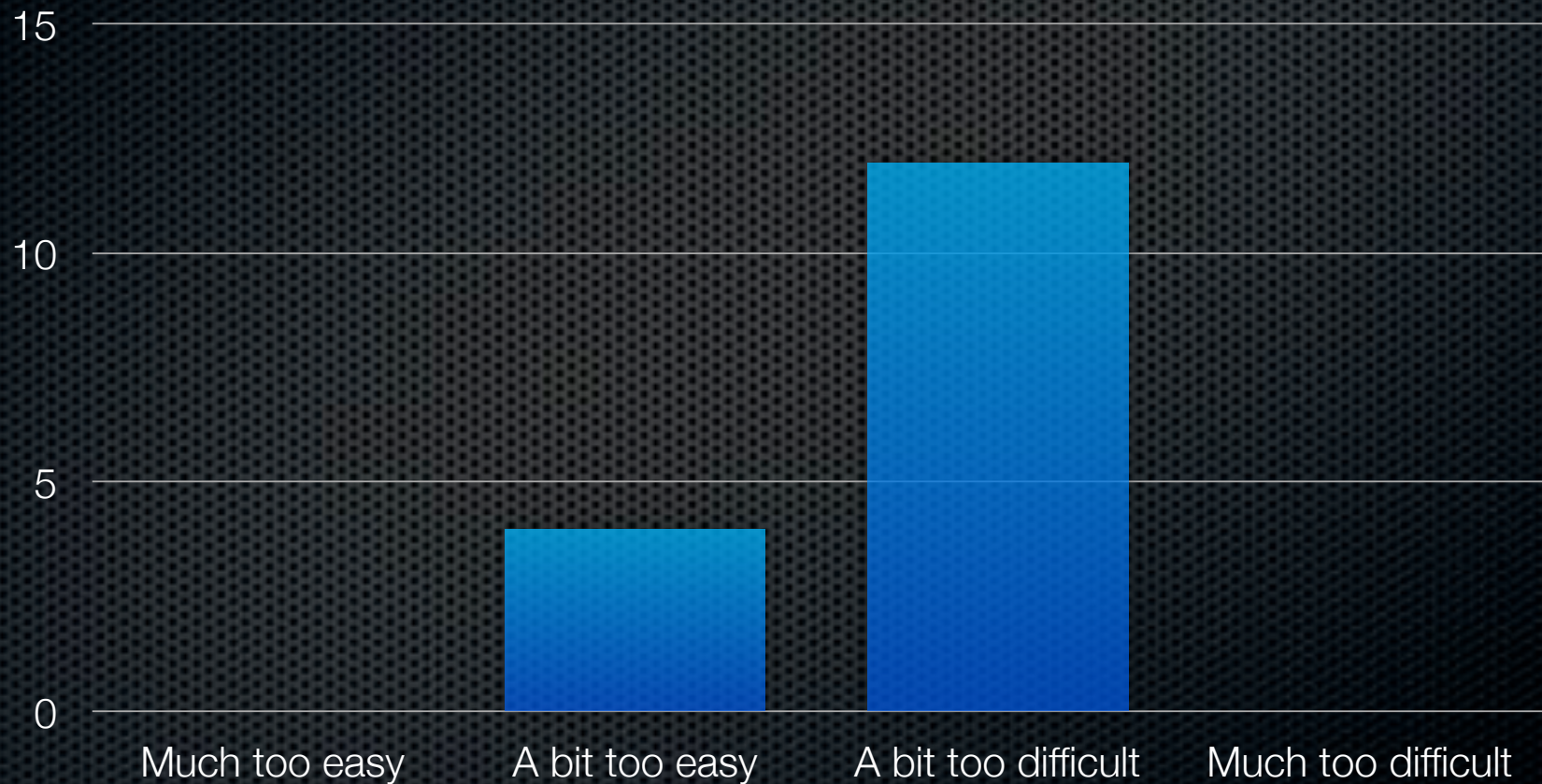
Daily Quizzes +/∂

- + Project-related pattern questions (3) ↔ ∂ Team questions aren't working (2)
- + Help to pay attention (3)
- + Diagrams (3)
- + Sync with lecture (2)
- + Get-to-know-you questions (2)
- ∂ More practice problems on quizzes (2) ↔ ∂ Fewer practice problems on quizzes (2)

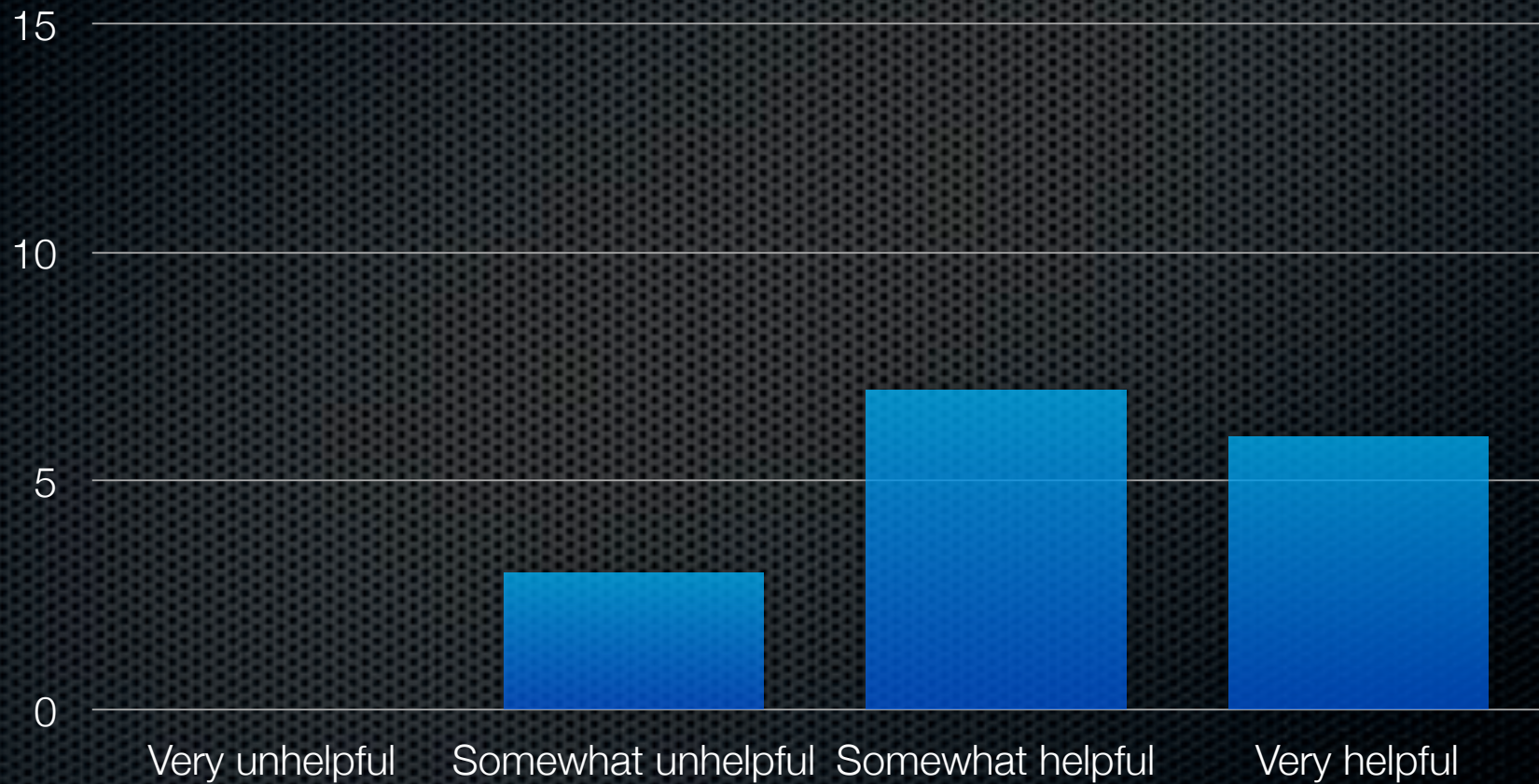
Reading Done



Homework Difficulty



Homework Helpfulness



Homework +/∂

+ Reinforcement (5)

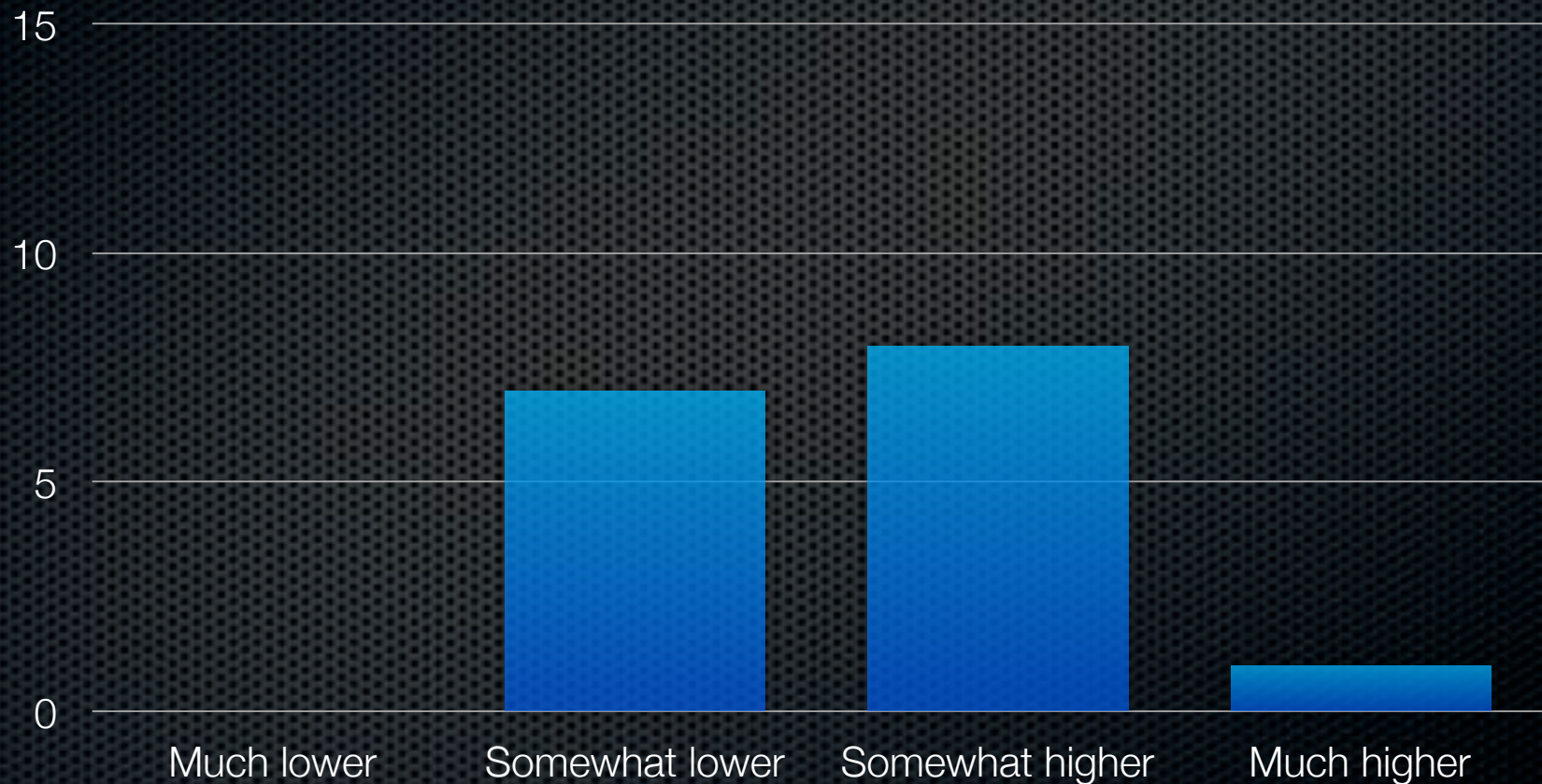
+ Hands on learning (3)

+ Good prep for
milestones (2)

+ Pattern questions (2)

∂ Not related to our
projects (2)

Workload Compared to Average Rose Course



Other Comments

- ✦ Direct meetings with me on team projects (1)
- ✦ Some team projects are a poor fit for the course (1)
 - ✦ Need smaller projects to be able to do them (1)
 - ✦ Need bigger projects for the principles to make sense (1)

Domain Model Refinement

Recall: Techniques for identifying conceptual classes

- Conceptual category lists
- Noun phrase identification
- Existing domain models

Conceptual Category List on NextGen POS, iteration 3

| Category | Examples |
|--|--|
| physical or tangible objects | <i>CreditCard, Check</i> |
| transactions | <i>CashPayment, CreditPayment, CheckPayment</i> |
| other systems external to ours | <i>CreditAuthorizationService, CheckAuthorizationService</i> |
| organizations | <i>CreditAuthorizationService, CheckAuthorizationService</i> |
| records of finance, work, contracts, legal matters | <i>AccountsReceivable</i> |

Noun Phrase Identification on NextGen POS, iteration 3

*Payment
Authorization
Request*

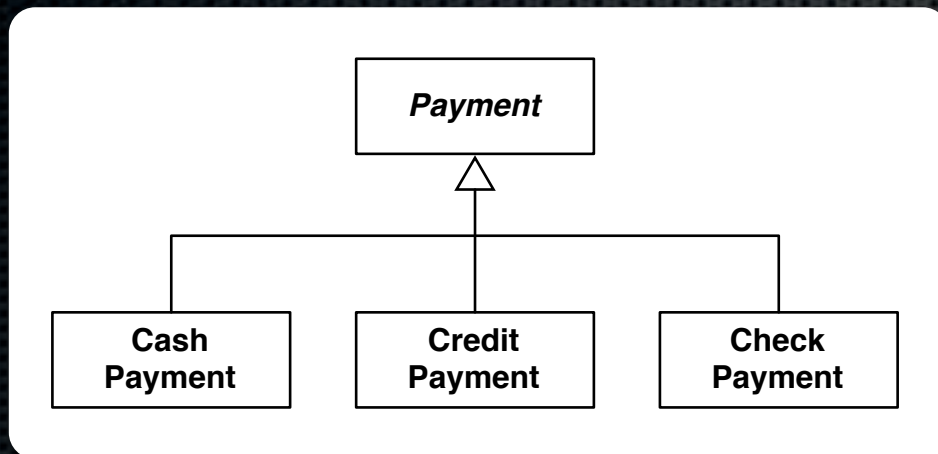
*Credit Account
Information*

*Payment
Authorization
Service*

*Payment
Approval*

- Use Case UC1: Process Sale
- Extensions:
6. Paying by credit:
1. Customer enters their **credit account information**.
 2. System sends **payment authorization request** to an external **Payment Authorization Service System**, and requests **payment approval**.
 - 2a. System detects failure to collaborate with external system:
 1. System signals error to Cashier.
 2. Cashier asks Customer for alternate payment.
 3. System receives **payment approval** and signals approval to Cashier.
 3. System receives **payment denial**:
 1. System signals denial to Cashier.
 2. Cashier asks Customer for alternate payment.
 4. System records the **credit payment**, which includes the payment approval.
 5. System presents credit payment signature input mechanism.
 6. Cashier asks Customer for a credit payment signature. Customer enters signature.
- 7c. Paying by check:
1. The Customer writes a **check**, and gives it and their **driver's license** to the Cashier.
 2. Cashier writes the driver's license number on the check, enters it, and requests **check payment authorization**.
 3. Generates a **check payment request** and sends it to an external **Check Authorization Service**.
 4. Receives a **check payment approval** and signals approval to Cashier.
 5. System records the **check payment**, which includes the payment approval.

Generalization-Specialization Class Hierarchy

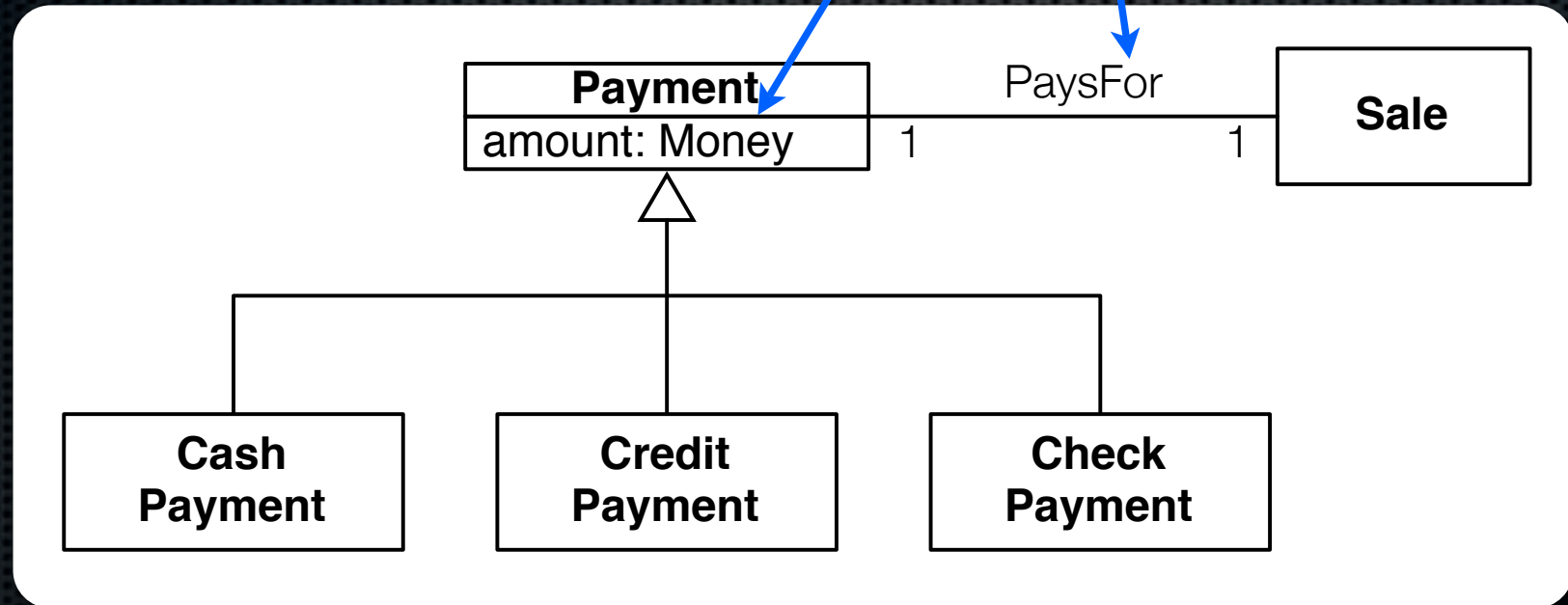


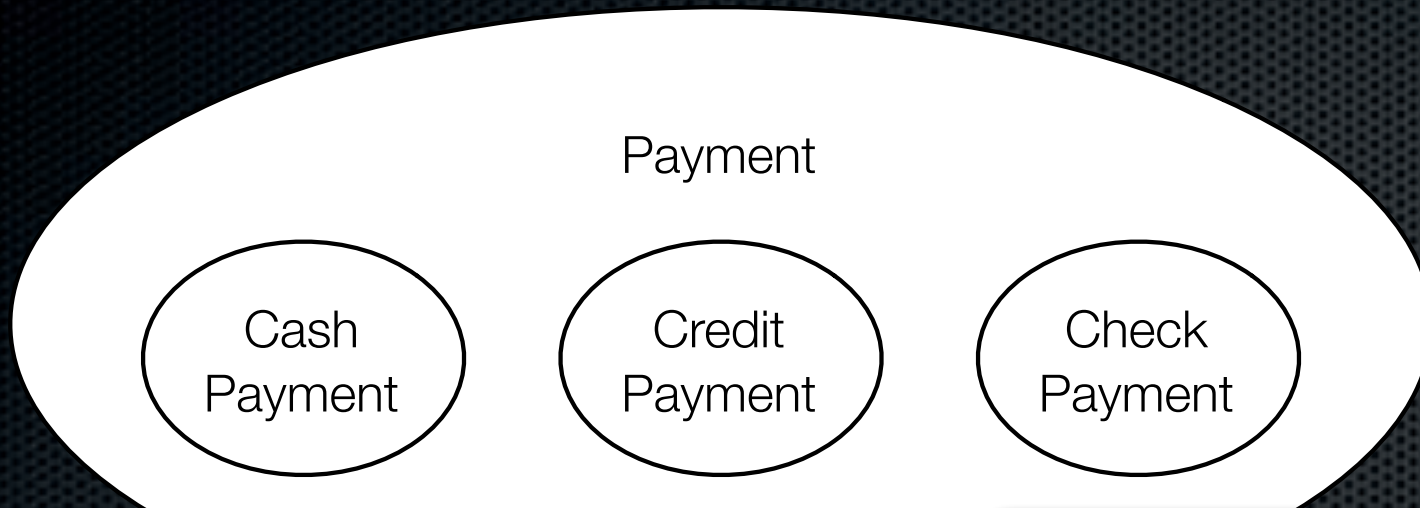
Why? Can understand concepts in more general terms. *Payment* gives our brains less to deal with.

- ✦ Conceptual classes, not software classes
- ✦ Domain modeling!
- ✦ *Generalization*: finding commonalities among concepts
 - ✦ *Superclass*: general concept
 - ✦ *Subclass*: specialized concept

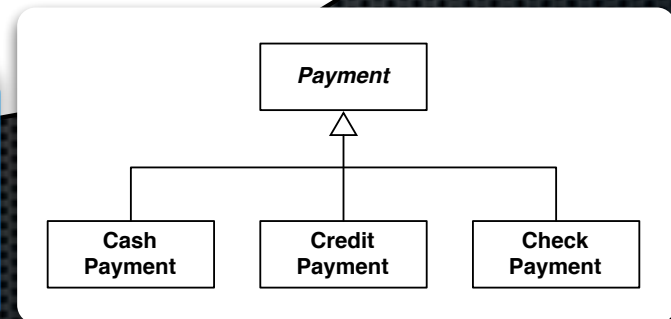
Generalization

Common features of Payments





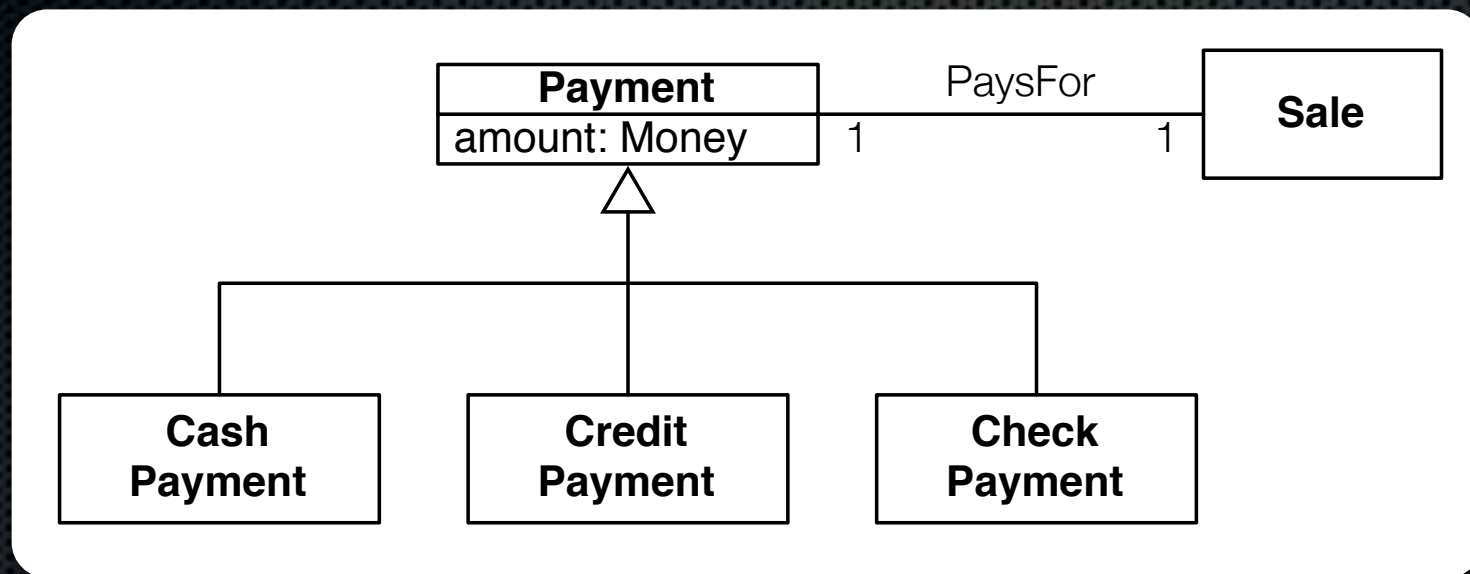
Is-a rule: Subclass **is a** superclass,
e.g., CashPayment is a Payment



Generalizations and Sets

All members of a conceptual subclass set are members of their superclass set

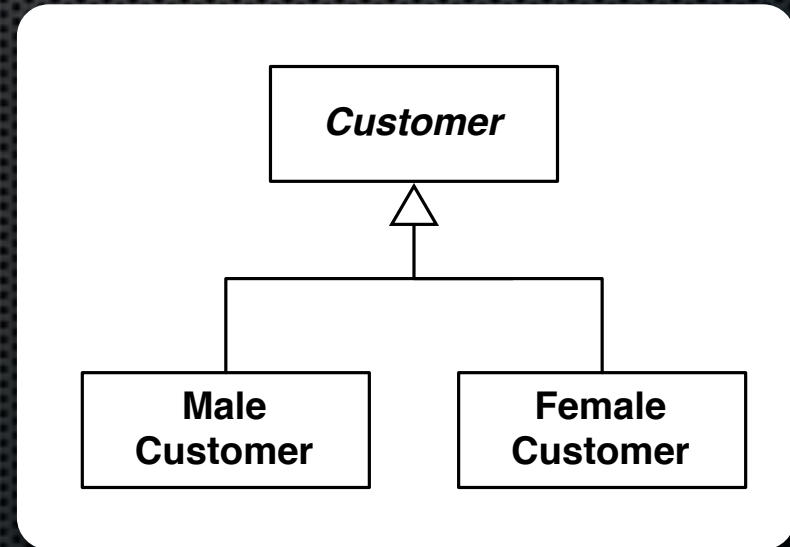
Subclass Conformance— The 100% Rule



The subclass must conform to all of the superclass's *attributes* and *associations*.

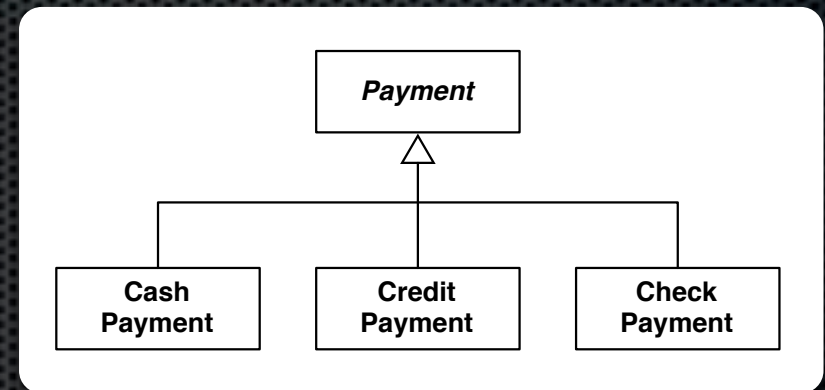
When should we define a Conceptual Subclass?

- ✦ Does this make sense...
 - ✦ for NextGen POS?
 - ✦ for other domains?



When should we define a Conceptual Subclass?

- When the subclass...
 - has additional attributes
 - has additional associations
 - is operated on or handled differently
 - represents an animate thing that behaves differently

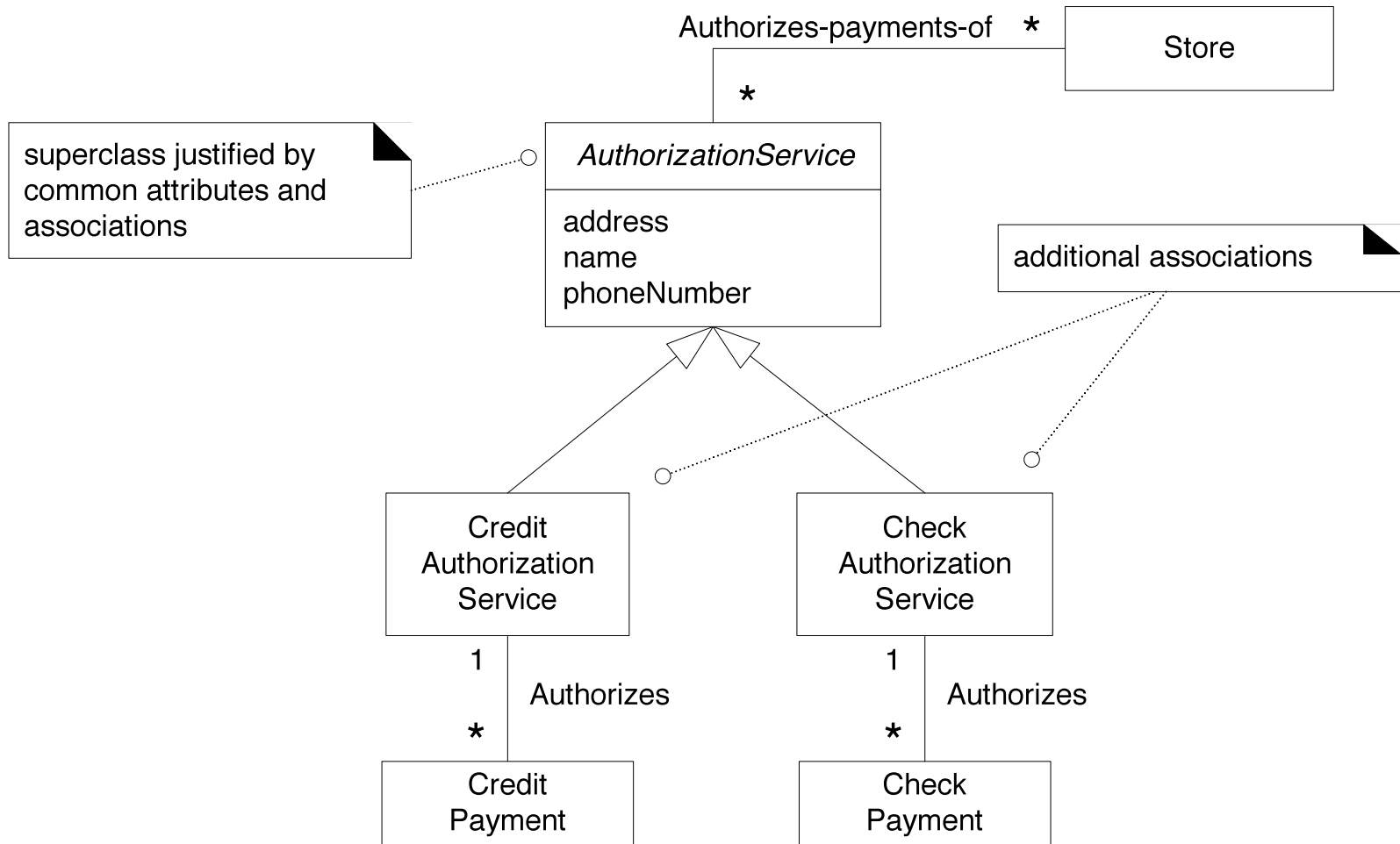


Which of these apply here?

When should we group classes and extract a superclass?

- Create a superclass when:
 - Potential *subclasses represent variations* of a similar concept (e.g., Video, Game → RentableItem)
 - Subclasses will conform to *100% and is-a rules*
 - There are *common attributes or associations* that could be pulled into superclass

Another Example

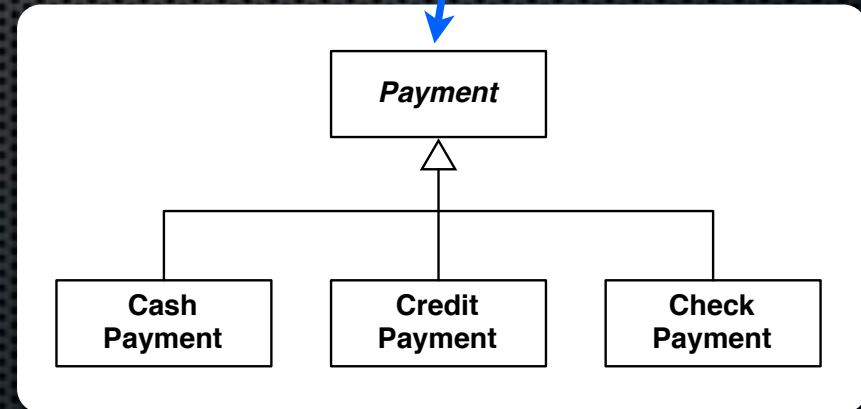


Abstract Conceptual Classes

- If every member of a class C must be a member of a subclass, then C is an *abstract conceptual class*

What does this mean in terms of our set idea?

Italics (or {abstract} keyword) indicate abstract class



Cartoon of the Day



Used with permission. <http://notinventedhe.re/on/2010-2-1/comic>

Thinking Ahead...
Work in teams on Q4

Q4