Object-Oriented Design and UML

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

March 31, 2022

Outline

Outline

Software Design

- Design starts mostly from/with requirements – evolving mostly from functionalities and other non-functional characteristics
  - In the waterfall model Design generally occurs after Requirements
  - In agile, design is performed during in each iteration
- To answer: How is the software solution going to be structured?
  - What are the main components – (functional composition) often directly from requirements' functionalities (e.g., use cases, user stories, scenarios)
  - How are these components related? – Possibly re-organize the components (composition/decomposition)
- Two main levels of design:
  - Architectural (high level) design
  - Detailed design
  - Different design concerns at different abstraction levels (e.g. classes vs. modules vs. entire system)
- How should we depict design – what notation/language?

Detailed Design

Discussed and to discuss

- Functional decomposition
- Database design
- Objected-Oriented desgn and Unified Modeling Language (UML)

Outline

Why model software?

- Engineers have always modeled things they are planning to build
- Displays a engineered system at a particular level of abstraction
- Helps one think clearly about the system
- Crucial in communicating to others the structure of a system
- Makes working in a team possible

Discussed models of database and UI design

Some History

- Models have always existed in software
  - In fact, at a point there were too many of them which made it difficult to translate between them
  - Companies using certain models would go out of business, rendering models useless
- The Object Management Group (OMG) a consortium of many companies drafted the Unified Modeling Language (UML) standard
  - This is now the de facto standard for most SW modeling
  - Now we are at UML version $\tilde{2}.5$

Outline

UML

UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system

- Consists of several different diagram types
- Can be used at different abstraction levels
- from business processes to individual language statements

Note: it's a language, not a method or procedure

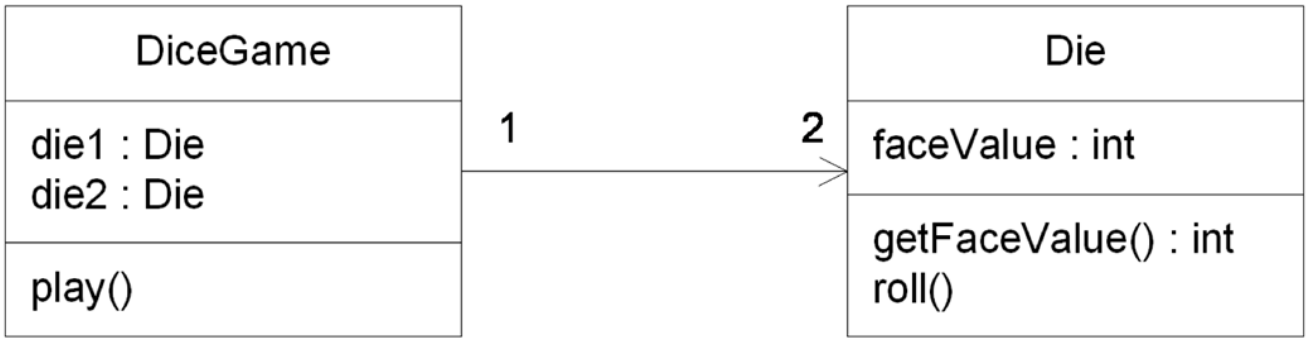
Modeling a Simple Application

Consider a dice game application

- Play a Dice Game: Roll two dice. If the dice value totals seven, player wins; otherwise, player loses

Modeling the Dice Game

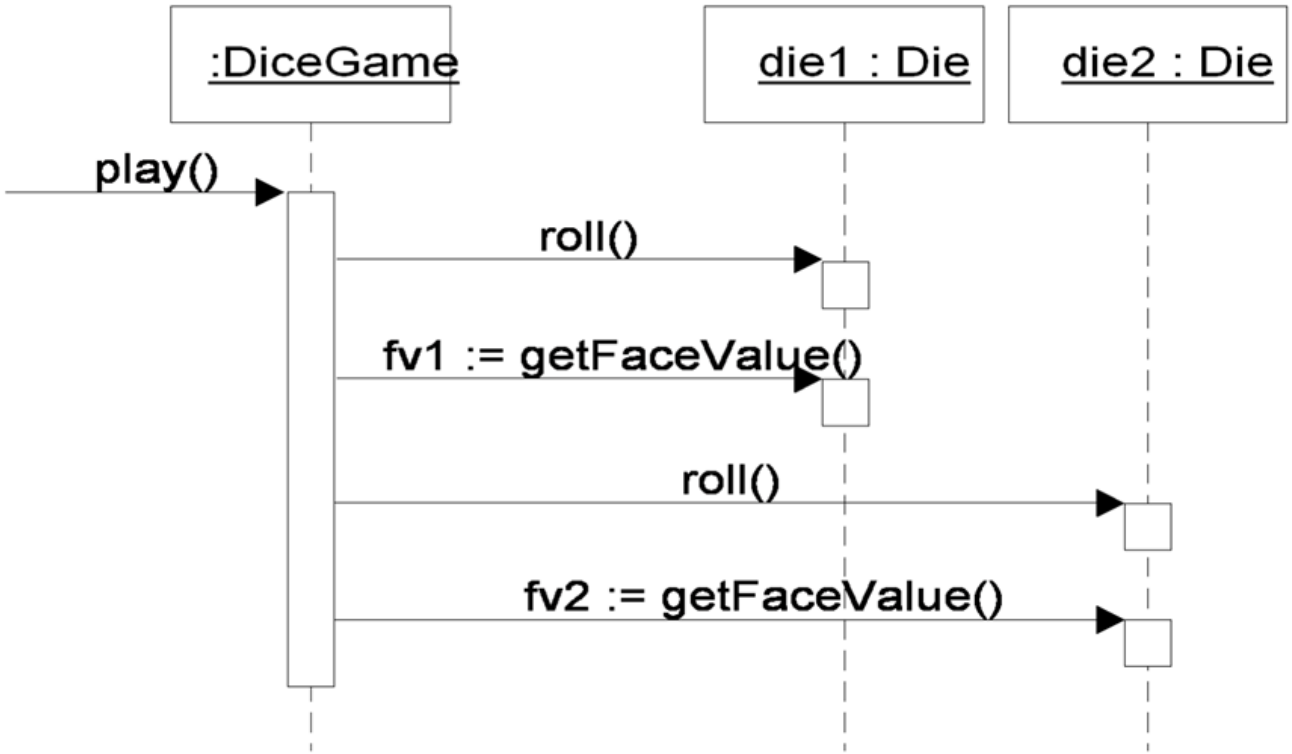
A UML class diagram



Note. This is part of the whole model (you may have more classes ...)

Modeling the Dice Game

A UML sequence diagram



Note. This is part of the whole model (you may have more classes and scenarios ...)

Outline

UML Diagrams

UML 2 defines 13 basic diagram types, divided into two general sets:

- Structural Modeling Diagrams – Used to model the "things" that make up a model, such as, the classes, objects, interfaces and physical components. In addition, they are used to model the relationships and dependencies between elements
- Behavioral Modeling Diagrams – Capture the varieties of interaction and instantaneous states within a model as it "executes" over time; tracking how the system will act in a real-world environment, and observing the effects of an operation or event, including its results

Structural Modeling Diagrams

- Class diagrams define the basic building blocks of a model: the types, classes and general materials used to construct a full model.
- Object diagrams show how instances of structural elements are related and used at run-time
- Component diagrams are used to model higher level or more complex structures, usually built up from one or more classes, and providing a well defined interface.
- Deployment diagrams show the physical disposition of significant artifacts within a real-world setting.
- and Package and Composite structure diagrams

Behavioral Modeling Diagrams

- Use Case diagrams are used to model user/system interactions. They define behavior, requirements and constraints in the form of scripts or scenarios.
- State Machine diagrams are essential to understanding the instant to instant condition, or "run state" of a model when it executes.
- Communication diagrams show the network, and sequence, of messages or communications between objects at run-time, during a collaboration instance.
- Sequence diagrams are closely related to communication diagrams and show the sequence of messages passed between objects using a vertical timeline.
- Timing diagrams fuse sequence and state diagrams to provide a view of an object's state over time, and messages which modify that state.
- and Activity, Interaction oveview diagrams

Outline

UML Object-Constraint Language (OCL)

- A language intended to provide some rules to a UML diagram
- Mathematical in nature, it specifies some properties that must be met in order for the model to be used appropriately
  - First order predicate calculus
  - Invariants, preconditions, postconditions etc.
  - Examples: all parameters must be > 0, result must not be an empty list

The Dice Game: Example OCL


```
context Die
inv: faceValue >= 1 and faceValue <= 6

context Die::getFaceValue
post: result = faceValue
```

Explaining the Example: Invariant

Definition

- An invariant is a constraint that should be true for an object during its complete lifetime.
- Invariants often represent rules that should hold for the real-life objects after which the software objects are modeled.
- Syntax

```
context <classifier>
inv [<constraint name>]: <Boolean OCL expression>
```

Explaining the Example: Postcondition

Definition

- Constraint that must be true just after to the execution of an operation
- Postconditions are the way how the actual effect of an operation is described in OCL.
- Syntax

```
context <classifier>::<operation> (<parameters>)
post [<constraint name>]:
<Boolean OCL expression>
```

Model-Driven Architecture

- MDA is a process that may use UML, or another modeling approach to bring development closer to the domain expert (user)
  - User expresses the application's needs in a model he or she understands
  - Code is generated based on the model
- Some refinement takes place from model that is completely in the customer's domain, to one that is closer to the actual deployment
  - Platform Independent Model (PIM)
  - Platform Specific Model (PSM)
  - Platform Definition Model (PDM)

Outline

Discussing Several UML Diagrams

Several UML diagrams perhaps are frequently used than the others,

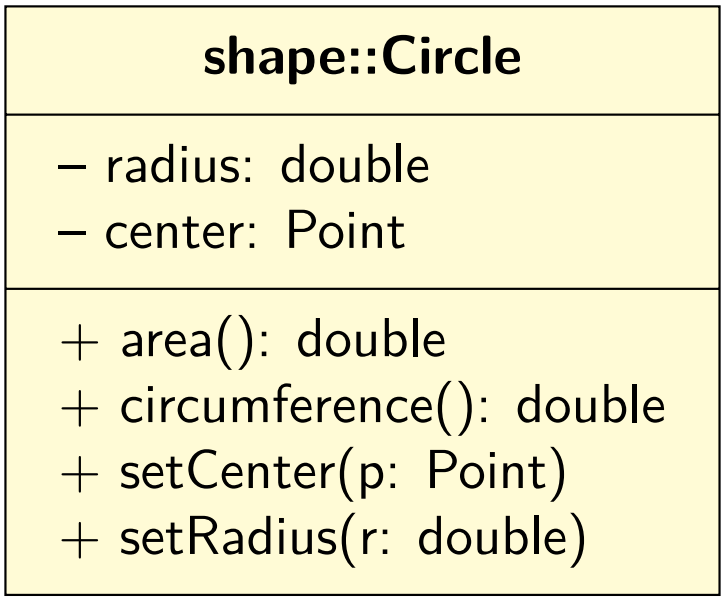
- Class diagrams
- Use Case diagrams
- Sequence diagrams

UML Class Diagram

- Probably the most popular diagram in UML
- Encodes classes and relationships between them
- An example of a structural diagram

Representing Class

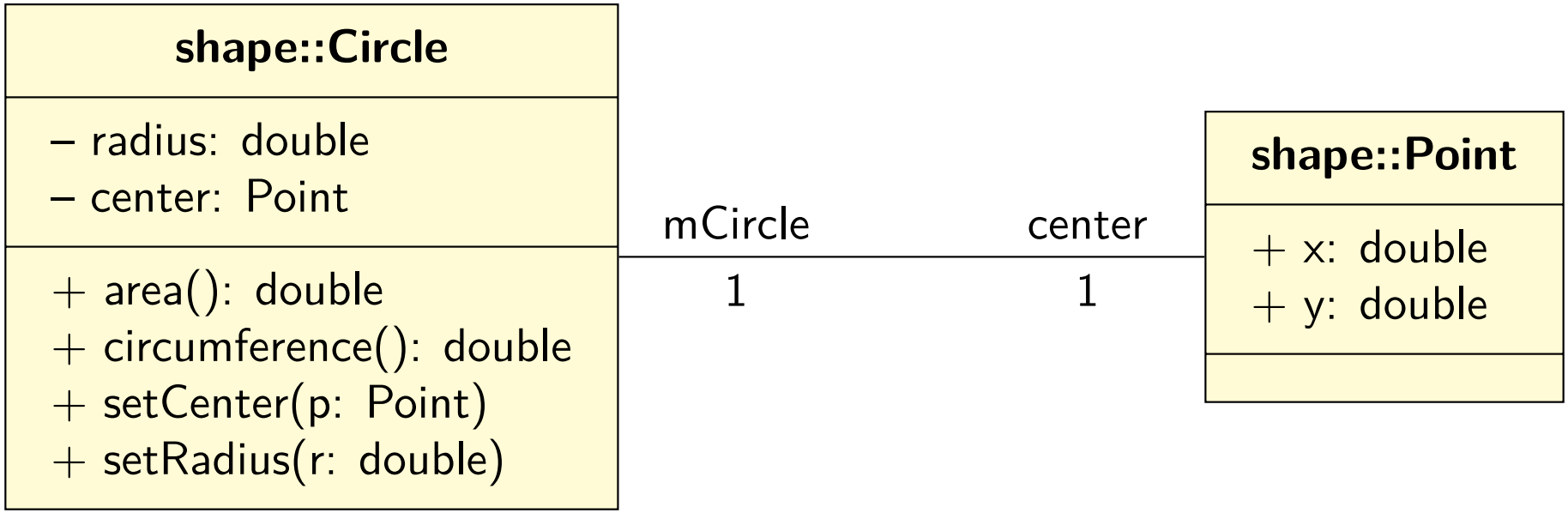
- attributes and operations, can be of several types of visibility:
  - + (public);
  - − (private);
  - # (protected);
  - ∼ (package)
- define an array using the [] syntax



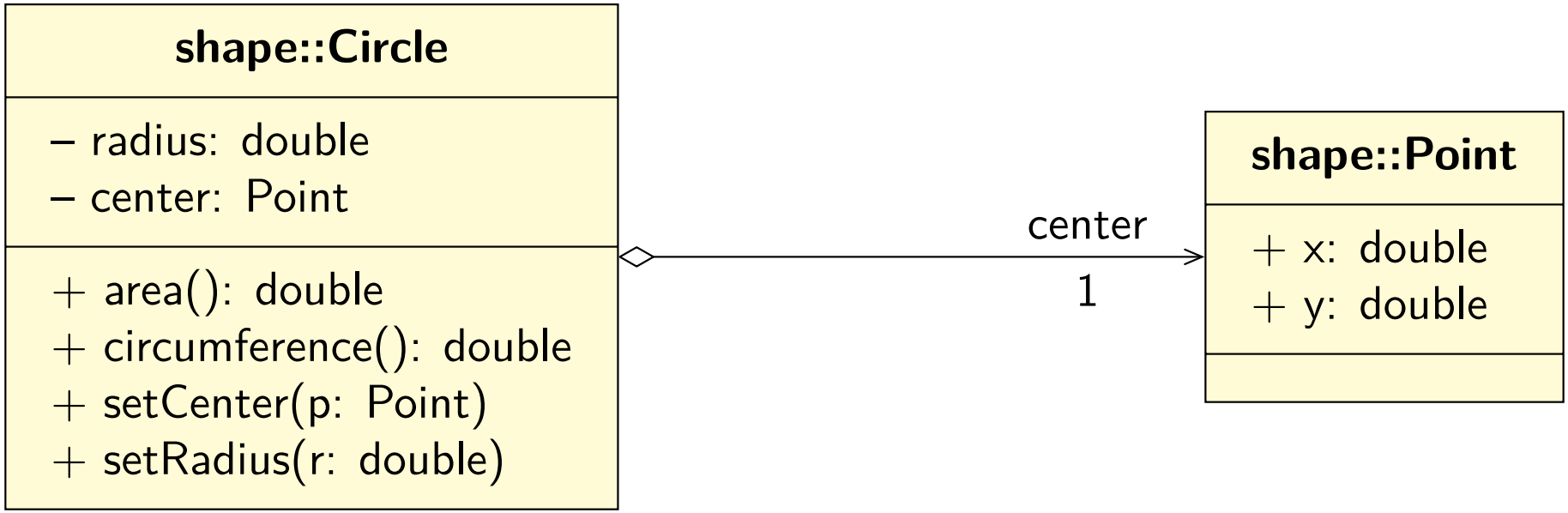
Representing Relations

- Association
- Aggregation
- Composition

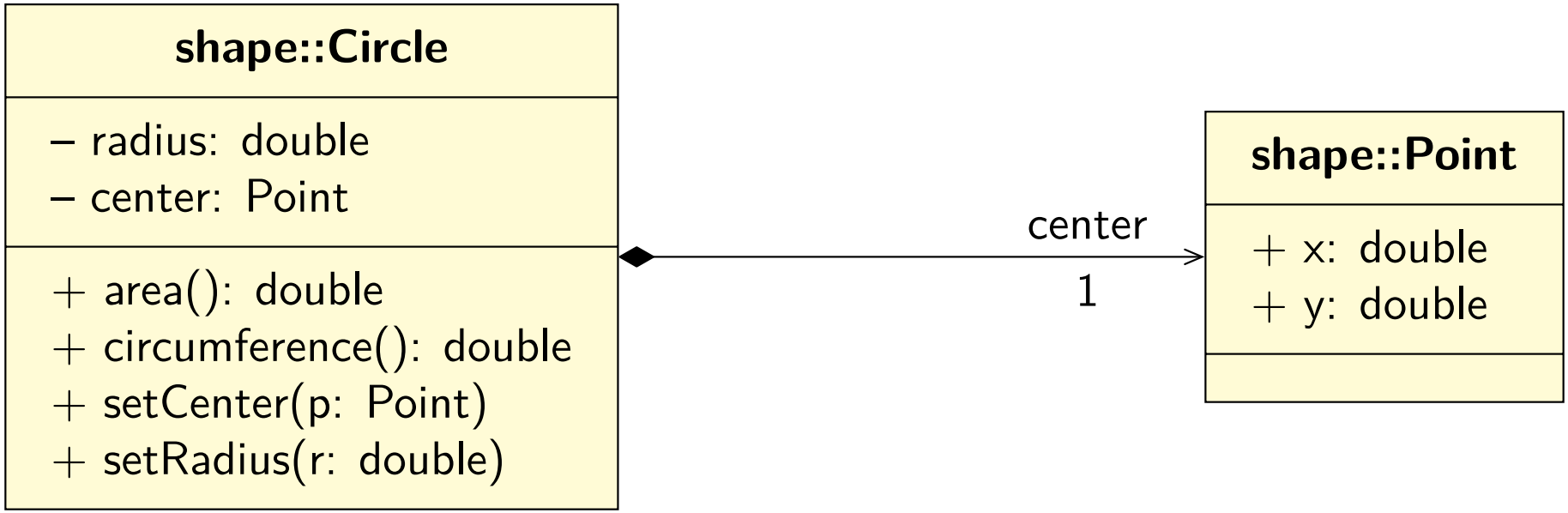
Representing Association



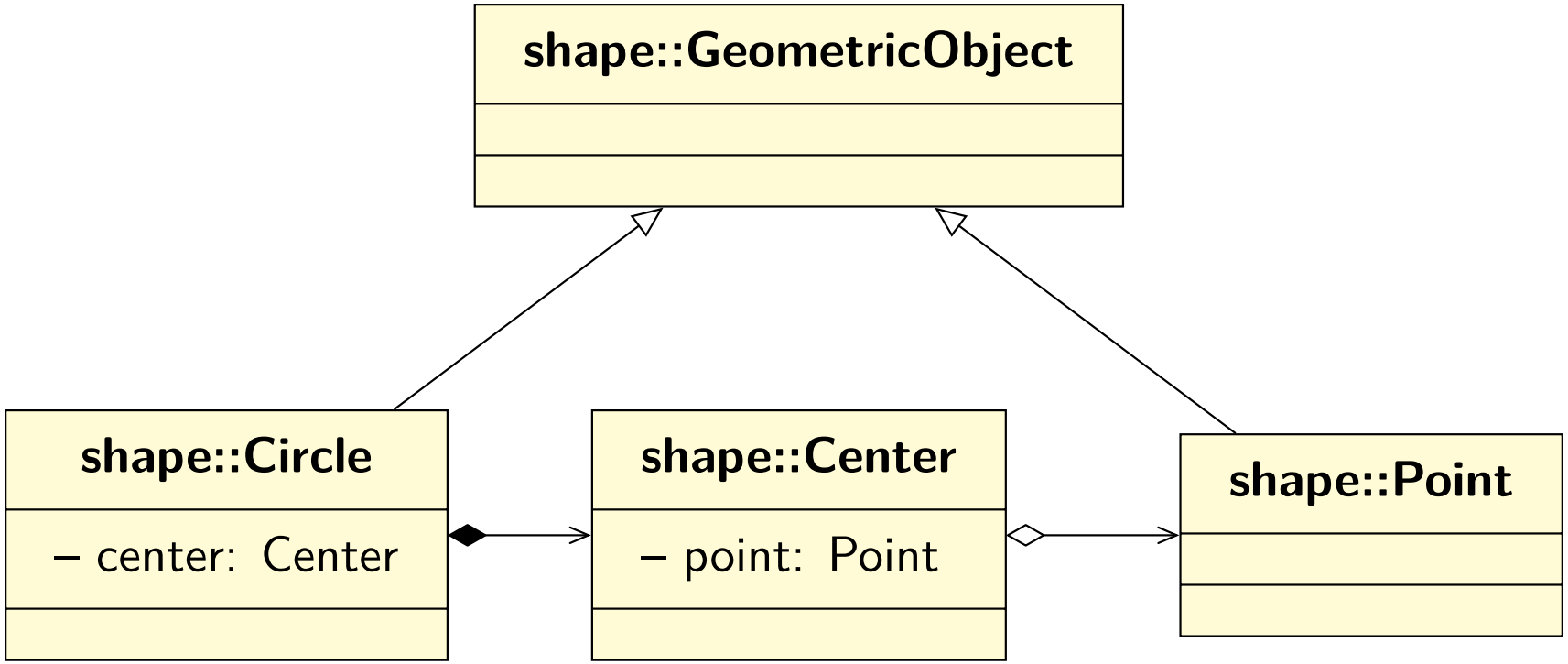
Representing Aggregation – Make Better Sense?



Representing Composition – Make Sense At All?



Putting These Together



- In this model, we encode that Center cannot exist without Circle.
- What else do we encode here?

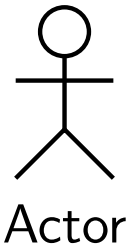
Use Case Diagrams

- Use Case diagram is behavioral
  - Recall that Class diagram is structural while Use Case and Sequence diagrams are behavioral
- Maps to user stories (i.e. requirements)
  - Describes the outside view of the system – from the point of view of a set of actors
  - Models system actions that yield an observable result
  - Simple, but effective for several purposes

Elements of Use Case Diagrams

Use case (with a name)

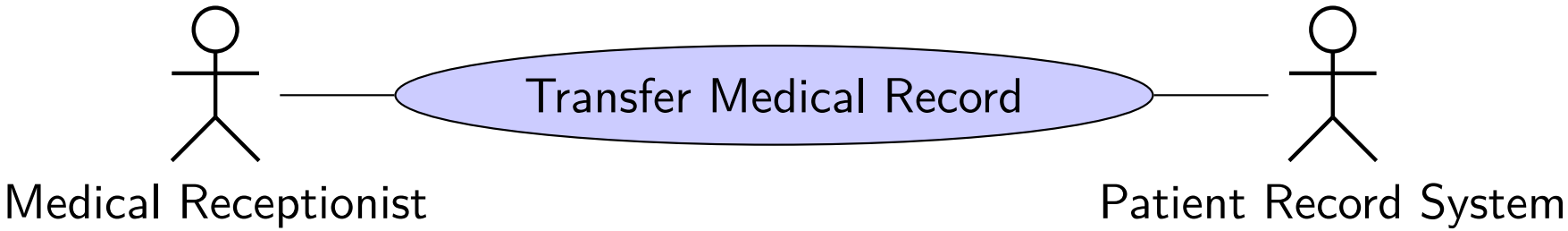
Actor (human or device that interacts with system)



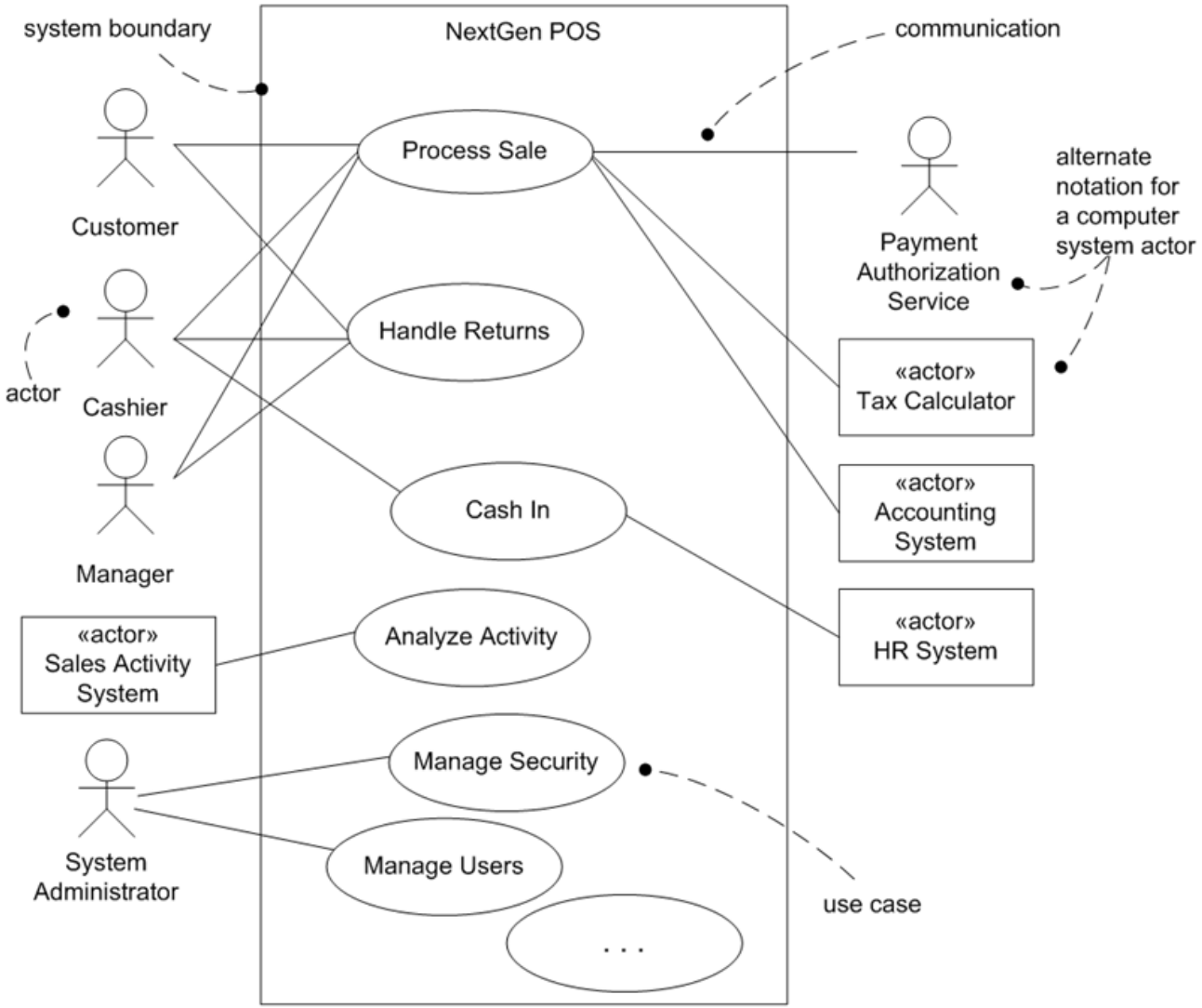
Rules for Use Case Diagrams

- Actors
  - e.g. Employee, Manager, AppUser
  - external to system (humans or devices)
  - interact with system
  - may appear in many use cases
- Use Cases
  - brief title of an interaction with the system

Use Case Diagram: A Simple Example



Use Case Diagram: More Complex Example



Outline

Why are use case diagrams important?

- Requirements elicitation and organization – e.g. to help show how different user stories are related
- Planning – e.g. to prioritize users or scenarios
- Testing – e.g. help in constructing acceptance tests with good coverage

Key ideas in software modeling with UML

- Names are important – give good names to classes in class diagrams, use cases, etc.
- Provide only essential details – avoid over-modeling
- Keep the model up to date – easy for the model to lose it's usefulness if it's outdated

Tool Support

Many specialized tools exists

- Some integrated in IDEs
  - Extensions for Eclipse or other IDEs (e.g. Sparx)
  - Visual Studio has native support for class and sequence diagrams
- Standalone (e.g. MS Visio)
- On the cloud (Draw.io etc.)

A lot depends of whether you want OCL or MDA/MDD capabilities

Summary and Questions

Object-Oriented Design using UML

- Overview of UML
- Diagrams
  - UML Class diagrams
  - UML Use Case diagrams

Outline

"Engineering Software as a Service" by Armando Fox and David Patterson (2nd Edition)
"Essentials of Software Engineering" by Frank Tsui, Orlando Karam, and Barbara Bernal(4th Edition)