Introduction to Software Engineering
(CS350)

Lecture 07

Jongmoon Baik

Requirement Modeling - I

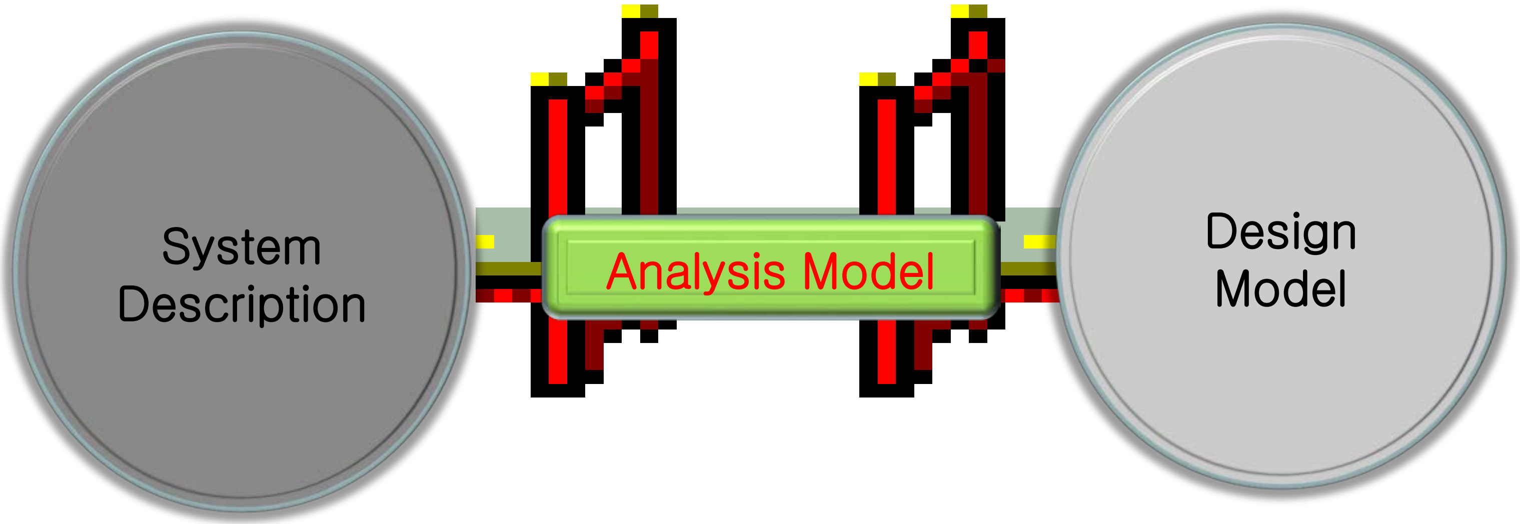
Scenarios, Information, and Analysis Classes

Requirements Analysis

- Requirements analysis
  - specifies software's operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet

- Requirements analysis allows the software engineer (called an *analyst* or *modeler* in this role) to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
  - build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.

A Bridge



Rules of Thumb

- The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high.
- Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system.
- Delay consideration of infrastructure and other non-functional models until design.
- Minimize coupling throughout the system.
- Be certain that the analysis model provides value to all stakeholders.
- Keep the model as simple as it can be.

Domain Analysis

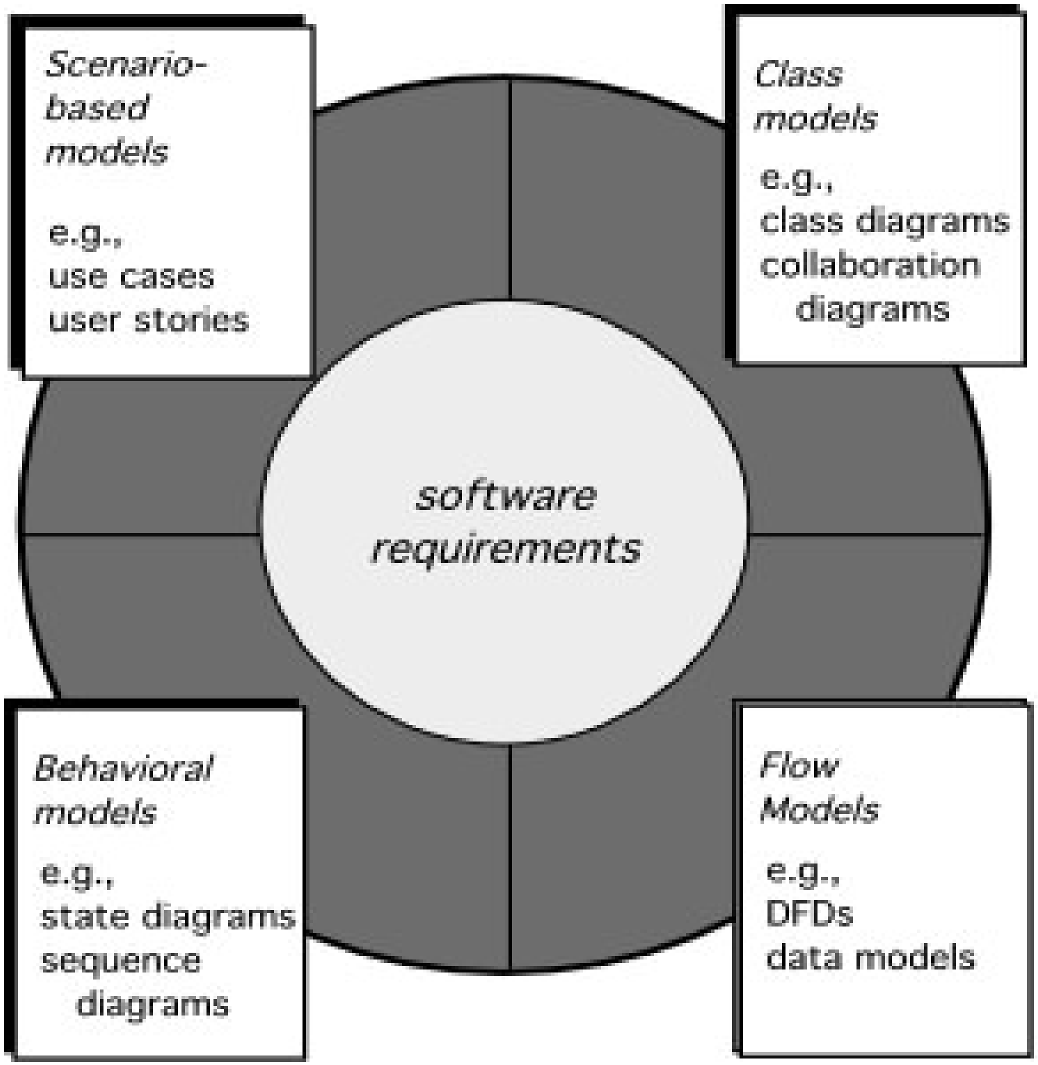
Software domain analysis is the identification, analysis, and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain . . . [Object-oriented domain analysis is] the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks . . .

Donald Firesmith

Domain Analysis

- Define the domain to be investigated.
- Collect a representative sample of applications in the domain.
- Analyze each application in the sample.
- Develop an analysis model for the objects.

Elements of Requirements Analysis



Scenario-Based Modeling

"[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)." Ivar Jacobson

(1) What should we write about?

(2) How much should we write about it?

(3) How detailed should we make our description?

(4) How should we organize the description?

What to Write About?

- Inception and elicitation—provide you with the information you'll need to begin writing use cases.
- Requirements gathering meetings, QFD, and other requirements engineering mechanisms are used to
  - identify stakeholders
  - define the scope of the problem
  - specify overall operational goals
  - establish priorities
  - outline all known functional requirements, and
  - describe the things (objects) that will be manipulated by the system.
- To begin developing a set of use cases, list the functions or activities performed by a specific actor.

How Much to Write About?

- As further conversations with the stakeholders progress, the requirements gathering team develops use cases for each of the functions noted.
- In general, use cases are written first in an informal narrative fashion.
- If more formality is required, the same use case is rewritten using a structured format similar to the one proposed.

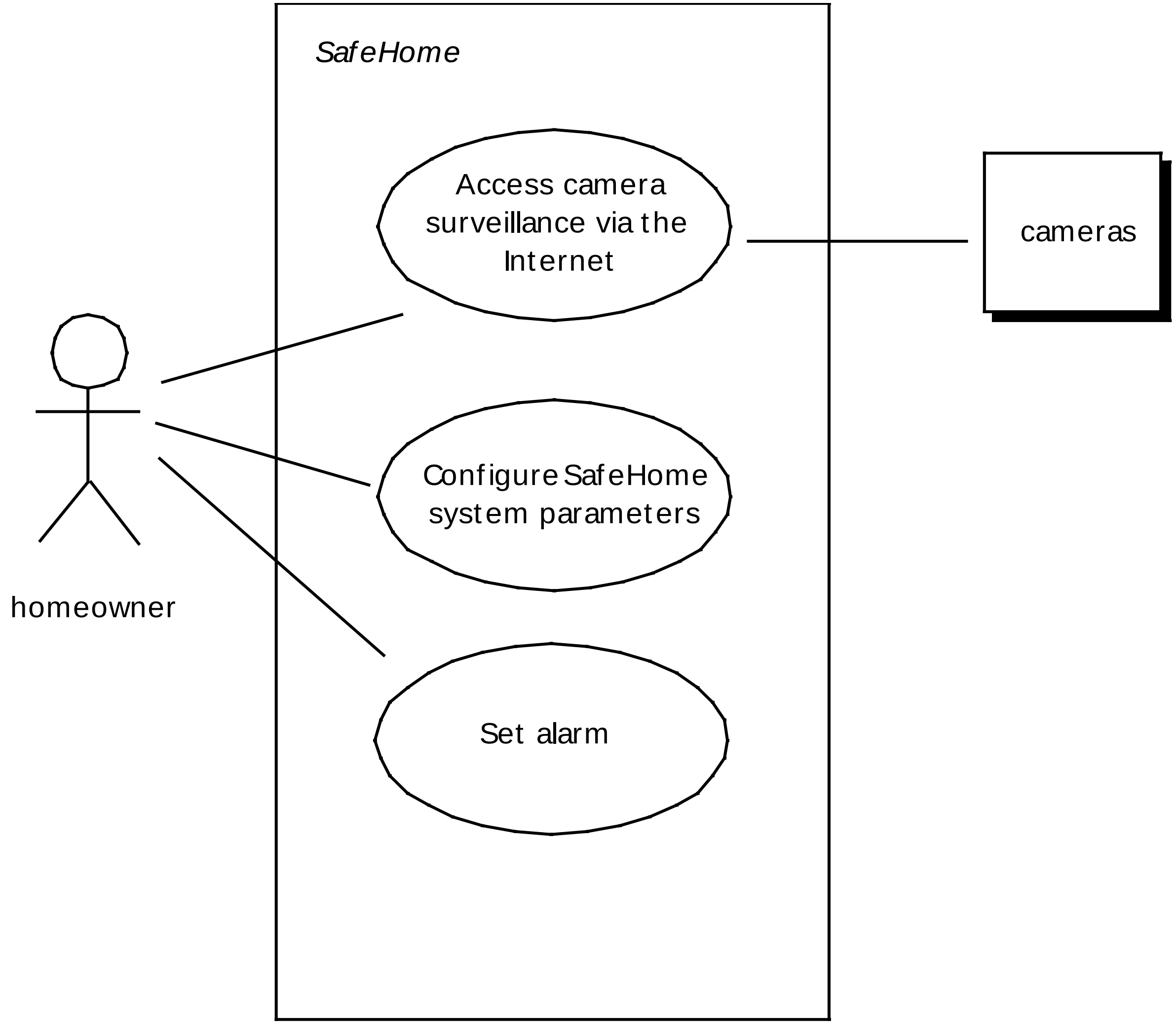
Use-Cases

- a scenario that describes a “thread of usage” for a system
- *actors* represent roles people or devices play as the system functions
- *users* can play a number of different roles for a given scenario

Developing a Use-Case

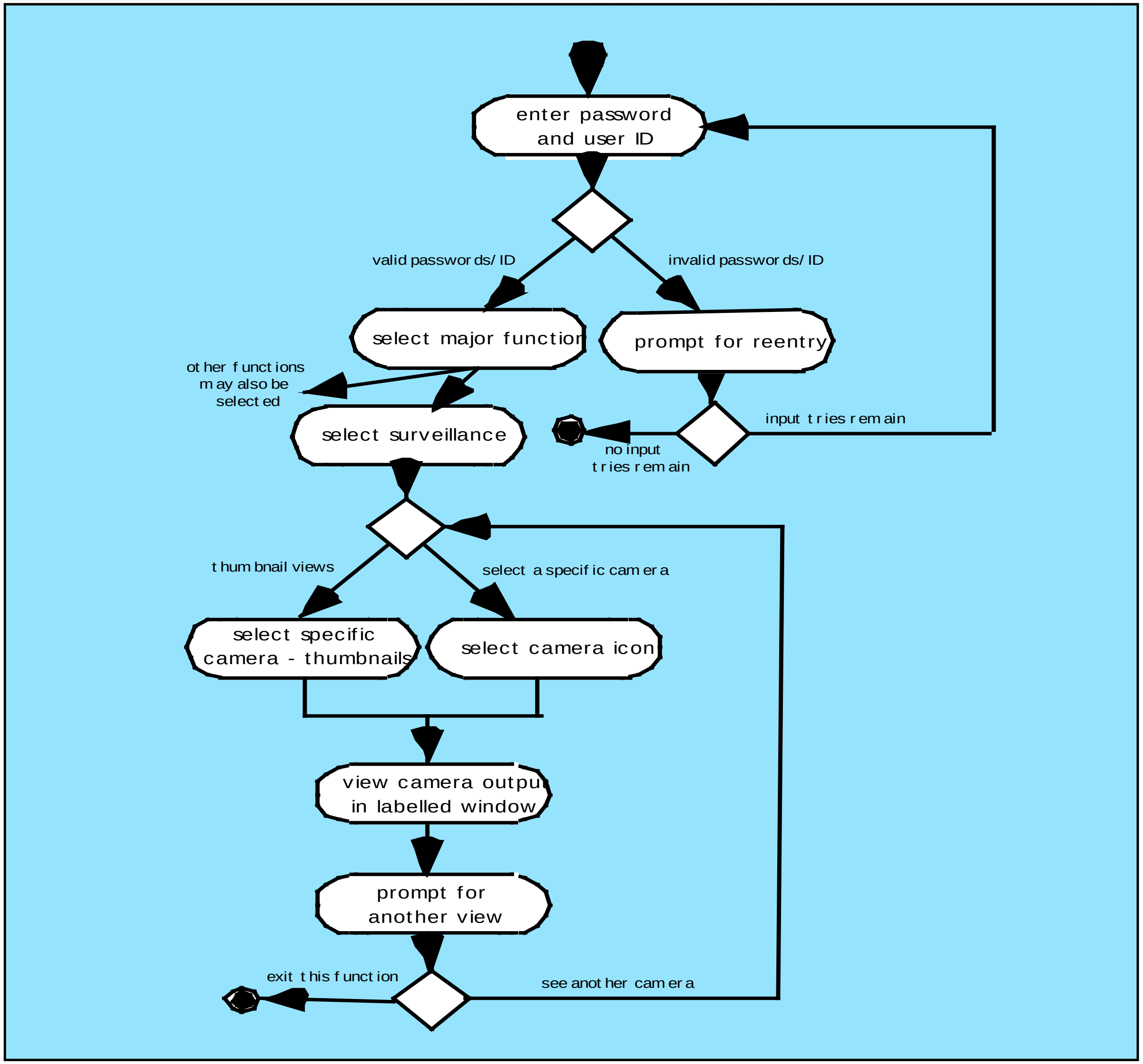
- What are the main tasks or functions that are performed by the actor?
- What system information will the actor acquire, produce or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

Use-Case Diagram



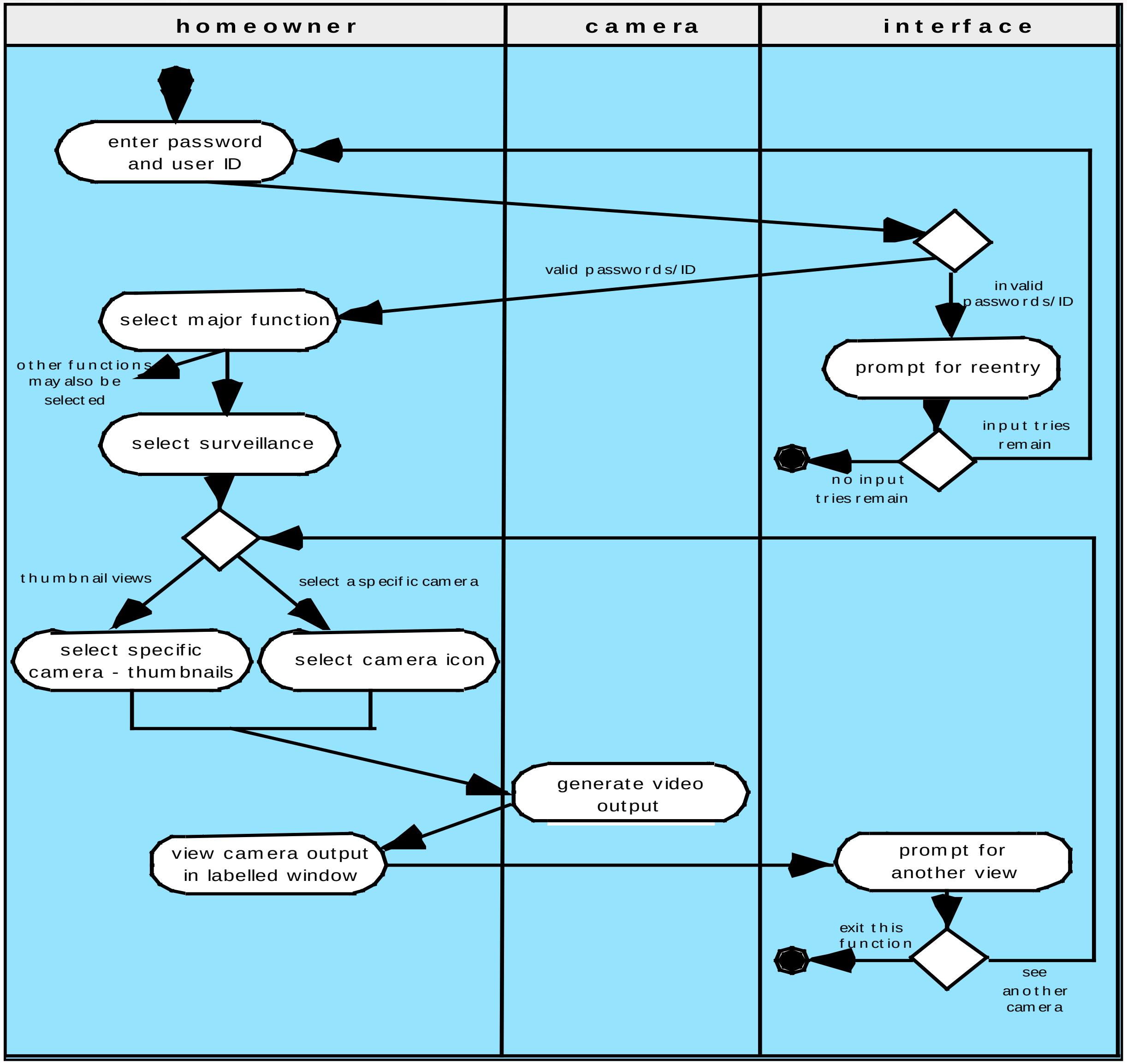
Activity Diagram

Supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario



Swimlane Diagrams

Allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are multiple actors involved in a specific use-case) or analysis class has responsibility for the action described by an activity rectangle



Data Modeling

- examines data objects independently of processing
- focuses attention on the data domain
- creates a model at the customer's level of abstraction
- indicates how data objects relate to one another

What is a Data Object?

- a representation of almost any composite information that must be understood by software.
  - *composite information*—something that has a number of different properties or attributes
- can be an external entity (e.g., anything that produces or consumes information), a thing (e.g., a report or a display), an occurrence (e.g., a telephone call) or event (e.g., an alarm), a role (e.g., salesperson), an organizational unit (e.g., accounting department), a place (e.g., a warehouse), or a structure (e.g., a file).
- The description of the data object incorporates the data object and all of its attributes.
- A data object encapsulates data only—there is no reference within a data object to operations that act on the data.

Data Objects and Attributes

A data object contains a set of attributes that act as an aspect, quality, characteristic, or descriptor of the object

object: automobile

---

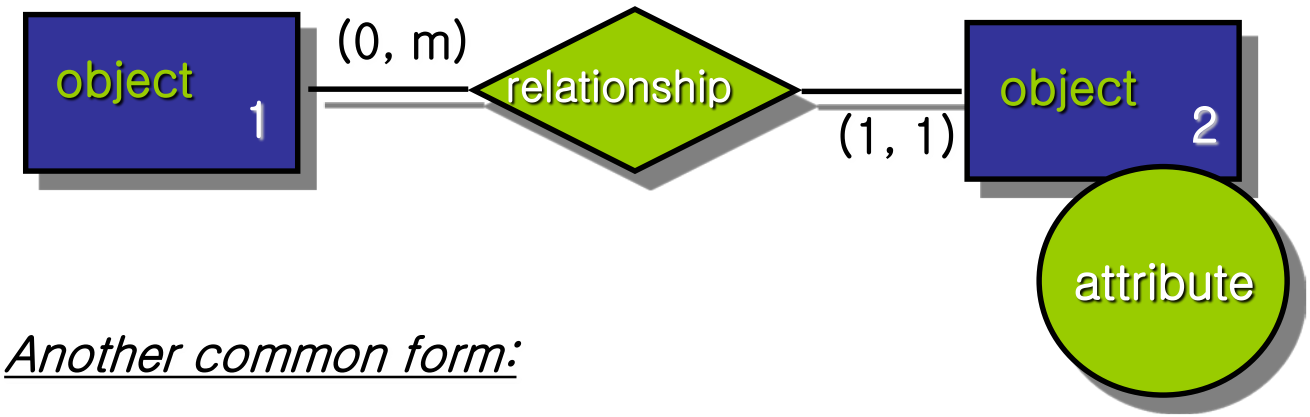
attributes:
- make
- model
- body type
- price
- options code

What is a Relationship?

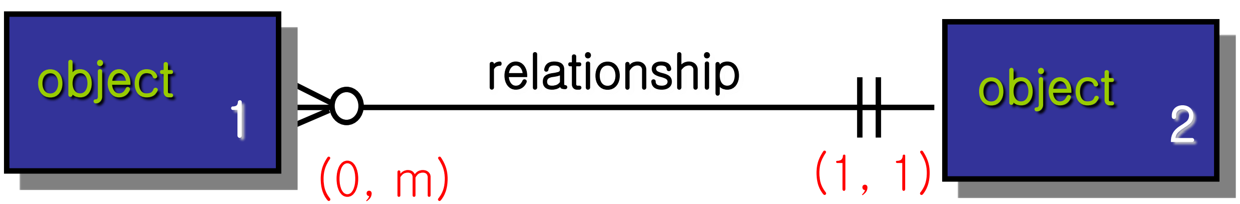
- Data objects are connected to one another in different ways.
  - A connection is established between **person** and **car** because the two objects are related.
    - A person *owns* a car
    - A person *is insured to drive* a car
- The relationships *owns* and *insured to drive* define the relevant connections between **person** and **car**.
- Several instances of a relationship can exist
- Objects can be related in many different ways

ERD Notation

One common form:



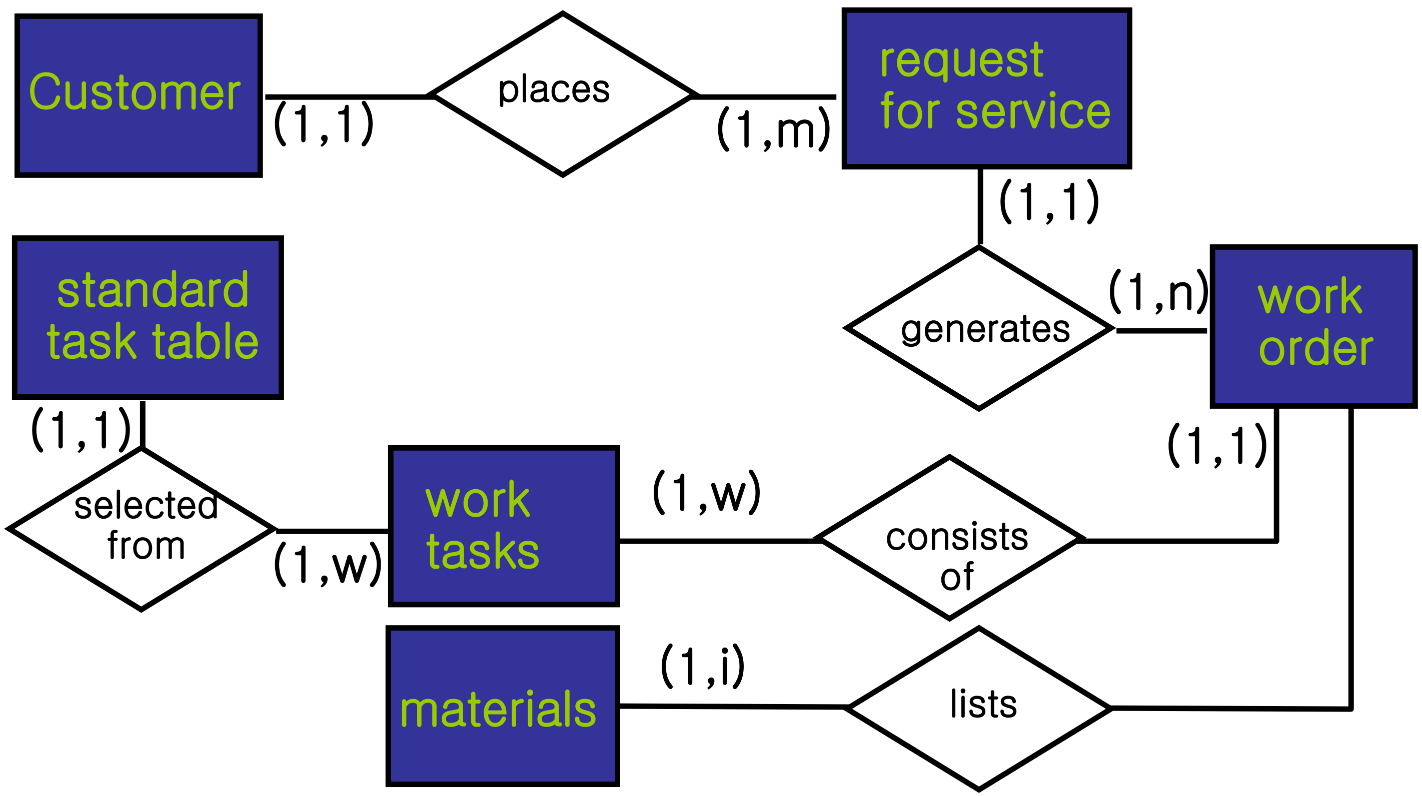
Another common form:



Building an ERD

- *Level 1*—model all data objects (entities) and their “connections” to one another
- *Level 2*—model all entities and relationships
- *Level 3*—model all entities, relationships, and the attributes that provide further depth

The ERD: An Example



Class-Based Modeling

- Class-based modeling represents:
  - objects that the system will manipulate
  - operations (also called methods or services) that will be applied to the objects to effect the manipulation
  - relationships (some hierarchical) between the objects
  - collaborations that occur between the classes that are defined.

- The elements of a class-based model include classes and objects, attributes, operations, CRC models, collaboration diagrams and packages.

Identifying Analysis Classes

- Examining the usage scenarios developed as part of the requirements model and perform a "grammatical parse" [Abb83]
  - Classes are determined by underlining each noun or noun phrase and entering it into a simple table.
  - Synonyms should be noted.
  - If the class (noun) is required to implement a solution, then it is part of the solution space; otherwise, if a class is necessary only to describe a solution, it is part of the problem space.

- But what should we look for once all of the nouns have been isolated?

Manifestations of Analysis Classes

- *Analysis classes* manifest themselves in one of the following ways:
  - *External entities* (e.g., other systems, devices, people) that produce or consume information
  - *Things* (e.g, reports, displays, letters, signals) that are part of the information domain for the problem
  - *Occurrences or events* (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation
  - *Roles* (e.g., manager, engineer, salesperson) played by people who interact with the system
  - *Organizational units* (e.g., division, group, team) that are relevant to an application
  - *Places* (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function
  - *Structures* (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects

Potential Classes

- *Retained information.* The potential class will be useful during analysis only if information about it must be remembered so that the system can function.
- *Needed services.* The potential class must have a set of identifiable operations that can change the value of its attributes in some way.
- *Multiple attributes.* During requirement analysis, the focus should be on "major" information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.
- *Common attributes.* A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.
- *Common operations.* A set of operations can be defined for the potential class and these operations apply to all instances of the class.
- *Essential requirements.* External entities that appear in the problem space and produce or consume information essential to the operation of any solution for the system will almost always be defined as classes in the requirements model.

Defining Attributes

- *Attributes* describe a class that has been selected for inclusion in the analysis model.
  - build two different classes for professional baseball players
    - **For Playing Statistics software:** name, position, batting average, fielding percentage, years played, and games played might be relevant
    - **For Pension Fund software:** average salary, credit toward full vesting, pension plan options chosen, mailing address, and the like.

Defining Operations

- Do a grammatical parse of a processing narrative and look at the verbs
- Operations can be divided into four broad categories:
  - (1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting)
  - (2) operations that perform a computation
  - (3) operations that inquire about the state of an object, and
  - (4) operations that monitor an object for the occurrence of a controlling event.

Class Diagram

SYSTEM
systemID verificationPhoneNumber systemStatus dalayTime telephoneNumber masterPassword temporaryPassword numberTries
program() display() reset() query() arm() disarm()

CRC Models

- *Class-Responsibility-Collaborator (CRC) modeling* [Wir90] provides a simple means for identifying and organizing the classes that are relevant to system or product requirements. Ambler [Amb95] describes CRC modeling in the following way:
  - A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

CRC Modeling

Class: FloorPlan

Description:

Responsibility:	**Collaborator:**
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

Class Types

- *Entity classes*, also called *model* or *business* classes, are extracted directly from the statement of the problem (e.g., FloorPlan and Sensor).
- *Boundary classes* are used to create the interface (e.g., interactive screen or printed reports) that the user sees and interacts with as the software is used.
- *Controller classes* manage a “unit of work” [UML03] from start to finish. That is, controller classes can be designed to manage
  - the creation or update of entity objects;
  - the instantiation of boundary objects as they obtain information from entity objects;
  - complex communication between sets of objects;
  - validation of data communicated between objects or between the user and the application.

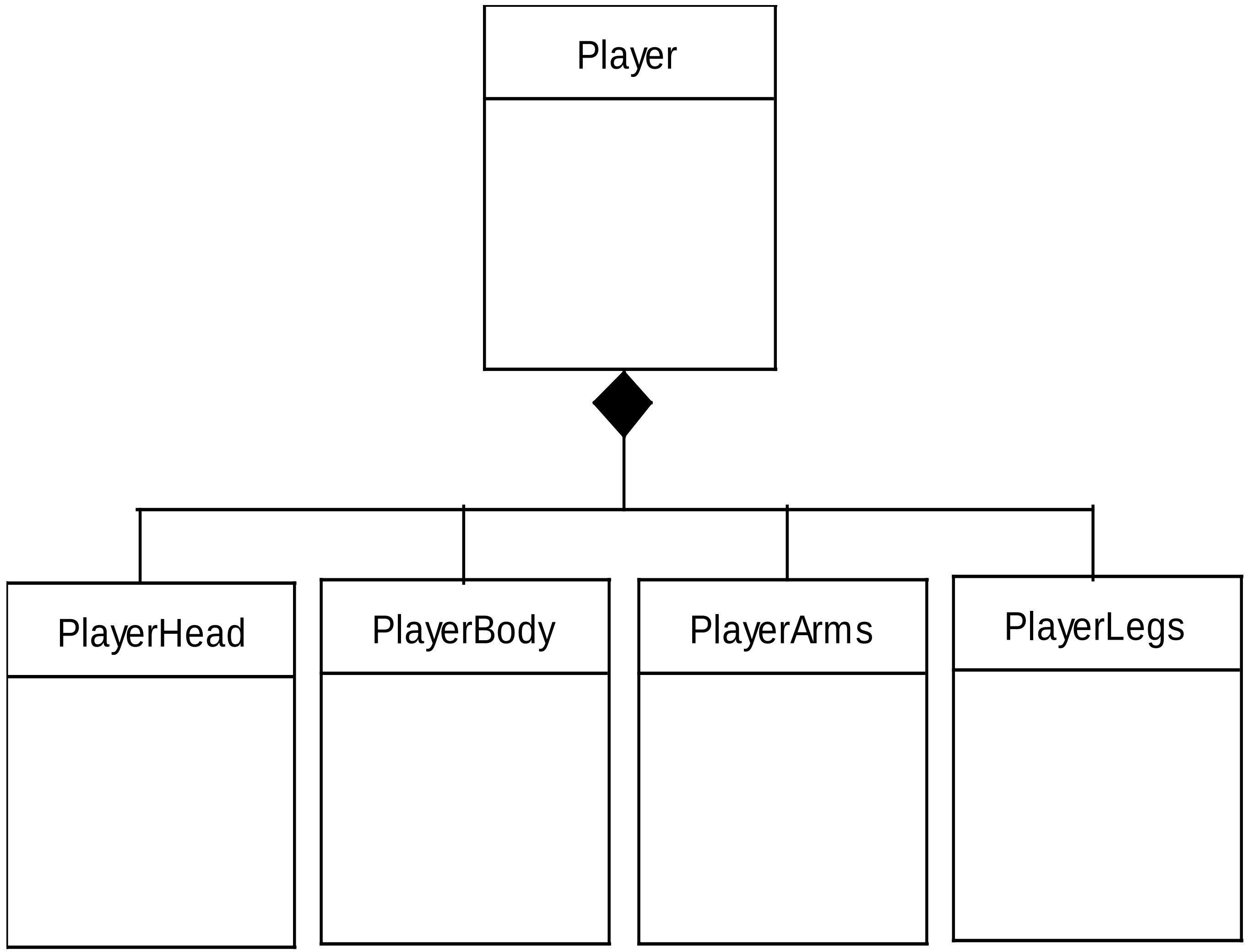
Guidelines for Allocating Responsibilities

- System intelligence should be distributed across classes to best address the needs of the problem
- Each responsibility should be stated as generally as possible
- Information and the behavior related to it should reside within the same class
- Information about one thing should be localized with a single class, not distributed across multiple classes.
- Responsibilities should be shared among related classes, when appropriate.

Collaborations

- Classes fulfill their responsibilities in one of two ways:
  - A class can use its own operations to manipulate its own attributes, thereby fulfilling a particular responsibility, or
  - a class can collaborate with other classes.
- Collaborations identify relationships between classes
- Collaborations are identified by determining whether a class can fulfill each responsibility itself
- three different generic relationships between classes [WIR90]:
  - the *is-part-of* relationship
  - the *has-knowledge-of* relationship
  - the *depends-upon* relationship

Composite Aggregate Class

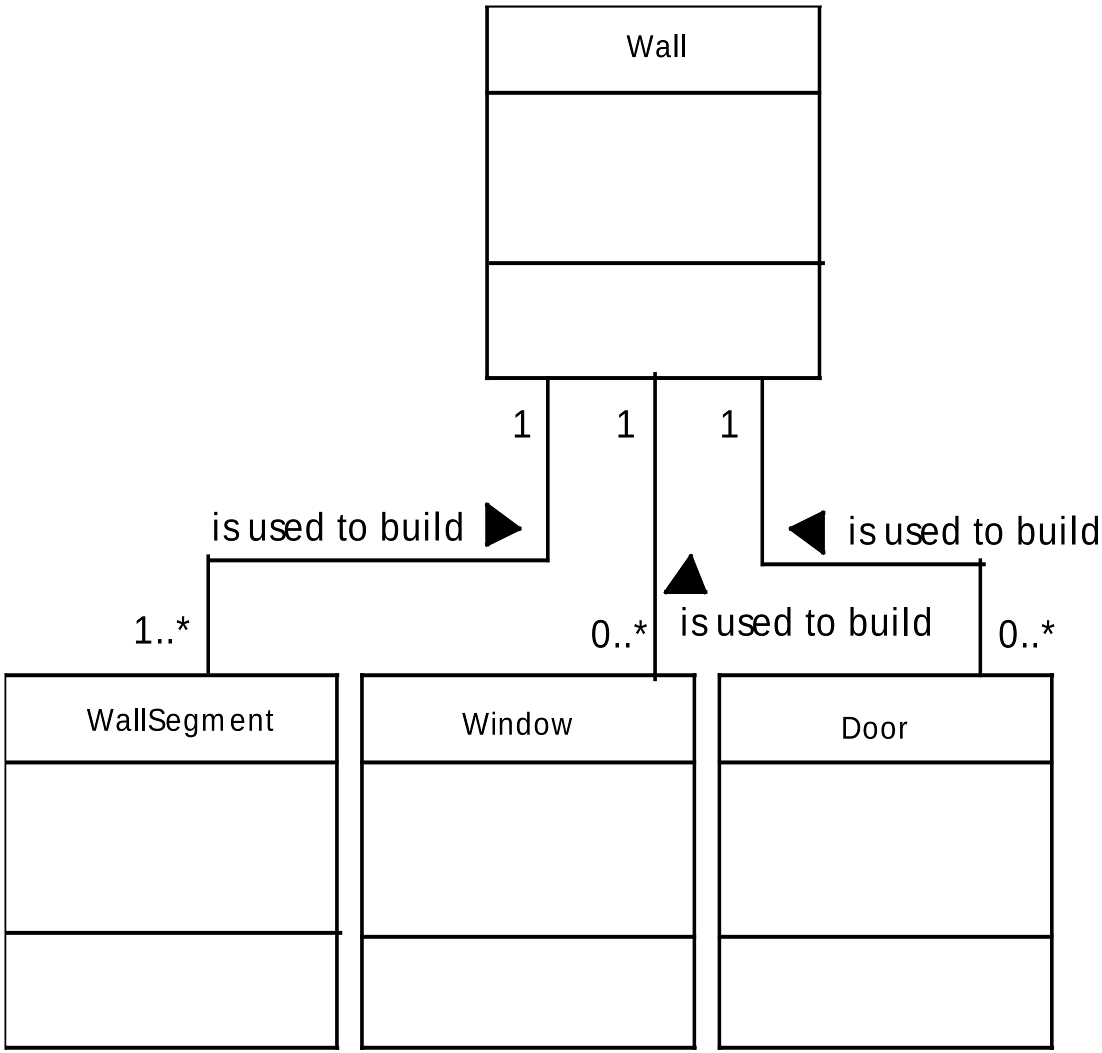


Associations and Dependencies

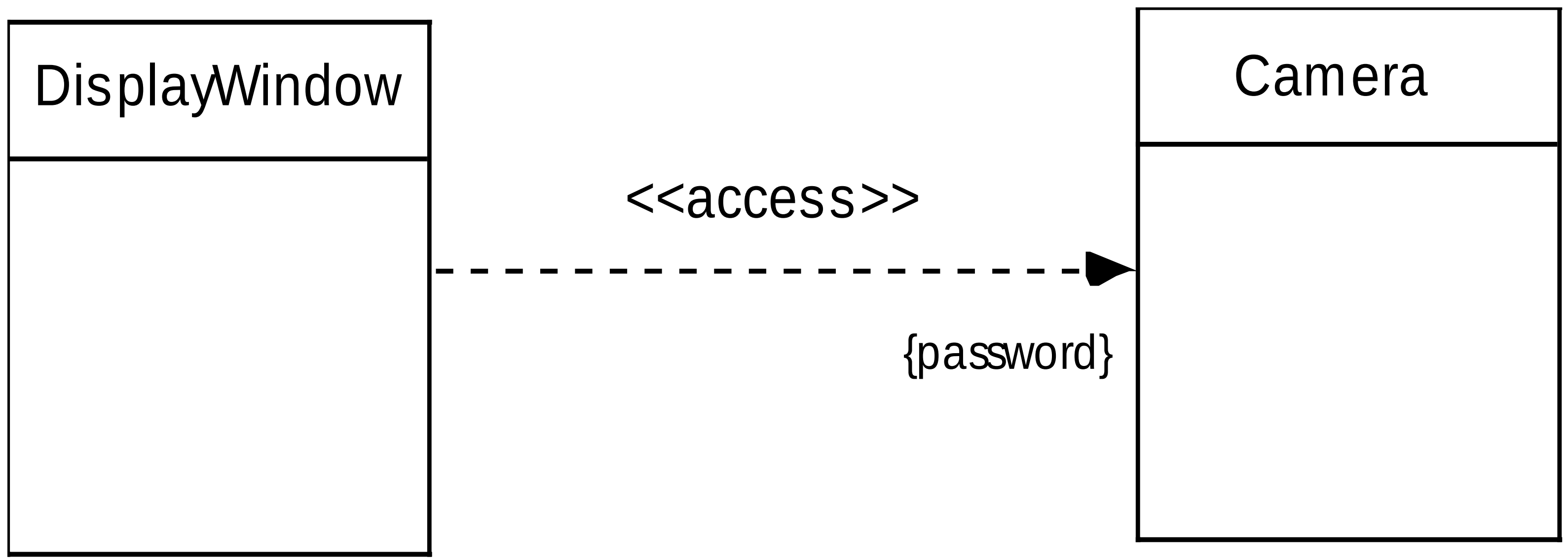
- Two analysis classes are often related to one another in some fashion
  - In UML these relationships are called ***associations***
  - Associations can be refined by indicating *multiplicity* (the term *cardinality* is used in data modeling

- In many instances, a client-server relationship exists between two analysis classes.
  - In such cases, a client-class depends on the server-class in some way and a ***dependency relationship*** is established

Multiplicity



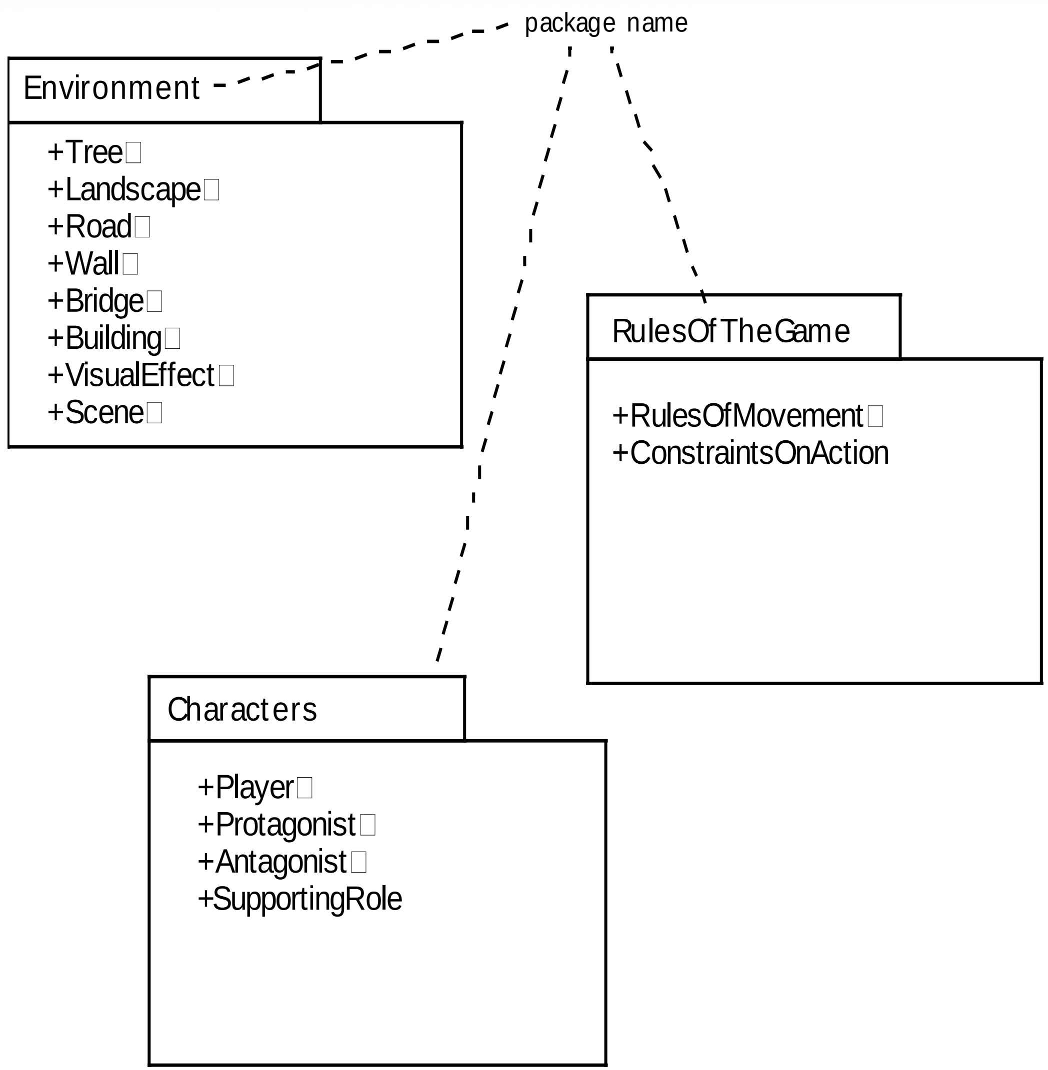
Dependencies



Analysis Packages

- Various elements of the analysis model (e.g., use-cases, analysis classes) are categorized in a manner that packages them as a grouping
- The plus sign preceding the analysis class name in each package indicates that the classes have public visibility and are therefore accessible from other packages.
- Other symbols can precede an element within a package. A minus sign indicates that an element is hidden from all other packages and a # symbol indicates that an element is accessible only to packages contained within a given package.

Analysis Packages



Q & A