Basic Structural Modeling

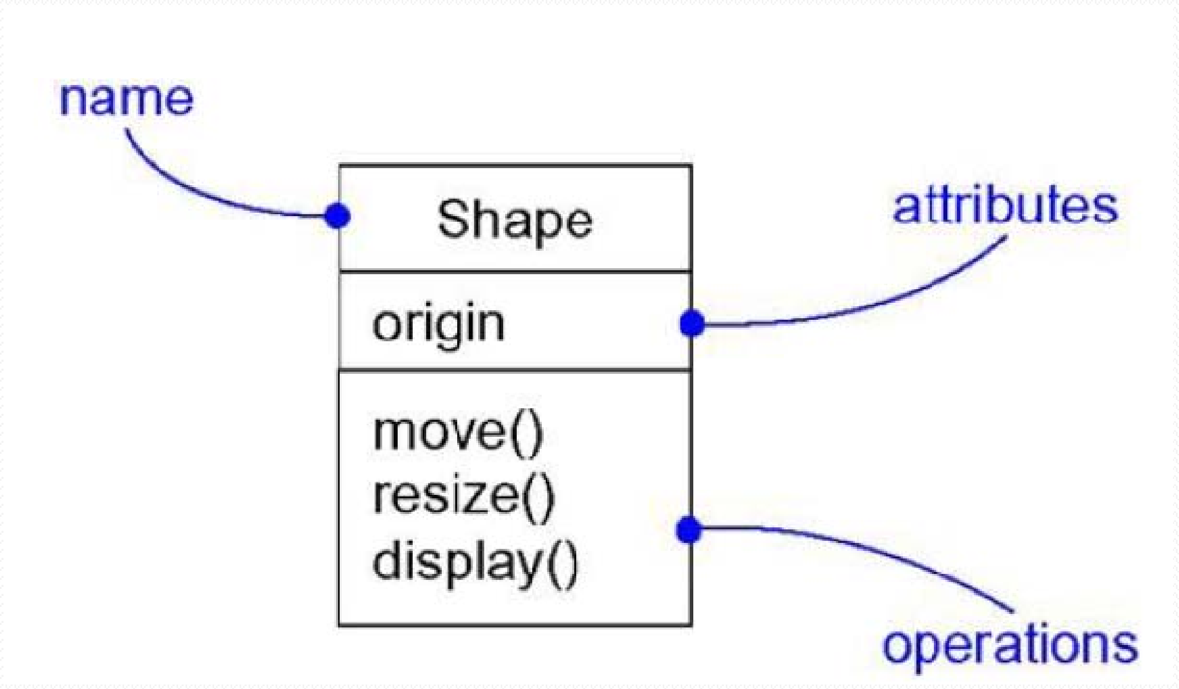
Introduction to Structural Modeling, Classes, Relationships, common Mechanisms, and diagrams.

Contents

Classes

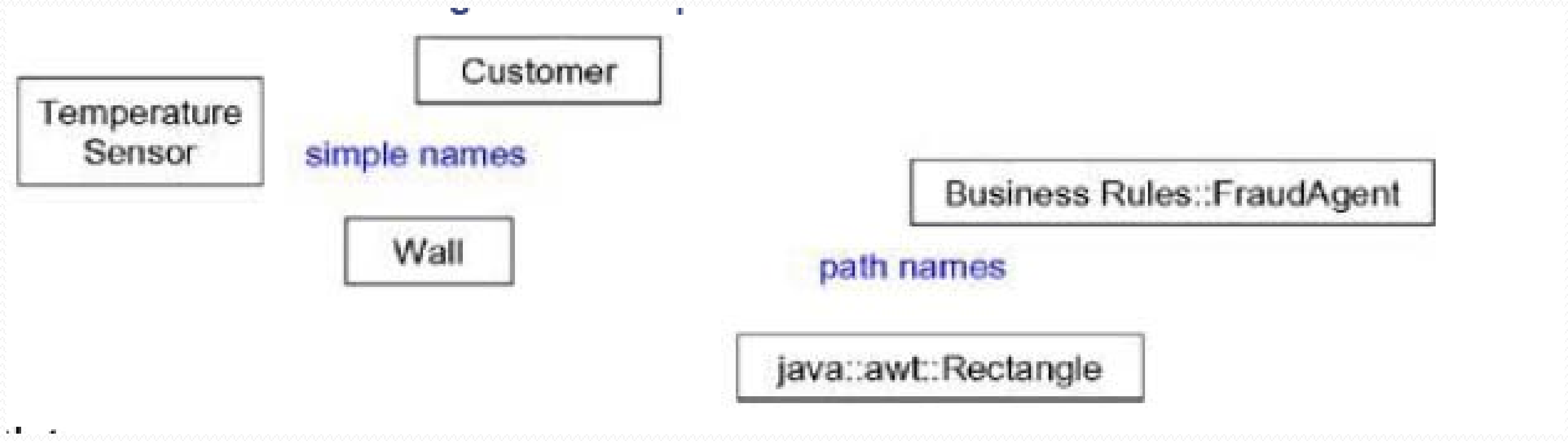
- Classes are the most important building block of any object-oriented system.
- A class is a description of set of objects that share the same attributes, operations, relationships and semantics.
- A class implements one or more interfaces.
- Graphically a class is represented as a rectangle.

Representation of a class

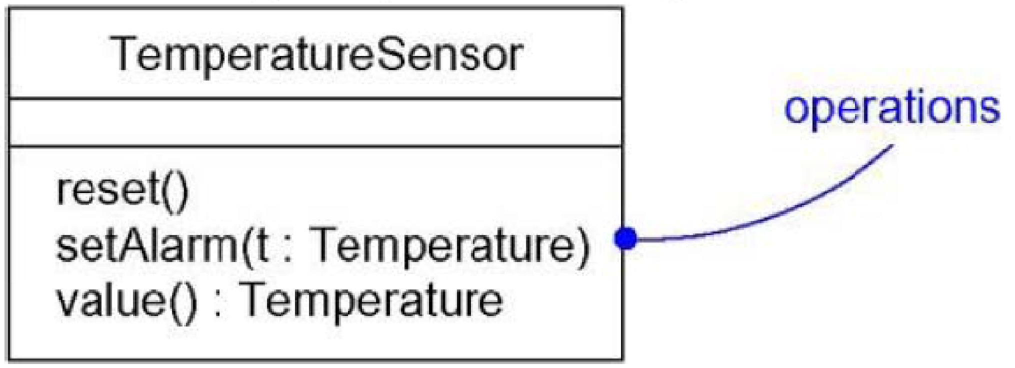


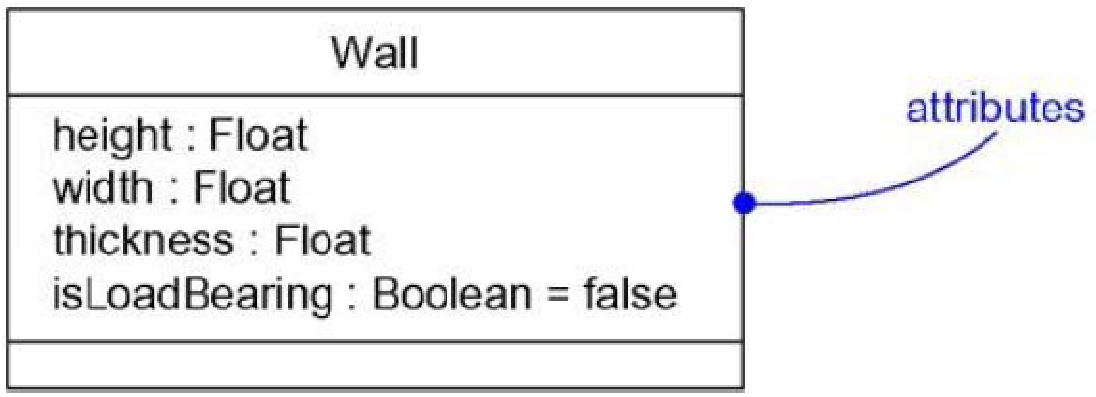
Simple name Vs Path name

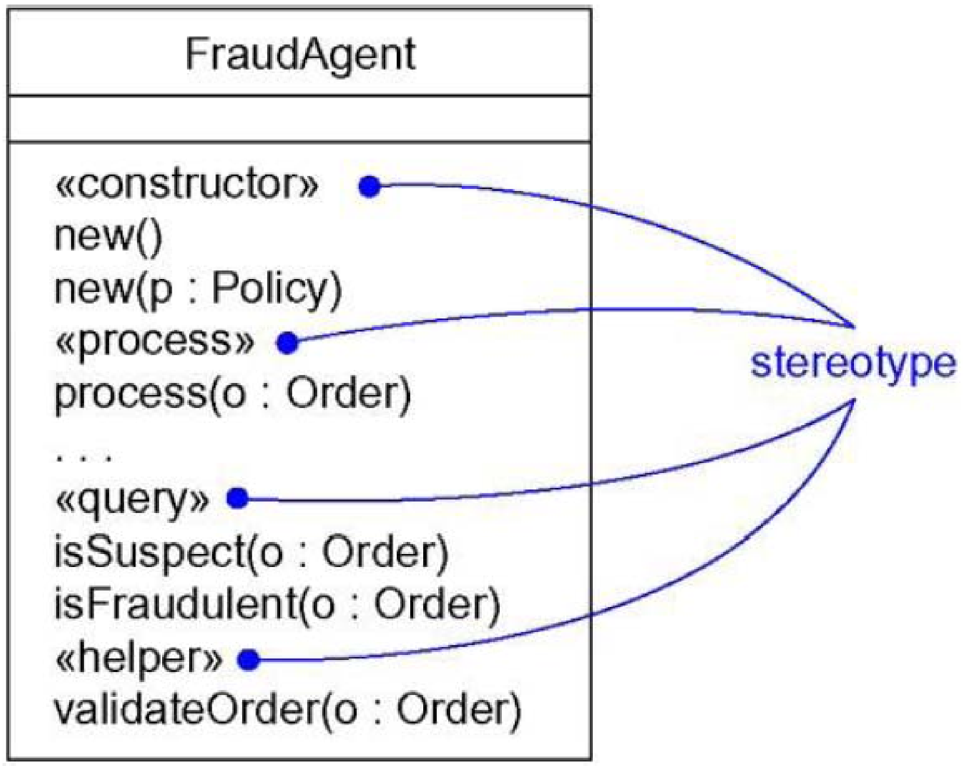
- A *name* alone is known as a *simple name;* a *path name* is the class name prefixed by the name of the package in which that class lives.

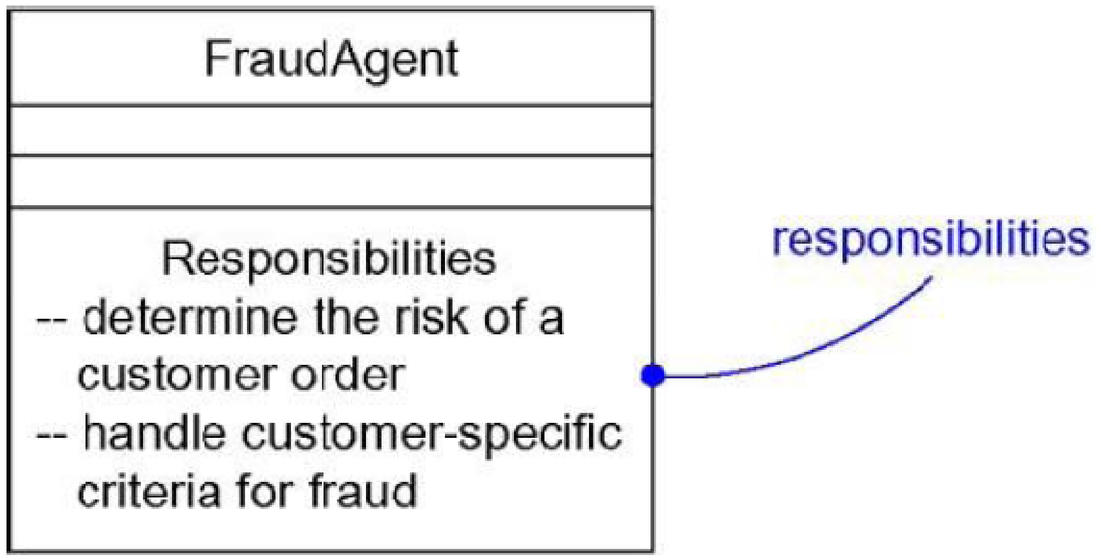


- An attribute is a named property of a class that describes a range of values.
- An operation is the implementation of a service that can be requested from any object of the class to affect behavior.
- A responsibility is a contract or an obligation of a class.


TemperatureSensor
reset()
setAlarm(t : Temperature)
value() : Temperature
operations


Wall
height : Float
width : Float
thickness : Float
isLoadBearing : Boolean = false
attributes


FraudAgent
«constructor»
new()
new(p : Policy)
«process»
process(o : Order)
. . .
«query»
isSuspect(o : Order)
isFraudulent(o : Order)
«helper»
validateOrder(o : Order)
stereotype


FraudAgent
Responsibilities
-- determine the risk of a customer order
-- handle customer-specific criteria for fraud
responsibilities

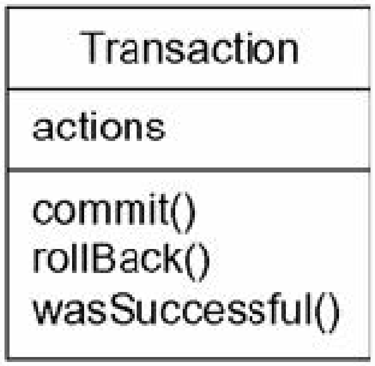
Common Modeling Techniques of a Class

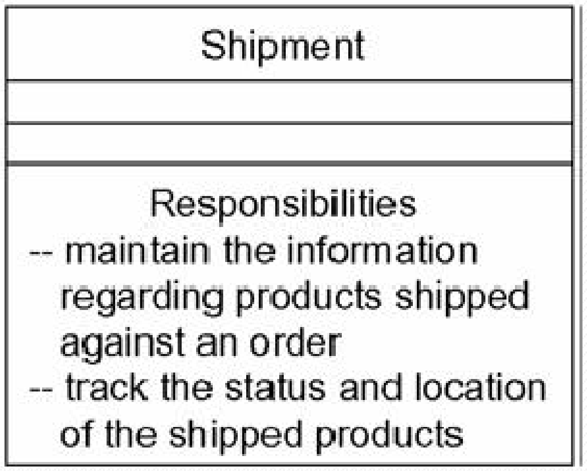
- Modeling the Vocabulary of a System
- Modeling the Distribution of Responsibilities in a System
- Modeling Non-software Things
- Modeling Primitive Types

To model the vocabulary of a system

- Identify those things that users or implementers use to describe the problem or solution.
- Use CRC cards and use case-based analysis to help to find these abstractions.
- For each abstraction, identify a set of responsibilities.
- Provide the attributes and operations that are needed to carry out these responsibilities for each class.

- **To model the vocabulary of a system**




Shipment
Responsibilities
-- maintain the information regarding products shipped against an order
-- track the status and location of the shipped products

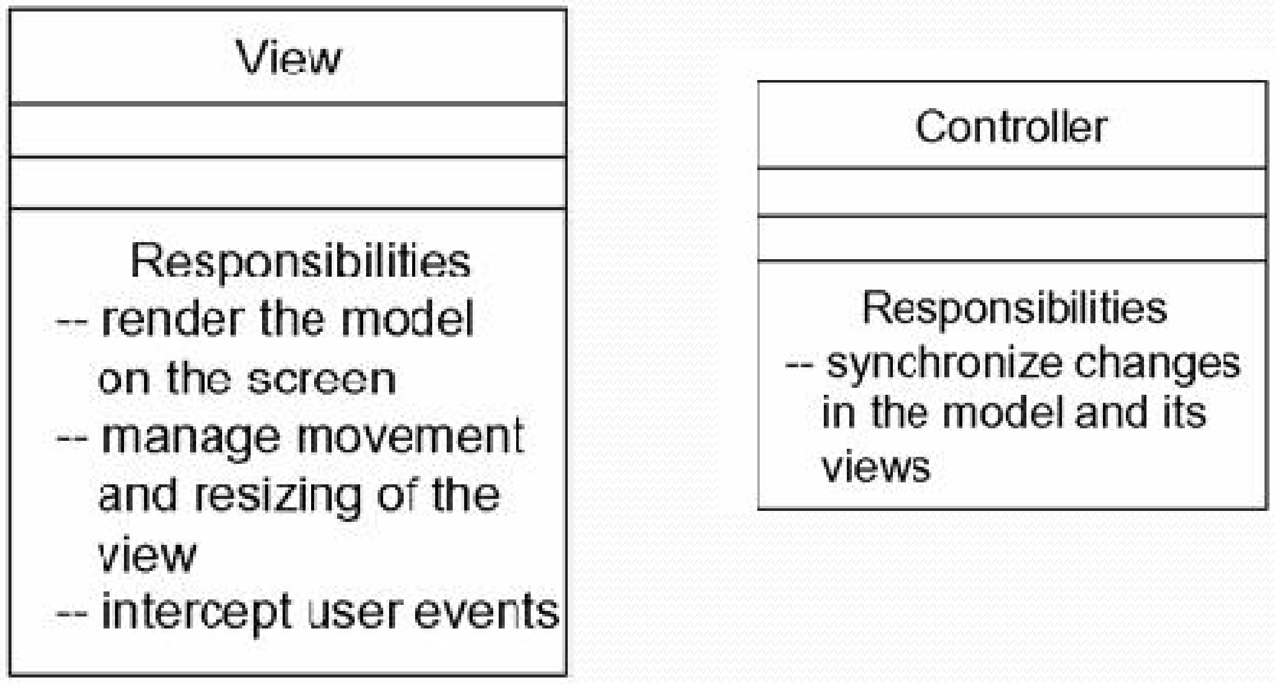
To model the distribution of responsibilities in a system

- Identify a set of classes that work together closely to carry out some behavior.

- Identify a set of responsibilities for each of these classes.

- Consider the ways in which those classes collaborate with one another, and redistribute their responsibilities accordingly so that no class within a collaboration does too much or too little.

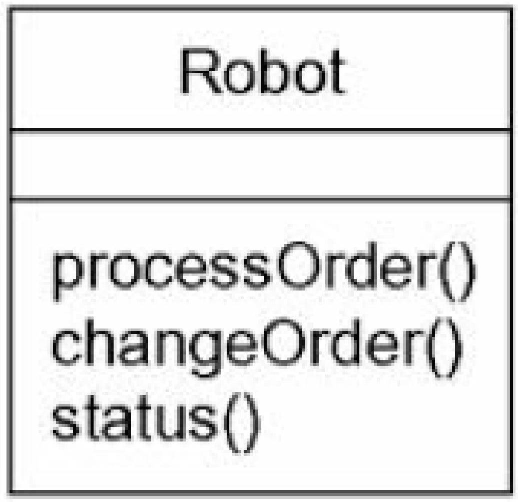
- To model the distribution of responsibilities in a system


View
Responsibilities
-- render the model on the screen
-- manage movement and resizing of the view
-- intercept user events
Controller
Responsibilities
-- synchronize changes in the model and its views

To Model Non-software Things

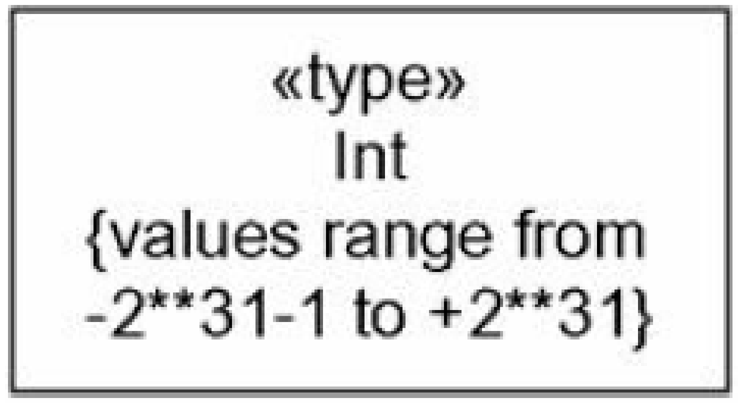
- Model the thing you are abstracting as a class.
- If you want to distinguish these things from the UML's defined building blocks, create a new building block by using stereotypes to specify these new semantics and to give a distinctive visual cue.
- If the thing you are modeling is some kind of hardware that itself contains software, consider modeling it as a kind of node, as well, so that you can further expand on its structure.

- **To Model Non-software Things**



To Model Primitive Types

- Model the thing you are abstracting as a type or an enumeration, which is rendered using class notation with the appropriate stereotype.
- If you need to specify the range of values associated with this type, use constraints.



Relationships

- A relationship is a connection among things.
- Graphically a relationship is rendered as a path, with different kinds of lines.
- Dependency, Association, Generalization and realization are the different types of relationships in UML.

Dependency

- A dependency indicates a semantic relation between two or more classes in which a change in one may force changes in the other although there is no explicit association between them.
- A stereotype may be used to denote the type of the dependency.
- It is a “using” relationship.

Association

- A semantic relationship between two or more classes that specifies connections among their instances.
- A structural relationship, specifying that objects of one class are connected to objects of a second (possibly the same) class.

Generalization

- Indicates that objects of the specialized class (subclass) are substitutable for objects of the generalized class (super-class).
  - “is kind of” relationship.

Relationships: Common Modeling Techniques

- **Modeling Simple Dependencies**
- **Modeling Single Inheritance**
- **Modeling Structural Relationships**

Reference

The Unified Modeling Language User Guide - *Grady Booch, James Rumbaugh, Ivar Jacobson* Addison-Wesley (International Student Edition)