UML Class Diagrams

PSC292 Engineering
Design Project

Organised by
Dr. Ashima Singh
Asstt. Professor, CSED

Complexity and Modeling

- Modeling is a well-accepted engineering technique for dealing with complexity
- Models let us focus on one view of the system while ignoring irrelevant details (a simplification of reality)
  - Example: Plumbers only care about plumbing, so we give them plumbing diagrams; electricians only care about electrical work, so we give them electrical diagrams, etc.
- Models can be created at different levels of abstraction
  - 10,000 ft. view vs. 10 ft. view
  - High-level: sketch of the building's exterior for the customer
  - Low-level: detailed electrical plan for the electrician

A model is an abstraction of a system, specifying the modeled system from a certain viewpoint and at a certain level of abstraction

Software Modeling

- Why do we create models of software systems?
- Models help us visualize and understand the system we're building
- Models are used to document the design of the system and communicate it to others
- Models serve as a guide during system construction

The Unified Modeling Language

- UML is a standard graphical notation for drawing models
- UML defines a variety of diagrams for
  - Structural modeling
  - Behavioral modeling
  - Physical modeling
- UML is an important skill
  - Frequently used in articles, papers, books, and specifications
  - Commonly used on OO development projects

Different ways of using UML

- Conceptual Modeling (a.k.a., Domain Modeling)
- Software Modeling

- Sketching
  - whiteboard discussion, document figures
- Blueprints
  - designers create UML models to guide construction
  - more complete than sketches
- Programming Language
  - execute UML models directly

- Forward Engineering
- Reverse Engineering

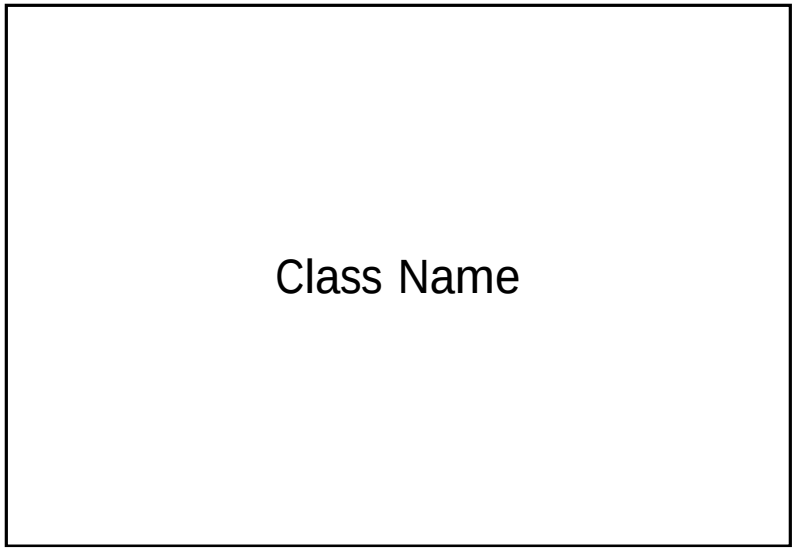
Overview of UML Diagrams

- Structural Diagrams
  - Class - static class structure
  - Object - runtime object structure
- Behavioral Diagrams
  - Use Case - interaction between system and its environment
  - Communication - dynamic message passing between objects
  - Sequence - dynamic message passing between objects
  - State - object states and transitions between them
  - Activity - algorithmic processes
- Physical Diagrams
  - Component - physical packaging of system as source and deployment files
  - Deployment - physical deployment of system on nodes

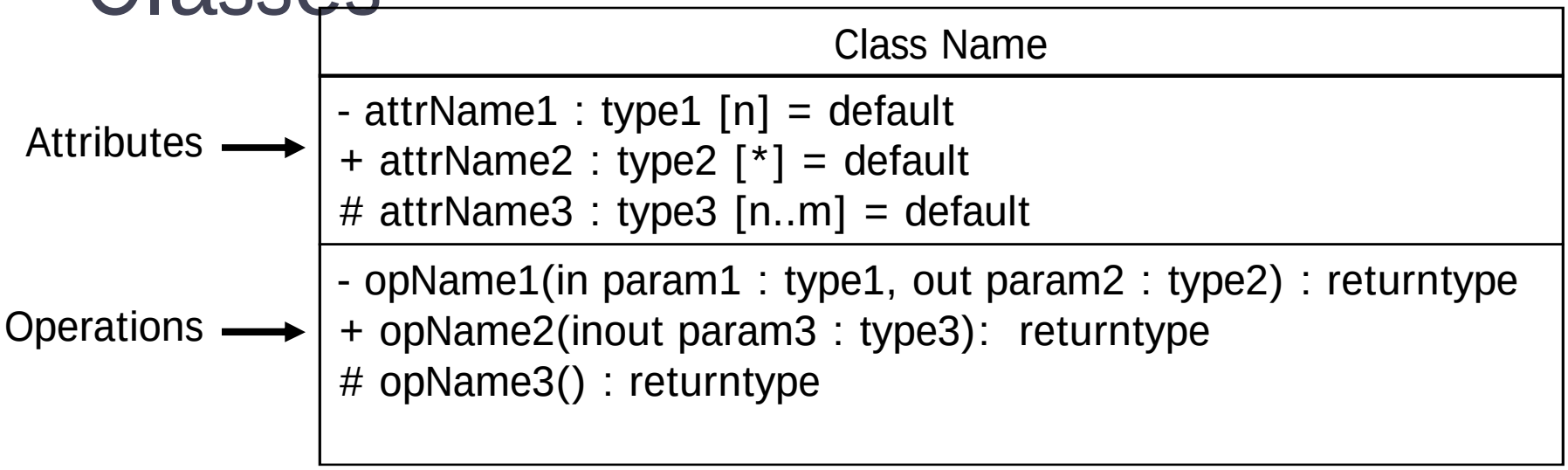
Class Diagrams

- Class Diagrams show the kinds of objects in a system and the static relationships between them
- Classes
  - Attributes
  - Operations
- Relationships
  - Association
    - Aggregation
    - Composition
  - Generalization
  - Dependency

Classes



Classes



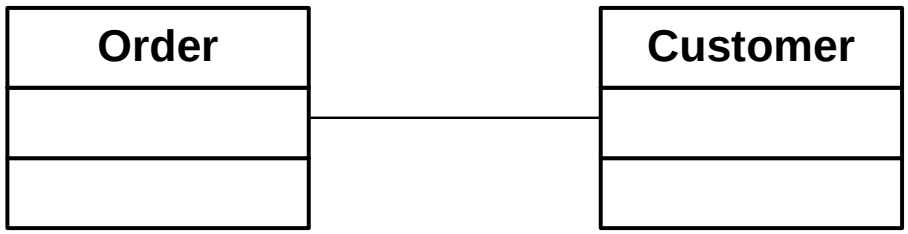
- private
+ public
# protected

Parameter types: in, out, inout

Relationships: Association

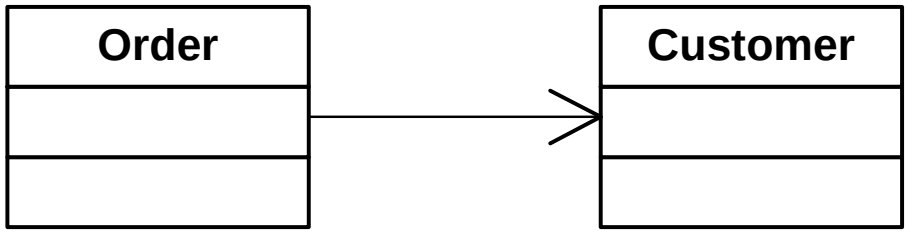
- Association is a structural relationship that specifies that objects of one class are connected to objects of another class.

- If there is an association between class A and class B, you can navigate from an object of type A to an object of type B, and vice versa
  - From an Order, you can get to its Customer
  - From a Customer, you can get to its Orders



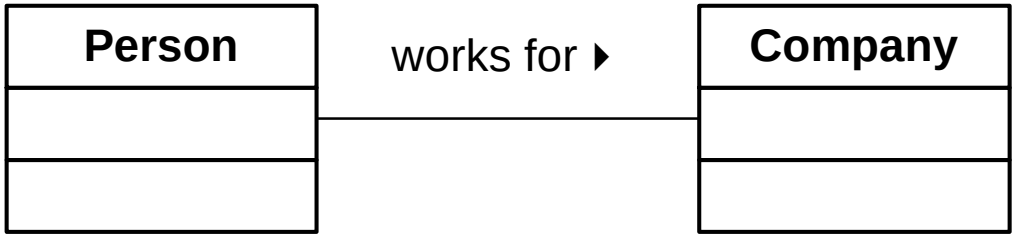
Relationships: Association

- An arrowhead may be used to limit navigation to only one direction



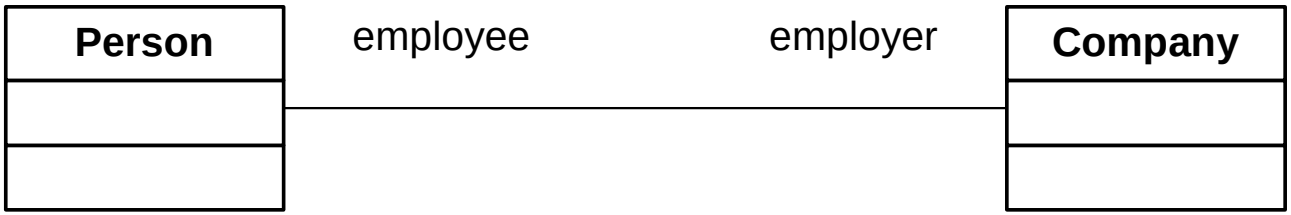
Relationships: Association

- An association may be given a name and a direction arrow that indicates how to read the name



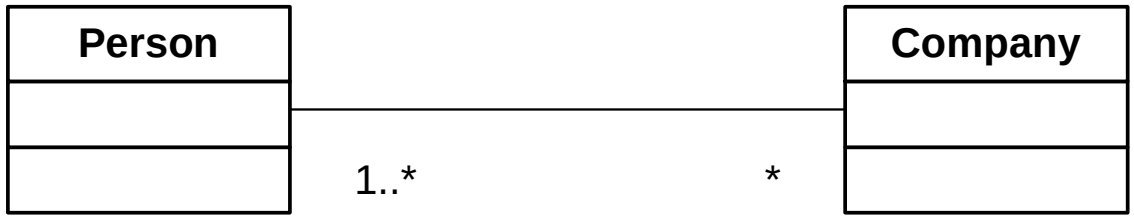
Relationships: Association

- Each end of an association may be given a "role name", indicating the role played by that class in the association (this is different than naming the association itself)



Relationships: Association

- Each end of an association may be given a "multiplicity"
- The multiplicity at one end indicates how many objects of that type are connected to each individual object at the other end
  - A Company has one or more employees
  - A Person works for zero or more companies

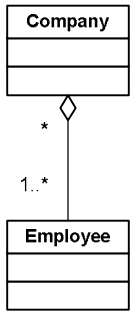


Relationships: Association

- Different ways to specify a multiplicity
  - A constant number (example: 3)
  - An asterisk * means any number (zero or more)
  - N..M specifies a range (example: 1..*)

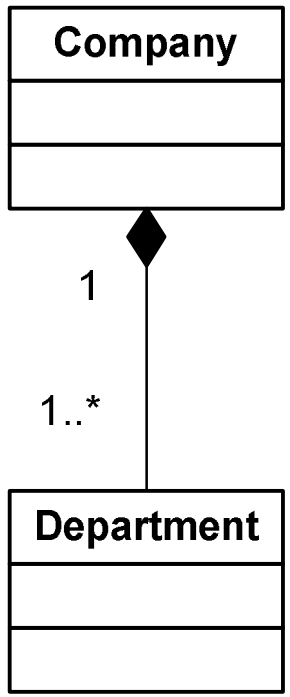
Relationships: Aggregation

- Aggregation is a special kind of association that represents a "whole/part" relationship
- The end with the diamond is the whole
- The end without the diamond is the part



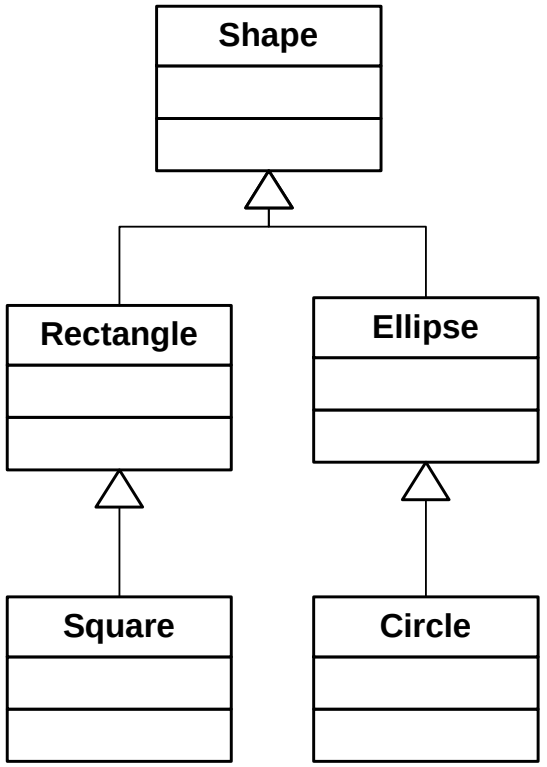
Relationships: Composition

- Like Aggregation, Composition represents a "whole/part" relationship
- Composition is used when the part does not exist independently from the whole
- For example, Department does not exist independently of its Company, but an Employee does
- The multiplicity on the “whole” end of a composition should always be 1, because a part can belong to only 1 whole



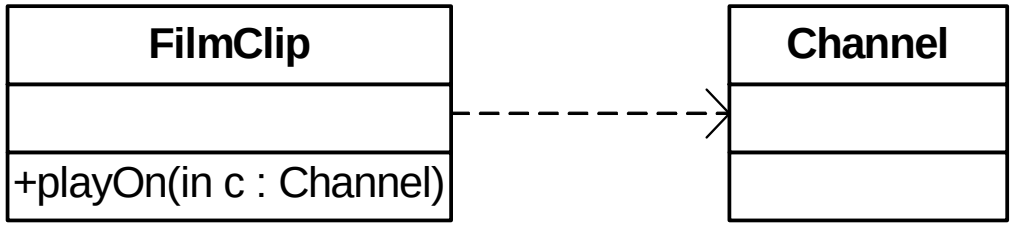
Relationships: Generalization

- Used to represent an "is-a" relationship between two classes
  - a Dog "is-a" Animal
  - a Truck "is-a" Vehicle
- Used to show inheritance relationships in software models
- Subclass inherits attributes and operations from superclass

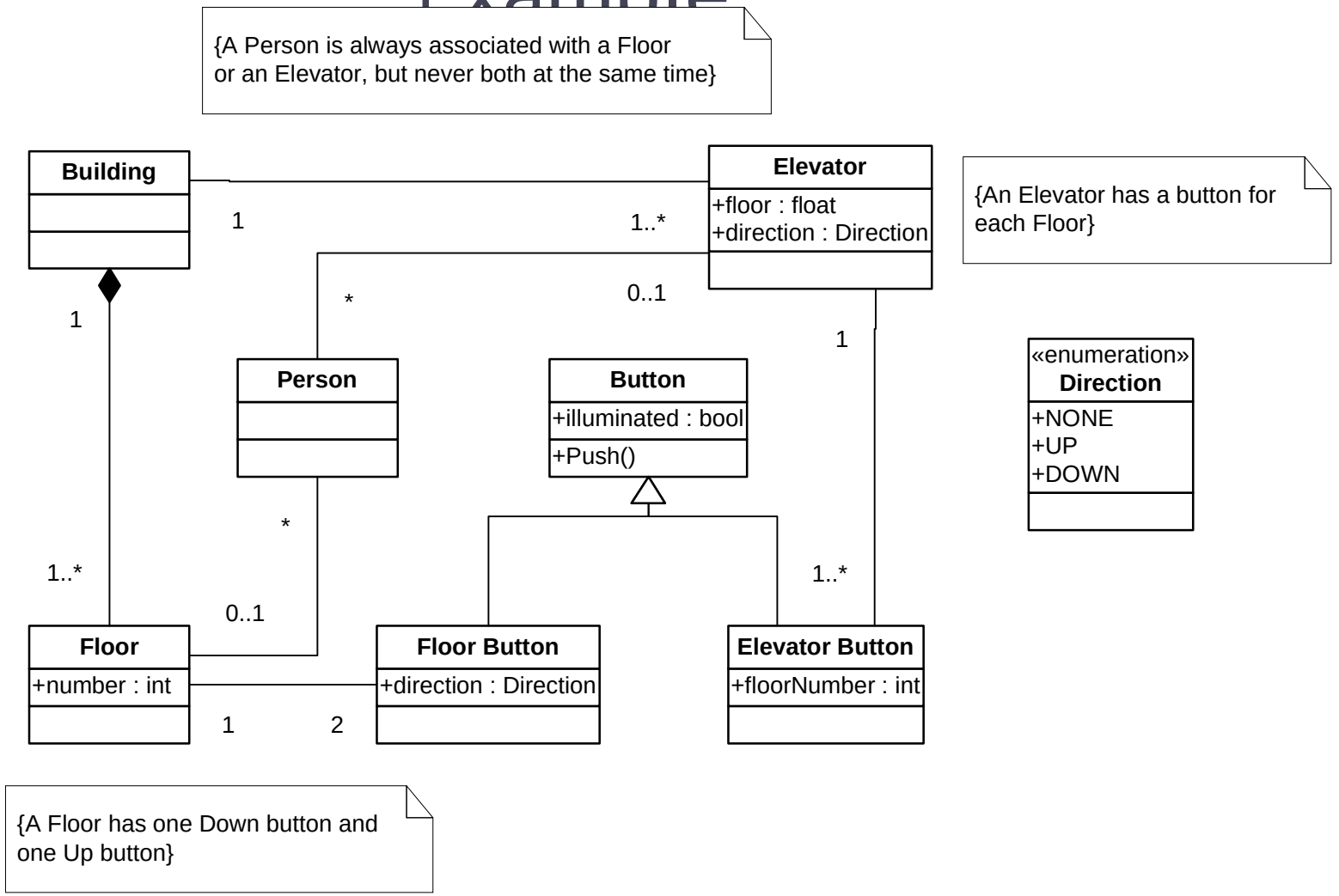


Relationships: Dependency

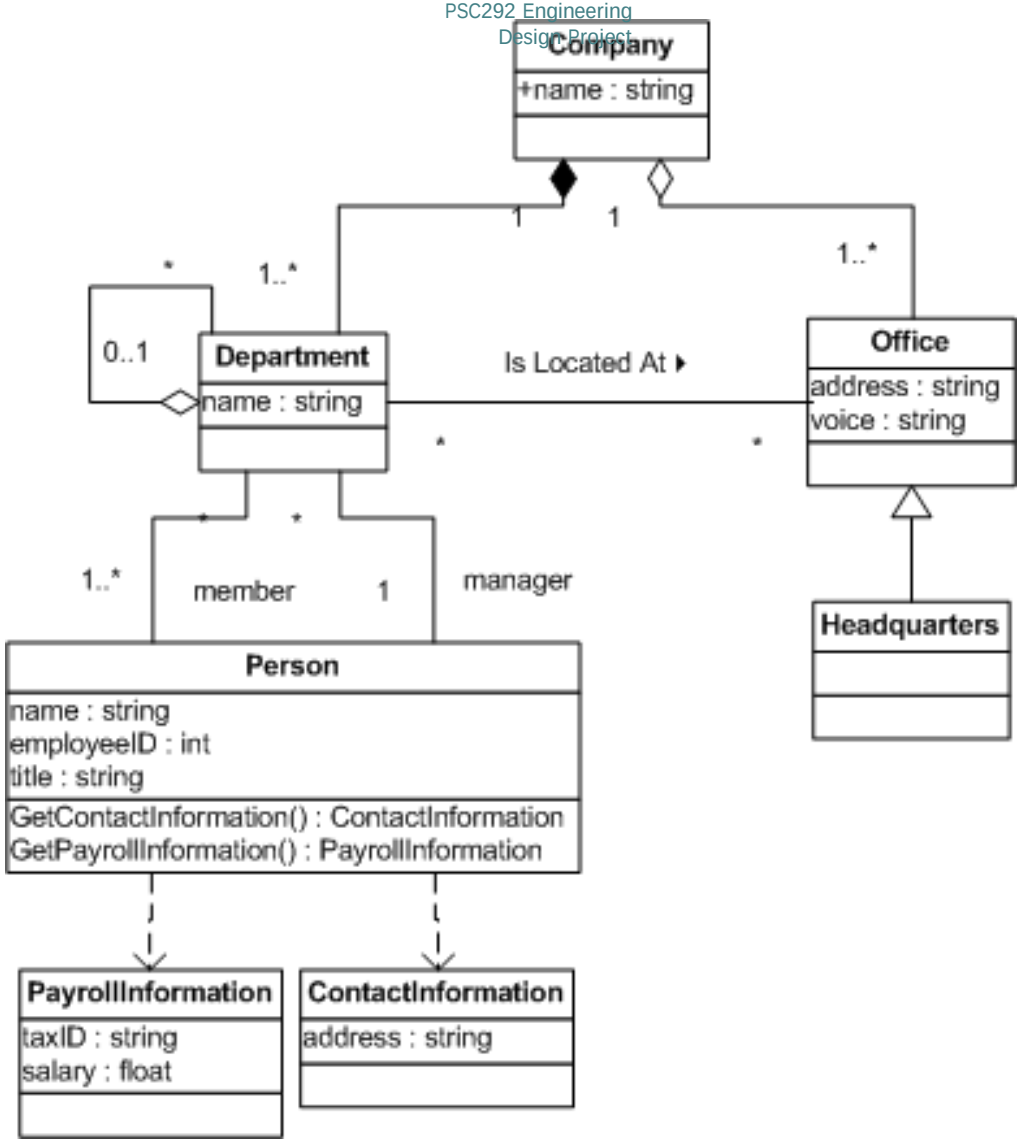
- Class A depends on class B if A "uses" B
- Common "uses" relationships
  - Operation on A has parameter of type B
  - Operation on A has return value of type B
  - Operation on A has a local variable of type B
  - Operation on A accesses static members of B
  - Operation on A accesses a global variable of type B
- If B is changed, A might also have to change



Class Diagram Example

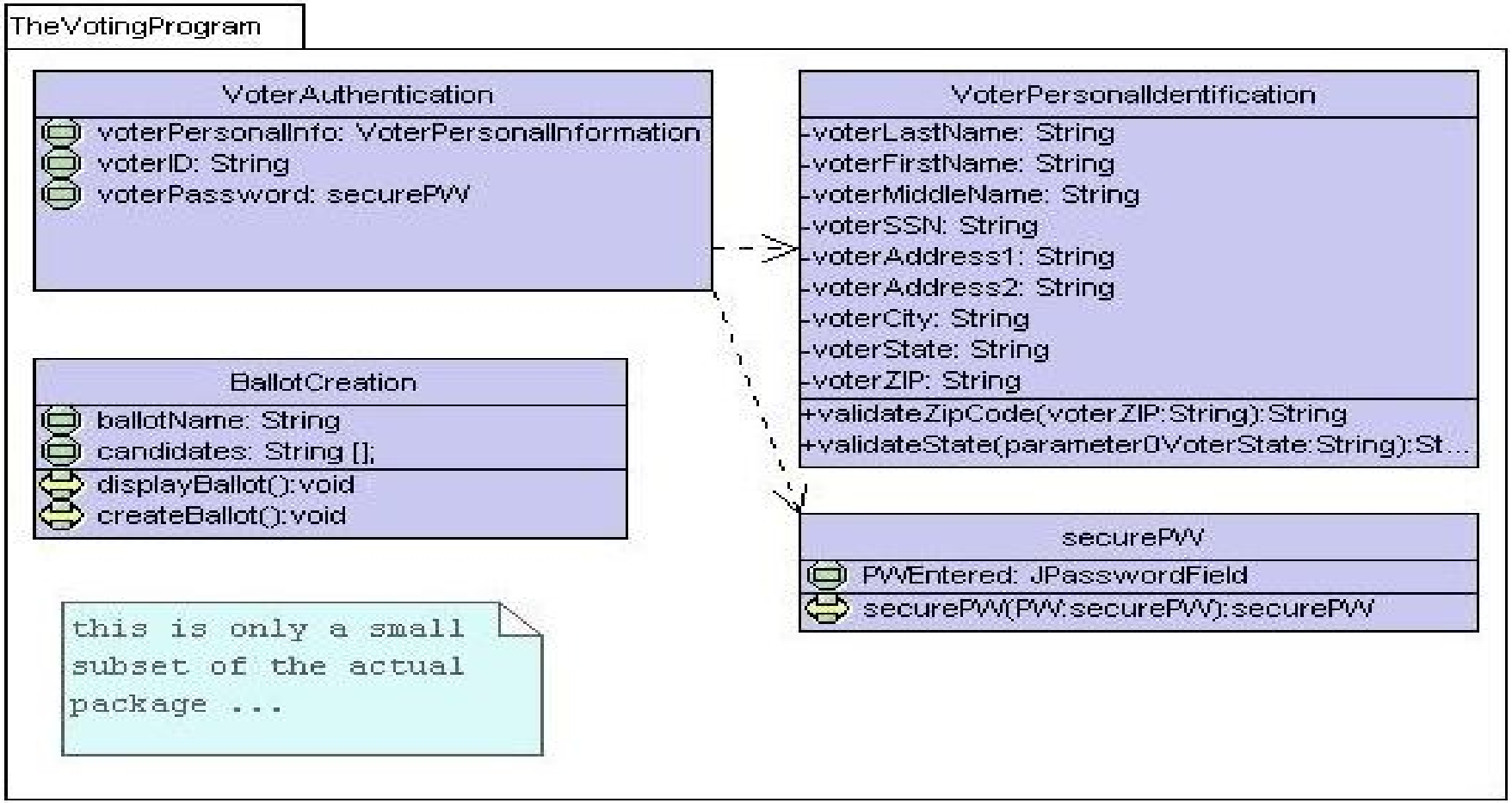


Class Diagram Example



Source: Booch et. al., The UML User Guide

PACKAGE


TheVotingProgram
VoterAuthentication
voterPersonalInfo: VoterPersonalInformation
voterID: String
voterPassword: securePW
VoterPersonalIdentification
-voterLastName: String
-voterFirstName: String
-voterMiddleName: String
-voterSSN: String
-voterAddress1: String
-voterAddress2: String
-voterCity: String
-voterState: String
-voterZIP: String
+validateZipCode(voterZIP:String):String
+validateState(parameter0VoterState:String):St...
BallotCreation
ballotName: String
candidates: String [];
displayBallot():void
createBallot():void
securePW
PWEntered: JPasswordField
securePW(PW:securePW):securePW
this is only a small subset of the actual package ...