The Unified Modeling Language (UML)

A Standard Graphical Modeling Notation

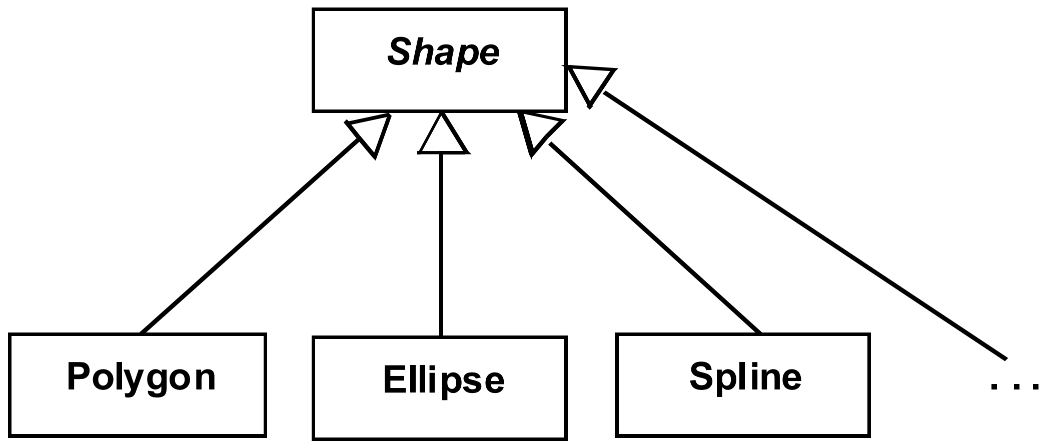
Outline

- Why Use UML
- History
- UML Characteristics
- Diagram Types
- Use Case Diagrams
- Class Diagrams

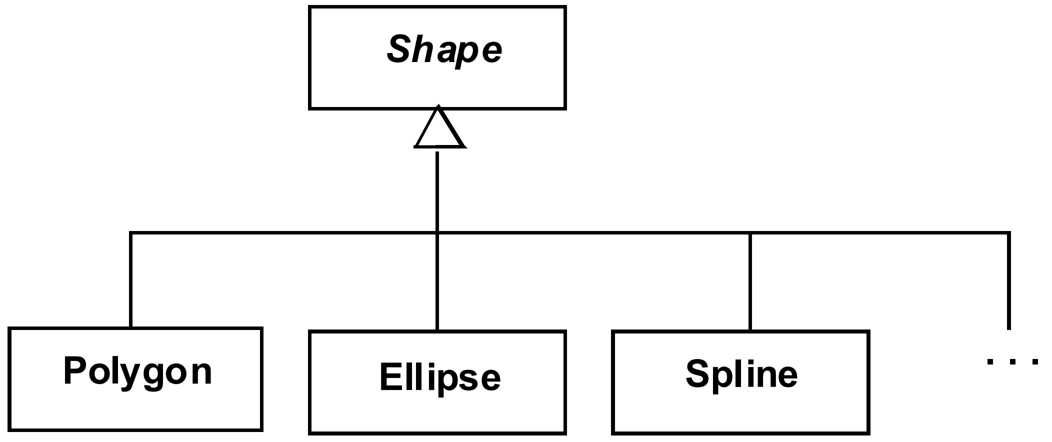
Why Use UML?

- Help analyze complex domains
- Help design complex systems
- Visualize analysis and design artifacts
- Clearly document development artifacts
- Is simple, yet expressive
- Can be applied to different processes

Some UML Examples



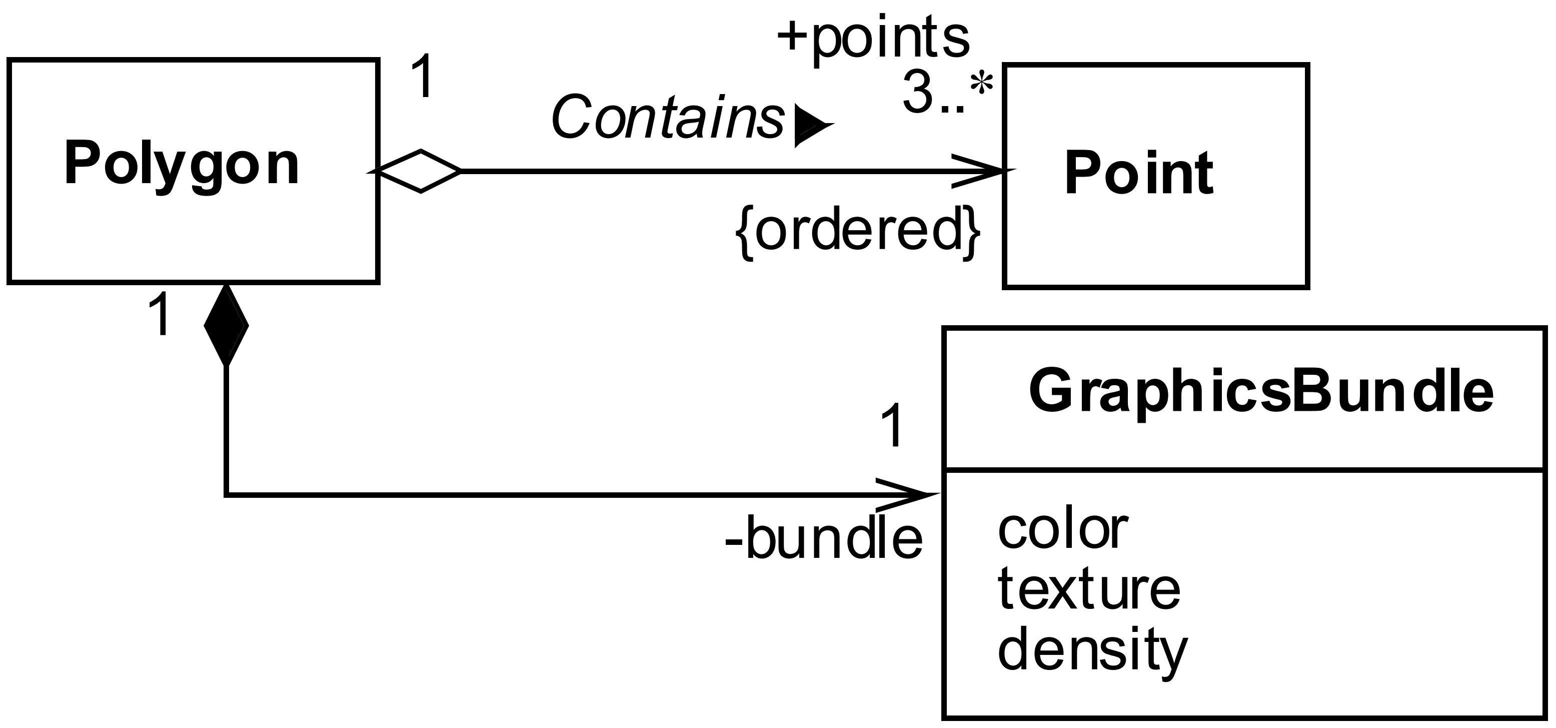
Separate Target Style



Shared Target Style

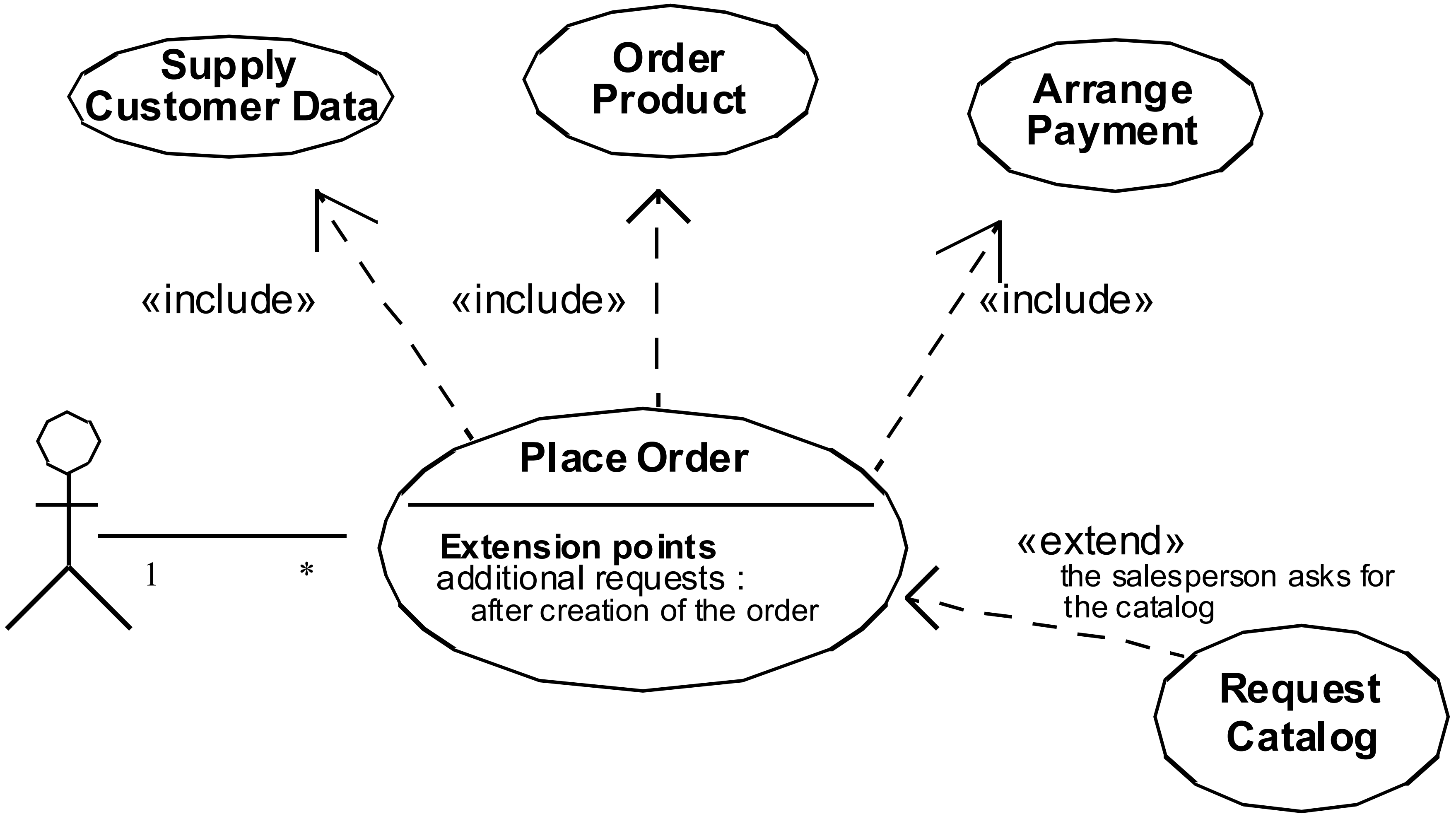
Source: OMG, *Unified Modeling Language Specification*, version 1.5. March 2003

Some UML Examples



Source: OMG, *Unified Modeling Language Specification*, version 1.5. March 2003

Some UML Examples



Source: OMG, *Unified Modeling Language Specification*, version 1.5. March 2003

History

- OO takes a foothold
- New OO modeling notations spring up
- Three amigos get together
  - Grady Booch
  - Ivar Jacobson
  - James Rumbaugh
- UML 0.8 is born

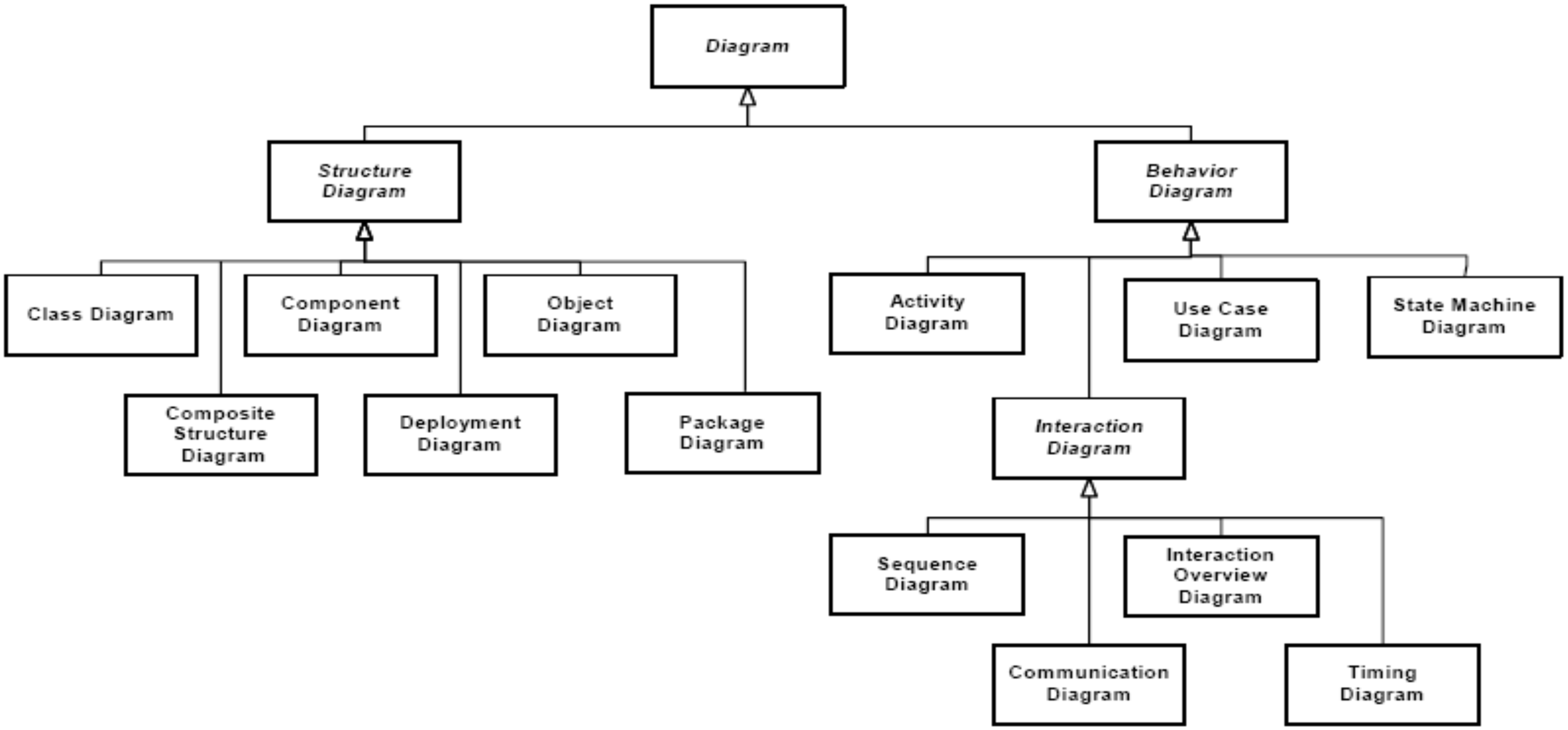
UML Characteristics

- Unified Modeling Language
- Standard graphical modeling notation
- Supported by formal semantics
- Current version: 2.0
- Wide use and acceptance in the IT industry
- Large tool support
  - Rational Rose, Together, Visio, Jude, …
  - … although no tool fully supports the standard
- Is *not* a process
- Process independent

Diagram Types

- Structure diagrams
  - Class diagram
  - Object diagram
  - Component diagram
  - Deployment diagram
  - Package diagram
  - Composite Structure diagram
- Behavior diagrams
  - Use Case diagram
  - Interaction diagrams
    - Sequence diagram
    - Interaction Overview diagram
    - Communications diagram
    - Timing diagram
  - State Machine diagram
  - Activity diagram

Diagram Types



Source: OMG, *Unified Modeling Language: Superstructure Specification*, version 2.0. August 2005

Use Case Diagrams

- Model user interaction with system
- Capture functional requirements
- Also used for
  - business modeling
  - component specification
- UML Spec does not include Use Case Specs

Use Case Definition

- *“A use case specifies a sequence of actions, including variants, that the system can perform and that yields an observable result of value to a particular actor.”*

  “The Unified Software Development Process”, Ivar Jacobsen, Grady Booch, Jim Rumbaugh*

*Author of “*Object-Oriented Software Engineering: A Use Case Driven Approach*”

Use Case Specification

Name: Create AddressEntry

Description: This use case allows the actor to create a new entry for an Address Book.

Preconditions:

1. An Address Book is open.

Steps:

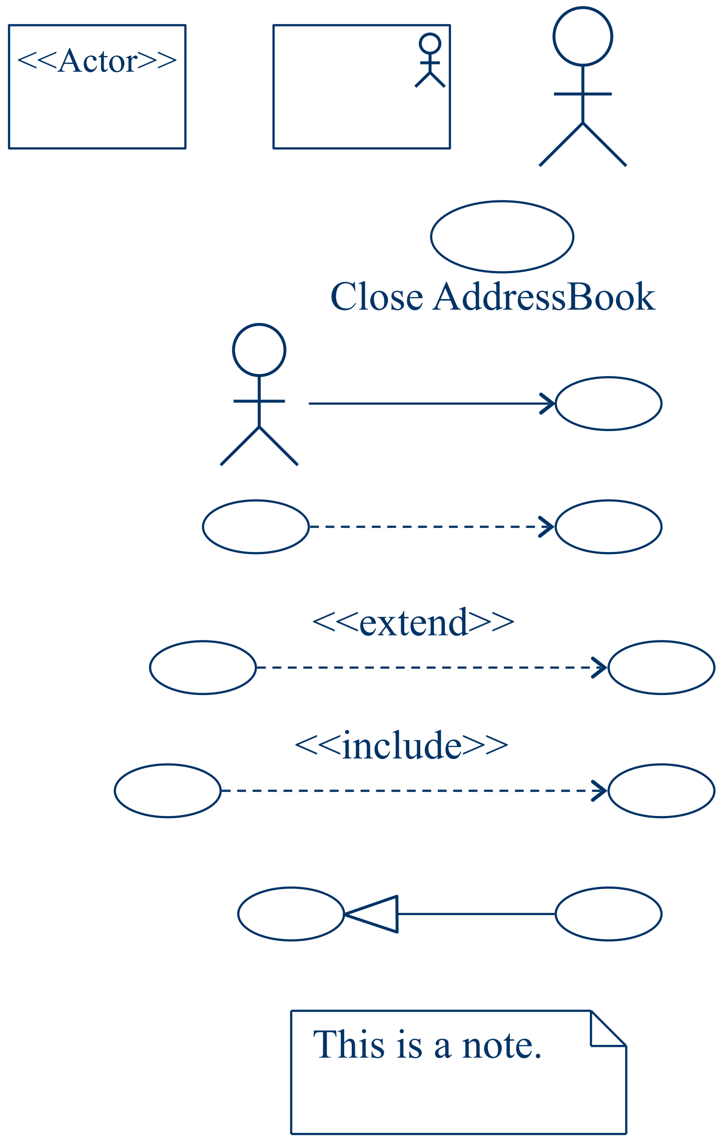
1. The actor requests to create a new AddressEntry.
2. The system creates a new AddressEntry and returns it to the actor.

Post Conditions:

1. An Address Book Entry is created.

See "*Writing Effective Use Cases*" and "http://alistair.cockburn.us/usecases/usecases.html"

Use Case Diagrams

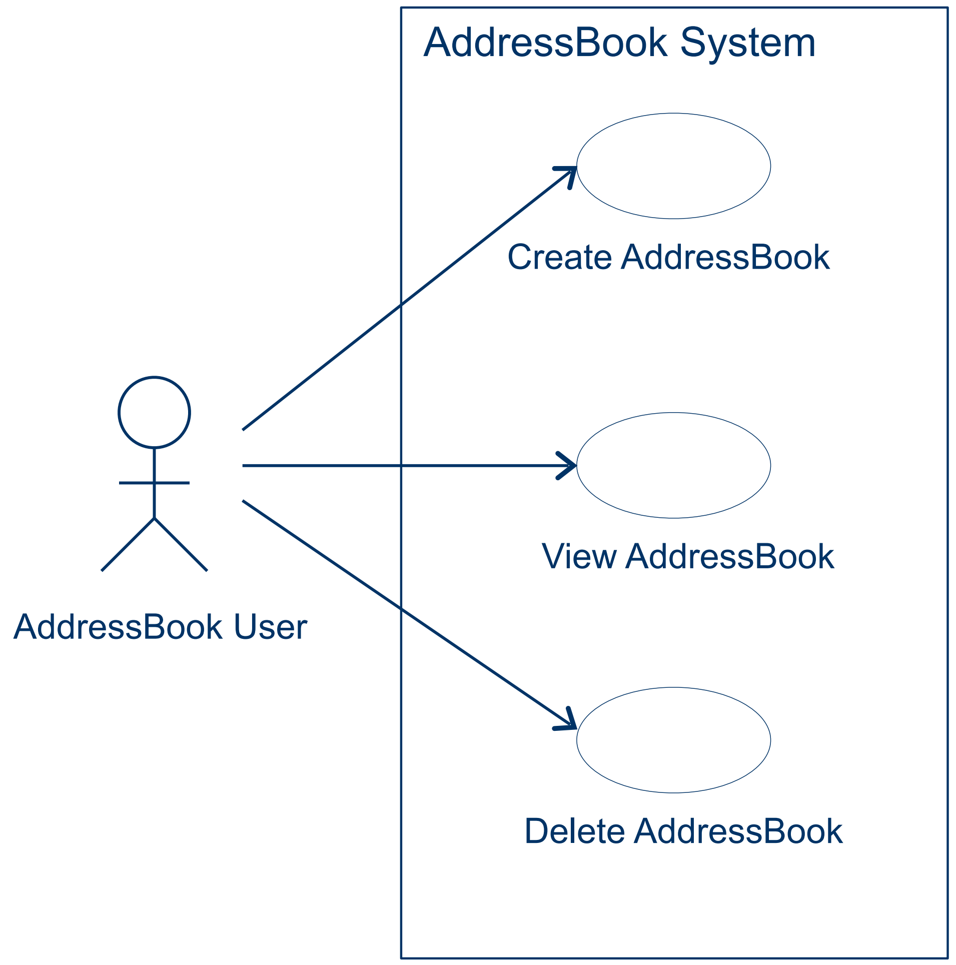


- Actor
- Use Case
- Association
- Dependency
  - extend
  - include
- Generalization
- Note (Comment)

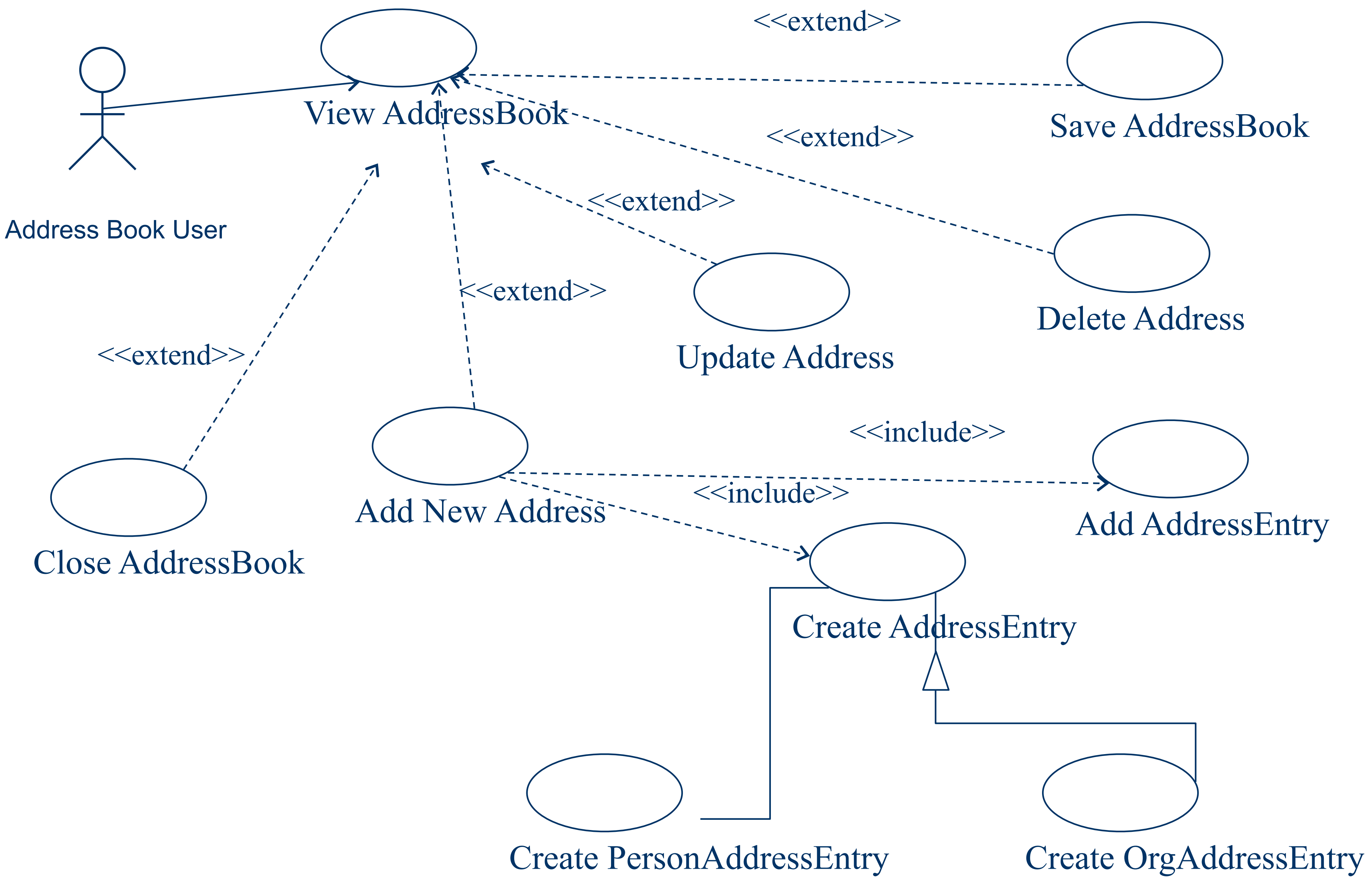
Address Book Example

- The Address Book System provides distributed access to a set of address books.
  - An Address Book is made up of Address Book Entries
  - An Address Book Entry contains a name, street address, phone number, email address
- The Address Book System shall:
  - allow users to *create*, *view*, *delete*, and *save* address books
  - allow users to *create, modify*, and *delete* entries in an address book

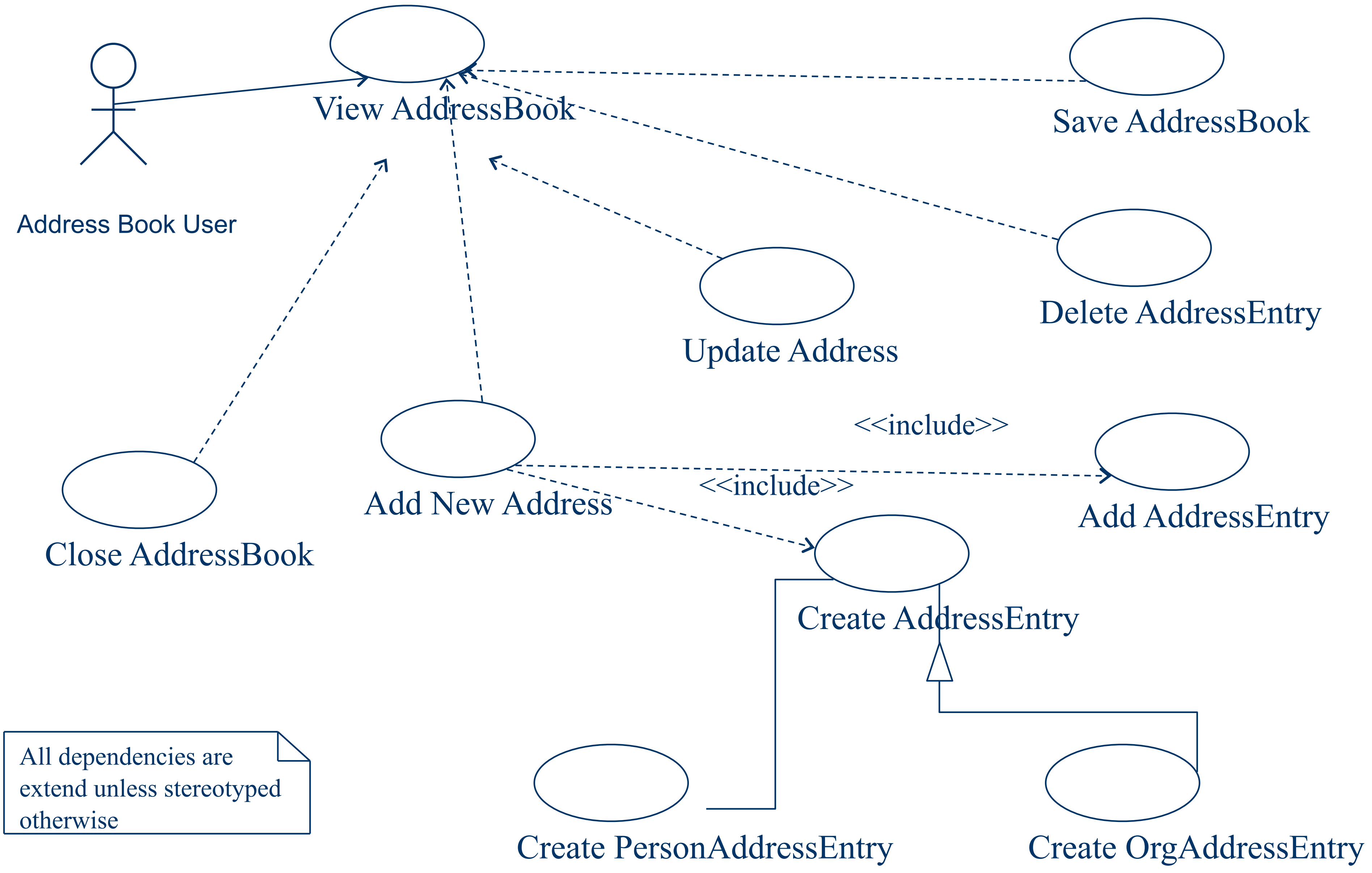
Use Case Diagram



Use Case Diagram



Use Case Diagram

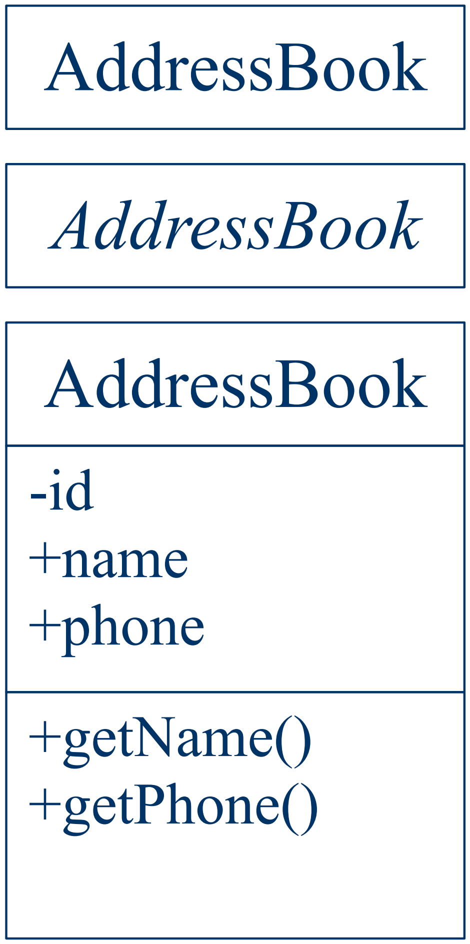


All dependencies are extend unless stereotyped otherwise

Class Diagrams

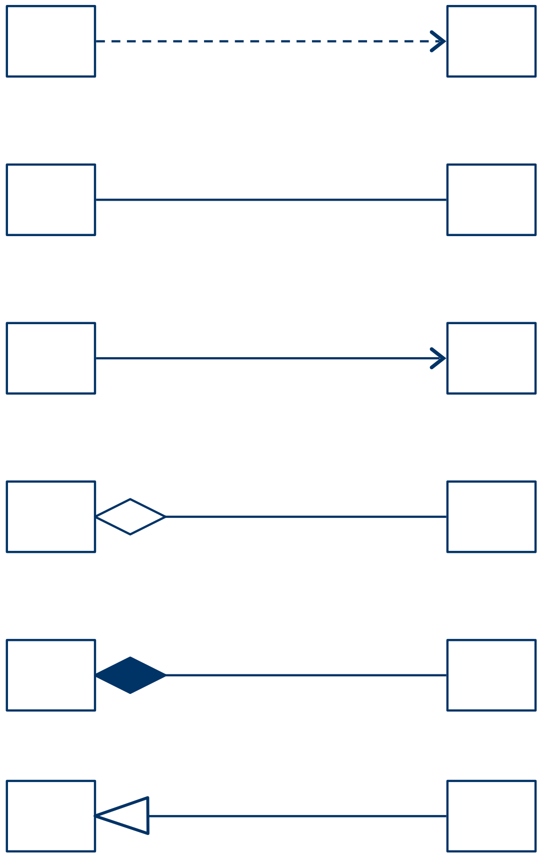
- Model system classes, interfaces, and class relationships
- Capture structural (vs. behavioral) info
- Used for
  - business domain modeling
  - logical design
  - implementation design

Class Diagrams



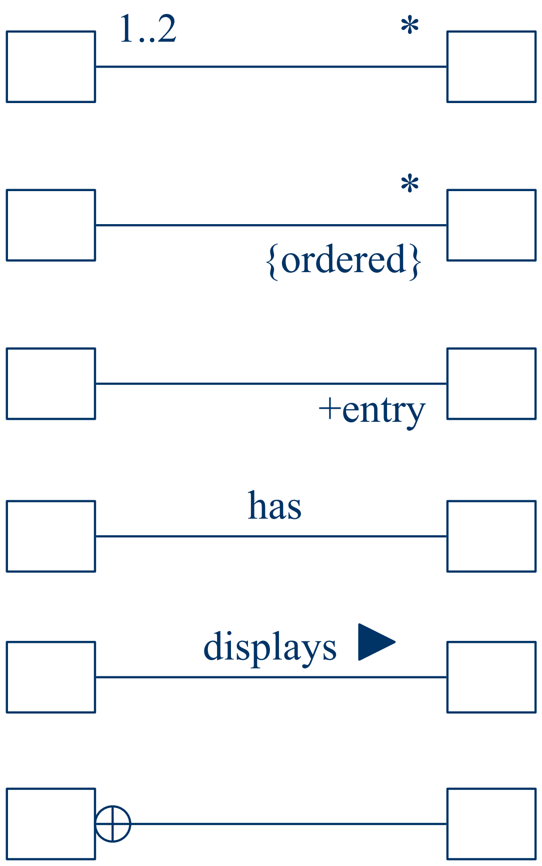
- Class
- Abstract Class
- Class with
- Attributes
- and
- Operations

Class Diagrams



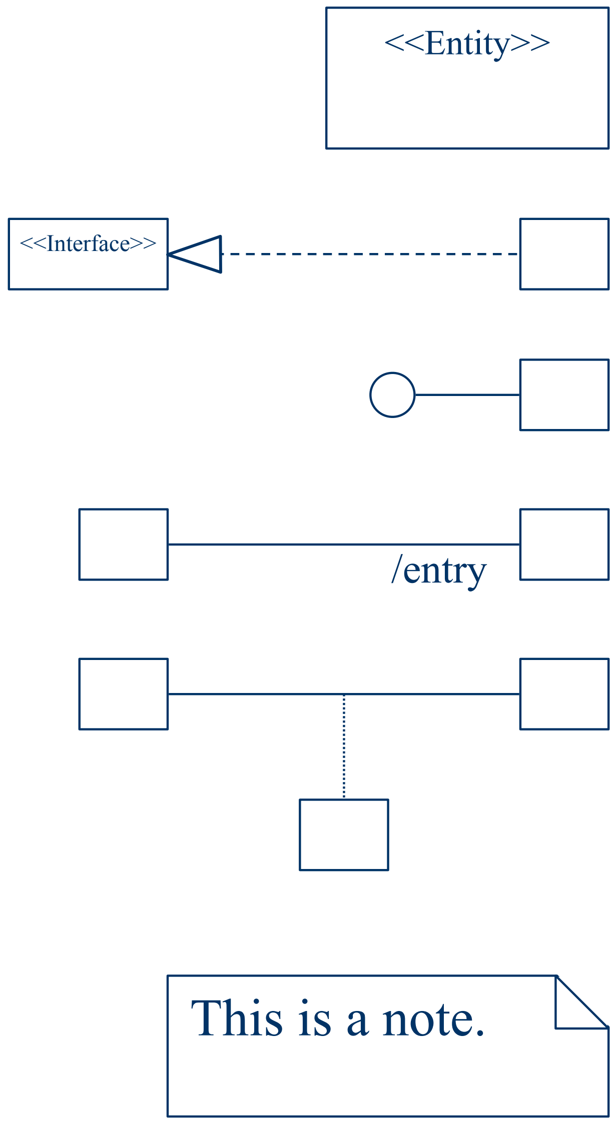
- Dependency
- Association
- Navigability
- Aggregation
- Composition
- Generalization

Class Diagrams



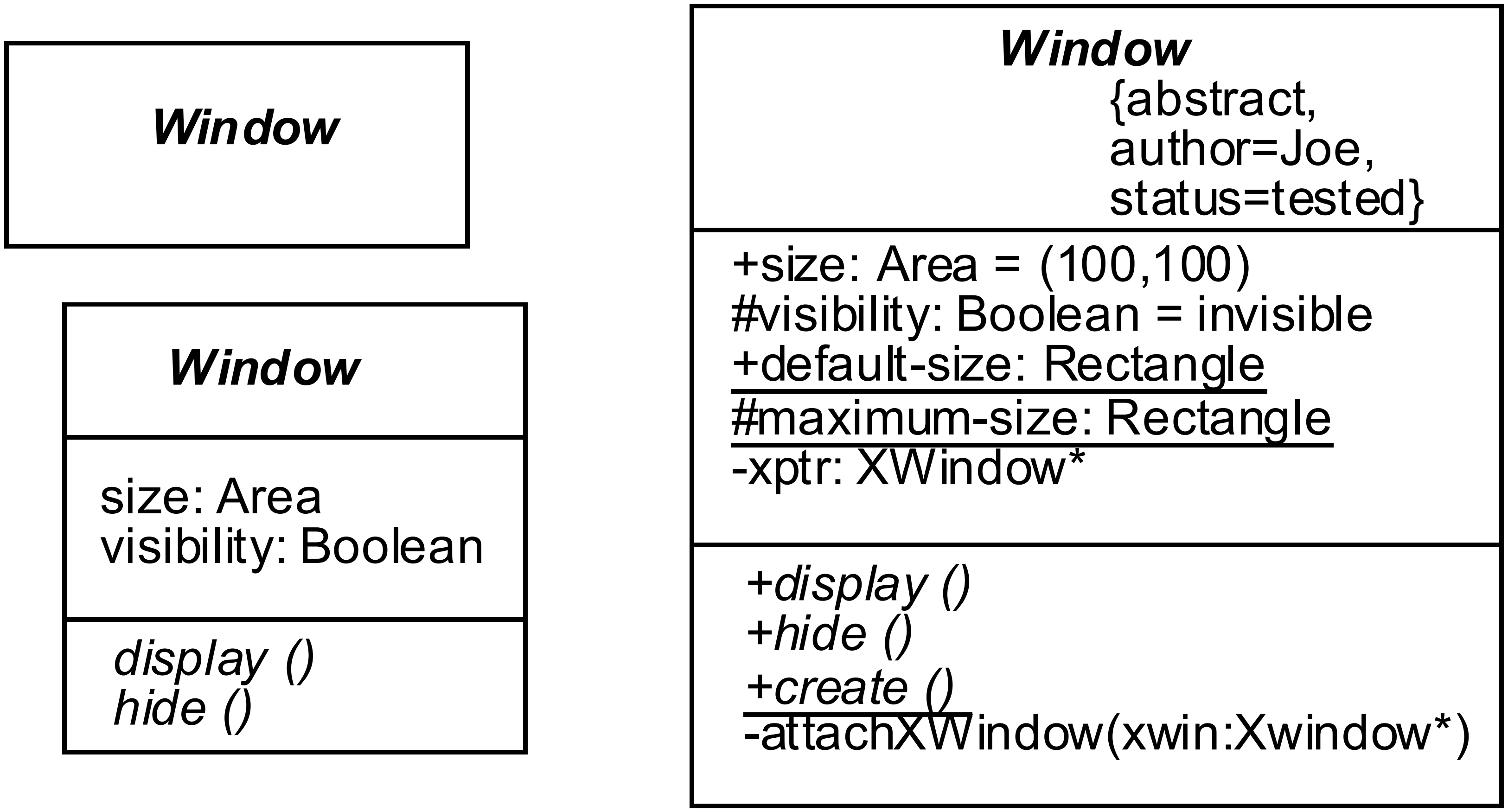
- Multiplicity
- Constraint
- Role
- Name
- Directionality
- Nesting

Class Diagrams



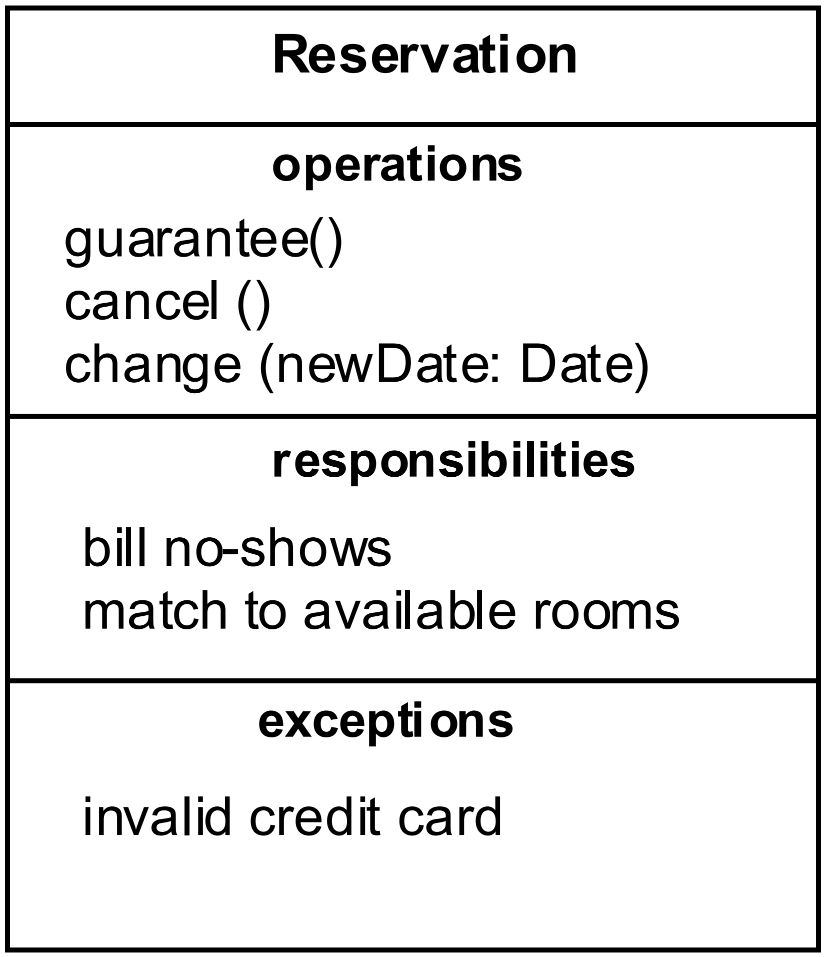
- Stereotype
- Realization
- Realization
- Derived Assoc.
- Association Class
- Note (Comment)

Class Model Element



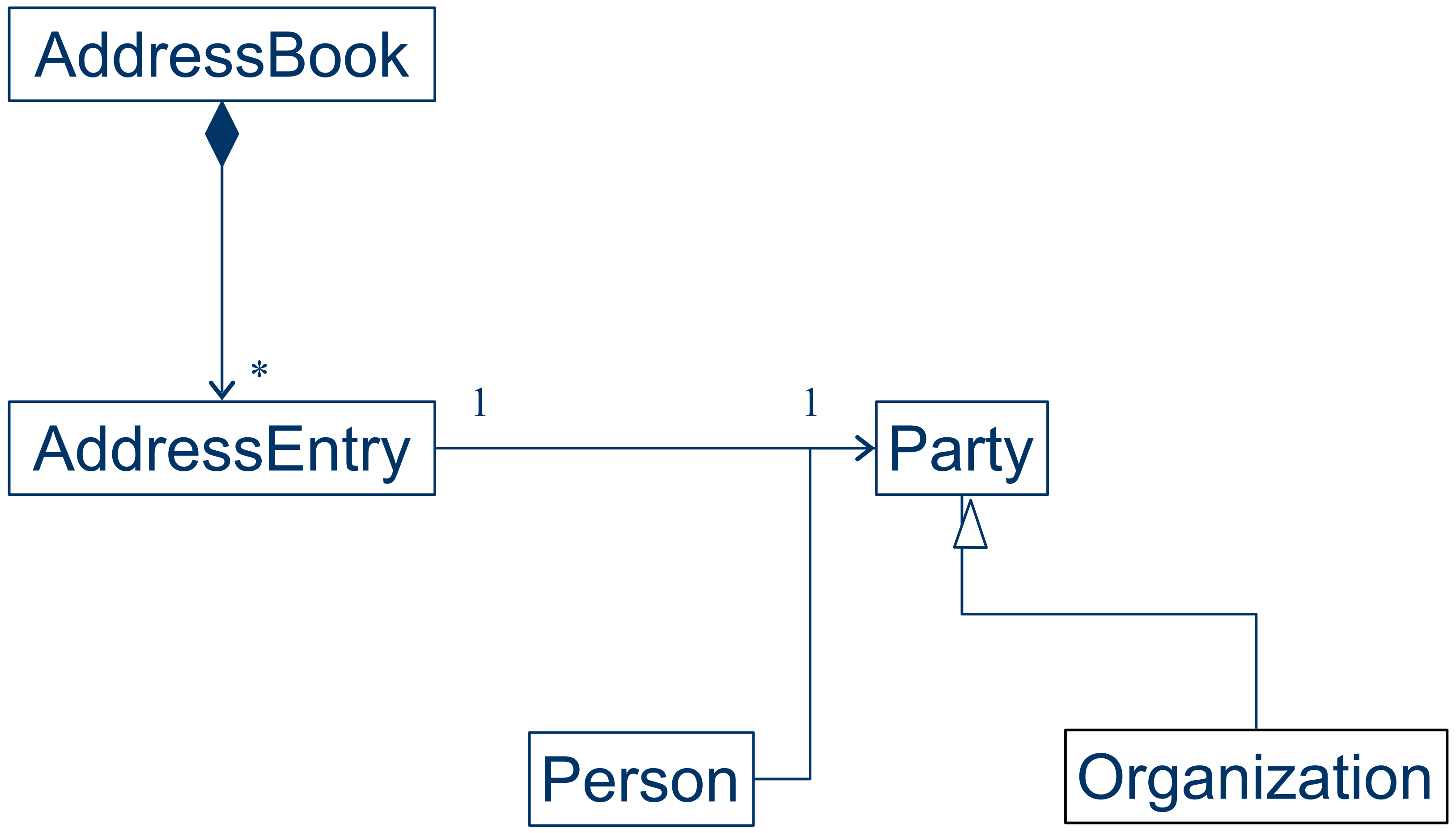
Source: OMG Unified Modeling Language Specification, version 1.5. March 2003

Class Model Element

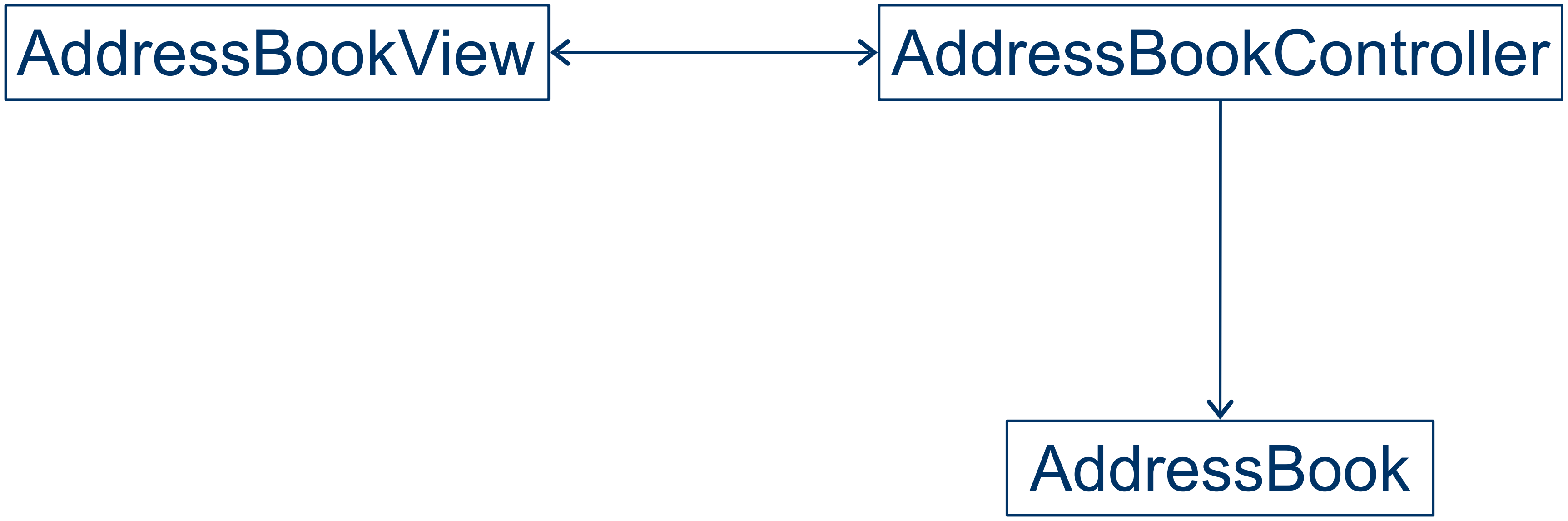


Source: OMG Unified Modeling Language Specification, version 1.5. March 2003

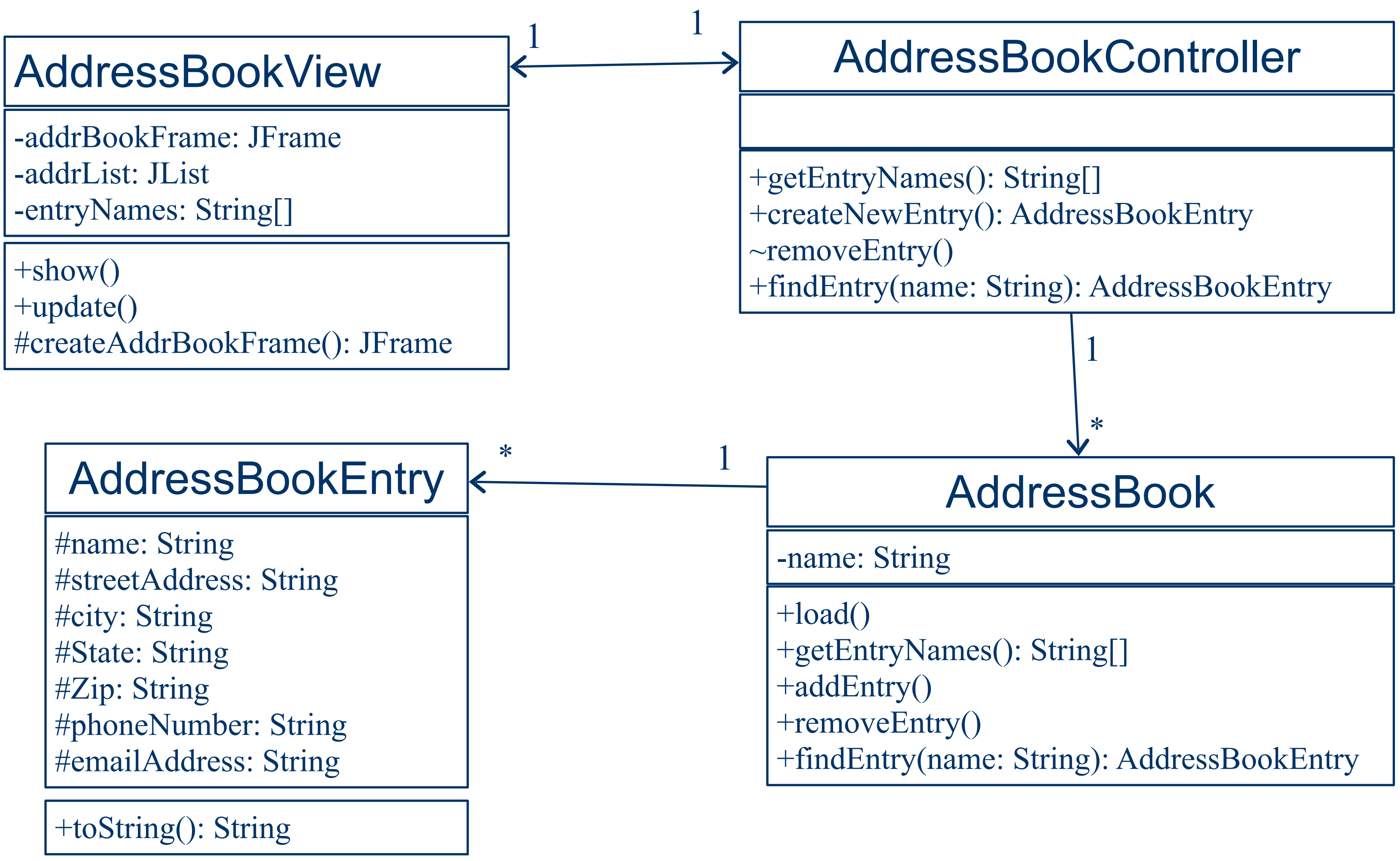
Address Book Domain



Logical Design



Implementation Design



AddressBookView
-addrBookFrame: JFrame
-addrList: JList
-entryNames: String[]
+show()
+update()
#createAddrBookFrame(): JFrame
1
1
AddressBookController
+getEntryNames(): String[]
+createNewEntry(): AddressBookEntry
~removeEntry()
+findEntry(name: String): AddressBookEntry
1
*
AddressBookEntry
#name: String
#streetAddress: String
#city: String
#State: String
#Zip: String
#phoneNumber: String
#emailAddress: String
+toString(): String
*
1
AddressBook
-name: String
+load()
+getEntryNames(): String[]
+addEntry()
+removeEntry()
+findEntry(name: String): AddressBookEntry