Overview of Unified Modelling Language (UML)

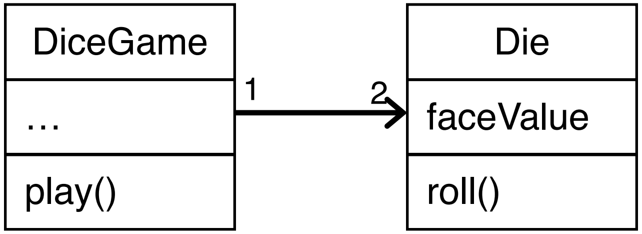
Topics Today

- Diagrams for Static model
  - Class diagrams
- Diagrams for Dynamic behavior
  - Sequence diagrams
  - Collaboration diagrams
  - State diagrams

Static and Dynamic Modeling

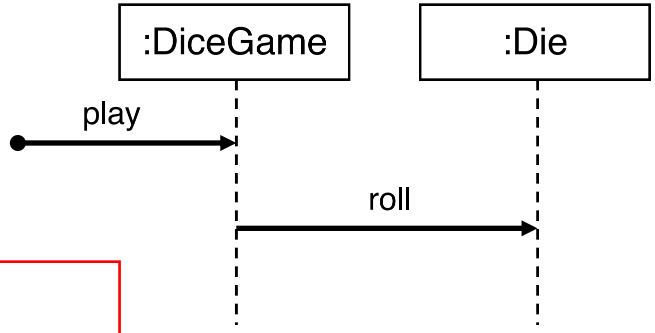
- Static Models: Help design the definition of packages, class names, attributes, and method signatures
  - □ class diagrams, package diagrams, deployment diagrams
- Dynamic Models: Help design the logic, the behavior of the code or the method bodies
  - □ sequence diagrams, communication diagrams, state diagrams, activity diagrams

static model



DiceGame
…
play()
1
2
Die
faceValue
roll()

dynamic model



:DiceGame
:Die
play
roll

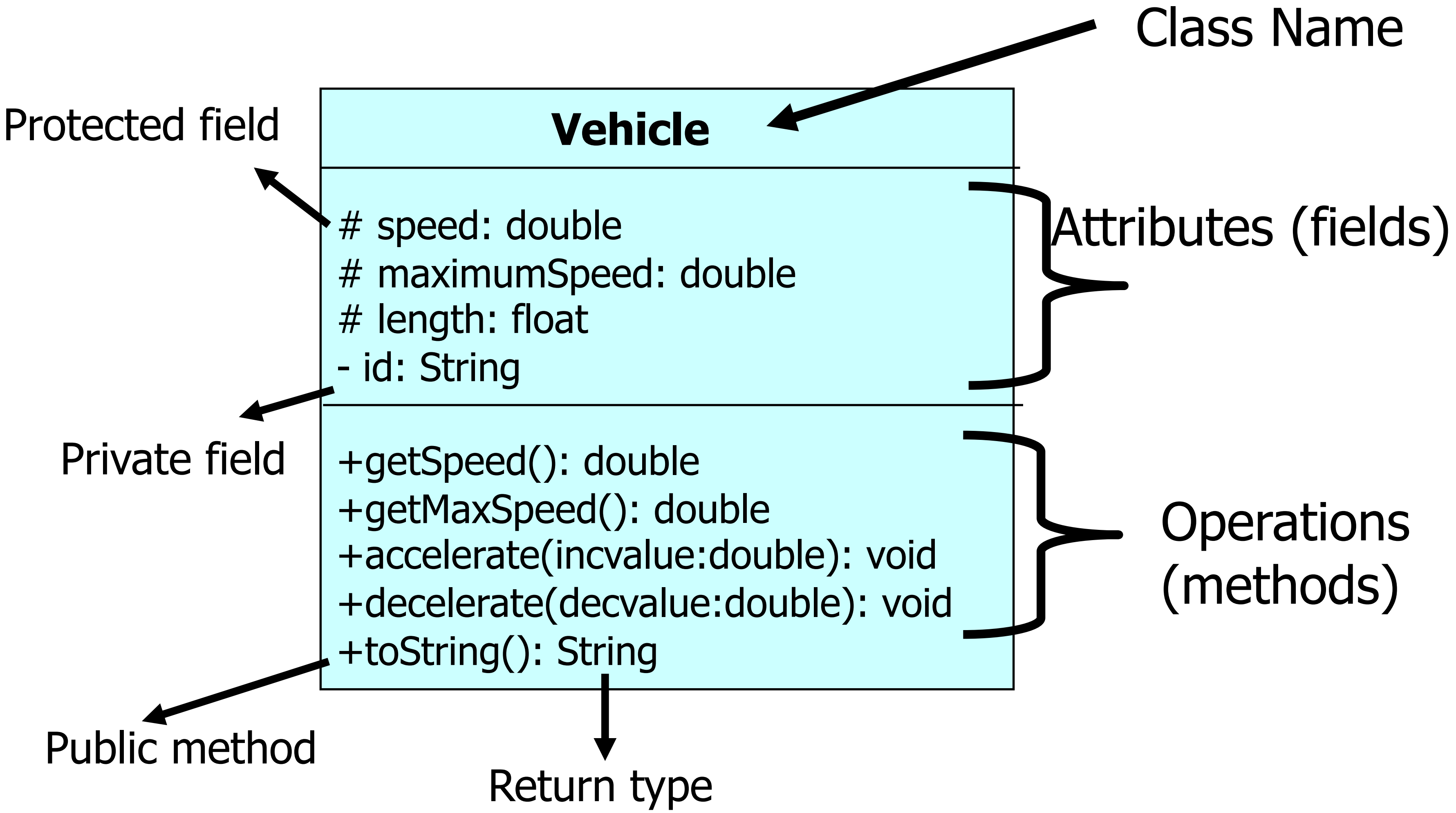
Create models in parallel !

Spend significant time doing interaction diagrams, not just class diagrams

Class Diagrams

- Represent the Static structure
- Classes/interfaces in the system and their relations
    - Inheritance
    - Realization
    - Associations
        - Multiplicity
        - Composition
        - Aggregation

Class Representation



Class Name

Protected field

Vehicle

# speed: double
# maximumSpeed: double
# length: float
- id: String

Attributes (fields)

Private field

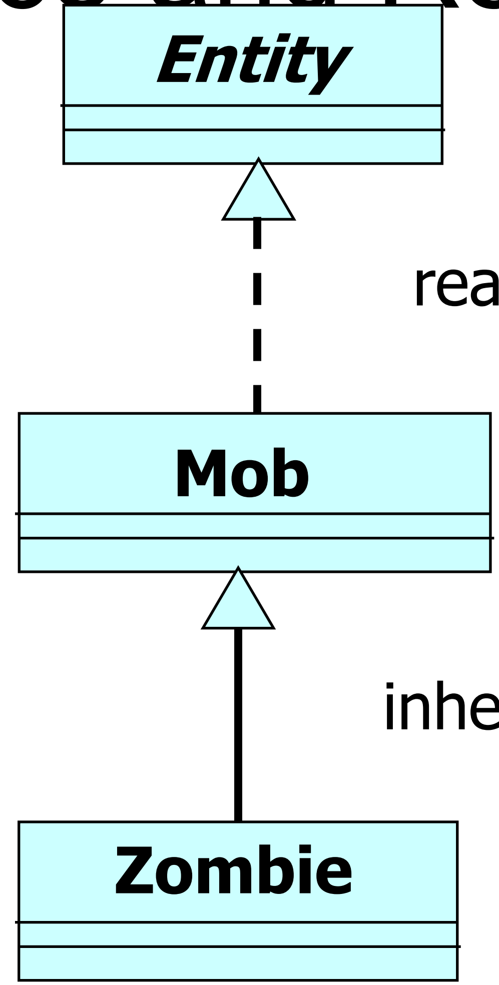
+getSpeed(): double
+getMaxSpeed(): double
+accelerate(incvalue:double): void
+decelerate(decvalue:double): void
+toString(): String

Operations (methods)

Public method

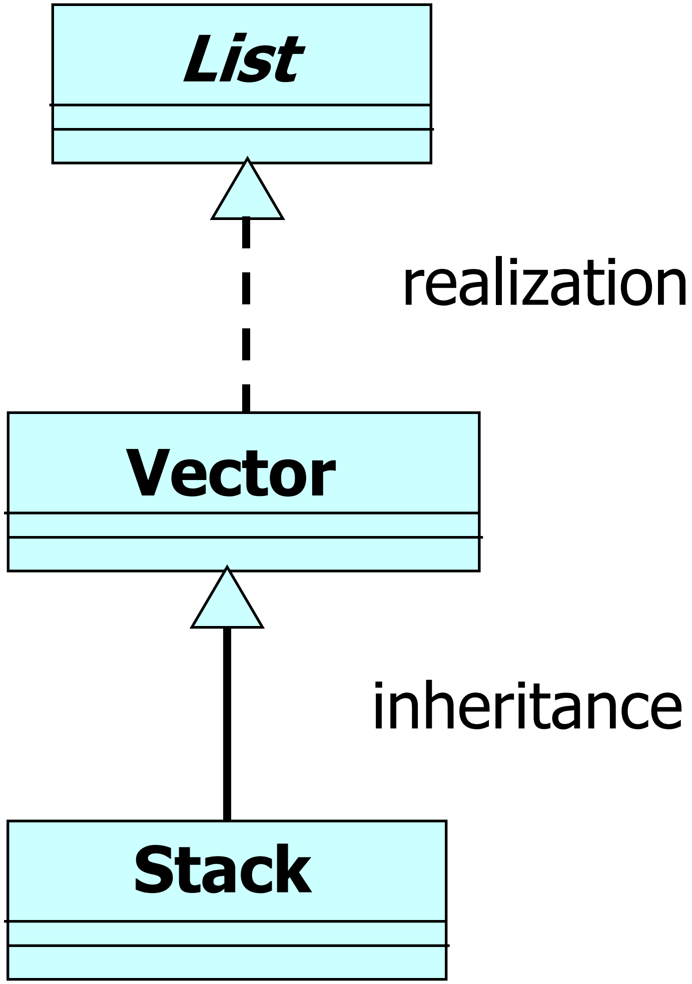
Return type

Inheritance and Realization

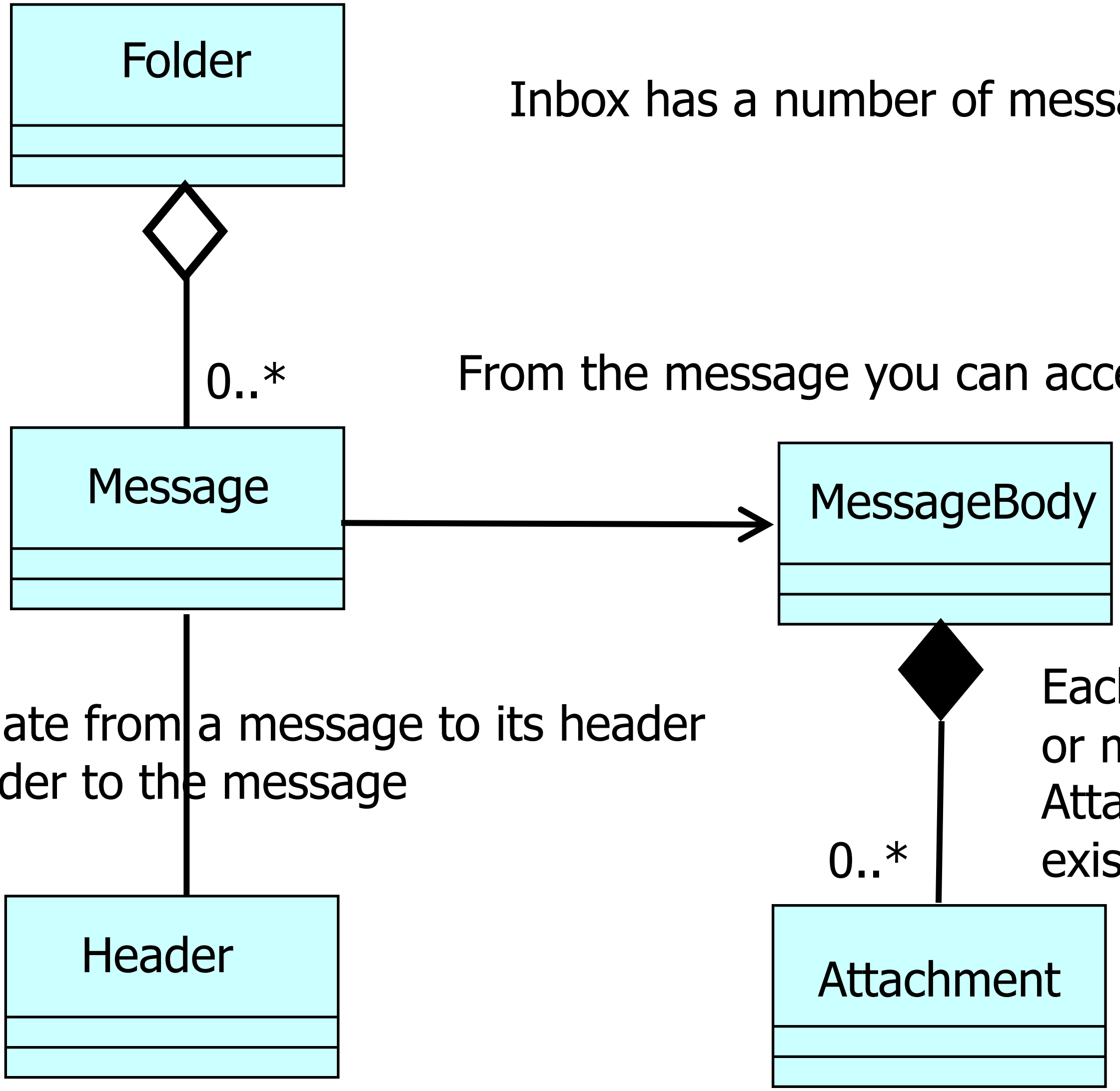


IS-A
relation

BAD!



Associations

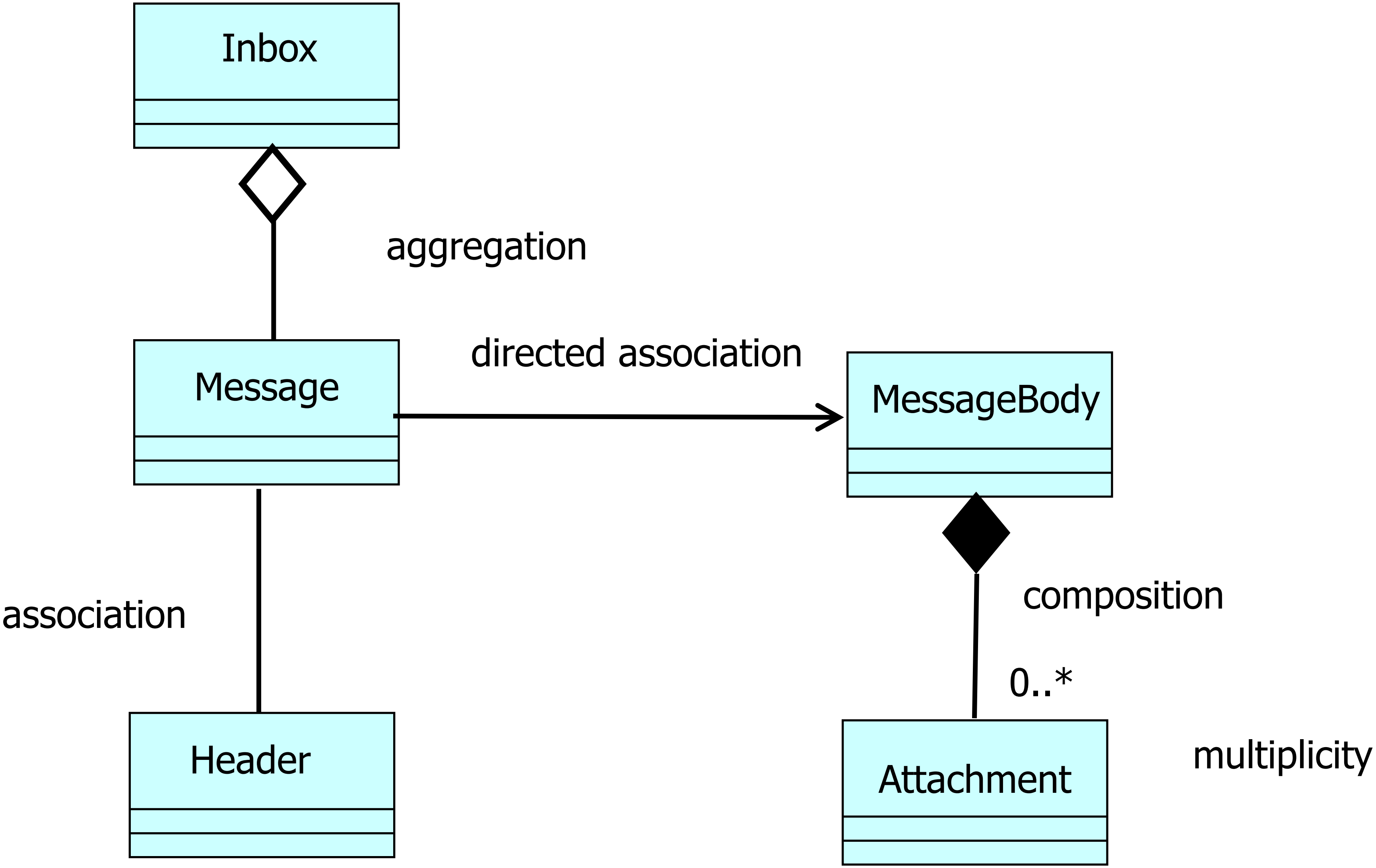


Inbox has a number of messages

From the message you can access to the body

You can navigate from a message to its header or from a header to the message

Each body has 0 or more attachments. Attachments cannot exist without a body


Inbox
aggregation
Message
directed association
MessageBody
association
composition
0..*
Header
Attachment
multiplicity

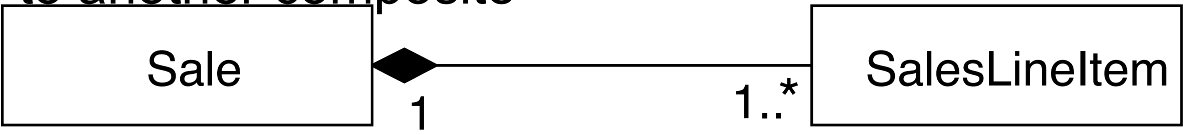
Relationship Summary

- Association
  - When two classes are connected to each other in any way
  - You can define the flow of the association by using a directed association. The arrowhead identifies the container-contained relationship
- Aggregation
  - When a class is formed as a collection of other classes,
  - It is also called a "**has a**" relationship.
- Composition
  - When there is a strong life cycle is associated between the classes.
  - part and whole live and die together
- Multiplicity
  - one to many, many to many, many to one, one to one, etc.
- Dependency
  - When one entity depends on the behavior of another entity.

Composition - Aggregation

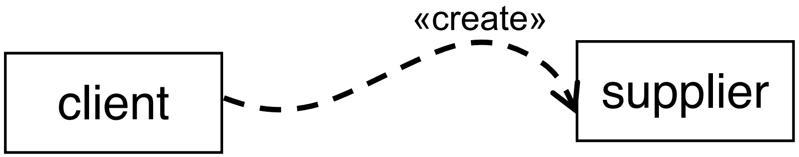
- Aggregation: A kind of association that loosely suggests whole-part relationships

- Composition: A strong kind of whole-part aggregation
  - e.g., GameBoard-Square
  - A composition relationship implies
    - An instance of the part belongs to only one composite instance at a time
    - The part must always belong to a composite
    - ***The composite is responsible for the creation and deletion of parts***
      - either by itself or by collaborating with other objects
  - If the composite is destroyed, its parts must either be destroyed, or attached to another composite

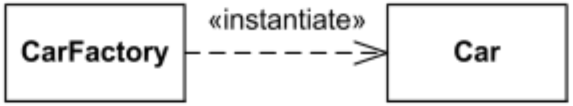


Dependency

- □ A client element has knowledge of another supplier element, and
- □ A change in the supplier could affect the client

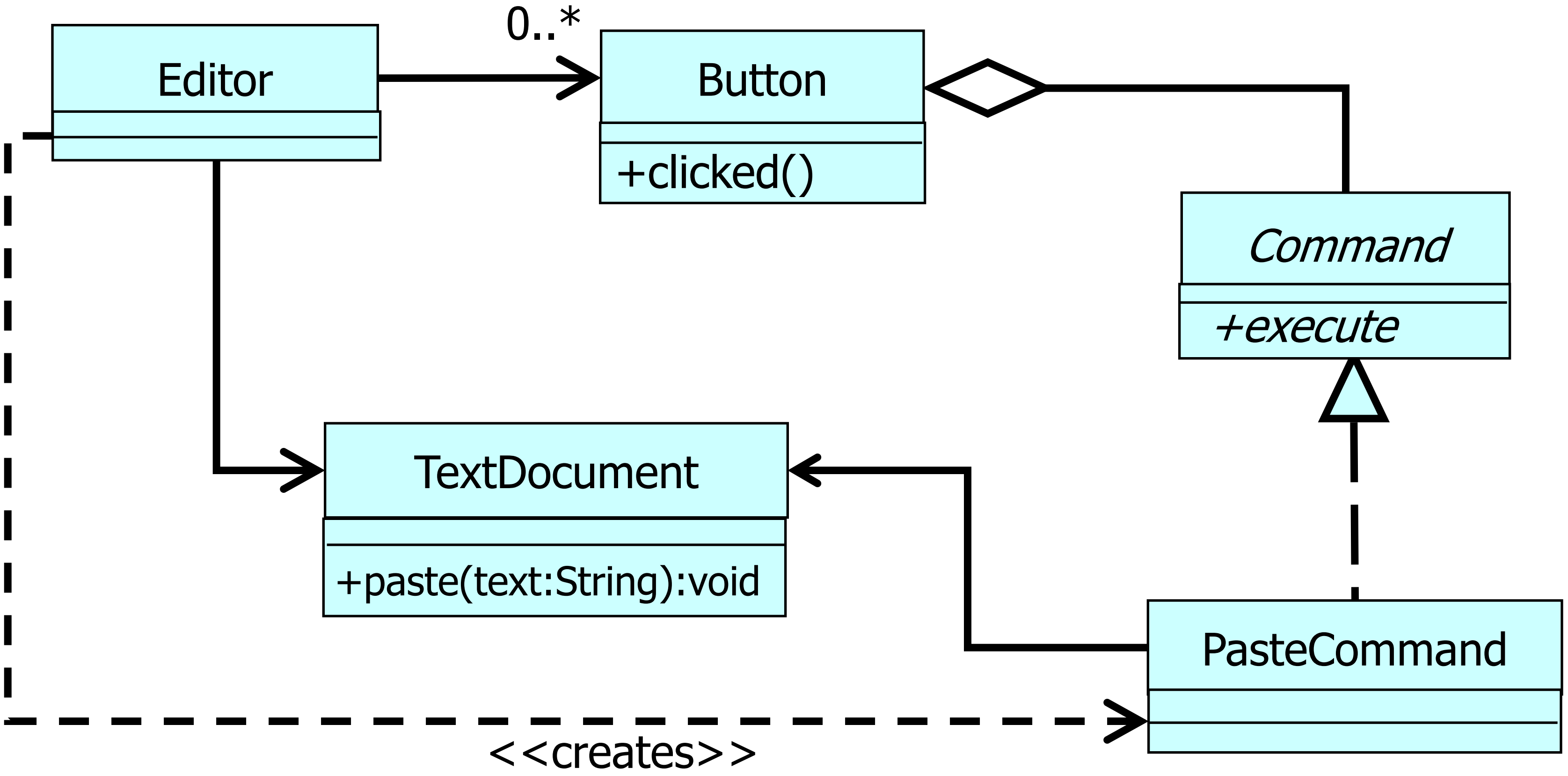


- ■ use for depicting global, parameter variable, local variable, and static-method dependency between objects
  - □ Class A depends on class B if the B is a parameter variable or local variable of a method of A.
- ■ To show the type of dependency, a label can be used



CarFactory depends on the Car class. Car class could be defined without the knowledge of CarFactory class, but CarFactory requires Car for its definition because it produces Cars

Sample Class Diagram

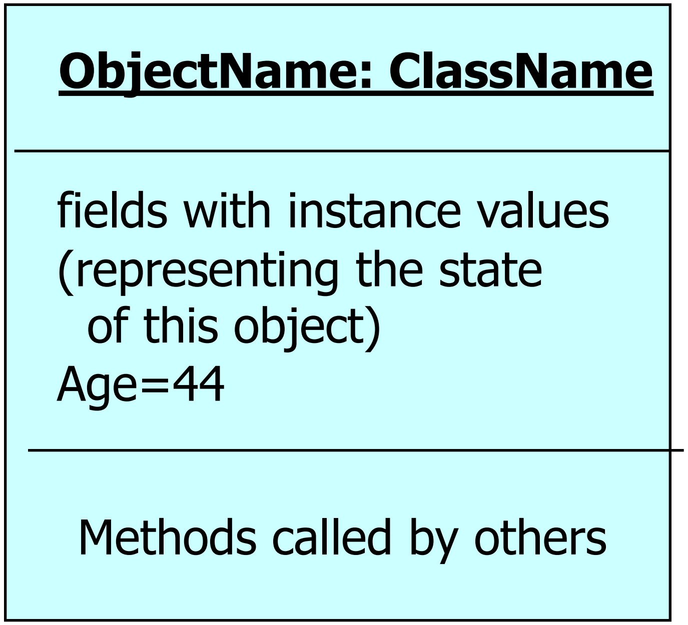


Dependency

Dynamic Behavior Diagrams

- Object diagram
  - to show the object interaction during runtime
- Sequence diagram
  - to show the sequence of actions that occur in a system
- Collaboration diagram
  - to show objects, the links connecting them, and the interactions over each link

Object Representation



ObjectName: ClassName

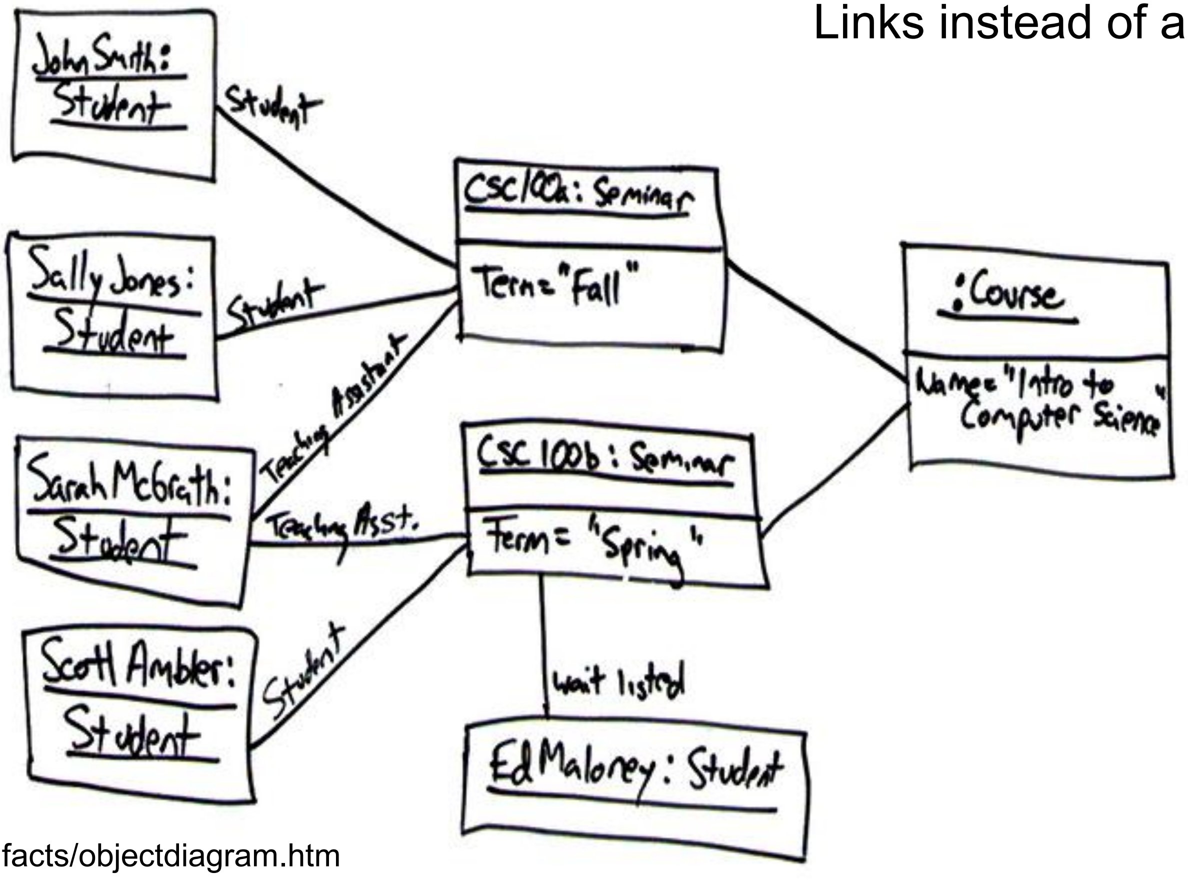
fields with instance values
(representing the state
of this object)
Age=44

Methods called by others

UML object diagrams

- they show objects and the connections between them.

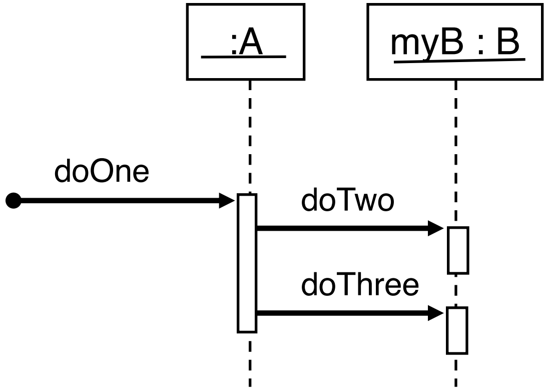
Links instead of associations



ttps://agilemodeling.com/artifacts/objectdiagram.htm

Representing Interactions

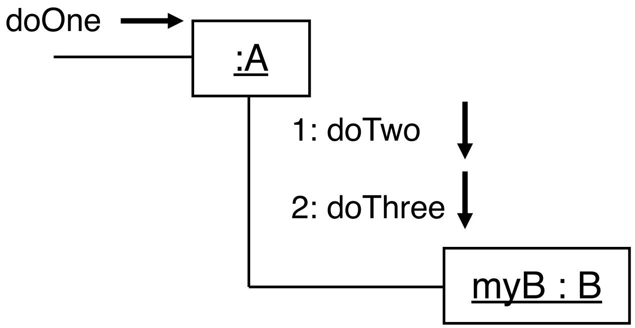
- Two common diagrams (both can express similar interactions):
  - □ Sequence diagrams: illustrate interactions in a kind of fence format
  - □ Communication diagrams: illustrate object interactions in a graph or network format



sequence diagram

```
public class A{
  private B myB = new B;
  public void doOne(){
    myB.doTwo();
    myB.doThree();
  }
  …
}
```

partial definition of class A

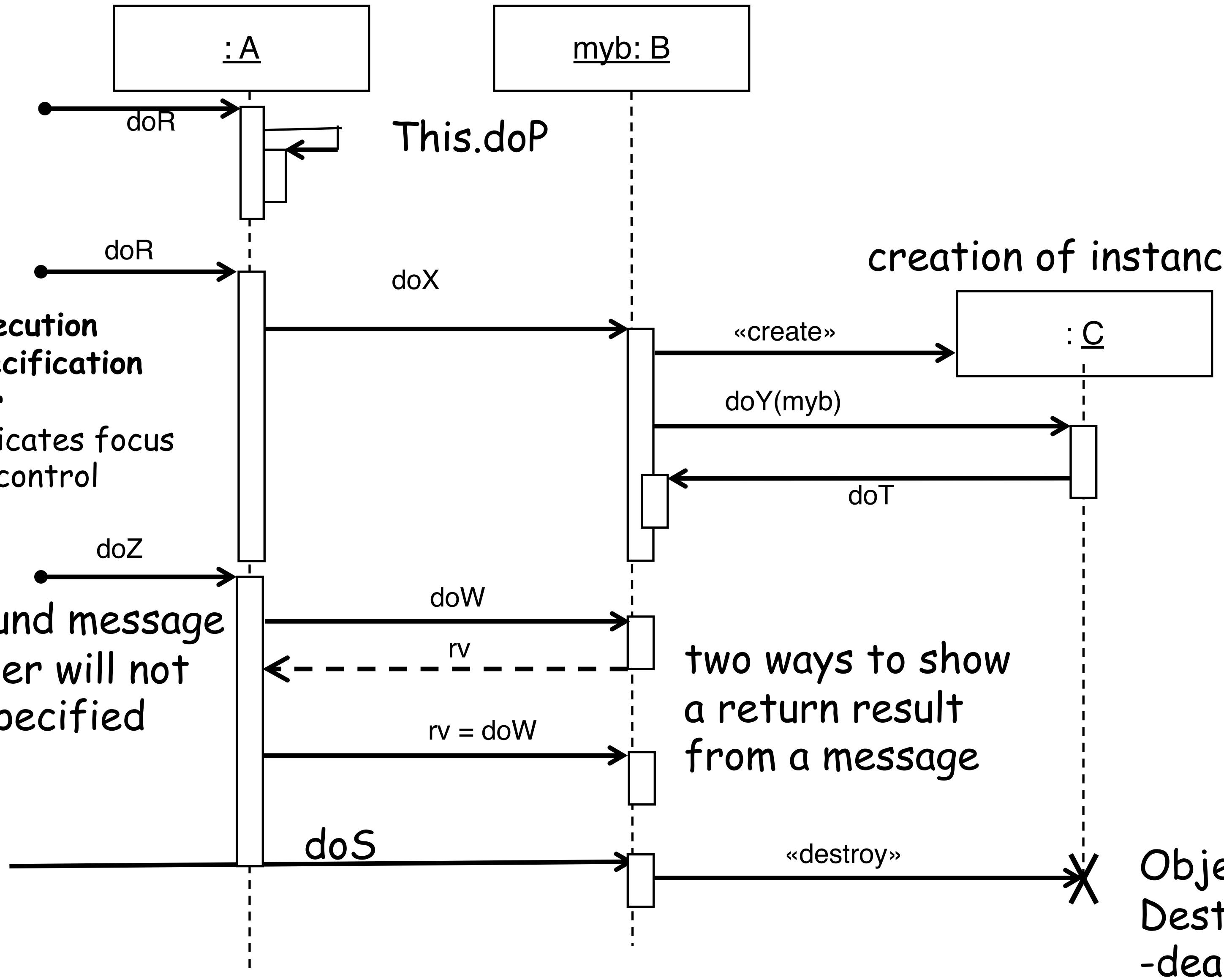


communication diagram

Sequence diagrams

- To define the sequence of actions that occur in a system
- Two dimension
  - shows the life of the objects
  - shows the sequence of the creation or invocation of the objects

Messages



This.doP

creation of instances

Execution specification bar
indicates focus of control

a found message
sender will not
be specified

two ways to show
a return result
from a message

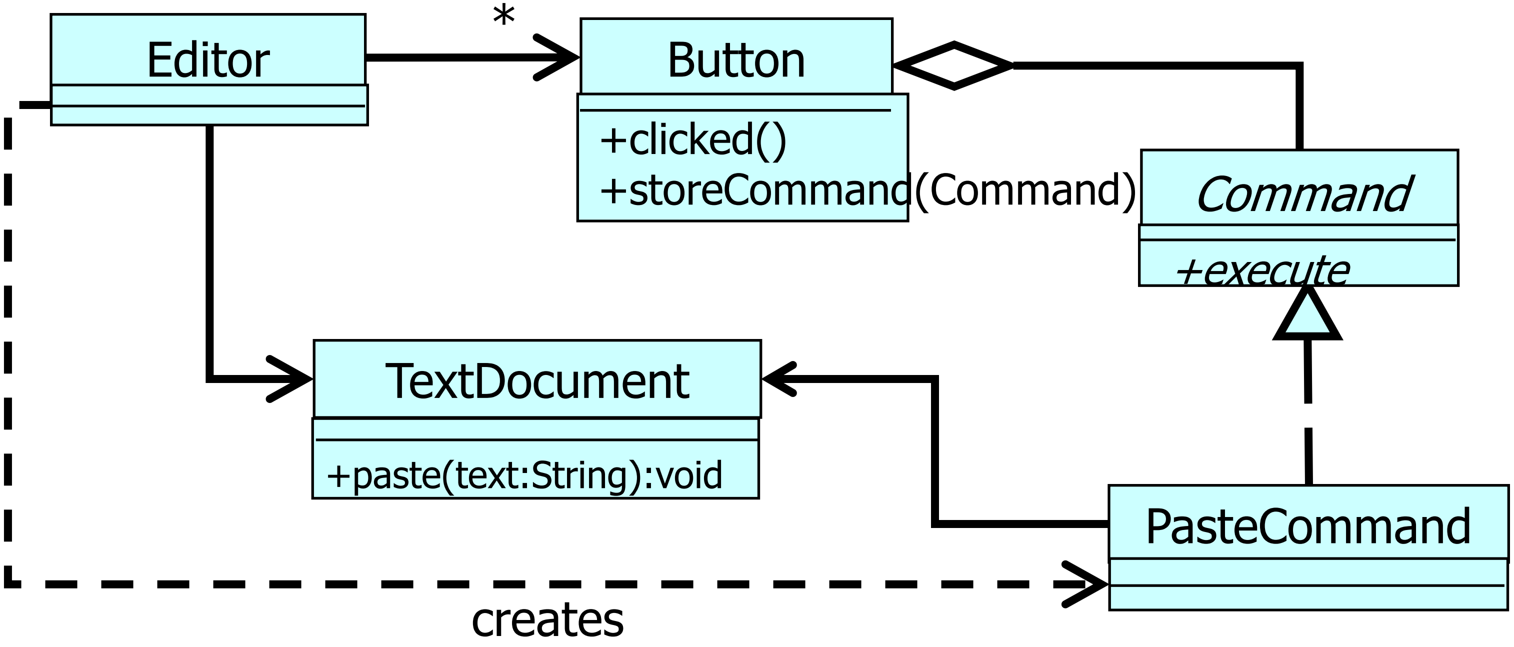
Object
Destruction
-dealloc or Free

Lifeline Boxes

: Sale	unnamed instance of class Sale
s1 : Sale	a named instance of class Sale
«metaclass» Font	represents the **class** Font. Used in representing static method invocations
sales: ArrayList<Sale>	represents an instance of ArrayList class, parameterized to hold Sale objects
sales[i] : Sale	represents one instance of class Sale, selected from the sales
: Store 1	1 implies this is a singleton

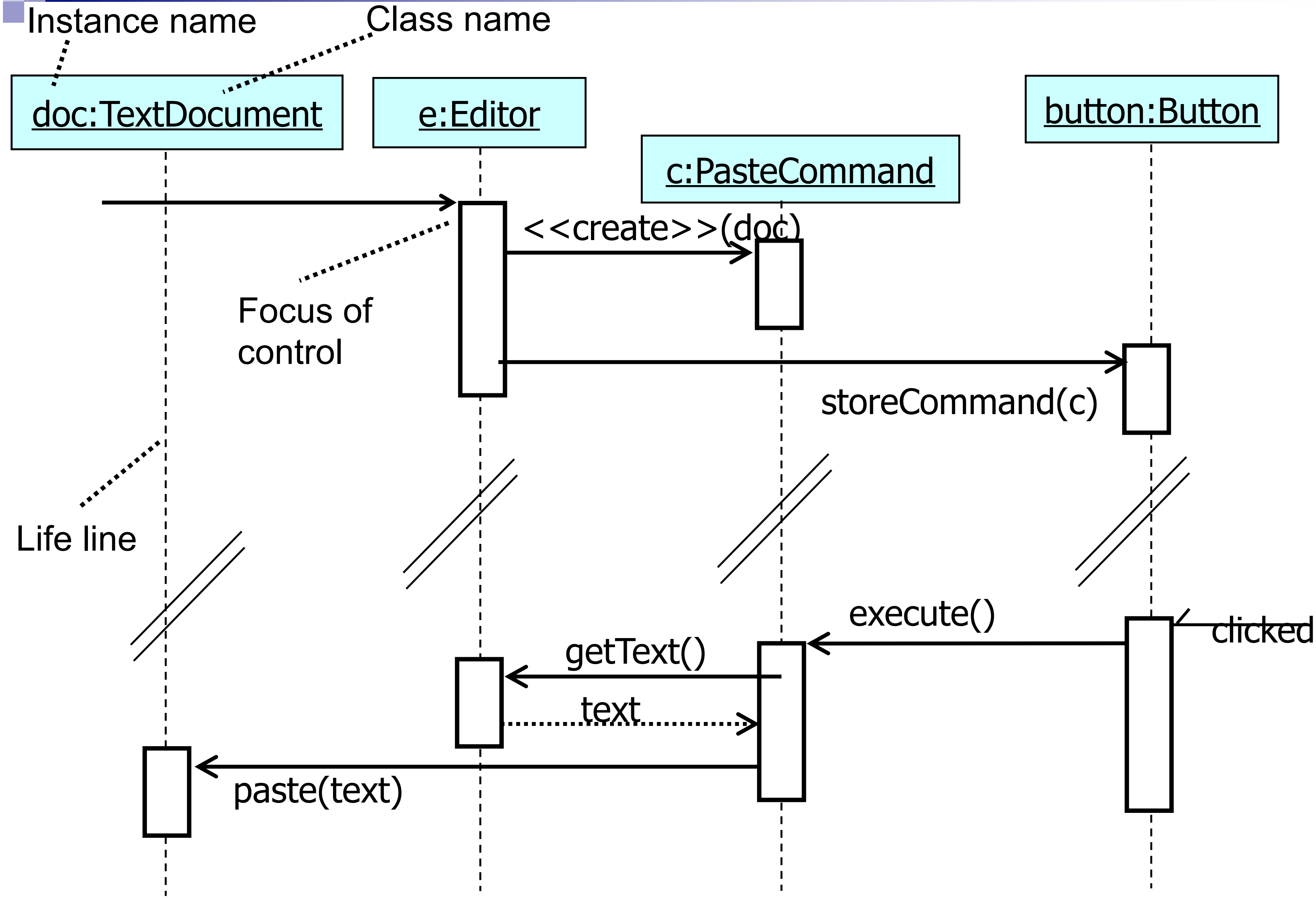
Lifeline boxes represent the participants in the interaction

Example



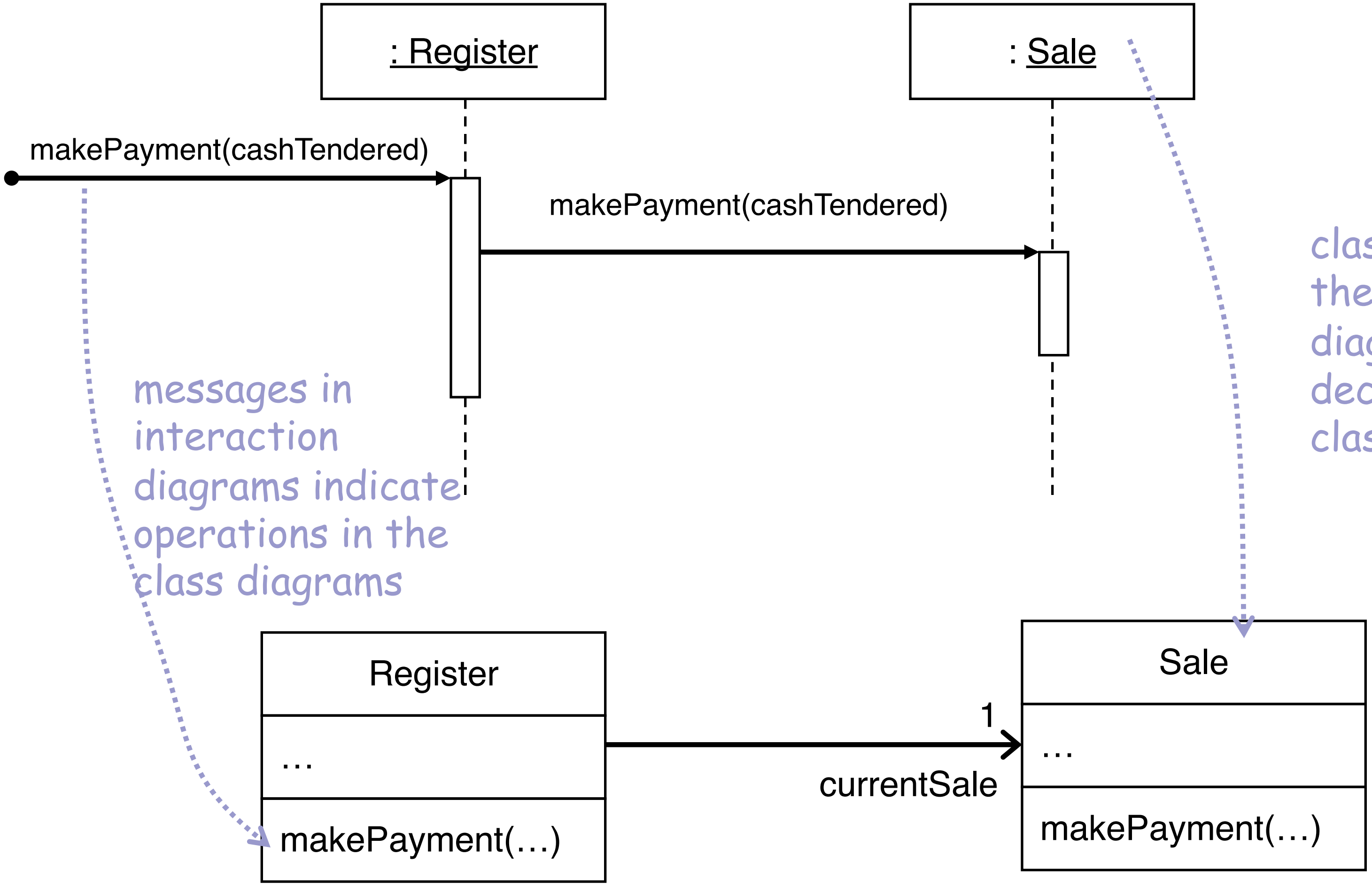
Let's describe a scenario of this system with a sequence diagram

Scenario: Issue a paste command and when clicked on the button perform the paste action (i.e. PasteCommand.execute invoked)



This sequence diagram is not in synch with the class diagram on prev page

Relationship Between Sequence & Class Diagrams



messages in interaction diagrams indicate operations in the class diagrams

classes identified in the interaction diagrams are declared in the class diagrams

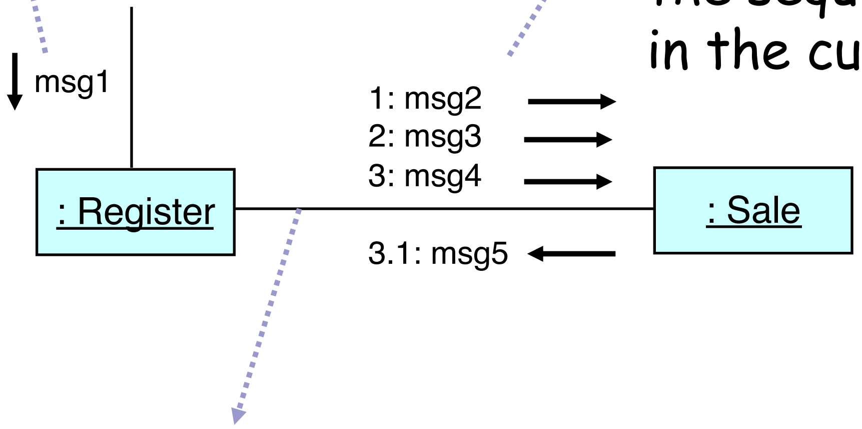
Communication Diagrams

- Collaboration diagrams in uml1
- Shows objects, links among them, interactions over each link
- Differences between sequence diagram
    - Shows links among objects besides interactions
    - More compact
        - Sometimes too compact to understand

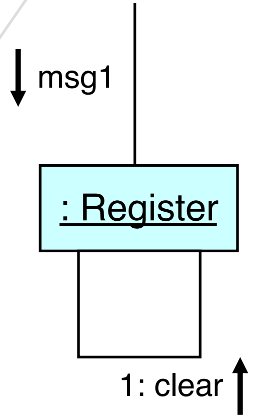
Communication Diagram

no need to number the starting message

All messages flow on the same link. Sequence number is added to show the sequential order of messages in the current thread of control.

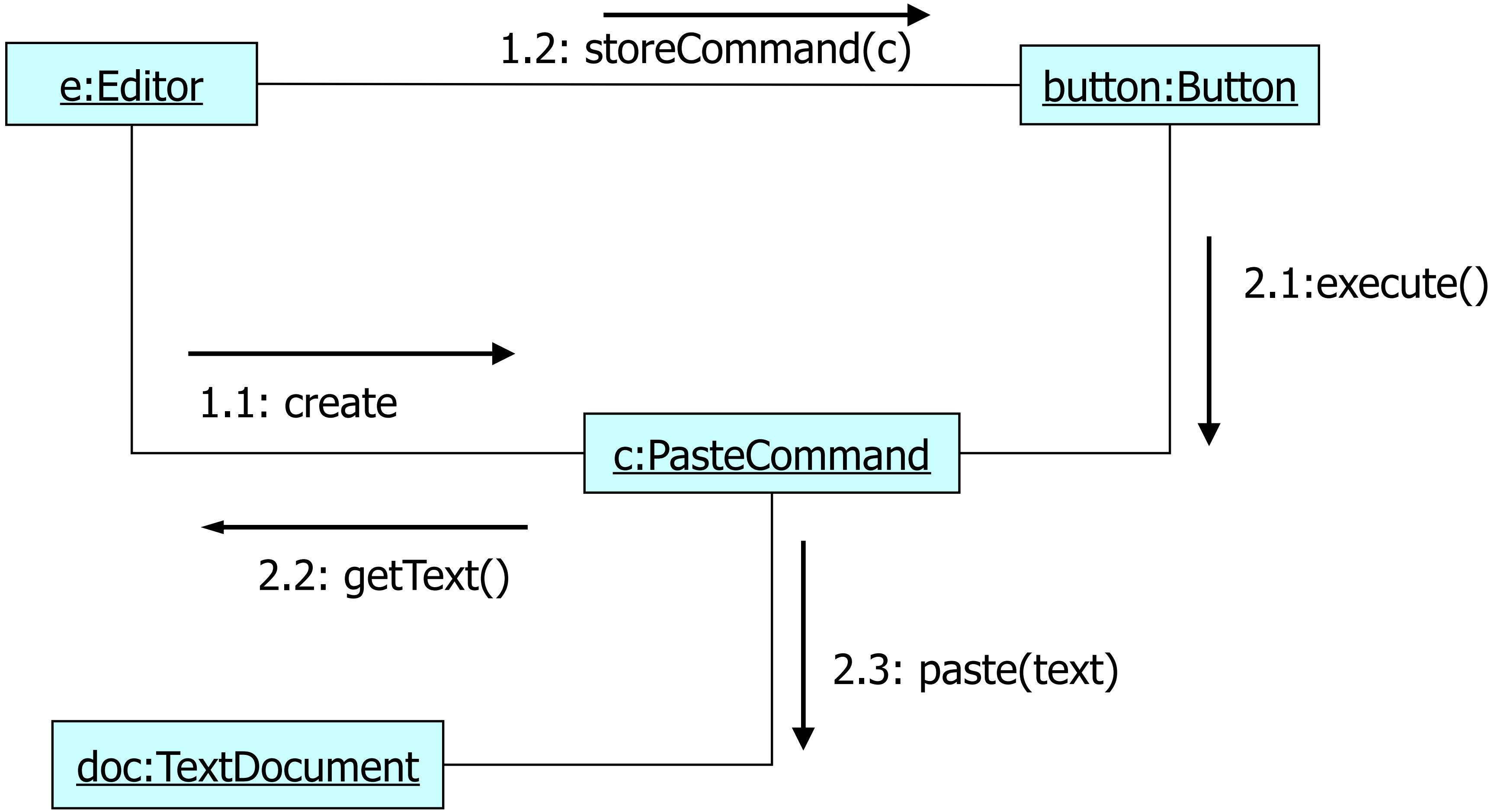


link line
a connection path between objects
informally, instance of an association



messages to this

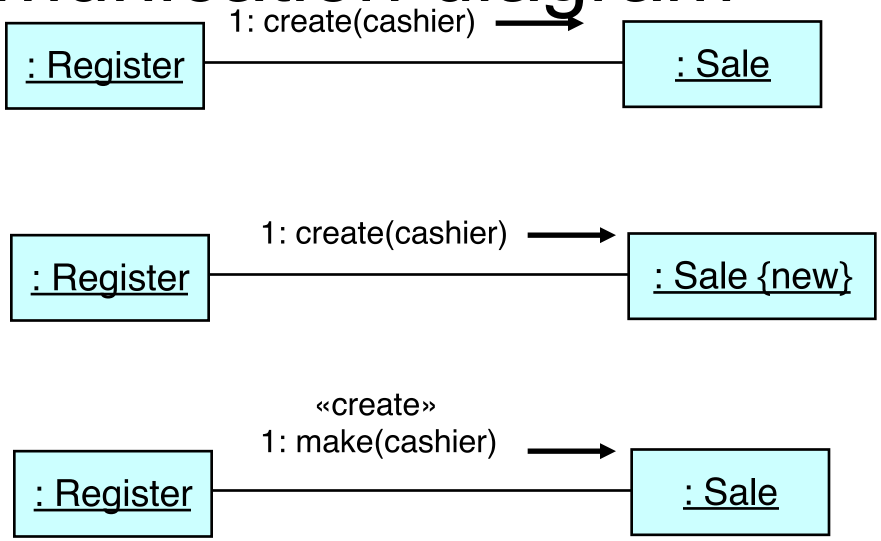
Collaboration/communication diagram



- Sequence numbers shows the time ordering
- Lines show the links between the objects

Instance Creation

- Three ways to show creation in a communication diagram



1: create(cashier)

: Register

: Sale

1: create(cashier)

: Register

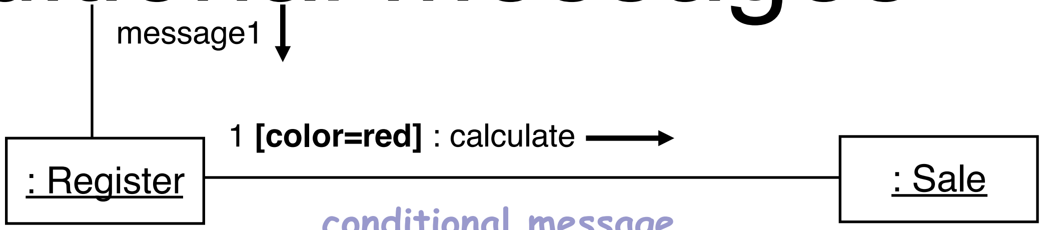
: Sale {new}

«create»
1: make(cashier)

: Register

: Sale

Conditional Messages



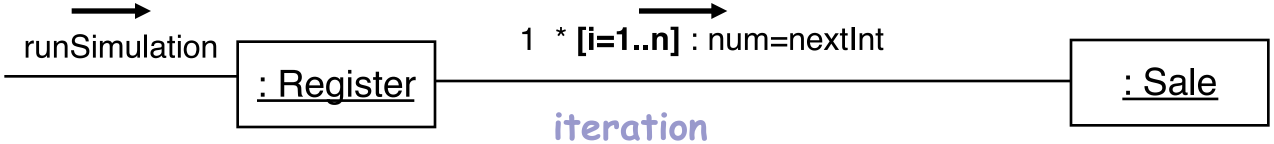
message1

1 **[color=red]** : calculate

: Register

: Sale

conditional message



runSimulation

: Register

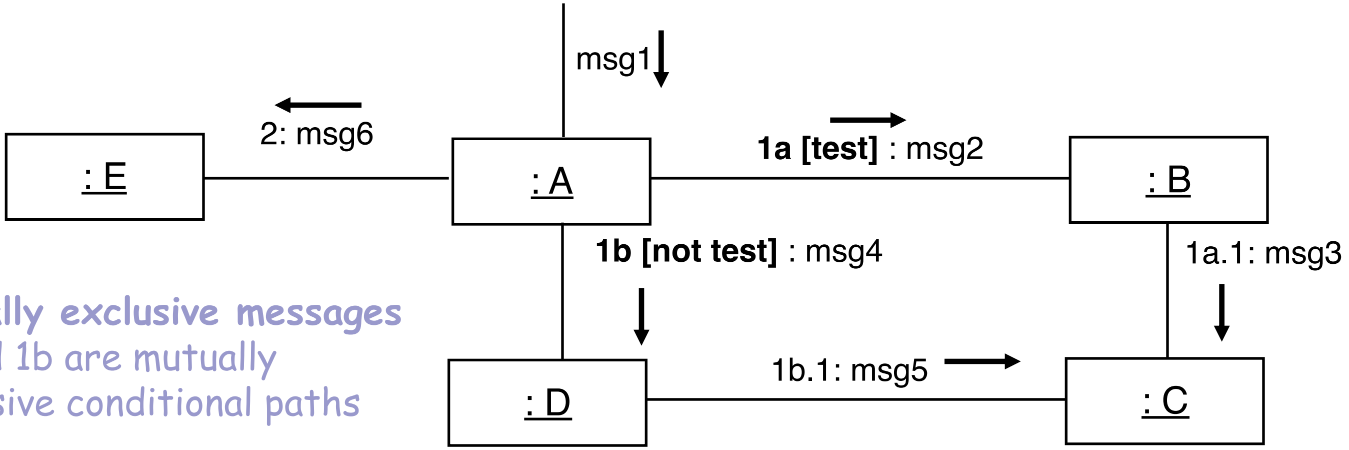
1 * **[i=1..n]** : num=nextInt

: Sale

iteration

indicated with a * and

an optional iteration clause



msg1

2: msg6

: E

: A

1a [test] : msg2

: B

1b [not test] : msg4

1a.1: msg3

: D

1b.1: msg5

: C

mutually exclusive messages

1a and 1b are mutually

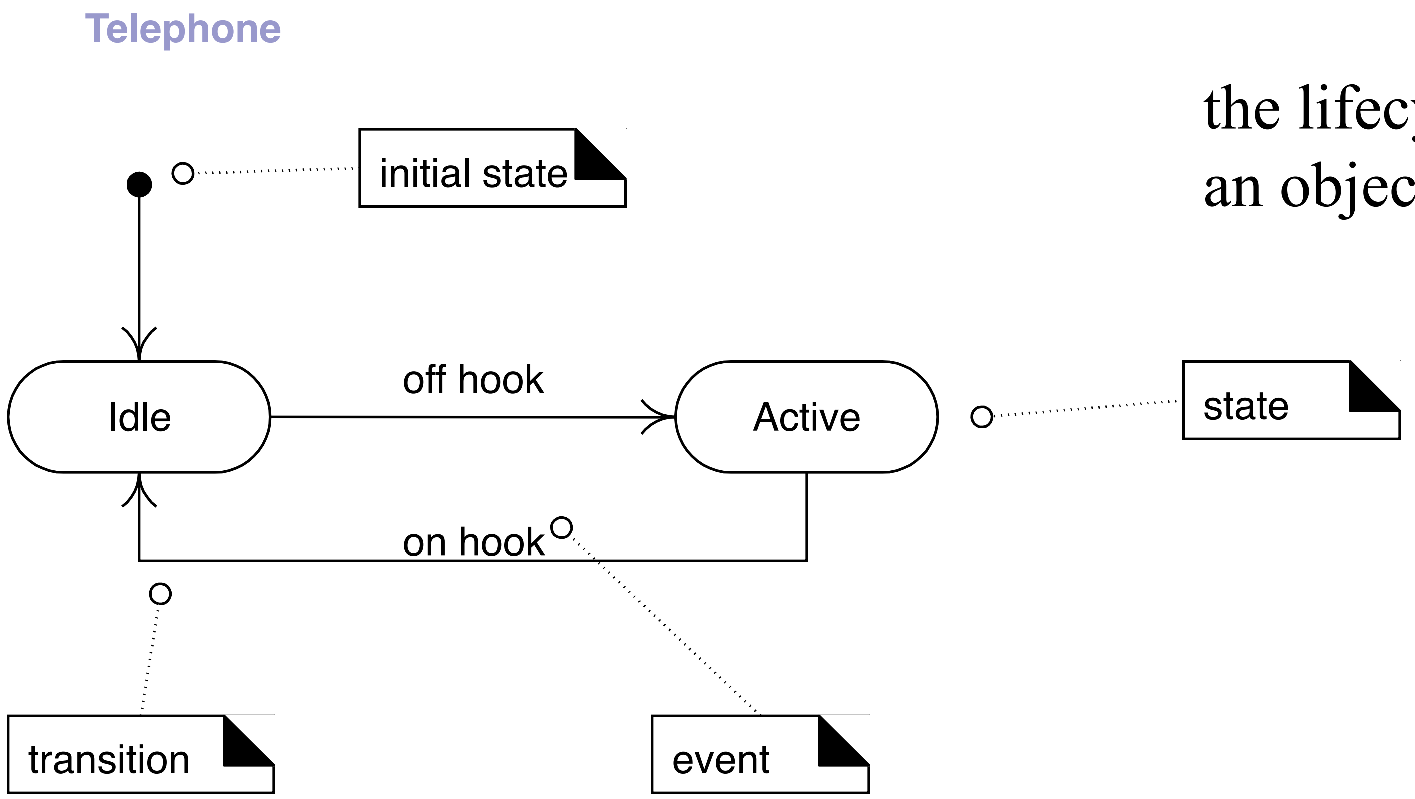
exclusive conditional paths

Strengths - Weaknesses

Type	Strengths	Weaknesses
Sequence	Clearly shows sequence or time-ordering of messages Large set of detailed notation options	Forced to extend to the right when adding new objects; consumes horizontal space
Communication	Space economical-flexibility to add new objects in two dimensions	More difficult to see sequence of messages. Fewer notation options

State Machine Diagrams

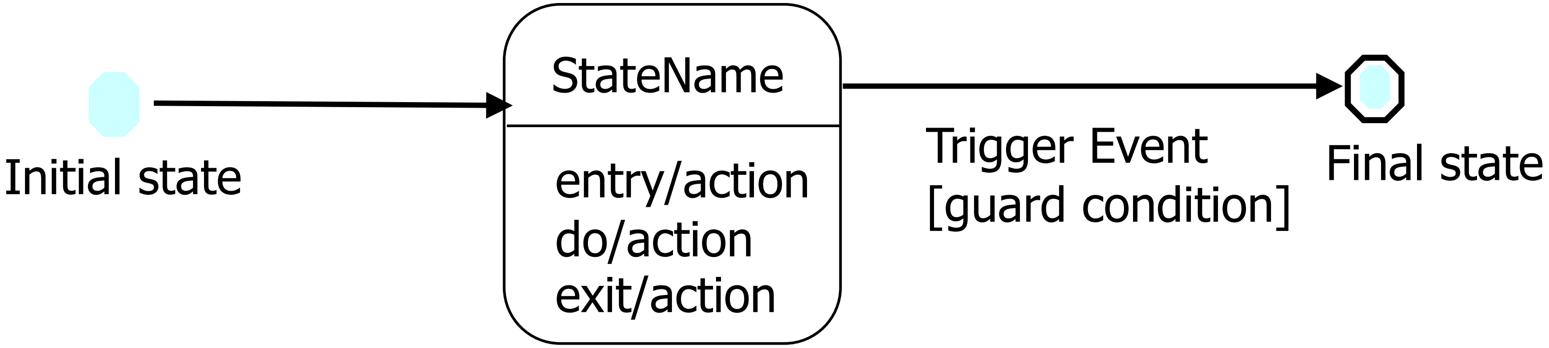
- shows the interesting events and states of **an** object, and
- Behavior of an object in reaction to an event



the lifecycle of an object

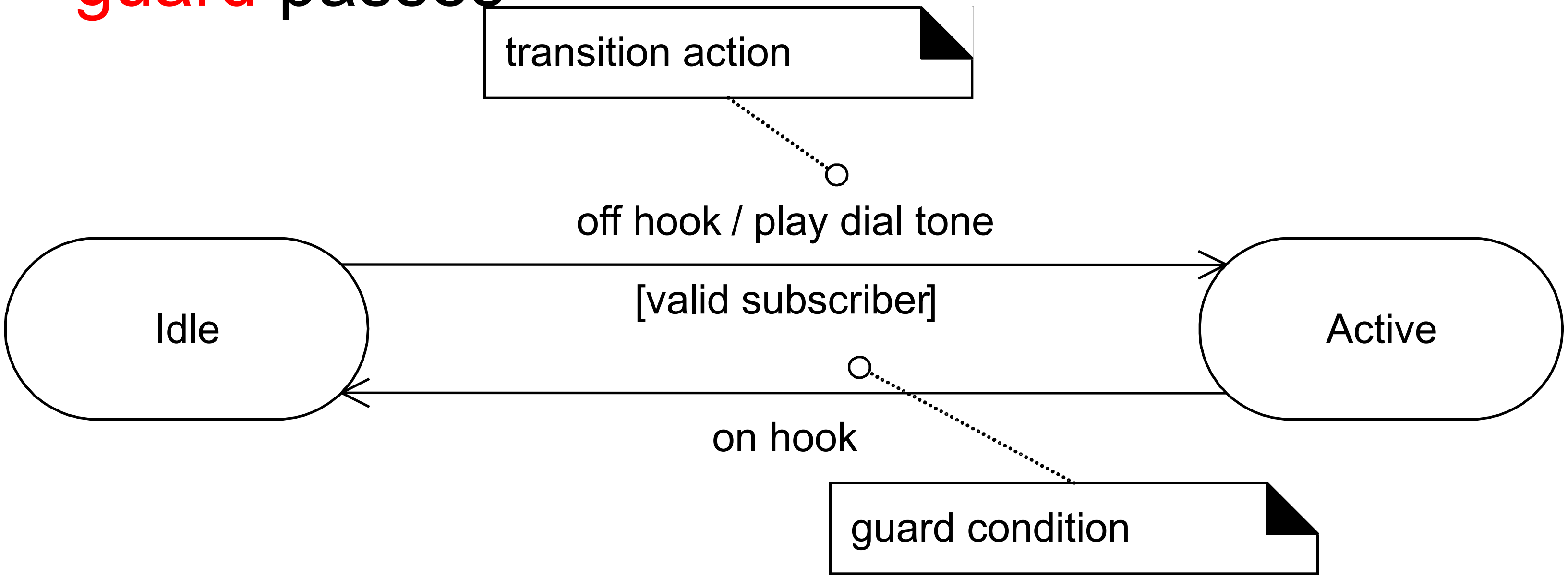
State Diagrams

- To demonstrate the behavior of an object through many use cases of the system
  - Shows states of an object (a state dependent object)
- States
  - Initial state, Final state, History state
  - Actions in a state denoted with **Entry, Exit, Do** actions
- Transitions
  - Triggered by events (e.g. receipt of a message)
  - May have guard conditions (boolean expressions)



Transition Actions and Guards

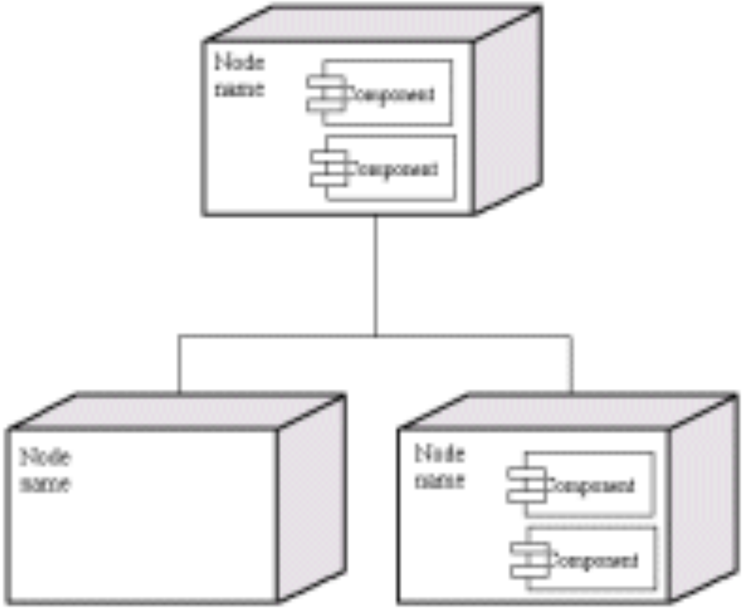
- A transition can cause an action to fire
  - fire action=May represent the invocation of a method
- The transition only occurs if the conditional guard passes



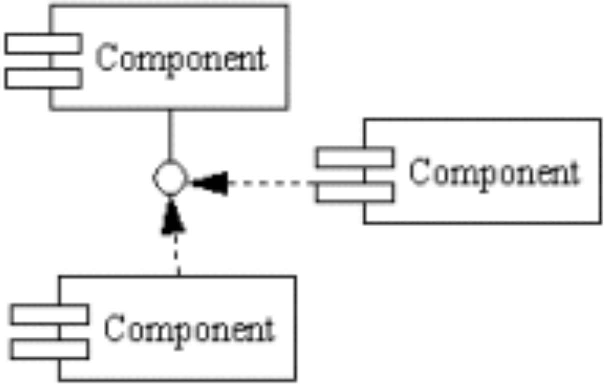
Modeling State Dependent Objects

- State Dependent Object: React differently to events depending on its state or mode
- State machines are applied in two ways:
  - To model the behavior of a complex reactive object in response to events
    - Physical devices controlled by software (e.g. phone)
    - Transactions and related business objects
    - Role mutators (e.g. person changing from employee to retired)
  - To model *legal sequences* of operations – protocol or language specifications
    - Communication protocols (TCP)
    - UI page/window flow or navigation (client side)
    - UI flow controllers or sessions (server side)
    - Use case system operations
    - Individual UI window event handling

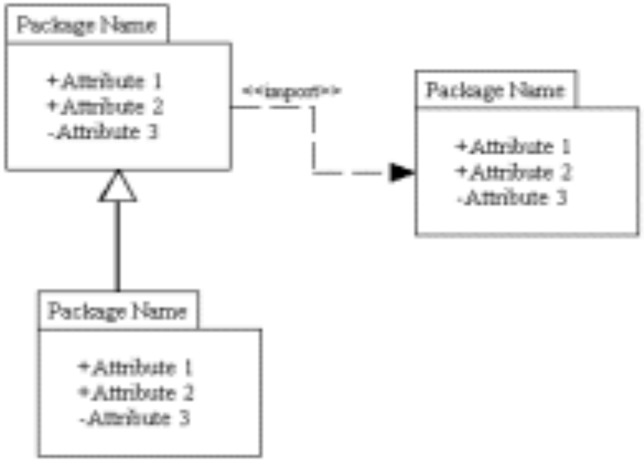
Sample other Diagrams



Deployment diagram



Component diagram



Package diagram