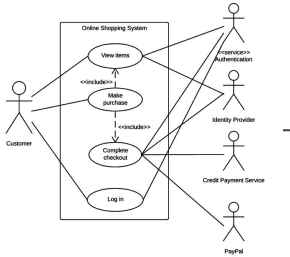
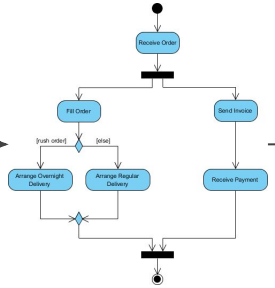
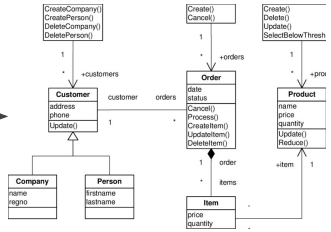
Software Analysis 2

Software Engineering
Prof. Maged Elaasar

Analysis Method with UML Diagrams



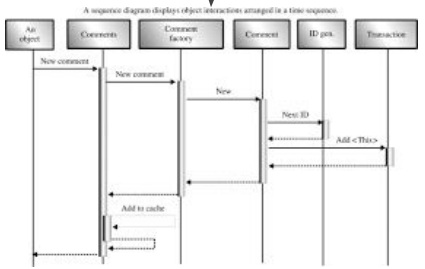




1. Identify system boundary, actors, external systems, and use cases with use case diagrams

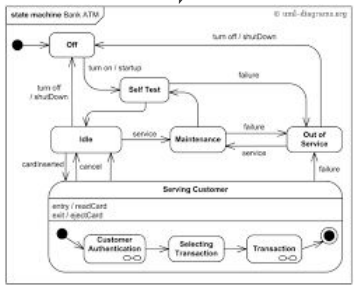
2. Identify actions for every system, and the control / data flow between them with activity diagrams

3. Identify classes and interfaces with their attributes, operations and relations with class diagrams



4. Capture how these entities interact with each other via messages using sequence diagrams

5. Flesh out the internal behavior of complex entities with state machine diagrams



3. Class Diagram

Class Diagram

1. Describes the white box implementation of a system in terms of types (classes, interfaces, enumerations and primitive types)

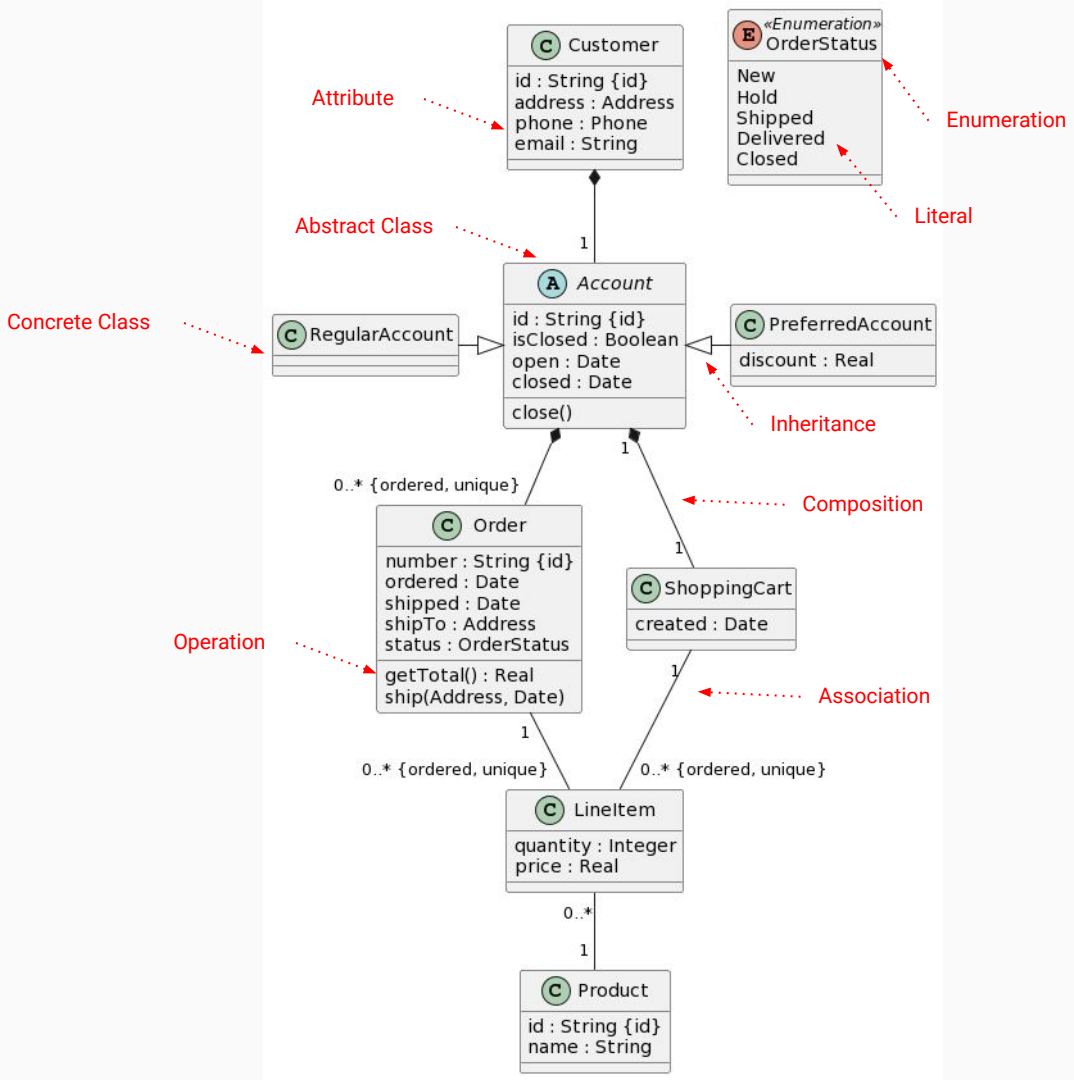
2. A type is defined by its structural features (attributes/fields) and behavioral features (operations/functions)

3. A set of relations can be described between the different types

4. Class diagrams can be created at different levels of abstraction (architecture, design, or implementation)

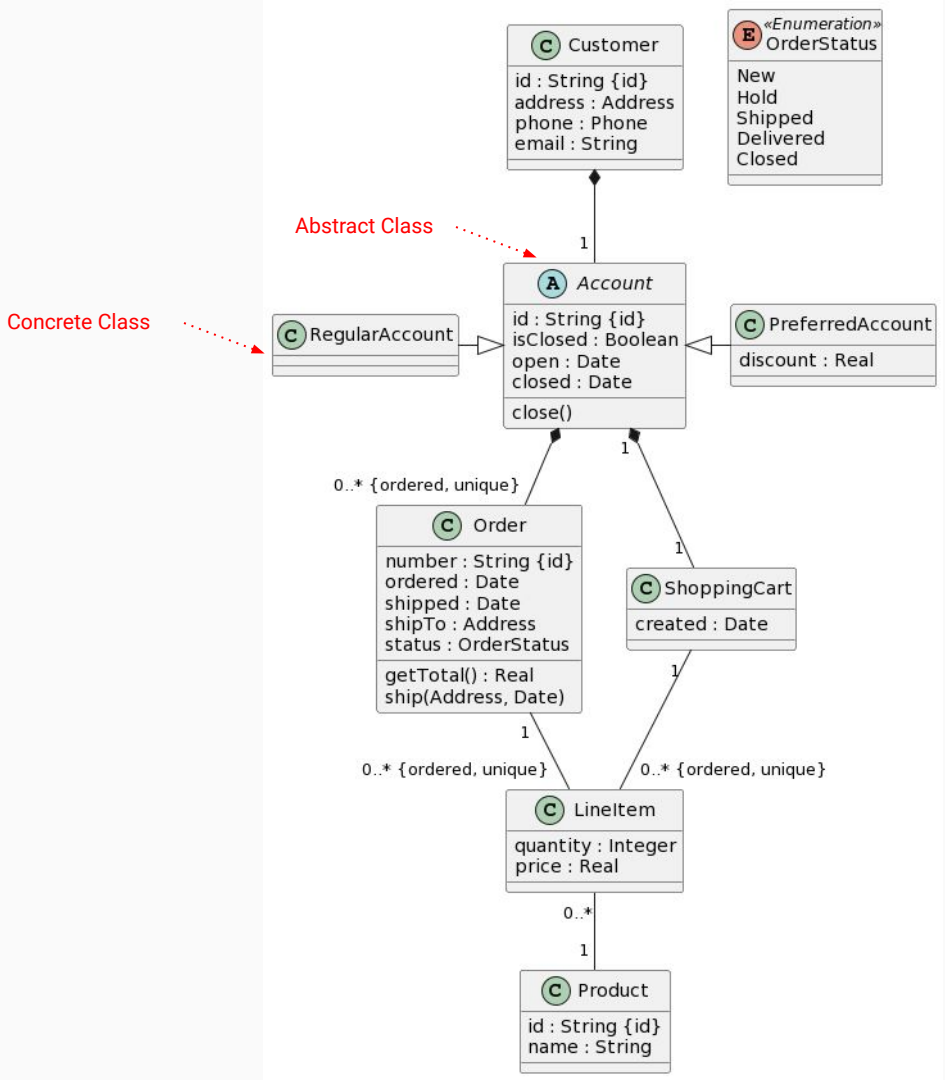
Key Elements

- Classifier
  - Abstract Class
  - Concrete Class
  - Enumeration
- Features
  - Attribute
  - Operation
  - Literal
- Relations
  - Inheritance
  - Association
  - Composition



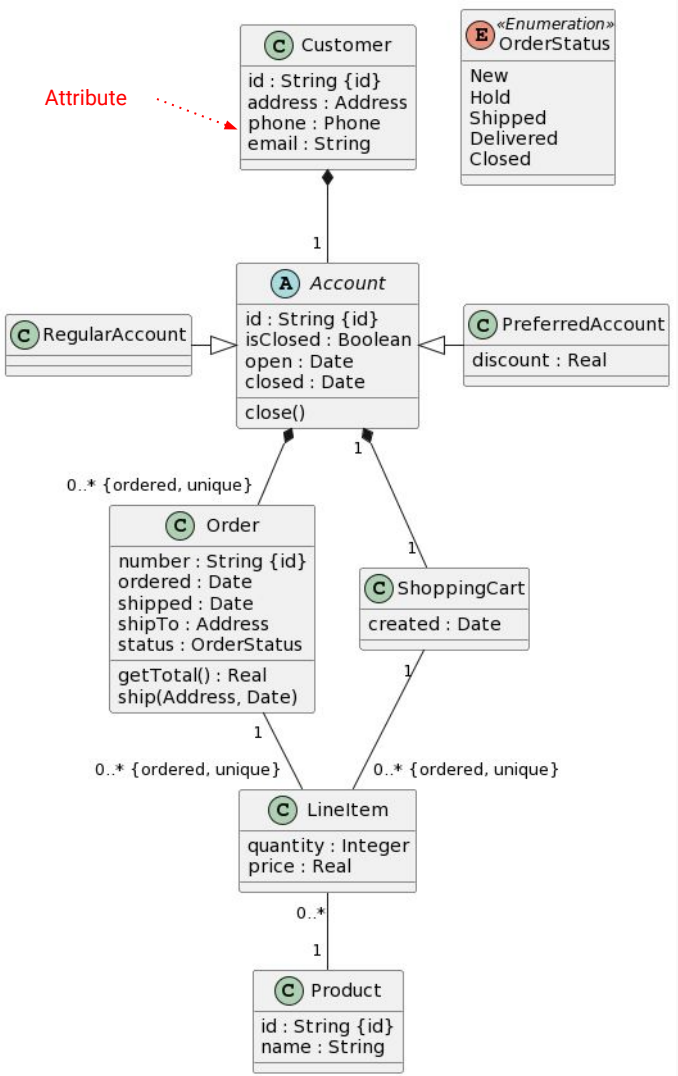
Class
(type of objects)

- Abstract Class
  - Cannot be instantiated
  - Name is in italic font
- Concrete Class
  - Can be instantiated
  - Name is not in italic font



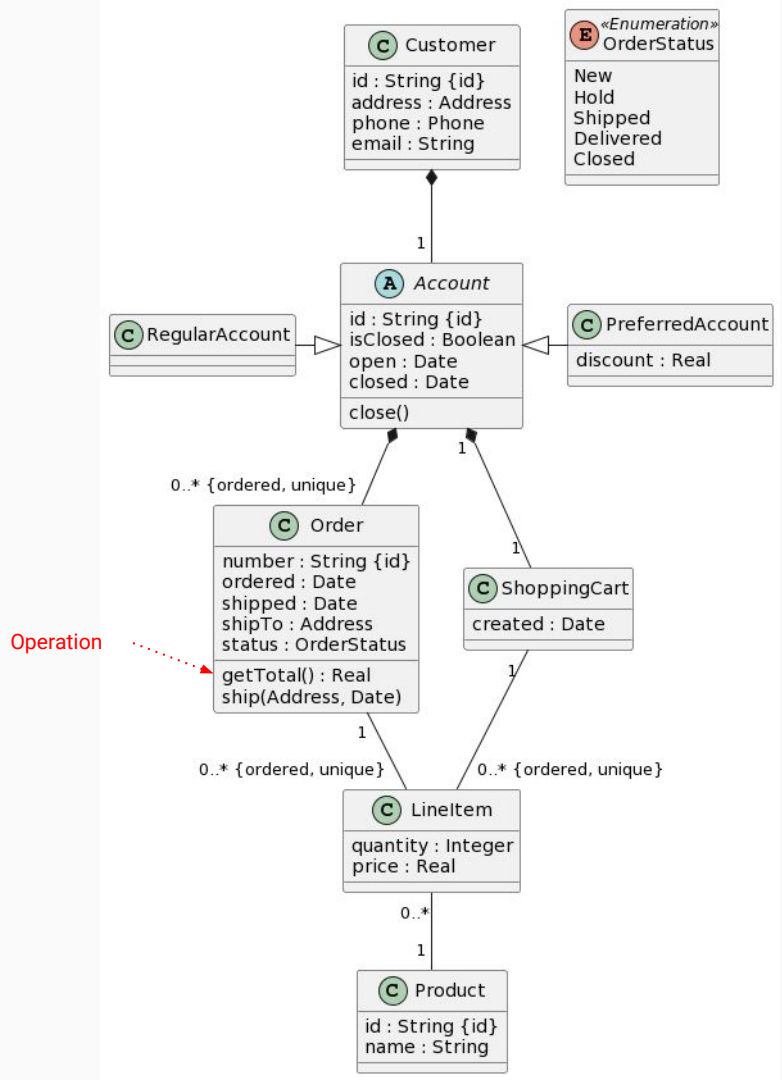
Attribute

- A structural feature of a Class
- Syntax: **name : Type [multiplicity] {modifiers}**
- Type can be primitive (String, Integer, Boolean, Rea, Date) or an enumeration
- Multiplicity can be exact (e.g., [1], [*]) or has [lower..upper] bounds (e.g., [2..4], [1..*])
- Modifiers can be comma-separated with possible values:
  - id (globally unique)
  - unique (a set)
  - ordered (a list)



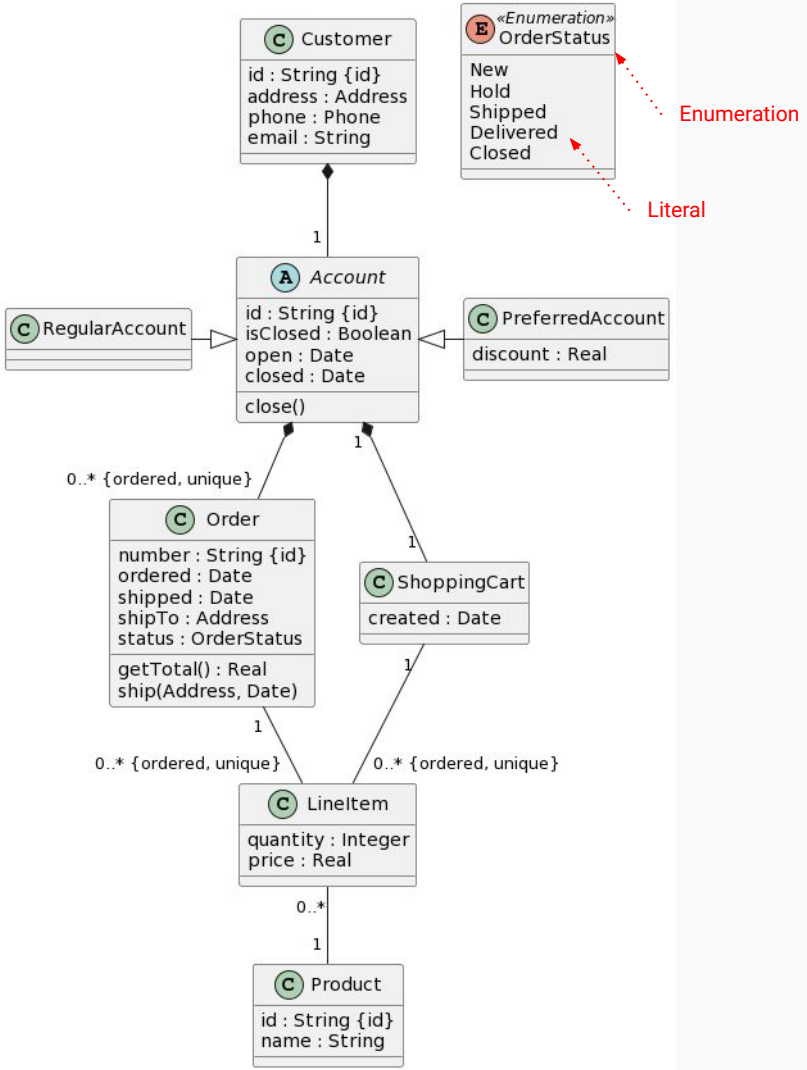
Operation

- A behavioral feature of a Class
- Syntax: **name (parameters) : Type [multiplicity] {modifiers}**
- Parameter have syntax **name : Type [multiplicity]**
- Type can be primitive (String, Integer, Boolean, Rea, Date) or an enumeration
- Multiplicity can be exact (e.g., [1], [*]) or has [lower..upper] bounds (e.g., [2..4], [1..*])
- Modifiers can be comma-separated with values {unique, and/or ordered}



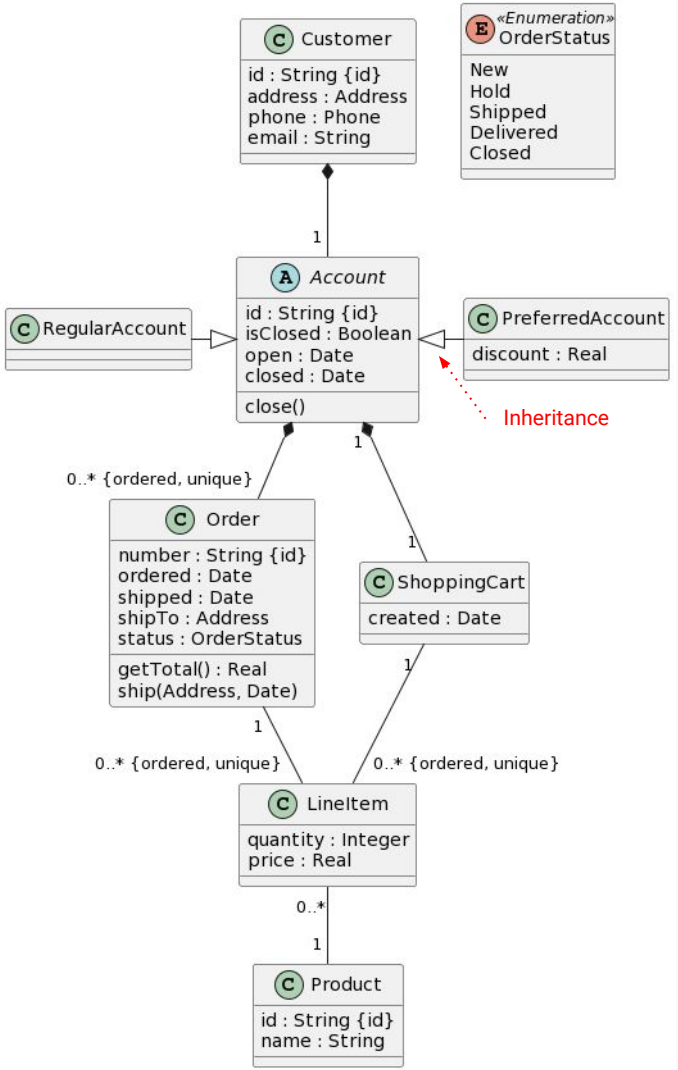
Enumeration (type of Literal)

- Defines a set of enumerated literals



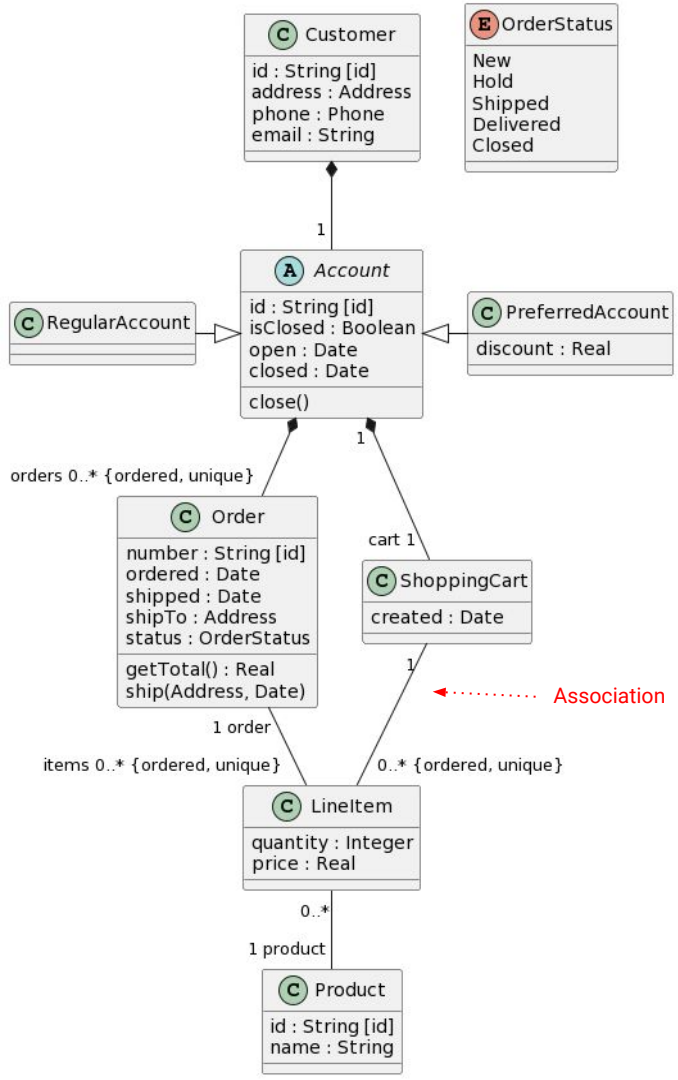
Inheritance (is-a)

- A relation from a subclass to a superclass
- Features of a superclass are visible in a subclass
- Notated with a solid line with a closed arrow



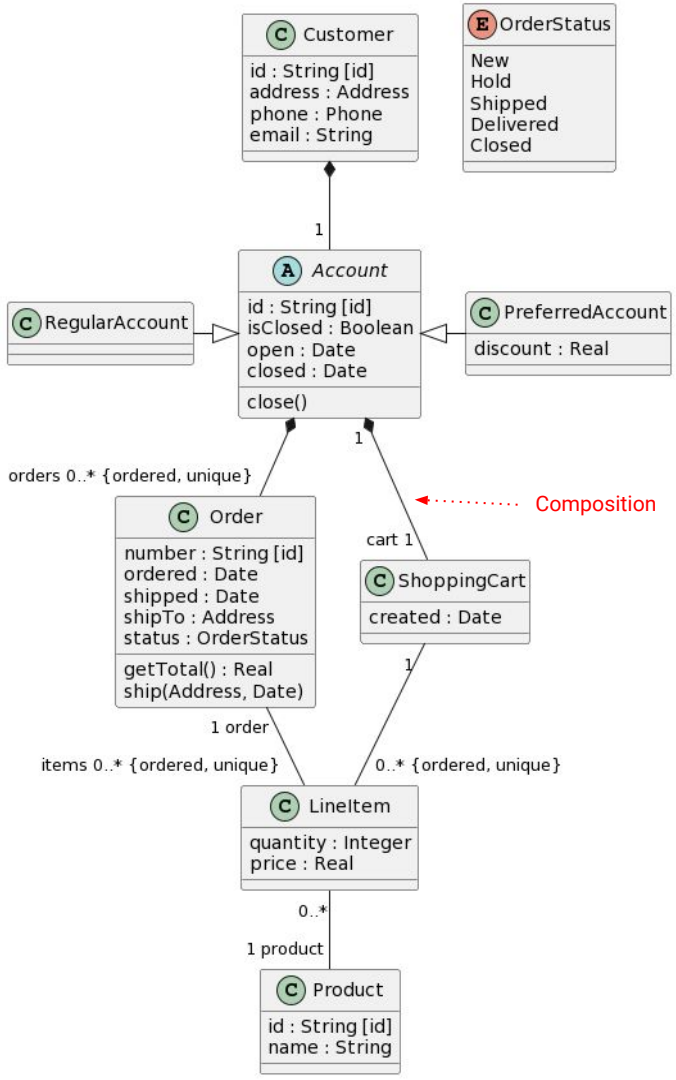
Association (has-a)

- A relation between classes that represents that one class has an attribute typed by another class
- Notated as a solid line with either no arrow (bidirectional) or an open arrow (unidirectional)
- An association end near one class can show the attribute that is typed by it but owned by the class at the other end
- The association can additionally have its own name



Composition

- An association where one end is a whole and the other end is a part (composed within that whole)
- Deleting an instance of a whole class deletes all instances of the part (composed within the whole)
- Notated similar to an association but the end near the whole class has a filled diamond

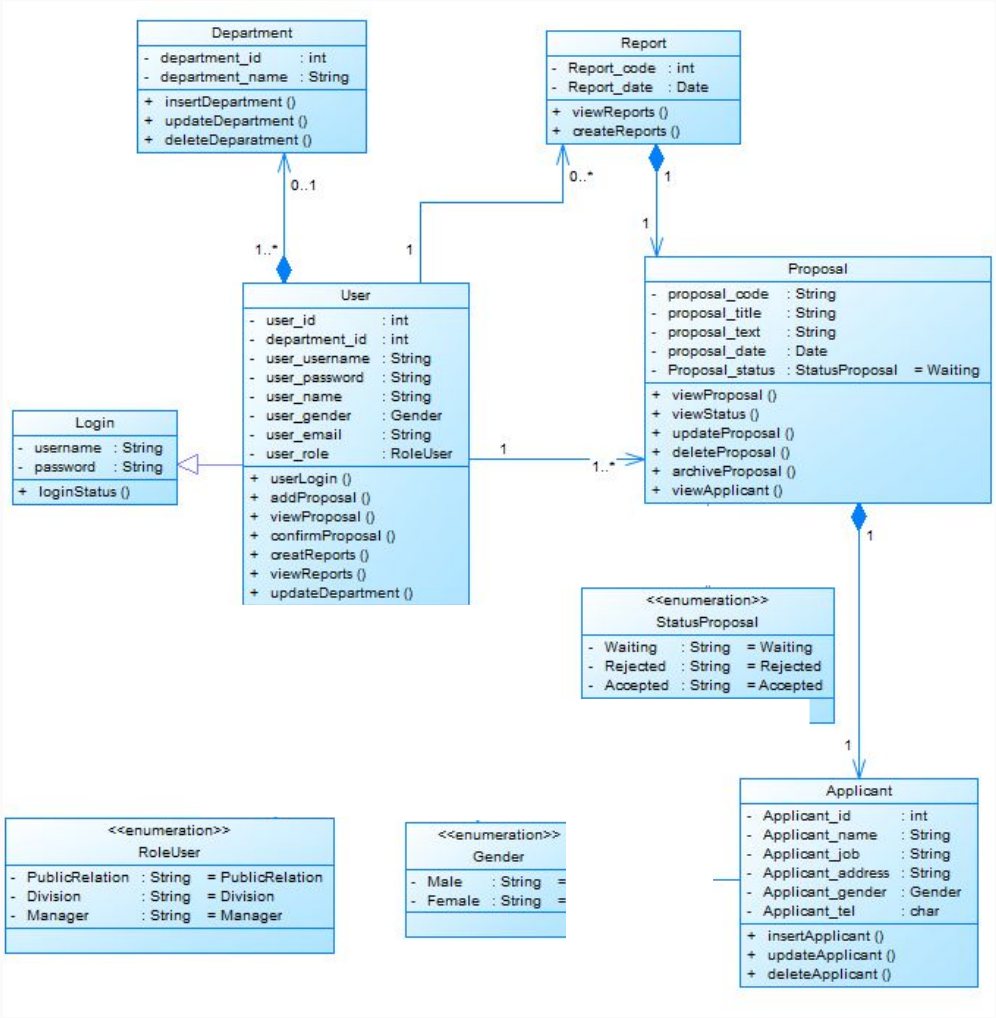


draw a class diagram

- The system shall include a representation of customer orders.
- Each order will have a single associated customer, and each customer can have multiple orders. Note that a customer is not required to have any orders.
- Each order will have at least one, and possibly multiple line items. Each line item is uniquely associated with a single order.
- Each line item represents a single product. Note that a product is not required to be represented in a line item. A product can be represented in multiple line items (even within the same quote).

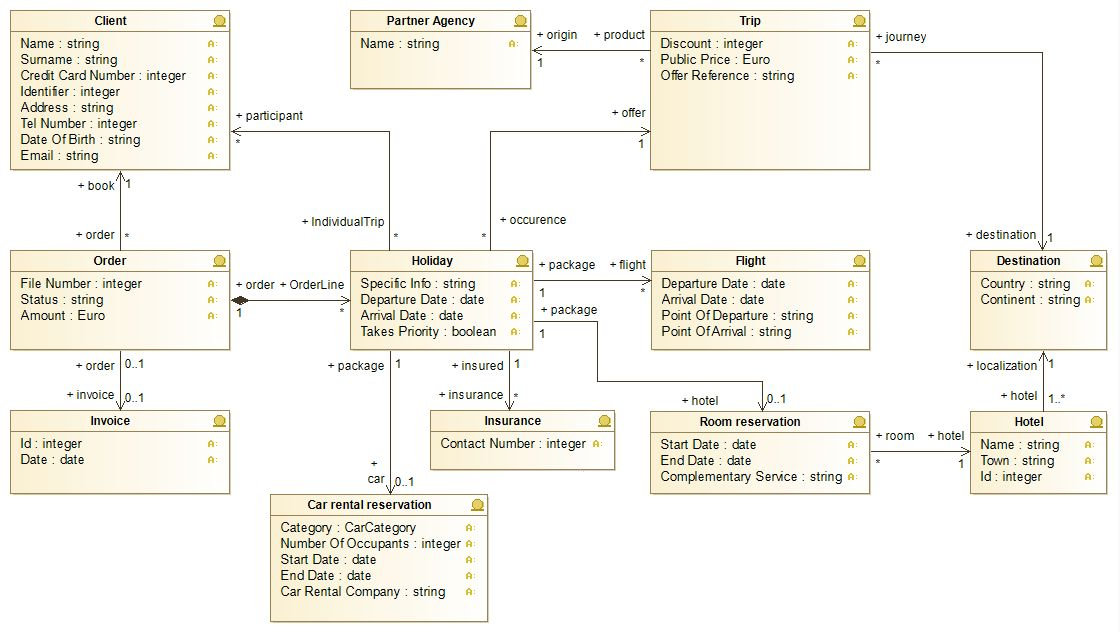
Read class diagram

- How many proposal can a User make?
- Does a user have to have a report?
- Is a User a Login or does a User have a login?
- Can a proposal change status?
- What else get deleted when a report gets deleted?



Read class diagram

- Can an order have multiple car rental reservations?
- Can a holiday have more than one hotel reservation?
- Can an order have multiple participating clients?



Class Diagram Quiz

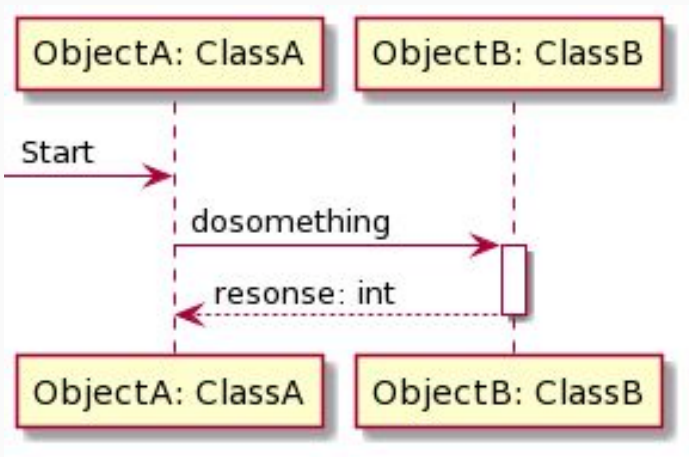
4. Sequence Diagram

Sequence Diagram

- Describes an interaction scenario among objects in a system
- The scenario is shown by an exchange of messages
- Messages are ordered in time (which flows downward)

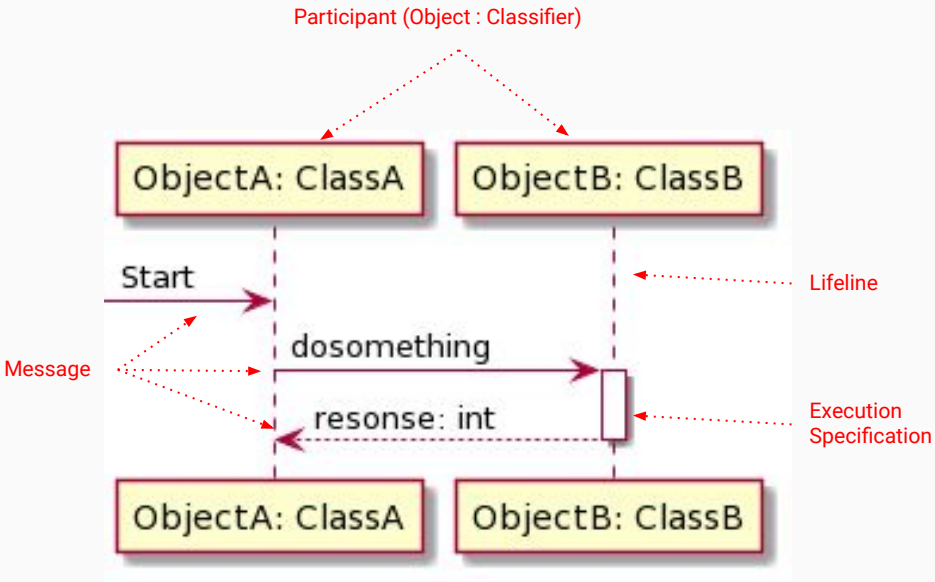
Sequence Diagram

- Captures the behavior in an scenario involving two or more objects



Key Elements

- Participants
- Lifelines
- Messages
- Execution Specifications
- Fragments (next slide)



Fragments

Condition

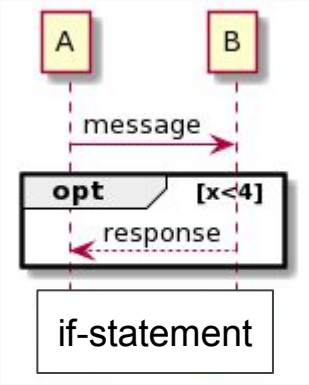
- opt: option
- alt: alternatives

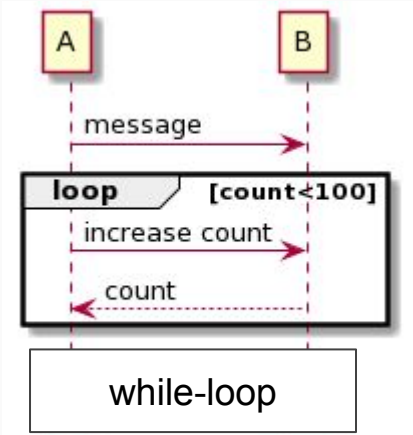
Loop

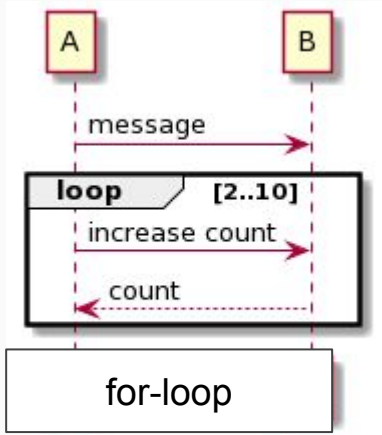
- loop [condition]
- loop [n] or loop [n..m]

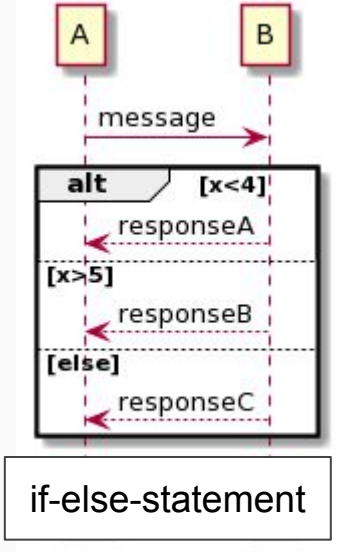
Parallel

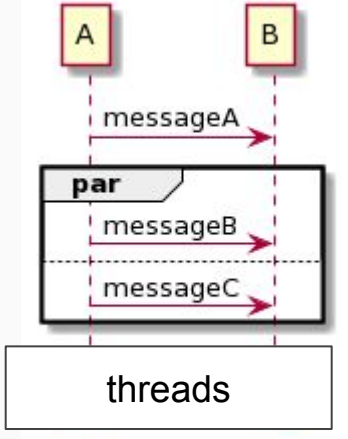
- par



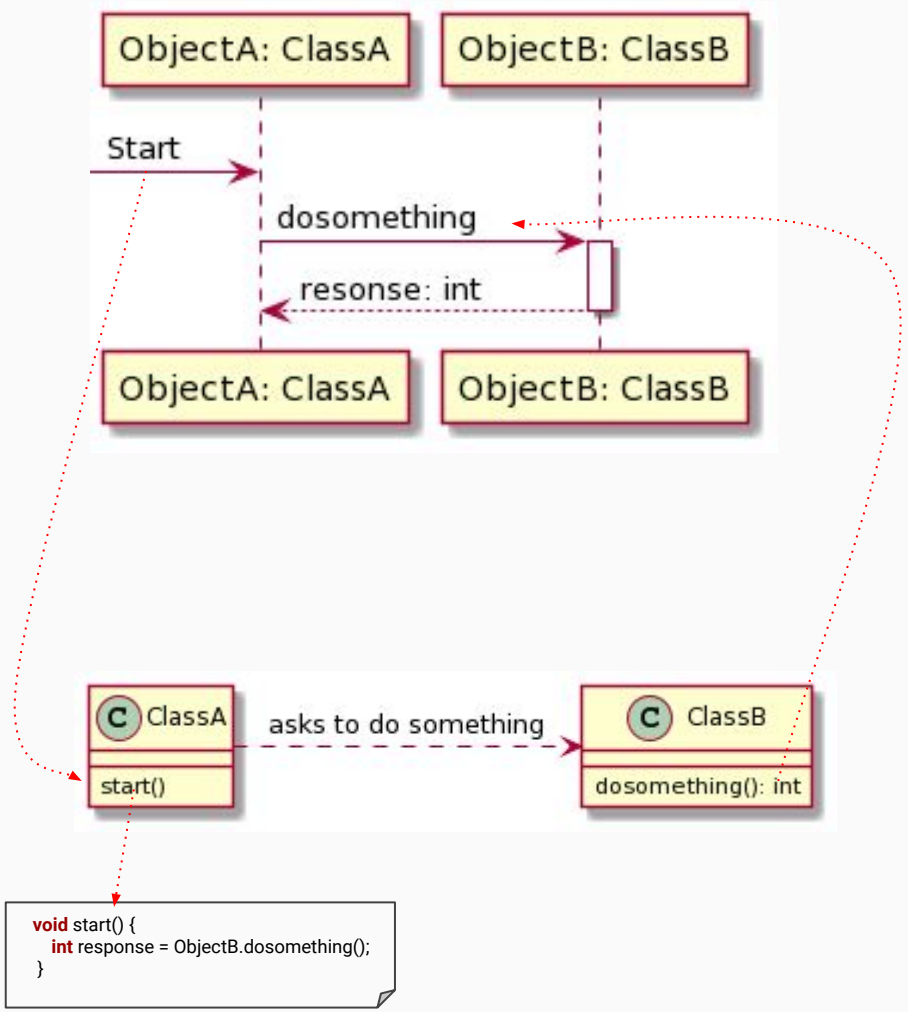






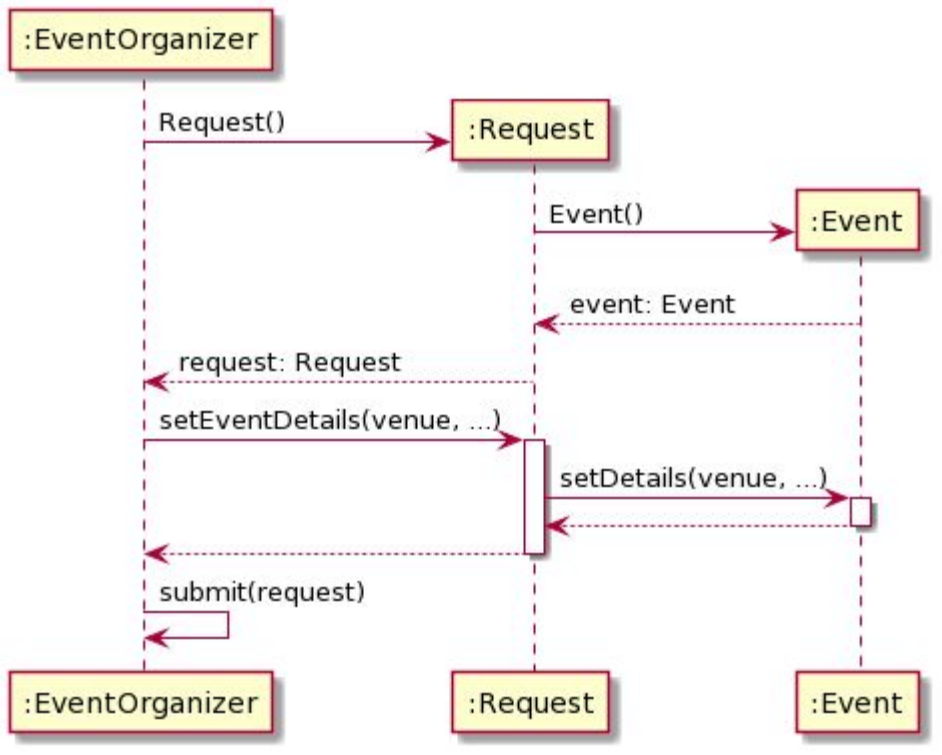


Sequence Diagram to Class Diagram

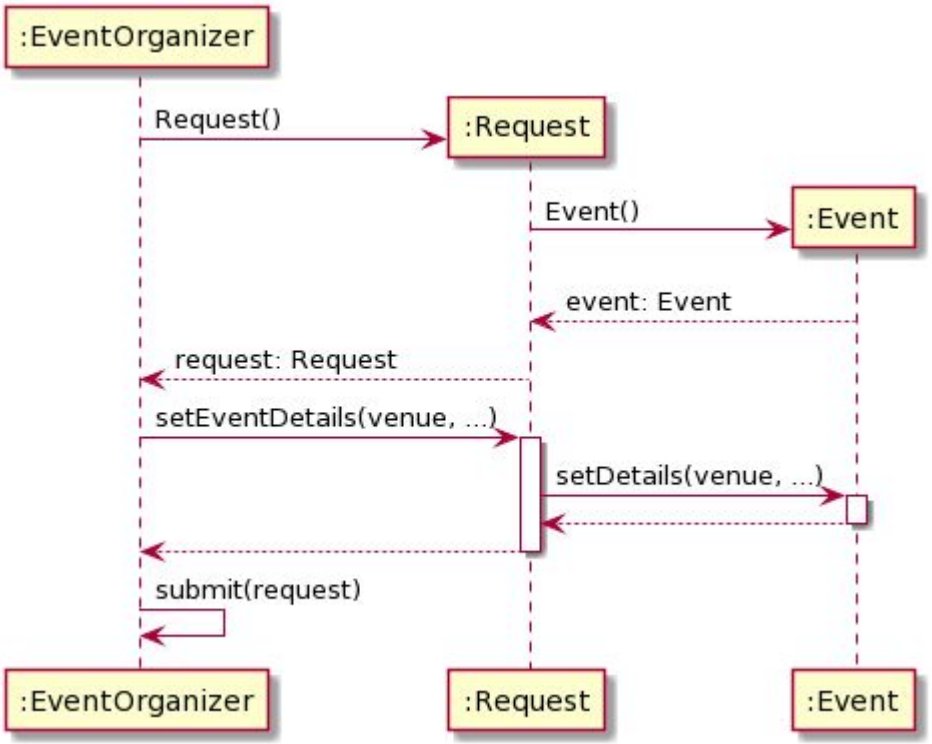


Example

An event organizer creates a request for a new event. She then specifies the event details (like venue, number of people, etc.) and submit it.



Example: Sequence Diagram to Code

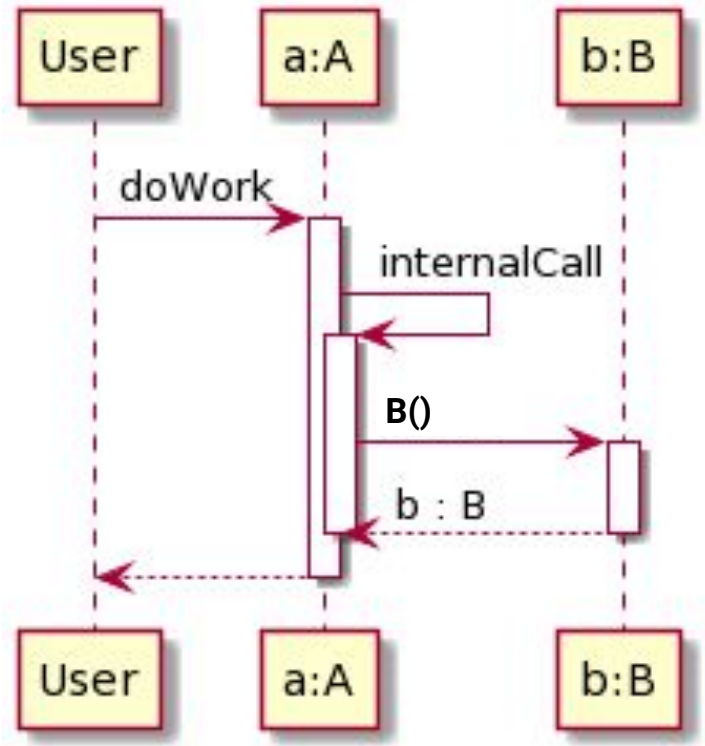



```
class EventOrganizer {
  void method() {
    Request request = new Request();
    request.setEventDetails(venue..);
    submit(request);
  }
  void submit(Request request)  {..}
}

class Request {
  public Request() {
    event = new Event();
  }
  void setEventDetails(Venue venue)  {
     event.setDetails(venue, ...);
  }
}

 class Event {
   public Event() {...}
    void setDetails(Venue venue)  {...}
 }
```

Example: Sequence Diagram to Code

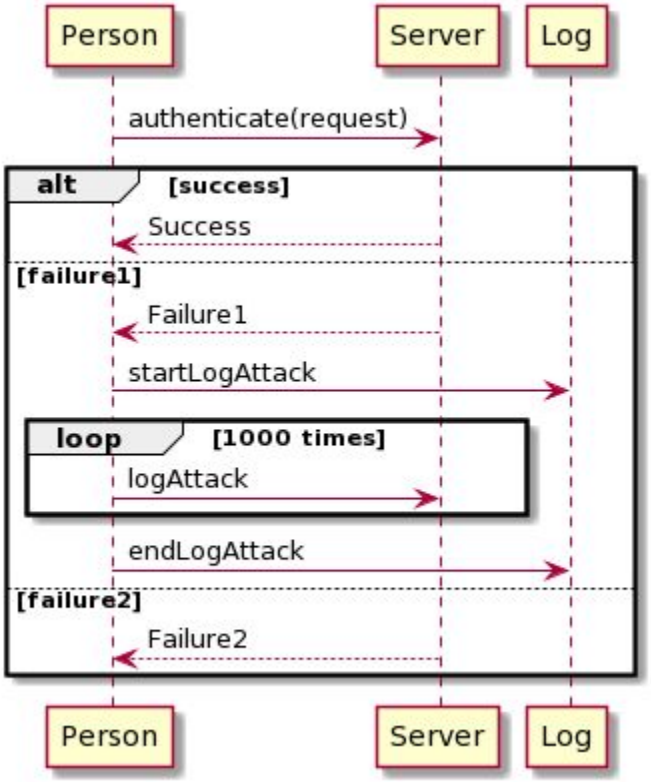



```
class User {
  void method() {
    a.doWork();
  }
}
```

```
class A {
  void doWork() {
    internalCall();
  }
  void internalCall() {
    B b = new B();
  }
}
```

```
class B {
  B() {}
}
```

Example: Sequence Diagram to Code




```
class Person {
  void method() {
    switch(Server.authenticate(request()) {
      case Success: break;
      case Failure1: {
         log.startLogAttack();
         for (int i=1; i<=1000; ++)
            server.logAttack();
         log.endLogAttack();
         break;
      }
      case Failure2: break;
   }
 }
```

```
class Server {
  Result authenticate(Request request) {
    if (success) return Success;
    if (failure1) return Failure1;
    if (failure2) return Failure2;
  }
   void logAttack() {}
}
```

```
class Log {
   void startLogAttack() {}
   void endLogAttack() {}
}
```

In Class Activity

Which traces are possible in the following sequence diagram?

[ ] c → a → b → d → e

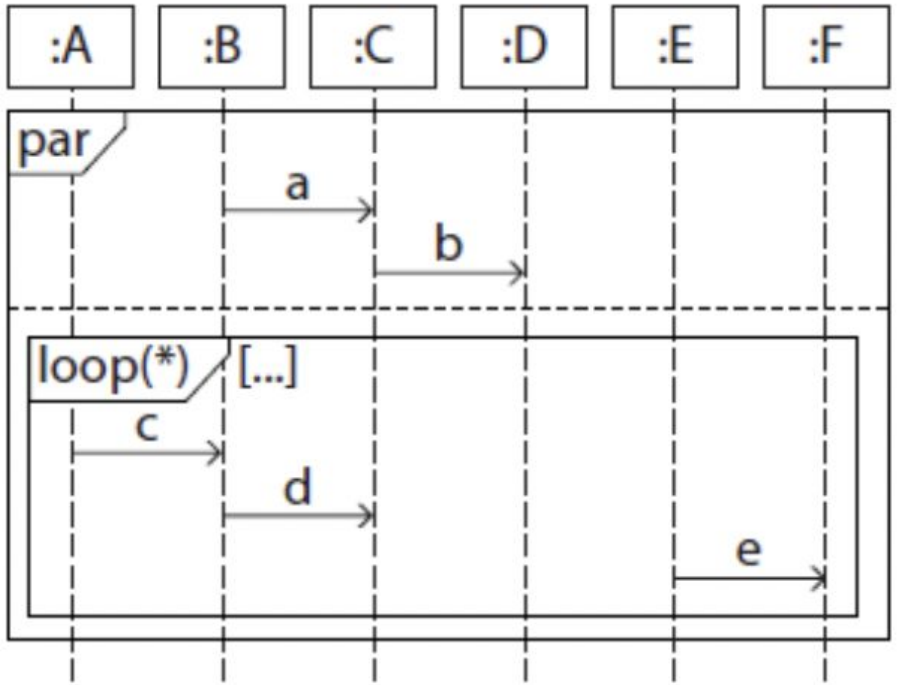
[ ] b → a

[ ] c → d → e → a → b

[ ] a → c → d → e → b

[ ] a → b → c → d → e → a → b

[ ] a → c → d → b → e



Sequence Diagram Quiz

5. State Machine Diagram

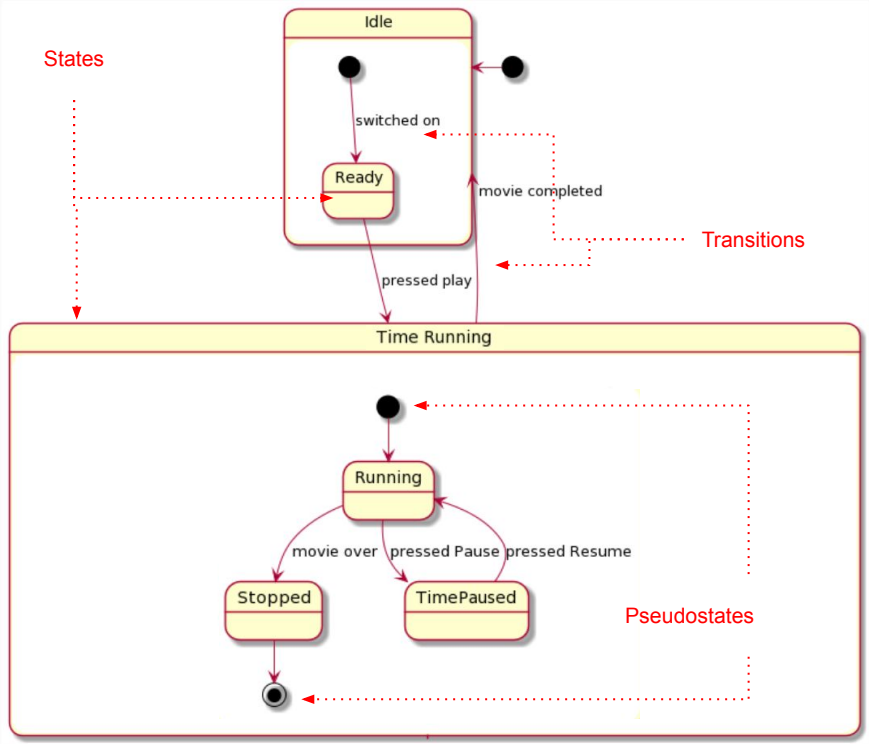
State Machine Diagrams

- Describes the states and their transitions within an entity (system or class)
- Describes the internal event-driven behavior within an entity as opposed to the interaction between entities
- Not all classes have behavior that need be described as a state machine

Key Elements

- States
- Transitions
- Pseudostates

Movie Player State Machine



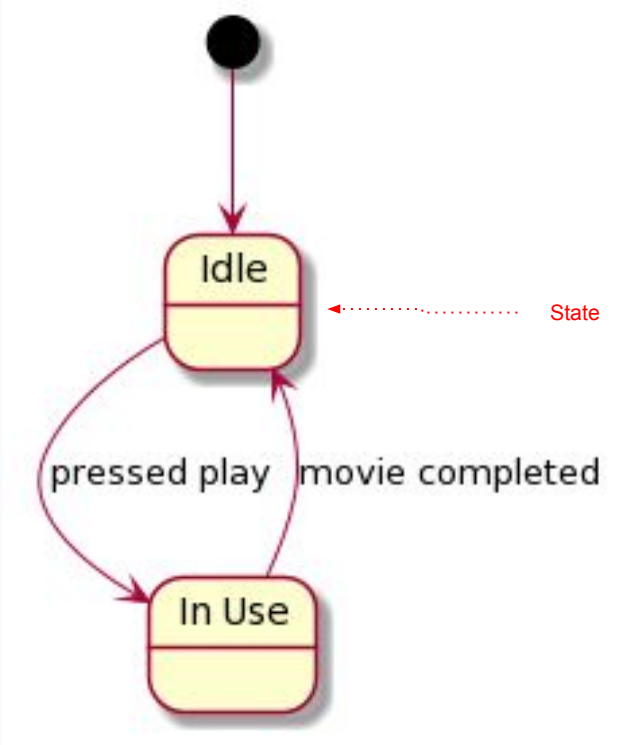
State

Represents a state of an entity being modeled

Types

- Simple
- Composite

Movie Player State Machine

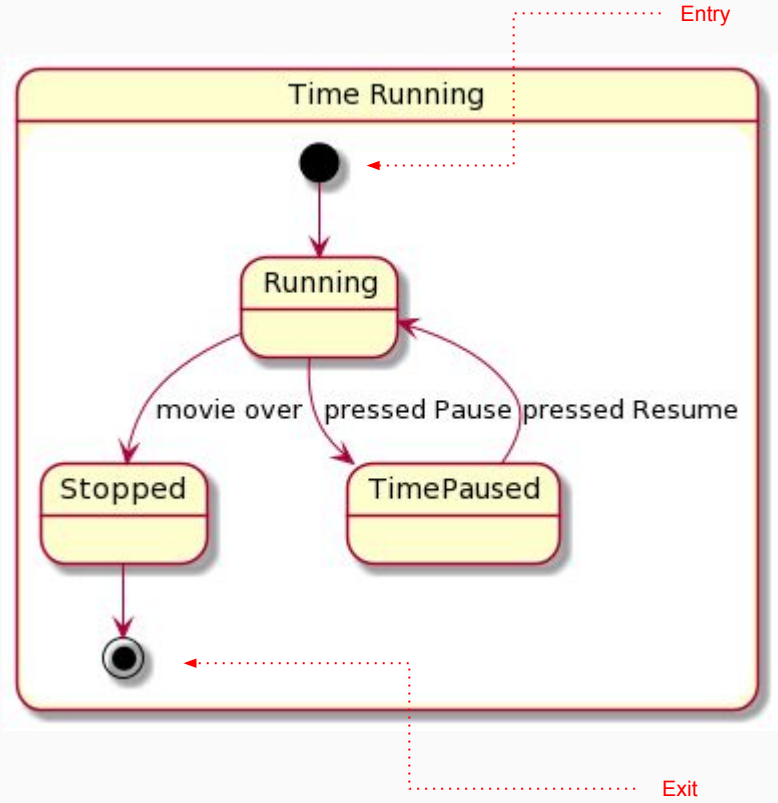


Pseudostate

A transient vertex in a state machine

Types

- **Enty**
- **Exit**

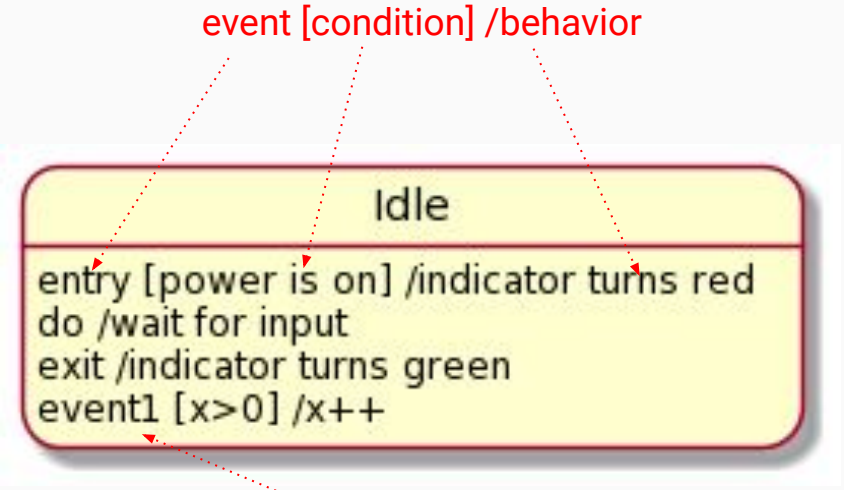


Simple State

A state that does not have sub states

Activities Compartment

- **entry**: behavior upon entry
- **do**: ongoing behavior
- **exit**: behavior upon exit
- **<event>** : some event while in do




```
void Idle() {
  entry();
  while (handleEvent() != false) {
    do();
  }
  exit();
}
```

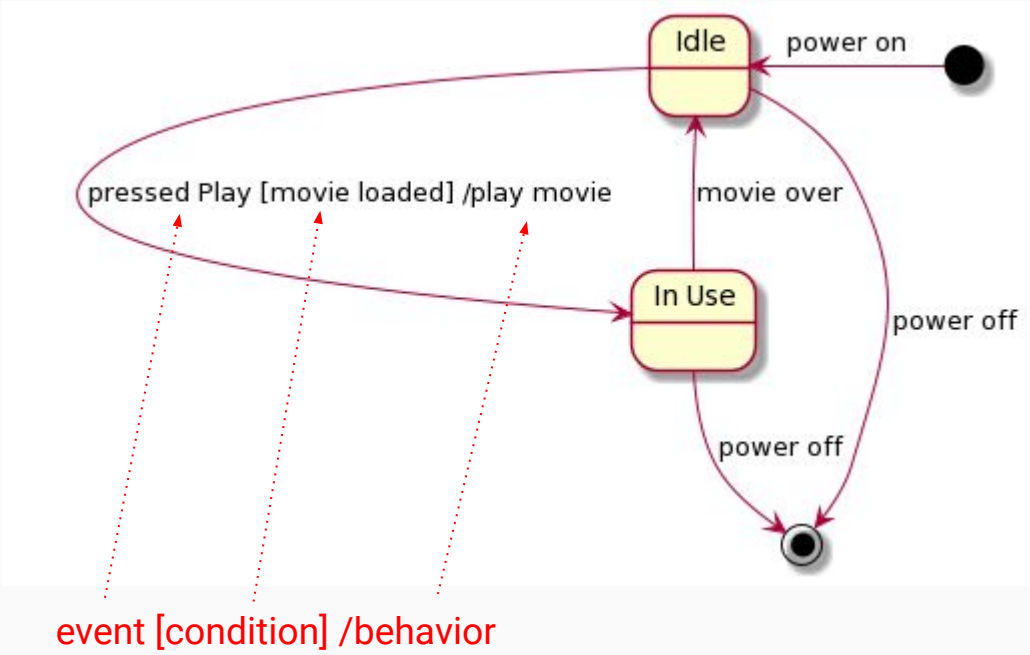
Transition

A movement from one state to another

Notation

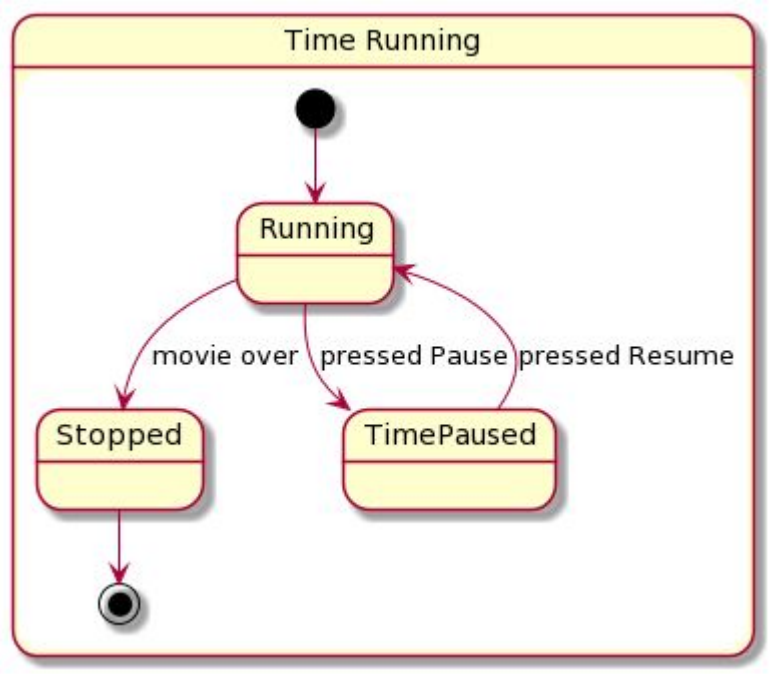
- **Arrow from source to target**

Movie Player State Machine

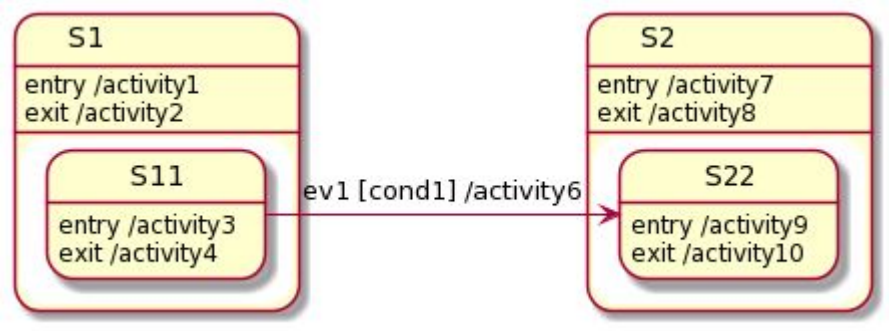


Composite State

A hierarchy of states within a higher abstraction state

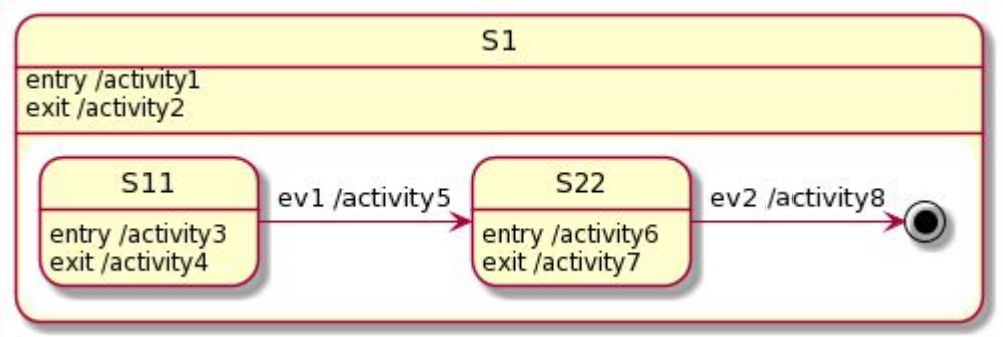


What would be the order of execution of the activities?



If `S11` is the active state, and event `ev1` occurred, and assuming condition `cond1` is true.

What would be the order of execution of the activities?



If `S11` is the active state, and event chain `ev1` , `ev2` occurred

State Machine Diagram Quiz

References

- Unified Modeling Language (UML) v2.5 Specification
- UML Distilled by Martin Fowler
- Applying UML and Patterns by Craig Larman
- Software Design: Modeling with UML by Neelam Dwivedi

Reading before next class:

- Software Architecture Patterns