SLIIT UNI

THE KNOWLEDGE UNIVERSITY

Lecture 7 - State-Chart (State-Machine) Diagram

SE2030 – Software Engineering

Session Outcomes

- Introduction to State Diagrams
- State Diagram symbols
  - States
    - Simple states
    - Composite states
  - Transitions
      - Call Event
      - Change Event
      - Time Event

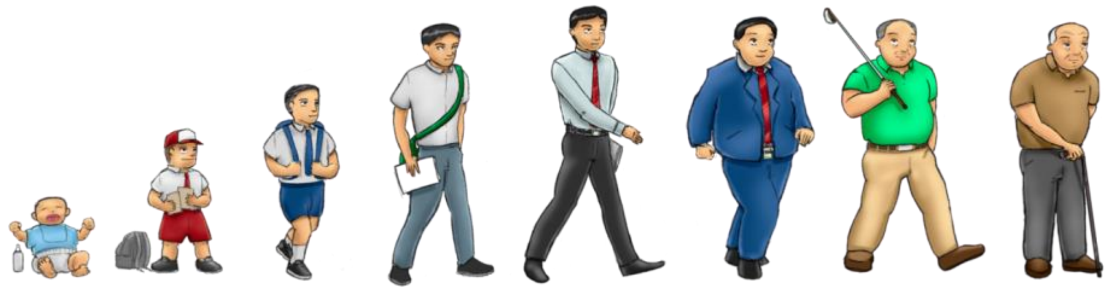
Lesson Learning Outcomes

- Define and explain the concept of states, events, and transitions in UML state-machine diagrams.
- Differentiate between simple states, composite states, and special states (initial and final) with examples.
- Identify and apply different types of triggers (call, change, time events) and guard conditions in state transitions.
- Construct state-machine diagrams to model dynamic behavior of objects in real-world scenarios such as seminars, alarms, and ATM systems.
- Evaluate and refine state-machine diagrams by applying best practices and avoiding issues such as black-hole or miracle states.

What is a State in General?

- **State** is a particular condition that someone or something is in at a specific time.

States of a Human Life



States of a Bulb

Determine states of these objects

- **A Fan**

- **A Car**

What are the States of an Object?

Objects has states …

Active

Idle

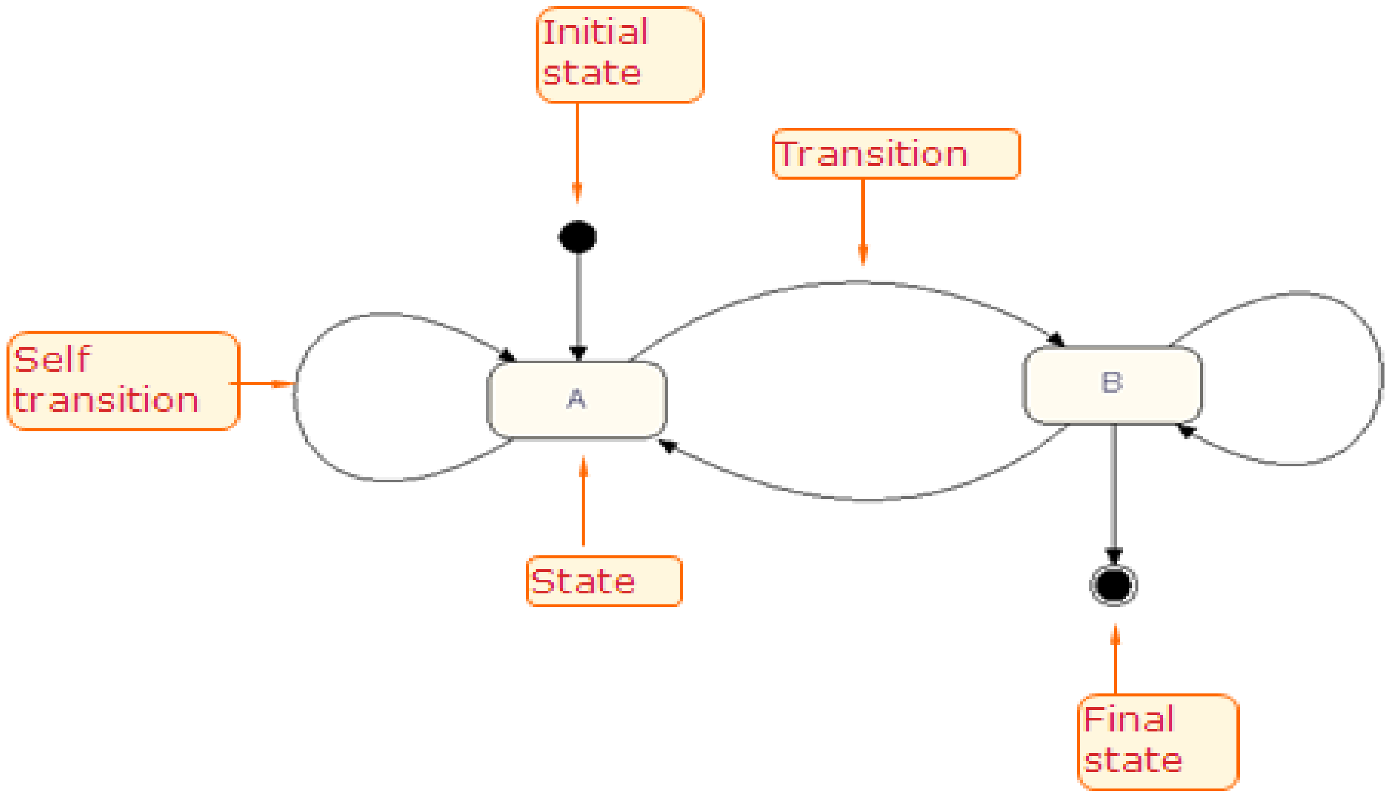
Waiting

UML State Machine Diagram

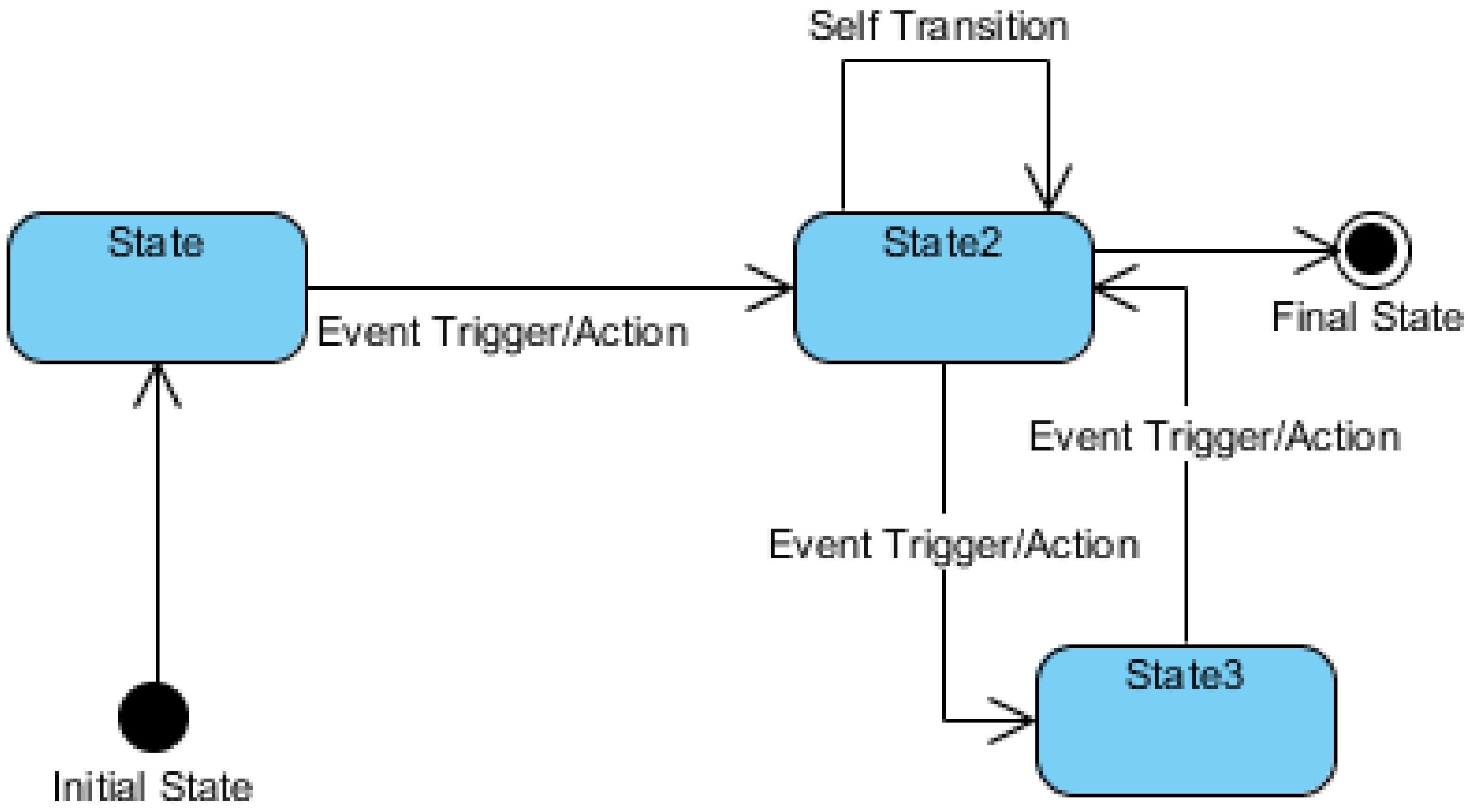
- **A state machine diagram models the behavior of a single object**, specifying the sequence of events that an object goes through during its lifetime.
- There is **only one state machine diagram for a class**.
- A state diagram is typically drawn for **only** for the classes which contains **significant dynamic behavior**.

State-Machine Diagram Notations

- States
- Transitions



State Machine Diagram Example



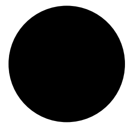
Source: https://www.visual-paradigm.com/tutorials/how-to-draw-state-machine-diagram-in-uml/

What is a State?

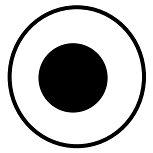
- A state represents a **state that an object is in at a particular time**.
- Shown as a **rectangle with rounded corners**, with the **state name shown within the state**.
- While in the state, the object satisfies some condition, performs some action, or waits for some event.

Special States

- Initial state (the object being constructed)
  - ✓denoted by a filled black circle

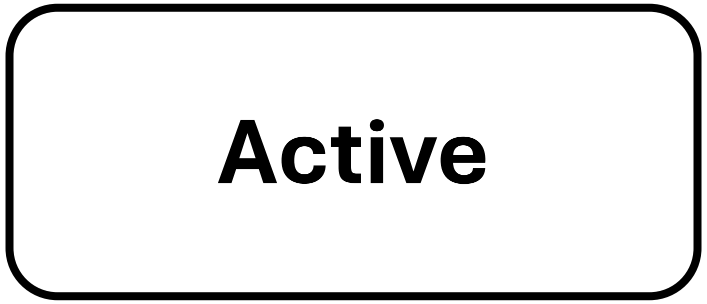


- Final state (the object being destructed)
  - ✓denoted by a circle with a dot inside

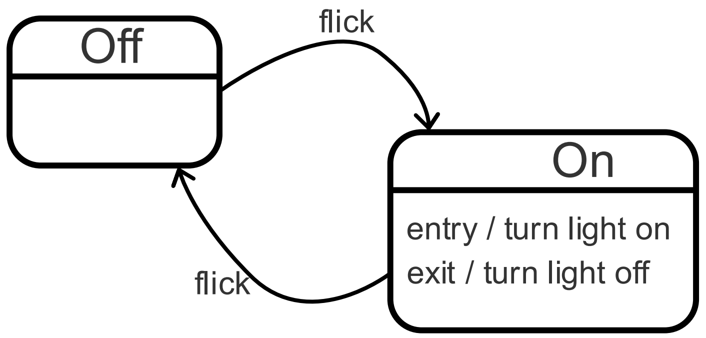


Types of States

- **Simple State** : Is a state that does not have sub states or compartments.

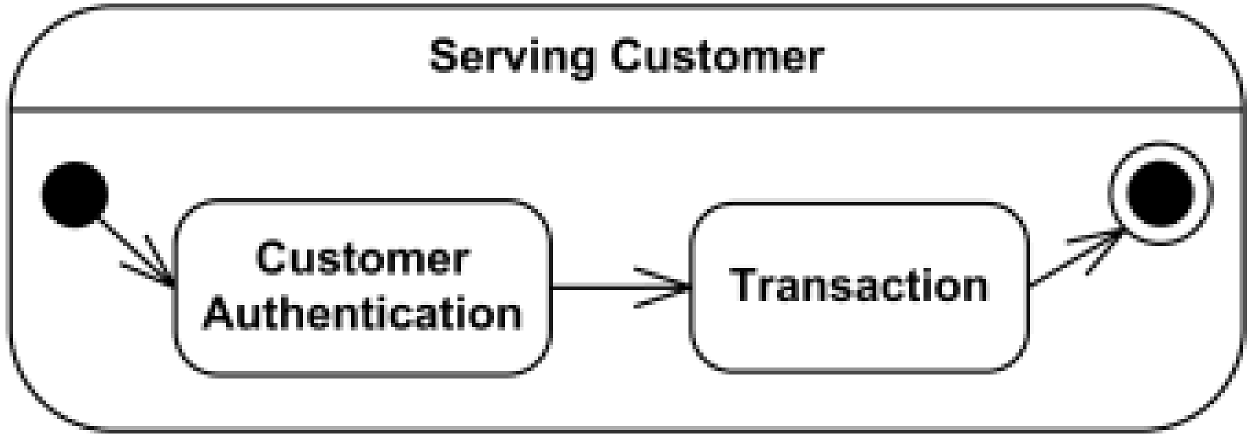


- **Simple State with compartments** :



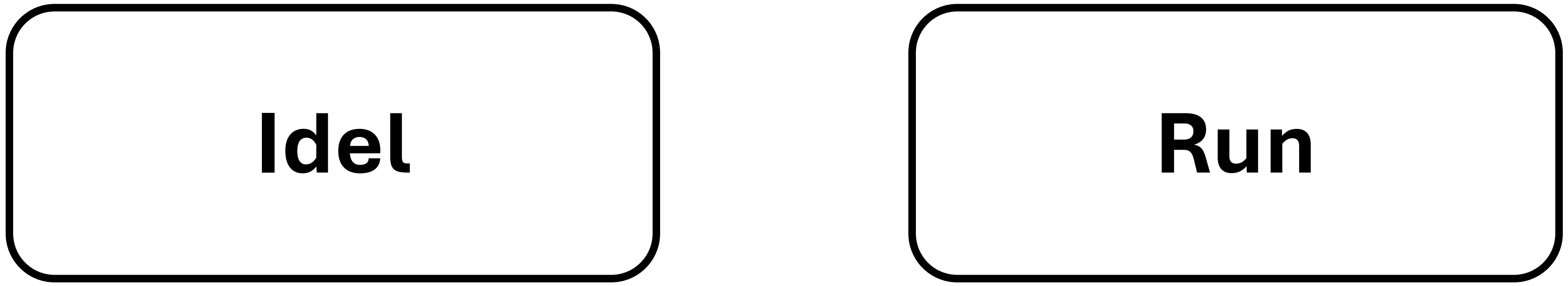
Types of States cont…

- **Composite State** :Is defined as a state that has sub states (nested states).



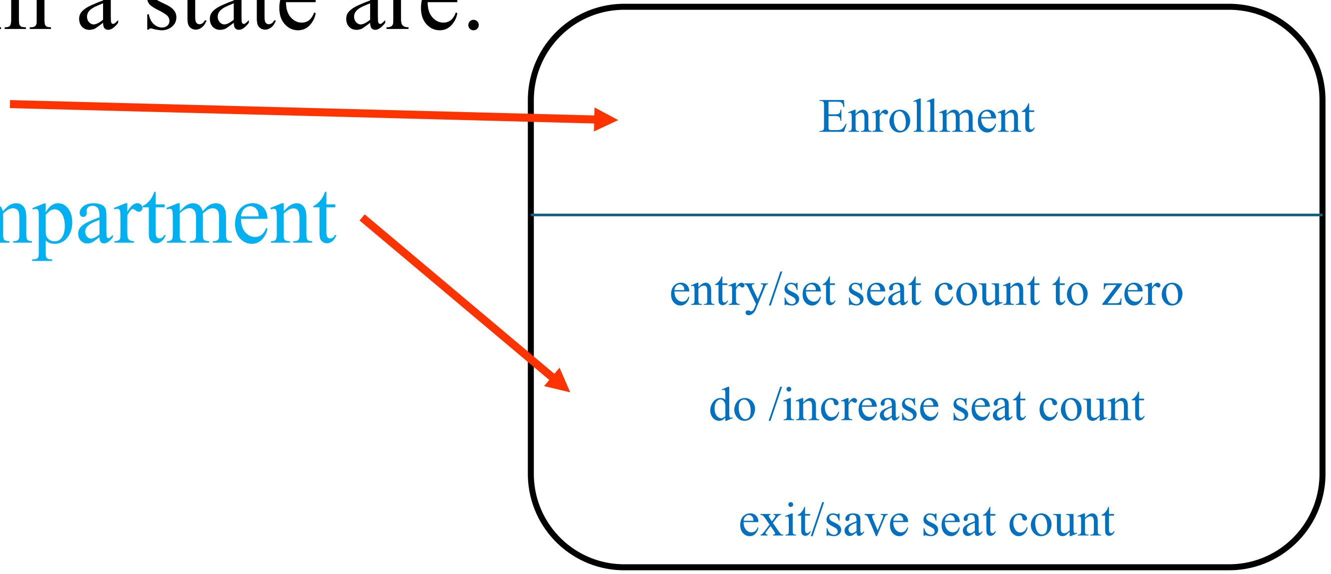
Simple States

- Simple state is a state without compartments.
- Indicate using a simple rounded rectangle and state name inside.



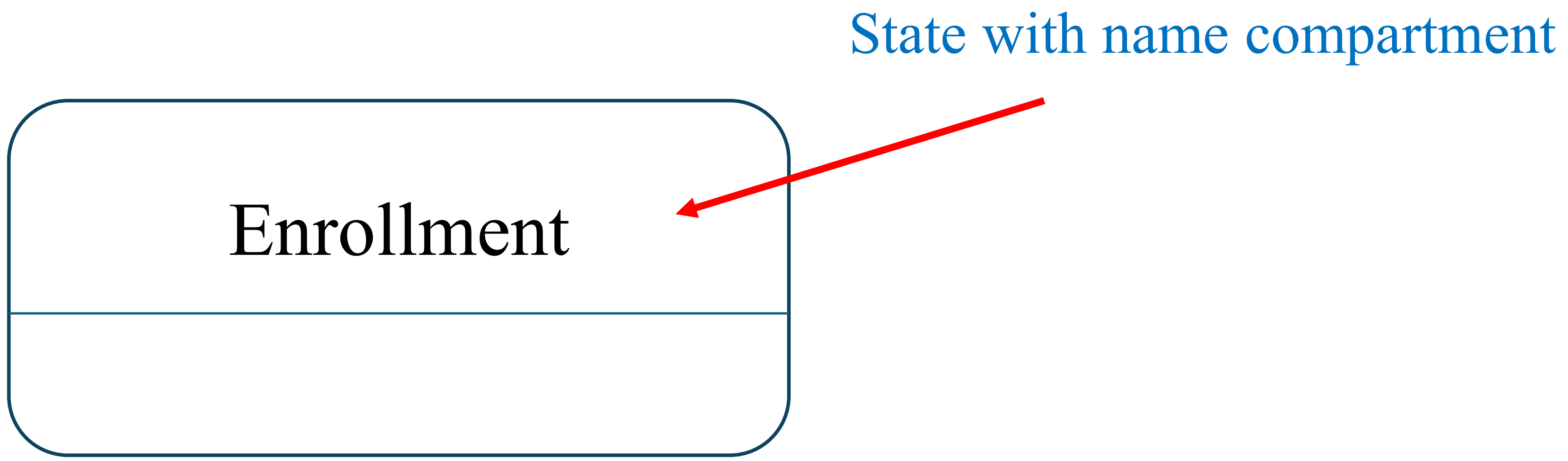
State with compartments

- A state may be subdivided into multiple compartments separated from each other by a horizontal line.
- Basic Compartments in a state are:
  - Name compartment
  - Internal behaviors compartment



Name Compartment

- This compartment holds the name of the state, as a string.



Internal Behaviors/Activities compartment

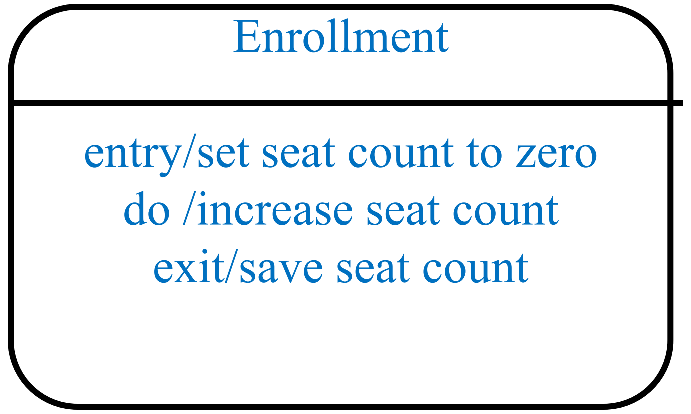
- This compartment holds a list of internal behaviors associated with a state.
- Each activity has the following format:

  <<behavior-type-label>> / <<action>>

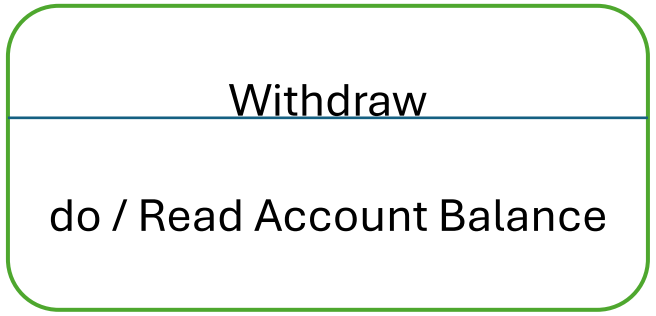
- The <behavior-type-label> identifies the situations under which will be invoked and can be one of the following:
  - **entry**
  - **exit**
  - **do**

Entry and Exit Activities/Behaviors

- **Entry** label identifies a Behavior, specified by the corresponding expression, which is performed upon entry to the State (entry Behavior).
- **Exit** label identifies a Behavior, specified by the corresponding expression, that is performed upon exit from the State (exit Behavior).



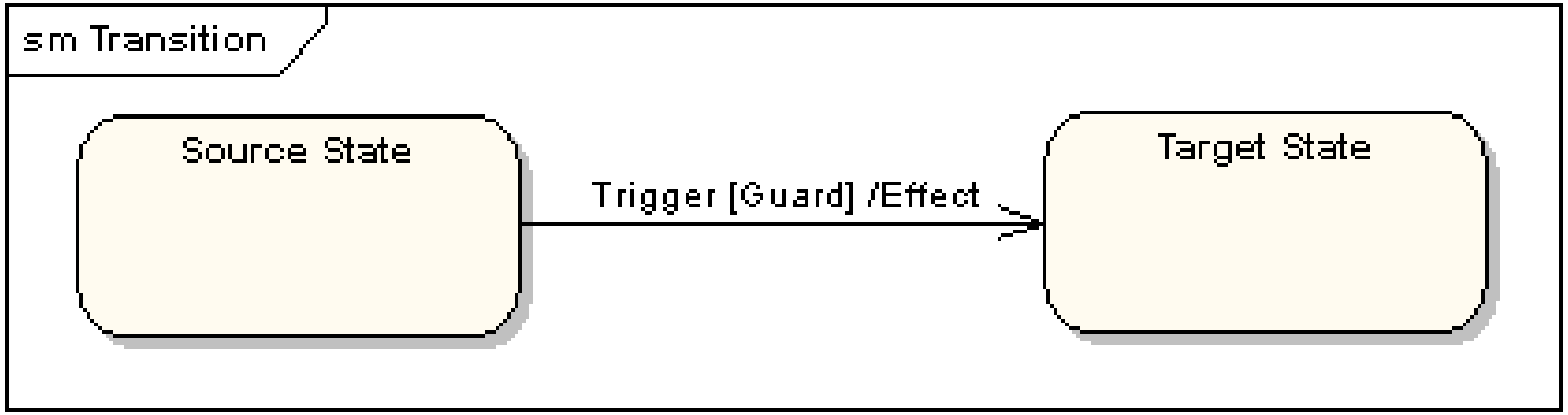
State Do behavior



- Performed within a state. Ongoing work that an object performs while in a particular state. This is started after the entry action is finished. The work automatically terminates when the state is exited.
- May be interrupted.
- Activities (do), by definition, cannot change the state of the object (Unlike actions Entry and Exit), so interrupting them will not corrupt the state of the object.

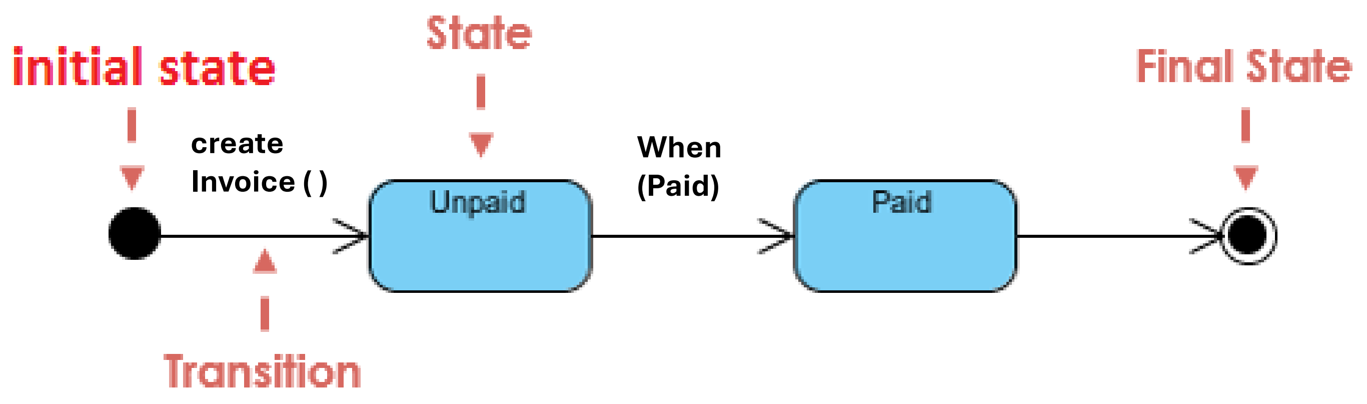
Transitions

- **Transitions** from one state to the next are denoted by **lines with arrow heads**.



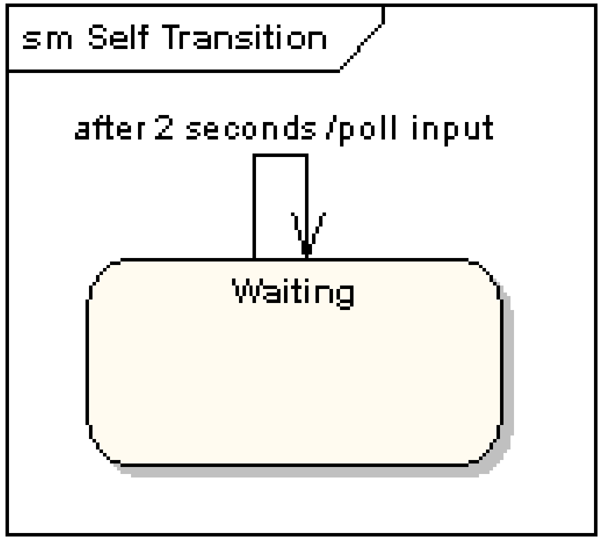
Elements of a Transition

- **Source State**: the state of the object before the transition.
- **Target State**: the state of the object after the transition.
- **Trigger**: the event that causes the transition



Self-Transitions

- A self transition is a **transition** that starts and ends in the same state.
- When a self transition is taken, the state in question is **exited** and **entered** again.
- Self transitions are commonly used to "restart" the current state, causing the exit actions to happen, followed by the entry actions.



Types of Triggers

- Mainly there are three types of triggers.
  - **Call Event** :Message (Parameters)
  - **Change Event** :When (Condition)
  - **Time Event** :After (Time Period)

Call Trigger / Event

- A Call-Event is denoted by the name of the triggering Operation.
- Used to represent a transition that occurs as a result of a message being received by the object.
- The arguments are optional.

Examples :

- buttonPressed().
- buttonPressed(buttonID).

Change Trigger / event – When (condition)

- Used to represent a transition that occurs when a condition / Boolean expression becomes true.
- Strictly, the condition should be a Boolean expression, but some Engineers use Plain English.
- Example : If the temperature T rises above 100° the object must change state:

  when(T > 100°).

  when (temperature exceeds 100 °).

Time Trigger / event – After (time duration)

- Time-Event is denoted with “after” followed by a Time Expression, such as “after 5 seconds.”
- The period can be expressed in any stated units.

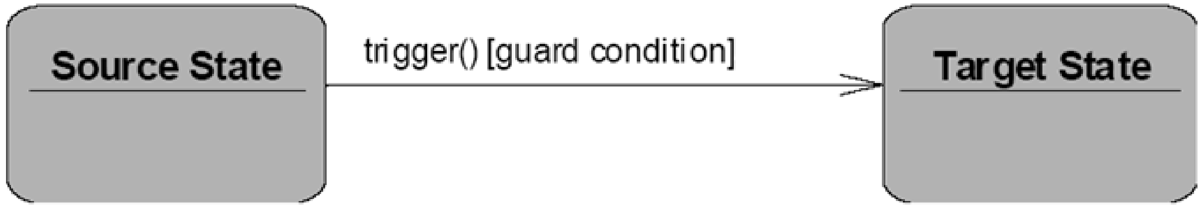
  Example : After 60 seconds the object must change state:

  after(1 minute)

  after(60 seconds)

Triggers with Guard Conditions

All the triggers can have guard conditions. But these are **optional**.



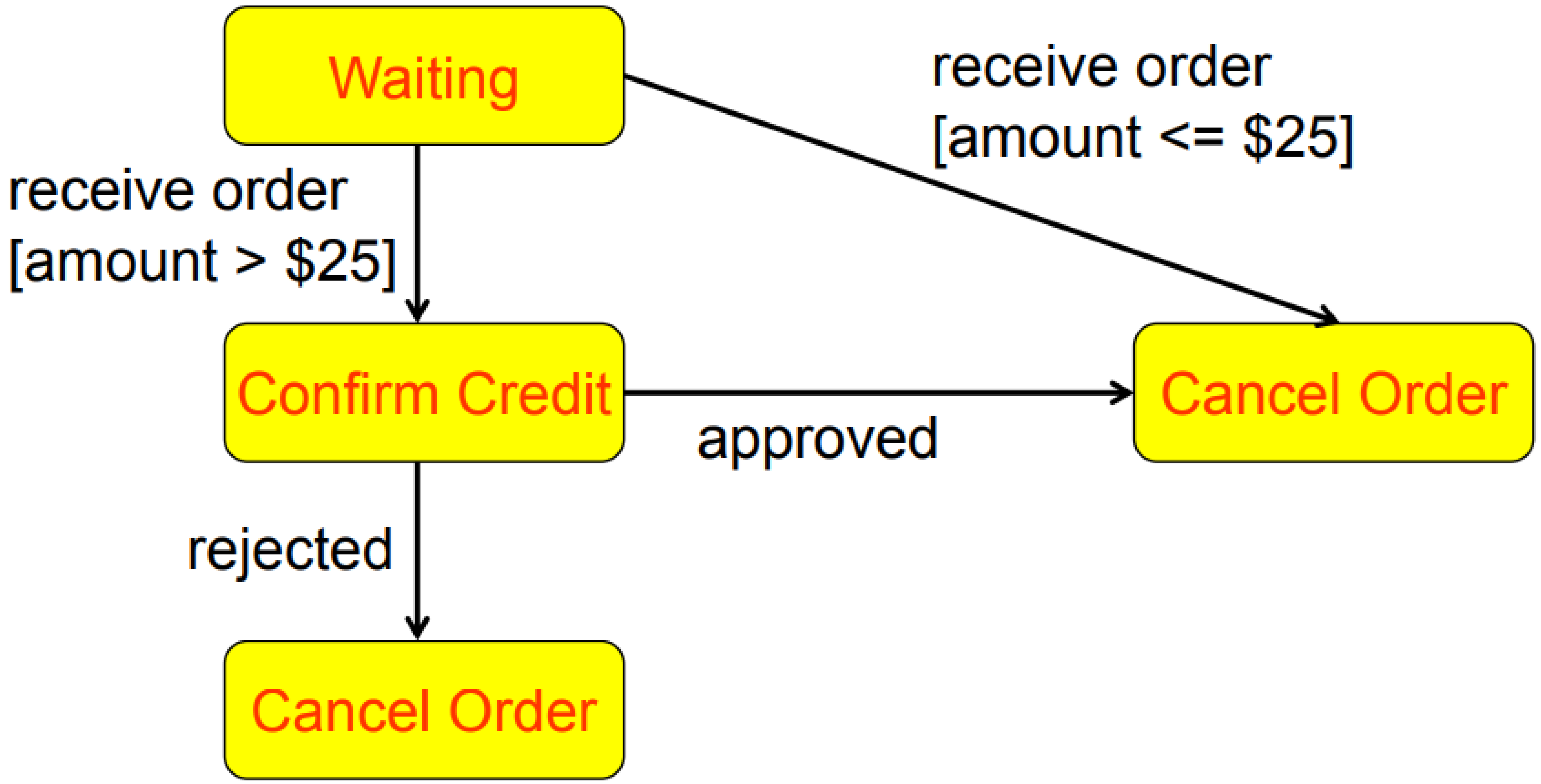
message() [Guard Condition]

message(arguments) [Guard Condition]

when(condition) [Guard Condition]

after(timespan) [Guard Condition]

Triggers with Guard Conditions - Example



Activity 01

Draw a state machine diagram for the Seminar class during registration.

- The seminar is first proposed and then scheduled.
- Then the Seminar will be opened for enrollment for the students. When enrollment starts, the seat capacity is zero.
- As the number of seats is limited, as long as the Seminar is open for enrollment, the remaining seats/size will be updated each time a student is enrolled in the seminar.

Activity 02

Draw the State Diagram for the following Scenario.

- Burglar alarm is initially at the state of resting .
- Then by setting the alarm, burglar alarm state may be changed to the state set.
- When the alarm is set, it may be turned off. This will allow the alarm to be resting. While the alarm is set it can be triggered, which will make it ring. When the alarm is ringing, it can be turned off. Then the alarm will be resting again. Draw a state diagram for the Burglar Alarm class.

Note: No need to model the final state.

Activity 03

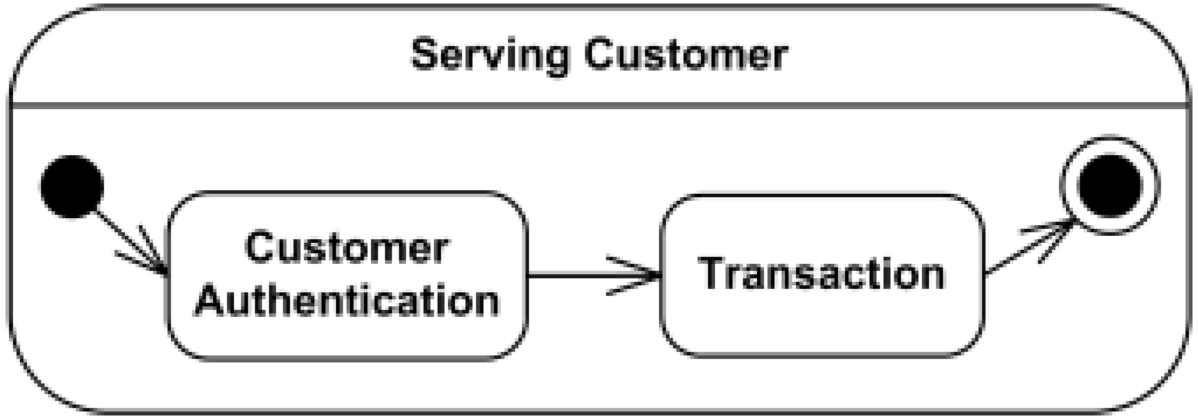
Draw the State Diagram for the following diagram

- You need to type a valid username and password to login to Courseweb. Once you are logged in you have access to unit contents.

- You can logoff once you have completed using the Courseweb. There is also a timeout of 5 minutes if you are inactive and you are automatically logged off. Draw a state diagram for the Courseweb user class.

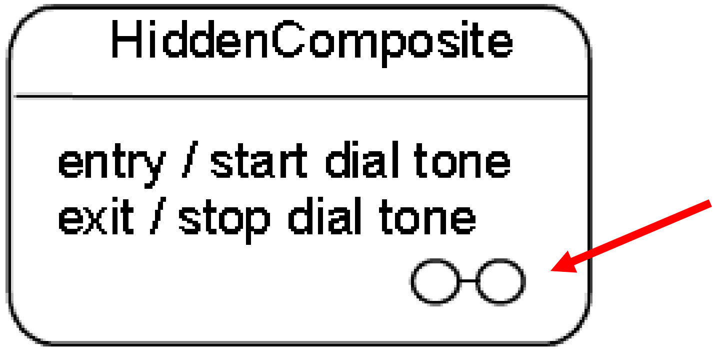
Composite State

- A state that has sub states (nested states).
- Sub states are in a separate compartment called "Decomposition" compartment.



Composite State – Hidden Decomposition Indicator

- In some cases, it is convenient to hide the decomposition of a composite state.
- For example, there may be a large number of states nested inside a composite state and they may simply not fit in the graphical space available for the diagram.
- In that case, the composite state may be represented by a simple state graphic with a special "composite" icon, usually in the lower right-hand corner.



Composite State with a hidden decomposition indicator icon

Activity 04

Draw a State Chart for the given ATM

- ATM is initially turned off, it is in the **Off state**. After the power is turned on, ATM performs startup action and enters **Self Test** state. If the test unsuccessful, ATM goes into **Out of Service** state, otherwise goes to the **Idle** state. In this state ATM waits for customer interaction.
- The ATM state changes from **Idle** to **Serving Customer** when the customer inserts debit or credit card in the ATM's card reader. On entering the **Serving Customer** state, the entry action **readCard** is performed. The state also has exit action **ejectCard** which releases customer's card on leaving the state, no matter what caused the transition out of the state.

Activity 04 cont…

- **Serving Customer** state is a composite state with sequential sub states **Customer Authentication**, **Selecting Transaction** and **Transaction**. **Customer Authentication** and **Transaction** are composite states by themselves which is shown with hidden decomposition indicator icon. These states follow in order within the composite state.
- **Serving Customer** state has a **transition** back to the **Idle** state after transaction is completed . While serving customer if failure occurs it will move into the **Out of Service** state. ATM machine can be turned off when it is in the **Idle** state.
- While in the **Out of Service** state when a maintenance is performed it will move back to **Self Test** state
- A transition from **Serving Customer** state back to the **Idle** state could be triggered by **cancel** event as the customer could cancel transaction at any time.

State Machine Diagram Guidelines

- **Keep the diagram simple**
  - If it is too complex, perhaps it should be broken down into separate diagrams.
- **Question "Black-Hole" States**
  - A black-hole state is one that has transitions into it but none out of it, something that should be true only of final states.
- **Question "Miracle" States**
  - A miracle state is one that has transitions out of it but none into it, something that should be true only of start points.

References

- UML 2 Bible
- Learning UML 2.0 by Kim Hamilton, Russ Miles
- Chapter 15UML 2 Bible

Thank You!