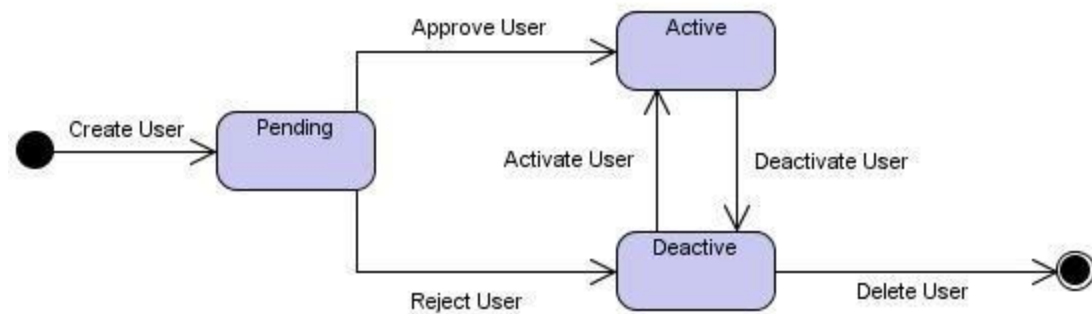


State chart Diagrams

StateChart Diagram Example



Objectives

- What are State Machines?
- What are Statechart Diagrams?
- Be able to model Statechart diagrams
- Be able to model super states and substates

Introduction

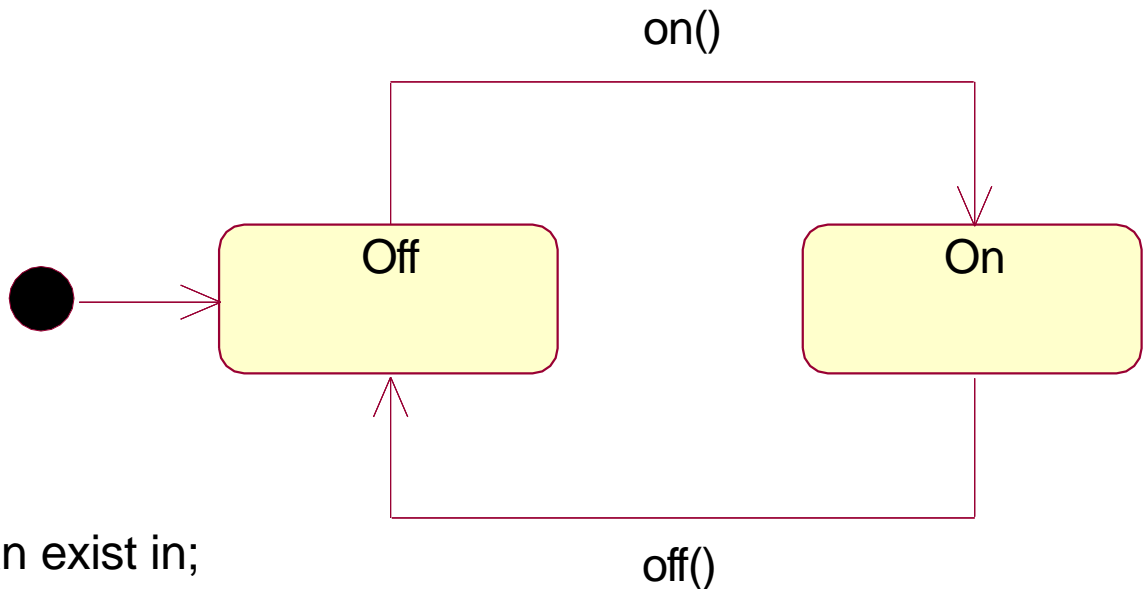
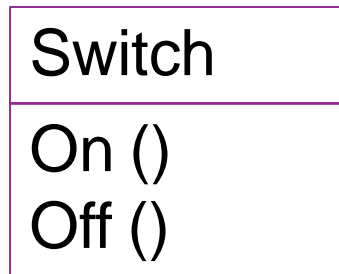
- A state chart diagram is a **state machine** that
 - Models the behavior of an individual object
 - Is a dynamic model of the system.
 - Is event driven.
- It captures the changes in an object throughout its lifecycle as they occur in response to internal and external events.
- The scope of a statechart is the entire life of one object.
- State machines come in two varieties:
 - Statechart diagrams.
 - Activity diagrams.

Statechart Diagrams

- UML state charts depict a single object at runtime:
 - a state chart shows the **lifecycle of a single object**, from construction to destruction, or part thereof.
 - Notation for state charts overlaps activity diagrams.
- A state machine has some number of states, and transitions between those states

What is a State?

- Switch object will have two states depending on the method call `on()`, `off()`

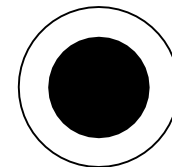
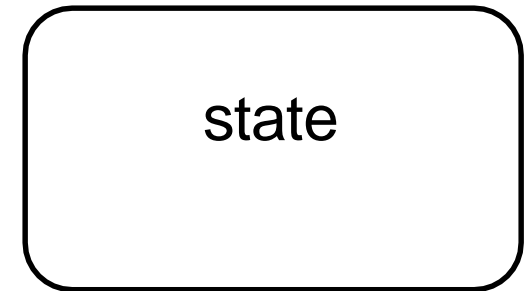
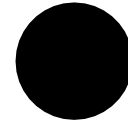


The state chart shows:

- the **states** that the object can exist in;
- the valid **transitions** between states;
- the **event** that causes each state transition.

State Symbols

- A UML state chart consists of.
 - A single initial state (filled black circle),
 - e.g. to represent the object being constructed:
- Intermediate state(s) to represent the value(s) of attribute(s) changing:
- A single final state (optional),
 - e.g. to represent the object being destructed:



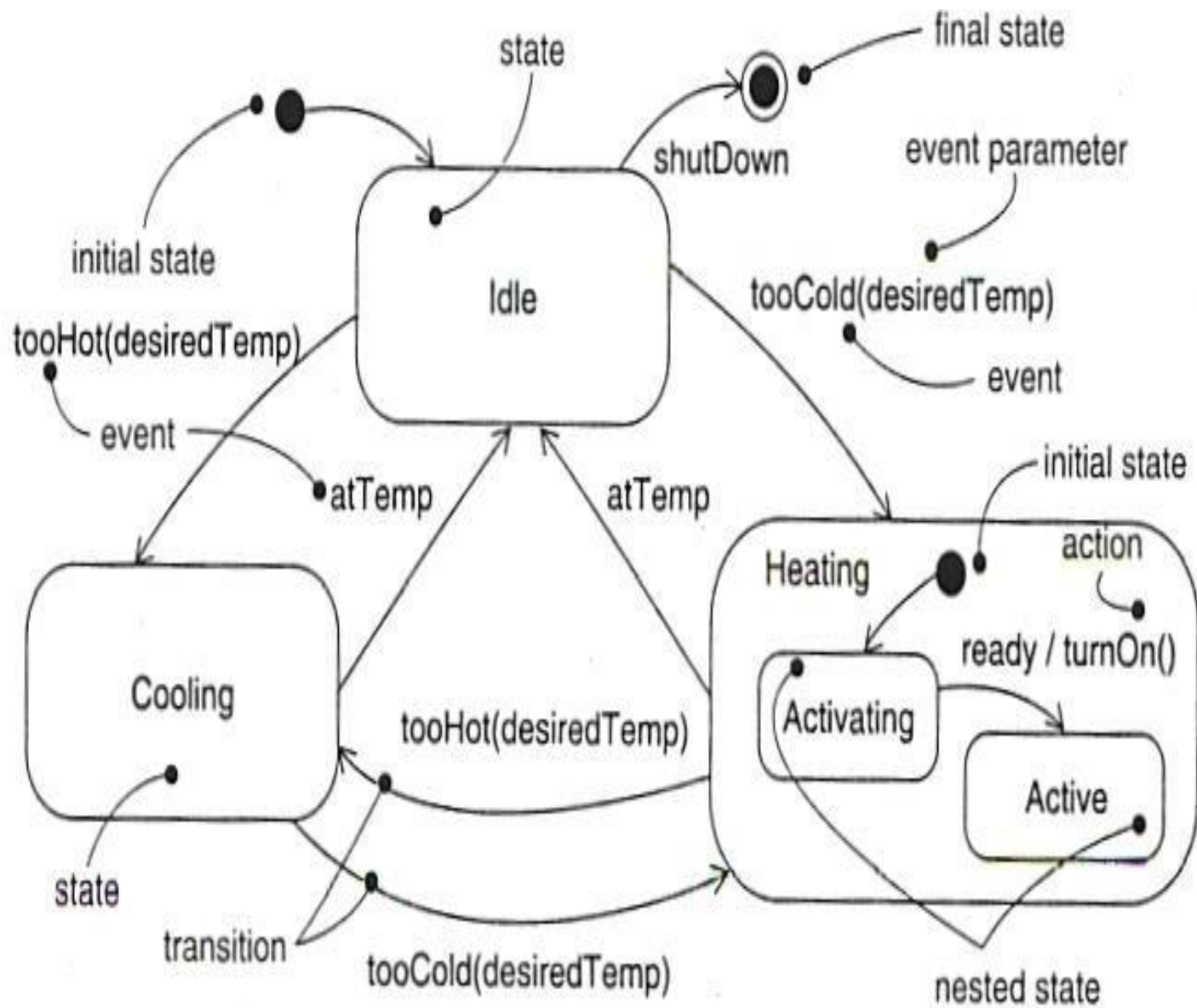



Figure 21-1: State Machines

State Transitions

- Represented by an arrow drawn between two states:
 - the arrow is labeled with the name of the event that triggers the transition.
- UML state diagrams have three types of events.
 - **Message**
 - **When ()**
 - **After**
- The source and destination states can be the same, in which case the transition is a **self-transition**

Trigger Events

- Trigger events cause transitions to occur, usually from one state to another.
- In a UML state chart, a trigger event can be one of the following:
 1. receipt of a **message** (i.e. a method within the object has been called);
 2. **when** some condition is satisfied (the condition can be written as a Boolean expression, or plain English);
 3. **after** a period of time has passed.

Transitions:Message

- Used to represent a transition that occurs as a result of a message being received by the object:

Message(arguments)

- The arguments are optional.
- Examples :
 - If a buttonPressed message is received the object must change state:
- **buttonPressed().**
- **buttonPressed(buttonID).**

Transitions: When()

- Used to represent a transition that occurs when a condition becomes true.

[When (condition)]

- Strictly, the condition should be a Boolean expression
- Example :
If the temperature Temperature rises above 100°
the object must change state:
[when(Temperature > 100°)]

Transitions: After()

- Used to represent a transition that occurs after a specified period of time has elapsed.

`after(time period)`

- The period can be expressed in any stated units.
- Example :
After 60 seconds the object must change state:
 - `after(60 seconds)`

State Diagrams

- In general, a State diagram is composed of:
 - a starting point (initial state);
 - a set of states, typically shown as rectangles with rounded corners;
 - a set of transitions between the states, typically shown as arrows drawn between pairs of states;
 - a set of events that cause the transitions;
 - an (optional) finishing point (final state).

Exercise 01

The burglar alarm may be set when the burglar alarm is resting. This event can be done by setting the alarm. 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.

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: can be one of `call()`, `signal()`, `when()` or `after()`.
- **Guard Condition** (optional): see next slide.
- **Action** (optional): something that is done as part of the transition.

Event Syntax

event-name (parameters-list) [guard-condition] /

action where:

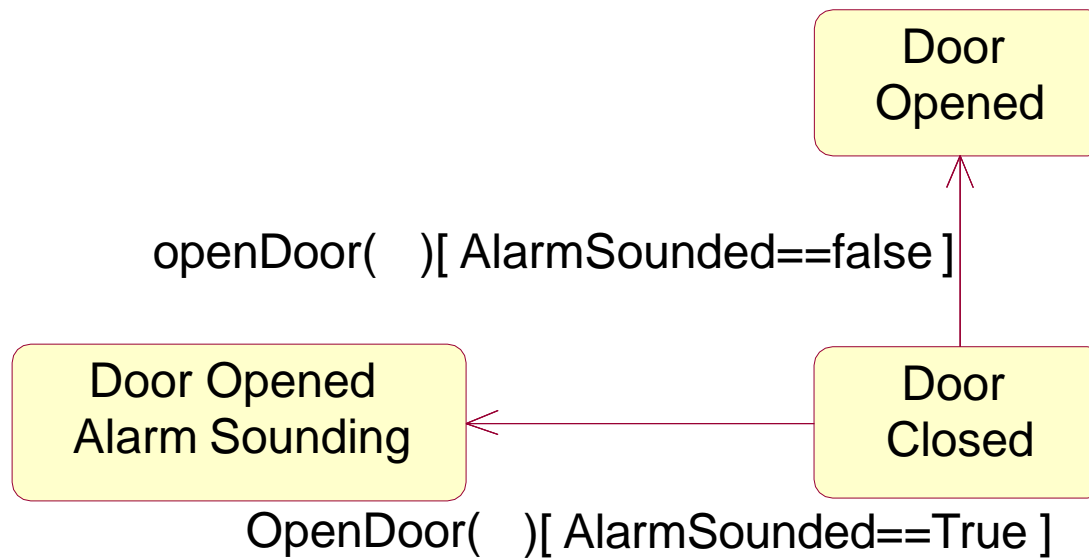
- 1) event-name - identifies the event
- 2) parameters-list - data values passed with the event for use by the receiving object in its response to the event
- 3) guard-condition - determines whether the receiving object should respond to the event
- 4) action-expression - defines how the receiving object must respond to the event

Types Actions

- **Entry** specifies an action that is *always* performed when entering a state:
 - e.g. /entry count = 0
 - **Un-interruptable**
- **Exit** specifies an action that is *always* performed when exiting a state:
 - e.g. /exit cleanUp()
 - **Un-interruptable**
- **Do** specifies an action that is *continually* performed when in a state:
 - e.g. /do checkForEvents()
 - **Interruptable**

Guard Condition

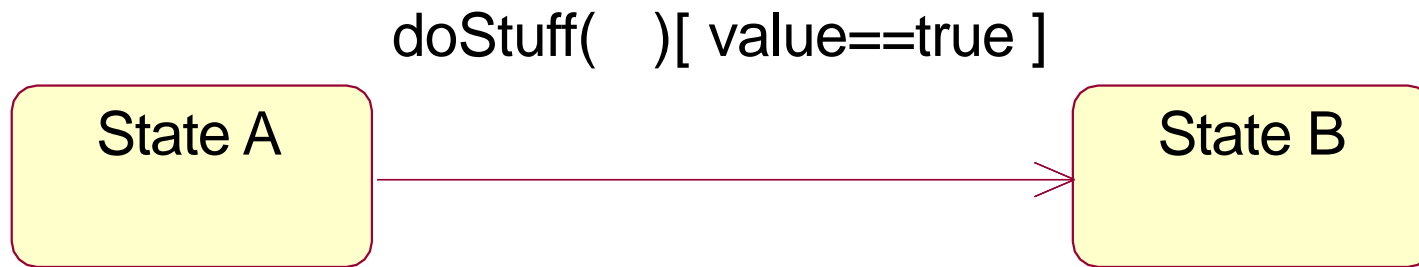
- It is possible for the **same event to cause a transition to different states** depending on some guard condition.



Guard Condition Rule 1

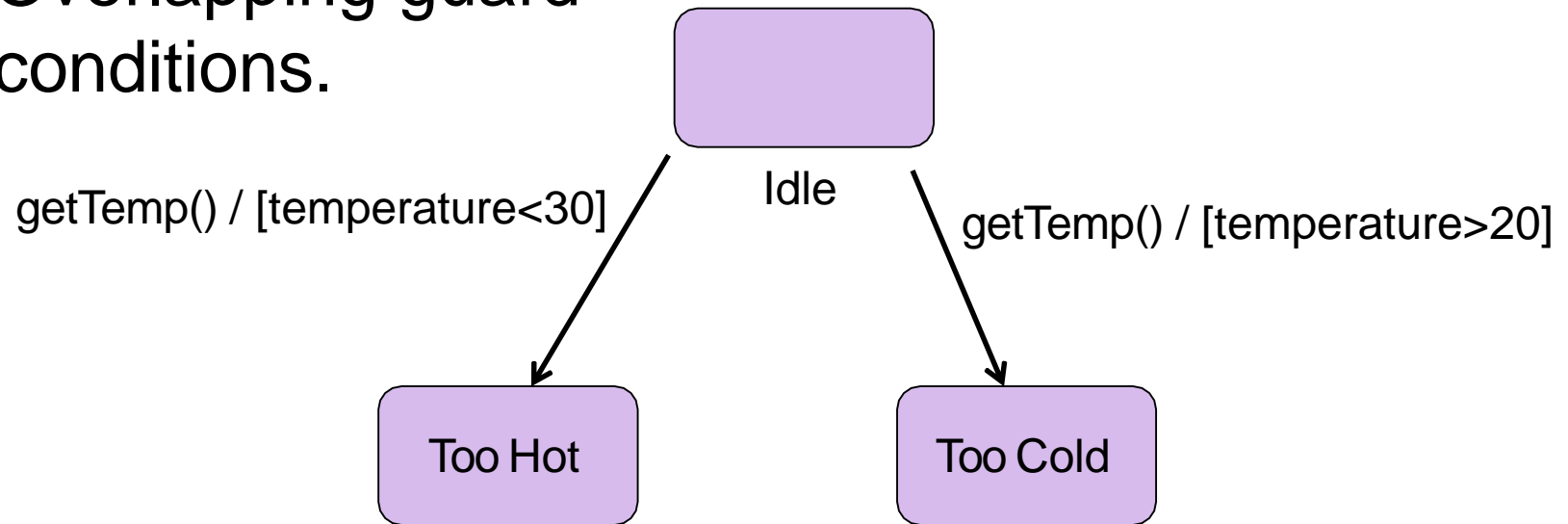
- The set of guard conditions should handle **all** possible outcomes.
 - Otherwise there will be ambiguity in the states.

Example: what happens if `value==false`?



Guard Condition Rule 2

- Overlapping guard conditions.



- What happens if the temperature is 25 degrees?
 - Guard conditions must be exclusive to ensure that there is no ambiguity

Exercise 2

- You need to type a valid username and password to login to Blackboard. Once you are logged in you have access to unit contents. You can log off once you have completed using the Blackboard. If you are inactive for 5 minutes you are automatically logged off.
- Draw a state diagram for the Blackboard user class.

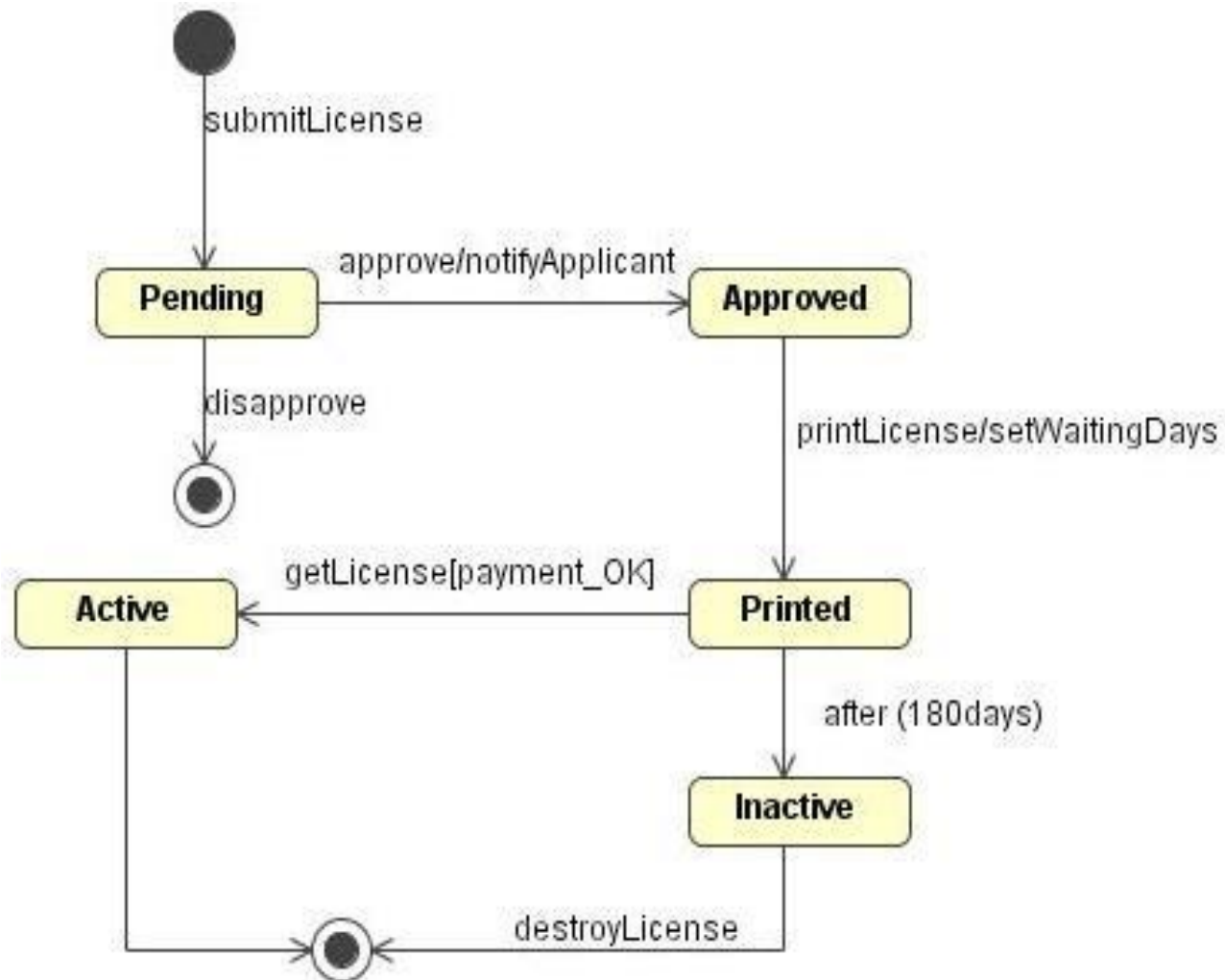
Exercise 3

An applicant apply for a driver license. All the new applications are considered as pending. If the submitted application meets all the requirements his application will be approved. Otherwise it will be rejected. Applicant will be notified via an sms about the approval. Then license will be printed and waiting time will be set. Applicant has to collect the license within 3 months by paying . If he does not collect it, after 3 months, license will be destroyed. Applicant has to re-apply. If he collects it within 3 months license will be active. All the active applications are destroyed after 1 year.

Draw a state transition diagram for the driver license application.

Event Actions Example

An applicant apply for a driver license. All the new applications are considered as pending. If the submitted application meets all the requirements his application will be approved. Otherwise it will be rejected. Applicant will be notified via an sms about the approval. Then license will be printed and waiting time will be set. Applicant has to collect the license within 3 months by paying . If he does not collect it, after 3 months, license will be destroyed. Applicant has to re-apply. If he collects it within 3 months license will be active. All the active applications are destroyed after 1 year.



Super States and Sub states

- UML provides a mechanism for representing an object life cycle as a hierarchy of state machines:
 - similar to step-wise refinement in algorithm design.
- The object lifecycle is divided into sub-cycles, each one represented by a different sub-state diagram.
- A sub-state diagram can have multiple entry points but they must all link to one sub-state start node.
- A sub-state diagram can have multiple exit points, each leading to any state outside..

Super States and Sub states

- UML provides two mechanisms for linking substates:
 - Drawing boundaries (super-states) around groups of sub-states;
 - Drawing a top-level diagram as an overview, and placing different groups of sub-states on different diagrams.



Substates

- A substate is a state that is nested inside a another state.
- A **simple state** has no substates.
- A **composite state** has substates. A composite state may contain either concurrent or sequential substates

Substates – Example 1

- Consider the behavior of an ATM. When there are no customers the ATM is in Idle state waiting for customers. When the ATM handles the customer transaction it is in Active state. While it is Active the ATM would validate the customer, select a transaction, process the transaction and then prints the receipt. After printing ATM goes back to the Idle state. While the cash store of the ATM is reloaded, it is in the Maintenance state. The customer might decide to cancel the transaction at any point.

Substates

- Sequential substates are the most common kind of nested state machine.
- In certain modeling situations you may want to model concurrent substates. Using these substates you can specify two or more substates that execute in parallel.

Substates – Example 2

Maintenance is decomposed into two concurrent substates, Testing and Commanding. Each of these substates is further decomposed into concurrent substates. That is, while in the Commanding state the object will be in the Waiting or Command state. While in the Testing state the object will be in the Testing Devices or Self Diagnosis state.

State Charts Vs Activity Diagrams

- A UML activity diagram can sometimes be used as an alternative to a state chart.
- A UML **state chart**:
 - shows all possible states and transitions (i.e. events) that can occur during the lifecycle of a single object;
 - all possible sequences of events can be traced via the transitions.
- A UML **activity diagram**:
 - can involve many different objects;
 - shows one possible sequence of transitions.

Thank you!