CSE 403
Lecture 9

UML State Diagrams

Reading:
*UML Distilled, Ch. 10*, M. Fowler

slides created by Marty Stepp
http://www.cs.washington.edu/403/
UML state diagrams

- **state diagram**: Depicts data and behavior of a single object throughout its lifetime.
  - set of states (including an initial start state)
  - transitions between states
  - entire diagram is drawn from that object's perspective

- similar to finite state machines (DFA, NFA, PDA, etc.)

- What objects are best used with state diagrams?
  - large, complex objects with a long lifespan
  - domain ("model") objects
  - not useful to do state diagrams for every class in the system!
State diagram example

- **Wait**:  Transition to **Lock**
  - **key turned [candle out] / release killer rabbit**
- **Lock**: Final state
  - **key turned [candle in] / open safe**
  - **candle removed [door closed] / reveal lock**
  - **initial pseudostate**

- **Safe closed**: Transition to **Wait**

**Transition**

- **State**
- **Initial pseudostate**
States

- **state**: conceptual description of the data in the object
  - represented by object's field values

- entire diagram is drawn from the central object's perspective
  - only include states / concepts that this object can see and influence
  - don't include every possible value for the fields; only ones that are conceptually different
Transitions

- **transition**: movement from one state to another

  - **signature** [guard] / activity
    - **signature**: event that triggers (potential) state change
    - **guard**: boolean condition that must be true
    - **activity**: any behavior executed during transition *(optional)*

- transitions must be mutually exclusive (deterministic)
  - must be clear what transition to take for an event
  - most transitions are instantaneous, except "do" activities
Internal activities

- **internal activity**: actions that the central object takes on itself
  - sometimes drawn as self-transitions (events that stay in same state)

- entry/exit activities
  - reasons to start/stop being in that state

<table>
<thead>
<tr>
<th>Typing</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry/highlight all</td>
</tr>
<tr>
<td>exit/ update field</td>
</tr>
<tr>
<td>character/ handle character</td>
</tr>
<tr>
<td>help [verbose]/ open help page</td>
</tr>
<tr>
<td>help [quiet]/ update status bar</td>
</tr>
</tbody>
</table>
User account management

State diagram example

- **New**
  - Transition: [isUnique](d()) create/
  - Transition: [isAccountDormant](d()) suspend/

- **Active**
  - Transition: [isCancelRequested](d()) cancel/
  - Sub-state: [isVerified](d()) activate/
  - Sub-state: [isUnique](d())
  - Transition: [isSuspensionRequested](d()) suspend/
  - Transition: [isPasswordAlert](d()) lock/
  - Transition: [isAccountDormant](d()) suspend/
  - Transition: [isResumeRequested](d()) resume/
  - Transition: [isLockExpired](d()) unlock/

- **Closed**
  - Transition: [isCancelRequested](d()) cancel/
  - Transition: [isPolicyViolated](d()) cancel/

- **Suspended**
  - Transition: [isCancelRequested](d()) cancel/

- **State machine User Account (protocol)**

Source: [uml-diagrams.org](http://uml-diagrams.org)
Super/substates

• When one state is complex, you can include substates in it.
  – drawn as nested rounded rectangles within the larger state

• *Caution:* Don't over-use this feature.
  – easy to confuse separate states for sub-states within one state
State diagram example

ATM software states at a bank
Java thread states

State diagram example

- **New**
- **Runnable**
  - **Ready**
  - **Running**
    - thread was selected by thread scheduler to run/
    - yield/
    - thread was suspended by thread scheduler/
  - **Timed Waiting**
    - sleep(sleeptime)/
    - wait(timeout)/
    - join(timeout)/
    - LockSupport.parkNanos()/
    - LockSupport.parkUntil()/
    - timeout elapsed/
  - **Waiting**
  - **Blocked**
    - wait for lock to enter syncro block or method
    - wait for lock to reenter syncro block or method
    - monitor lock acquired/
  - **Terminated**
    - thread terminated/
Implementing states

• What are some ways to write code to match a state diagram?
  – state tables (pseudo-code)
  – nested if/else
  – switch statements
  – state enums
  – State design pattern

```java
public void HandleEvent (PanelEvent anEvent) {
    switch (CurrentState) {
    case PanelState.Open :
        switch (anEvent) {
            case PanelEvent.SafeClosed :
                CurrentState = PanelState.Wait;
                break;
            }
        break;
    case PanelState.Wait :
        switch (anEvent) {
            case PanelEvent.CandleRemoved :
                if (isDoorOpen) {
                    RevealLock();
                    CurrentState = PanelState.Lock;
                }
                break;
            }
        break;
    case PanelState.Lock :
        switch (anEvent) {
            case PanelEvent.KeyTurned :
                if (isCandleIn) {
                    OpenSafe();
                    CurrentState = PanelState.Open;
                } else {
                    ReleaseKillerRabbit();
                    CurrentState = PanelState.Final;
                }
                break;
            }
        break;
    }
}
```
• **state pattern**: An object whose sole purpose is to represent the current "state" or configuration of another larger object.
  - A behavioral pattern.
  - Often implemented with an `enum` type for the states.
  - Each object represents one specific state for the larger object.
  - The larger object sets its state in response to various mutations.
  - Allows various observers and interested parties to quickly and accurately know what is going on with the larger object's status.

• Analogous to the notion of *finite state machines*.
  - Set of states (nodes)
  - Set of edges (mutations that cause state changes)
/** Represents states for a poker game. */
public enum GameState {
    NOT_STARTED, IN_PROGRESS, WAITING_FOR_BETS, DEALING, GAME_OVER;
}

/** Poker game model class. */
public class PokerGame {
    private GameState state;

    public GameState getState() { return state; }

    public void ante(int amount) {
        ...
        state = WAITING_FOR_BETS;  // change state
        setChanged();
        notifyObservers(state);
    }
}