# CSC 217 Lab 06

Finite State Machines

## Lab Overview

- Deadlines and Reminders
- FSM Reminders
- Activities
    - Course Name FSM
    - InvalidTransitionException
    - Testing CourseNameValidatorFSM
    - Implementing CourseNameValidator (state pattern)
    - Test
    - Deploy
- Lab Wrap-Up

## Deadlines and Reminders

- Deadlines
  - Lab 6 due week of October 18th
  - Project 1 Part 2
    - Deadline 10/14 @ 11:45pm
    - Late Deadline 10/16 @ 11:45pm
- Reminders
  - Jenkins Console Output for coverage/test related PMD notifications that might block TS tests
  - Project commit messages are graded (lab commit messages will be reviewed)
    - Keep them professional!

## **Finite State Machines**

Depict the states that an object may be in and the input that leads to transitions between states

Transitions

- Modeled with the input, or *trigger*, that leads to a state change
- Can also have a *guard*, which is a boolean expression to check before transitioning
- And can have an *activity*, which is a behavior that is executed during the transition

## FSM Transitions

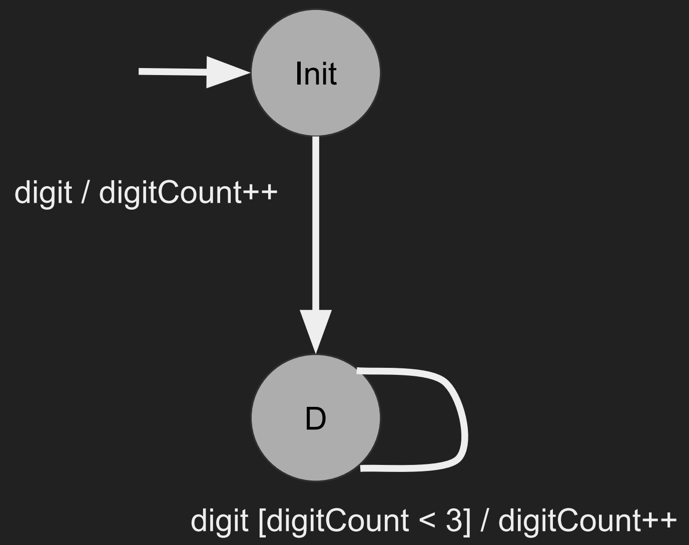
Suppose you have a series of states for a similar type of input

- Ex: A string starts with 1 to 3 digits

That series of states could be collapsed to a single state with a *guard* and *activity*

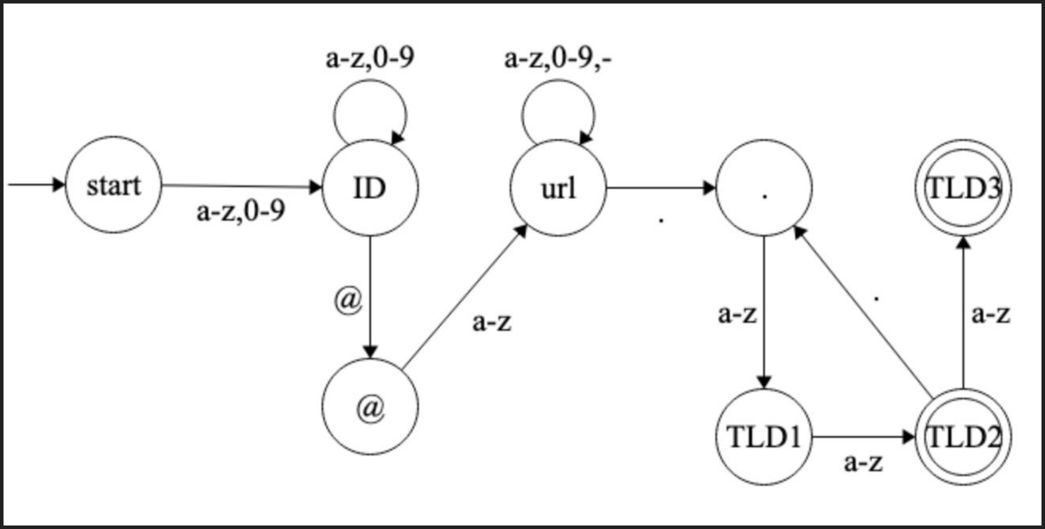
Format:

trigger [guard] / activity



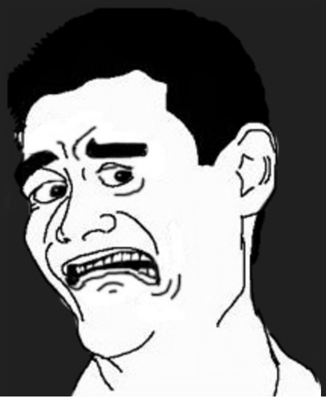
## Two Ways To Implement

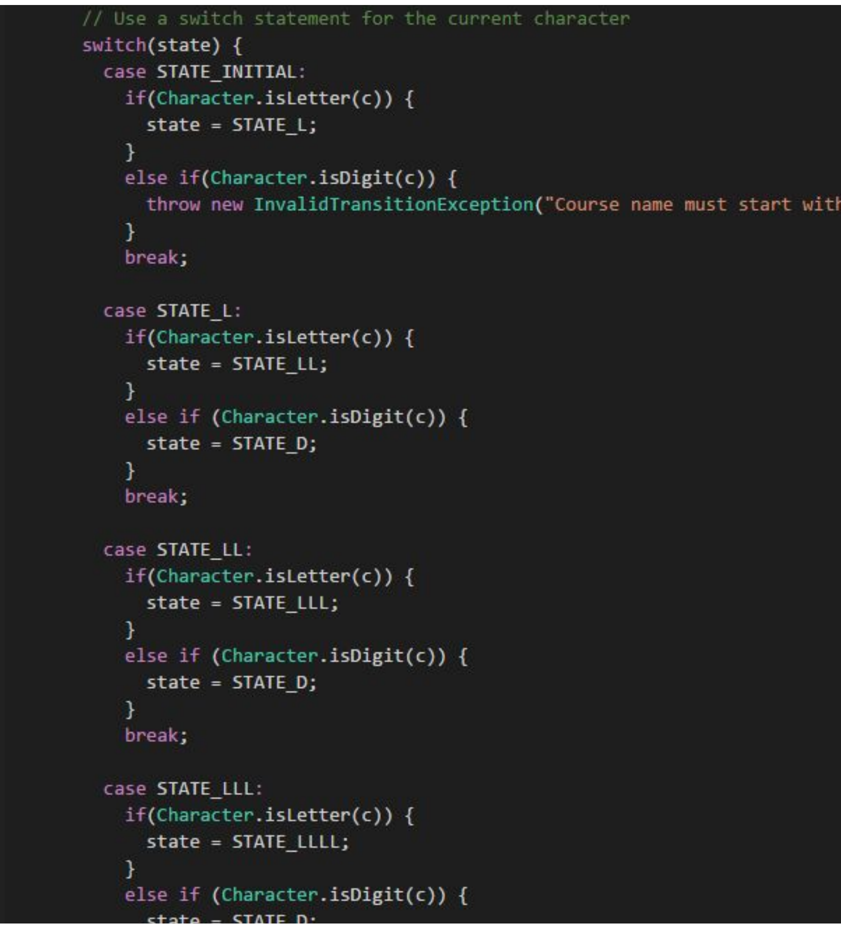
- Use a big ol' switch statement
- Use the state design pattern



## Big Switch Statement

- Hard to read
- Hard to maintain
- Gross
- Not a cool one





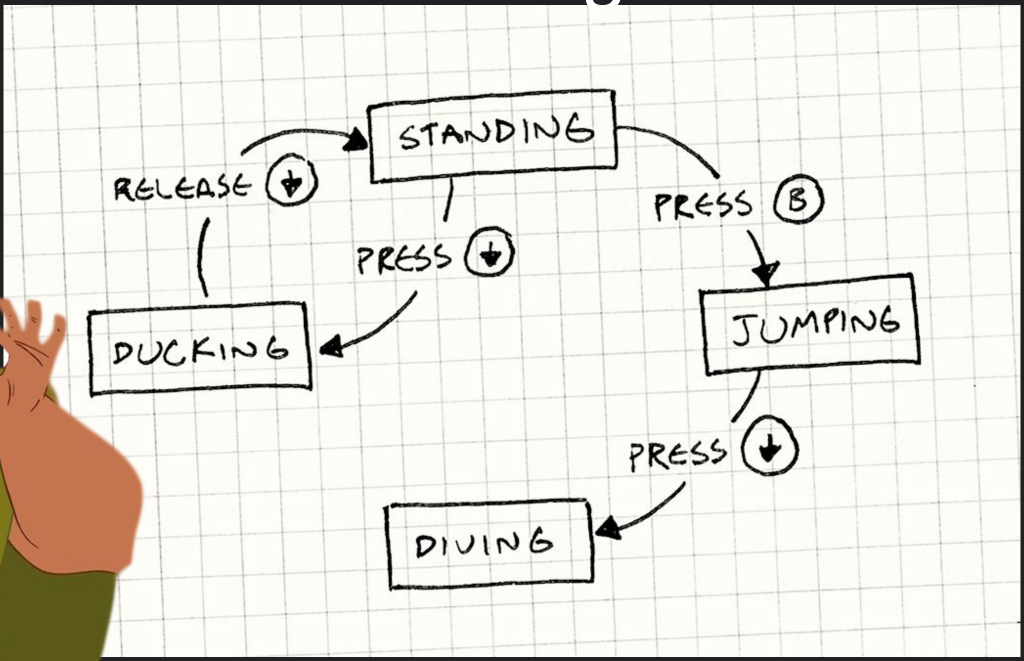
```
// Use a switch statement for the current character
switch(state) {
  case STATE_INITIAL:
    if(Character.isLetter(c)) {
      state = STATE_L;
    }
    else if(Character.isDigit(c)) {
      throw new InvalidTransitionException("Course name must start with
    }
    break;

  case STATE_L:
    if(Character.isLetter(c)) {
      state = STATE_LL;
    }
    else if (Character.isDigit(c)) {
      state = STATE_D;
    }
    break;

  case STATE_LL:
    if(Character.isLetter(c)) {
      state = STATE_LLL;
    }
    else if (Character.isDigit(c)) {
      state = STATE_D;
    }
    break;

  case STATE_LLL:
    if(Character.isLetter(c)) {
      state = STATE_LLLL;
    }
    else if (Character.isDigit(c)) {
      state = STATE_D;
```

## The State Design Pattern



http://gameprogrammingpatterns.com/state.html

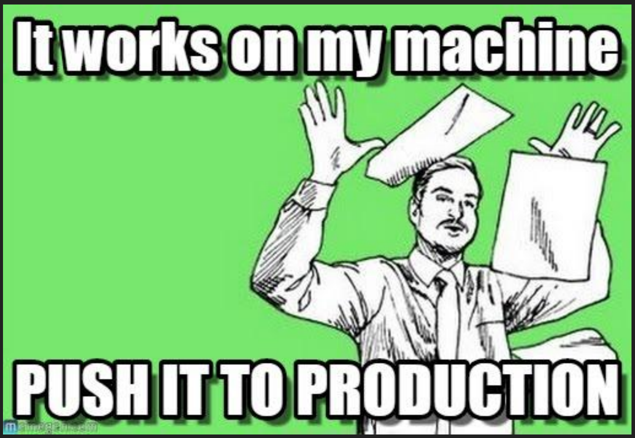
## The State Design Pattern

- States are objects
- The FSM is field of a class
- When a state runs it updates the field with a new State on a transition
- Splits up and orders logic
- Easy to maintain and extend
- Can be done with static or dynamic state classes
- Cool kids use it

## Inner Classes

- Classes can be declared whenever
- Hide internal representation of something with private internal classes
- [We will use this in conjunction with the state design pattern]

## Lab Overview



Lab activities are in Moodle

- Course Name FSM
- InvalidTransitionException
- Testing CourseNameValidatorFSM
- Implementing CourseNameValidator (state pattern)
- Test
- Deploy

All while working with your team!

## Wrap-Up

General Wrap-Up

- Deadline Reminder (see board)
- Exchange contact information with your partner
- Make a plan for finishing up the lab

Participation Outside of Lab (Guess which the teaching staff prefer?)

- If you pair program/design, **note that in the commit comments so everyone gets credit!**
- If you split the work, at least one contribution by each partner

REMINDER: We are expecting a significant contribution from all team members outside of lab!

- If you pair program/design, you **MUST** note it in your commit messages or there will be deductions
- Students who don't allow their partners to contribute will receive deductions
- Students who don't contribute will receive deductions

## Record Tasks & Owners

Tasks only get done when someone owns them!

Identify the tasks required to complete Lab 6

- Edit README.md to list the tasks required to complete Lab 6 (at top of README - should come before Lab 5 tasks)
- Add an owner to each task
- Add a deadline to each task

Deadlines should be at least 48 hours before the lab deadline so team members can help out and finish the lab if a team member runs into issues.

Notify team early if you run into problems with your tasks!