Command ======================================

* code
  + Switch => dictionary<action>
  + Identify that actions are unique blocks of code
  + “we should package them into commands”
* Slides
  + Command pattern
* code
  + Package actions => commands
* Slides
  + Hidden benefits
  + Experience usage with commands

Façade ==========================

* Code
  + See dispay of options and choice + switch in 2 places. Its complex. Lets package that complexity behind a façade
* Slide
  + Façade pattern
  + Entertainment Example
* Code
  + Display Options
* Slide Experiences

Observer ================================

* Code
  + Scanner is calling register
* Slide
  + Hollywood principle & sub comp calling parent comp bad
  + Observer
* Code
  + Change scanner => IConsumeSkus
  + Refactor into events
  + Refactor that into actions

Adapter ==================================

* Code:
  + What if we had different scanners? Identify concrete reference in scanner command
* Slide
  + Depend on abstractions not concrete
  + Adapter pattern
* Code
  + Iscanner
  + Make a new scanner without scan method
  + Create adapter to new scanne
* Slide
  + Experiences

Strategy =========================================

* Code
  + Identify tax code / switch; no commands because states id diff algorithms for calc
* Slide
  + Strategy Pattern
  + ….?
  + Model refactoring
* Code
  + Strategies
  + Still case statement!

Factory =============================================

* Code
  + Depending on state… we want to use a diff strategy
* Slide
  + Factory pattern
  + Types of Factories
  + UML ?
* Code
  + Factory Method
  + Factory Class
  + Factory class to use Dictionary of Funcs….