# CS 417
## Design Patterns

## UML Dynamic Behavior

Dr. Chad Williams
Central Connecticut State University

# Topics

- Modeling dynamic behavior
  - Sequence diagrams
  - State diagrams

# Modeling dynamic behavior

- Class diagrams model static relationships between classes
  - The structure of object model that will be created
- Dynamic modeling describes
  - The interaction of objects and
  - The order of events and actions

# Sequence diagrams

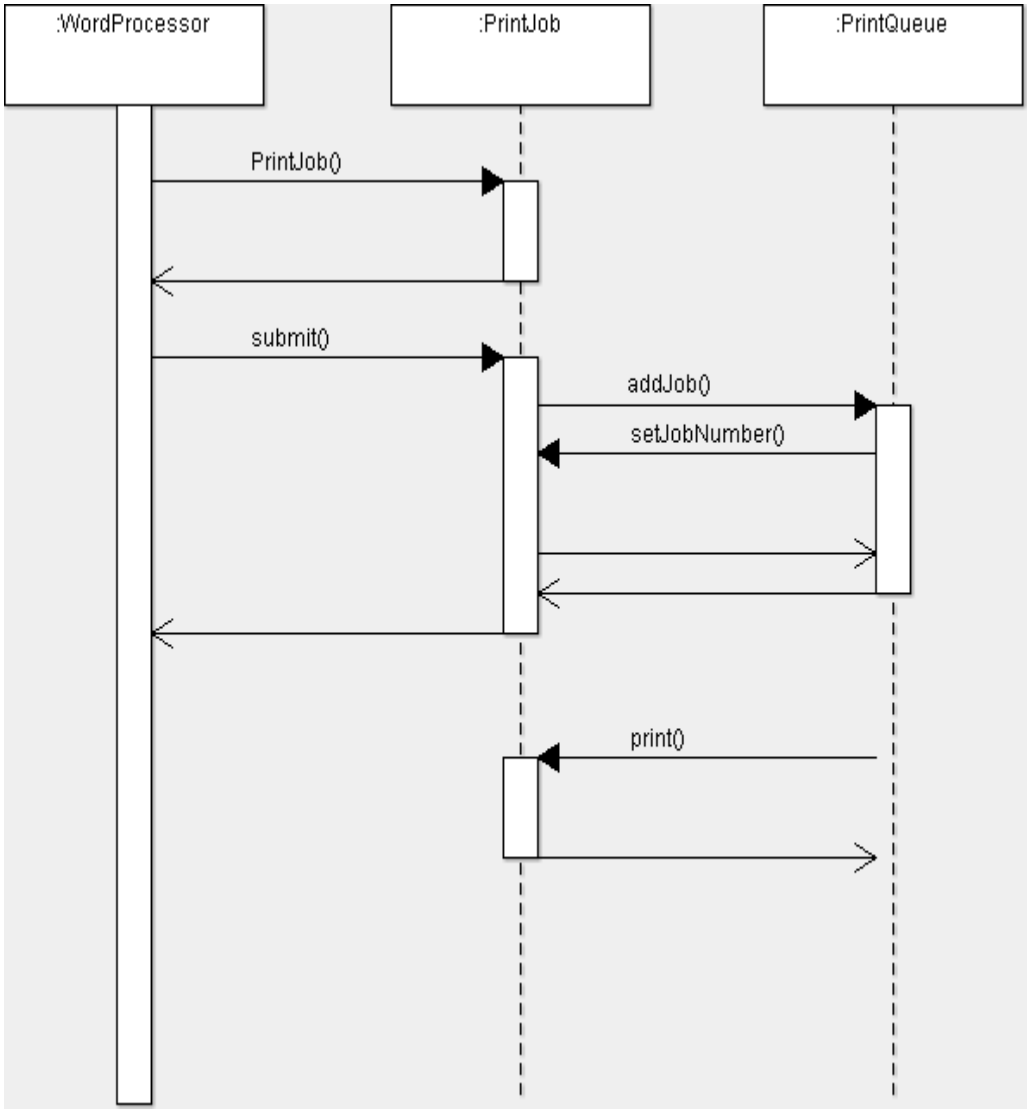
- Purpose is to highlight
  - interactions between objects
  - time ordering
- Diagrams capture
  - When objects are created
  - Methods invoked on each object
  - Order of invocation
  - The life of a function with respect to the functions it calls

# Reading a sequence diagram

- y-axis represents time in downward direction
- x-axis contains columns of the objects involved in the represented interaction
- Object columns generally ordered by sub ordinance with initial call being on left
- Solid vertical boxes indicate function is being executed
- Solid horizontal lines with solid arrow indicate object being created, or method invocation
- Dashed or solid horizontal lines with just lines of arrow indicate a method return

# Example sequence diagram

1) The object `WordProcessor` creates instance of `PrintJob`
2) `WordProcess` invokes `submit` method of `PrintJob`
3) `PrintJob` `addJob` itself to a queue on instance of `PrinterQueue`
4) The `PrinterQueue` object invokes the `setJobNumber`() method of `PrintJob` to assign a number to the new print job
5) `setJobNumber` method returns
6) `addJob` method returns
7) `submit` method returns
8) Sometime later when the print job becomes first in the queue the `print` method is invoked on the `PrintJob` object
9) The `print` method returns



# Group work

- Classes
  - Student
  - Registrar (class listings, placement)
  - Class (spots open, students registered)
- Create sequence diagram
  - Student looks up classes meeting some criteria, and registers for a class

# State diagram

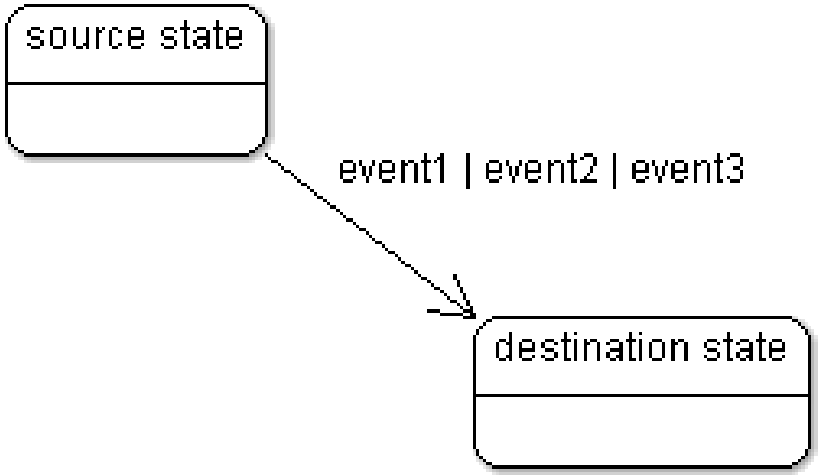
- Depicts the control of flow using concepts of *states* and *transitions*
- **State** – represents condition or situation that an object satisfies
- **Transition** – indicates relationship between objects on which an action on the first object leads to a change in state

# State diagram notation


A state

An initial state

A final state

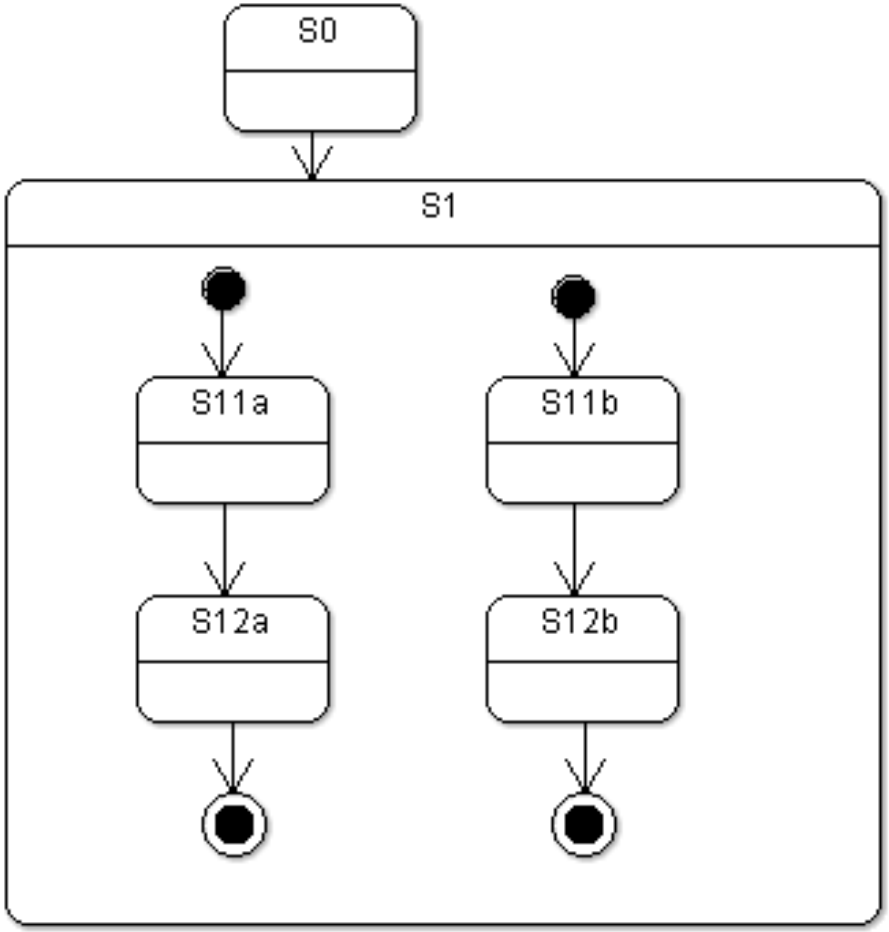


# State diagram flow

- Object begins at initial state
- While terminal state not reached
  - If triggerless transition move to next state
  - Otherwise wait until transition event occurs and then change to new state

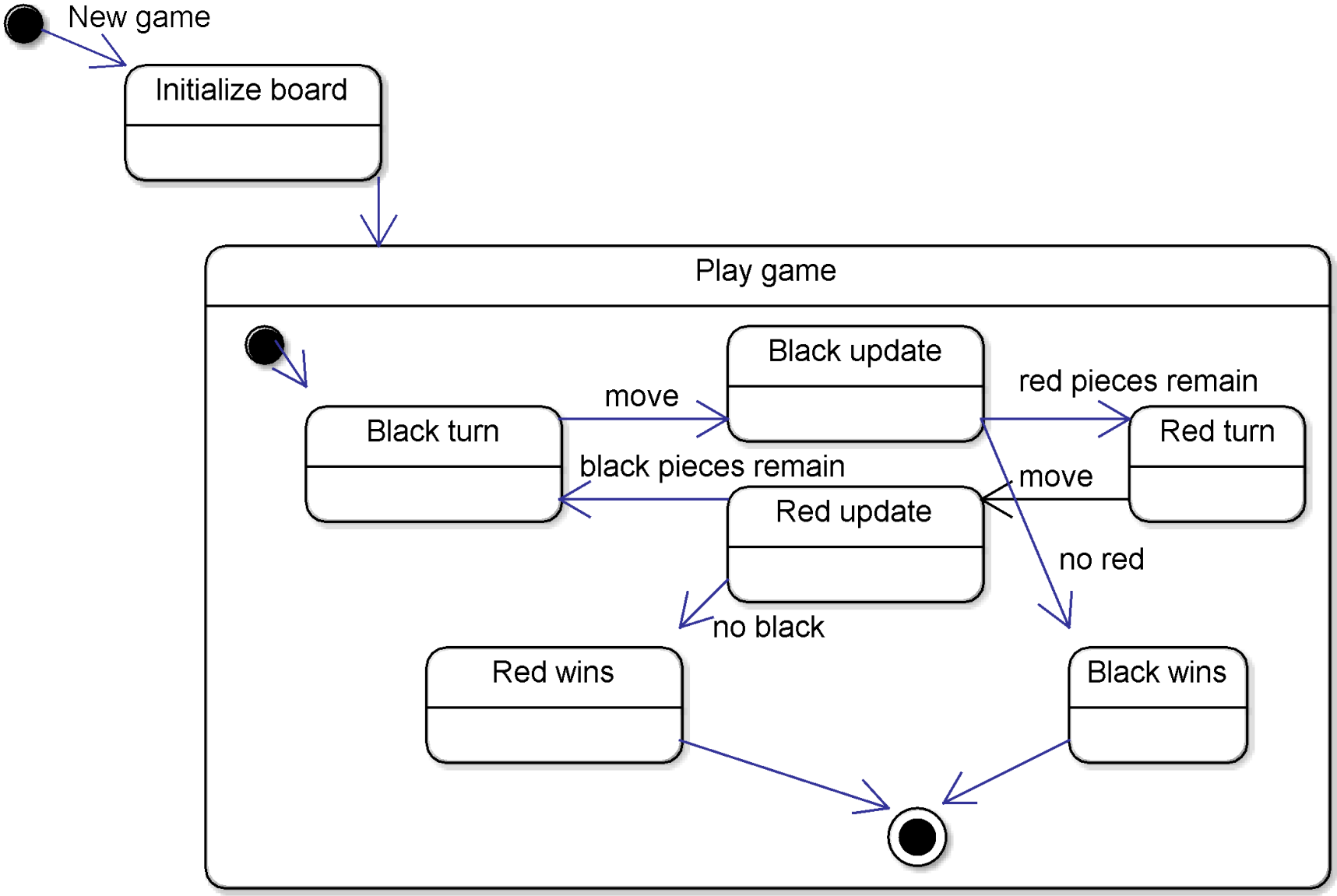
# Nested state diagrams

- A nested state represents a composite state – a state made up of other states
- Often used to describe transitions between higher level states while still capturing lower level details
- Also can be used to show concurrency
  - Entering main state simultaneously enters all initial states



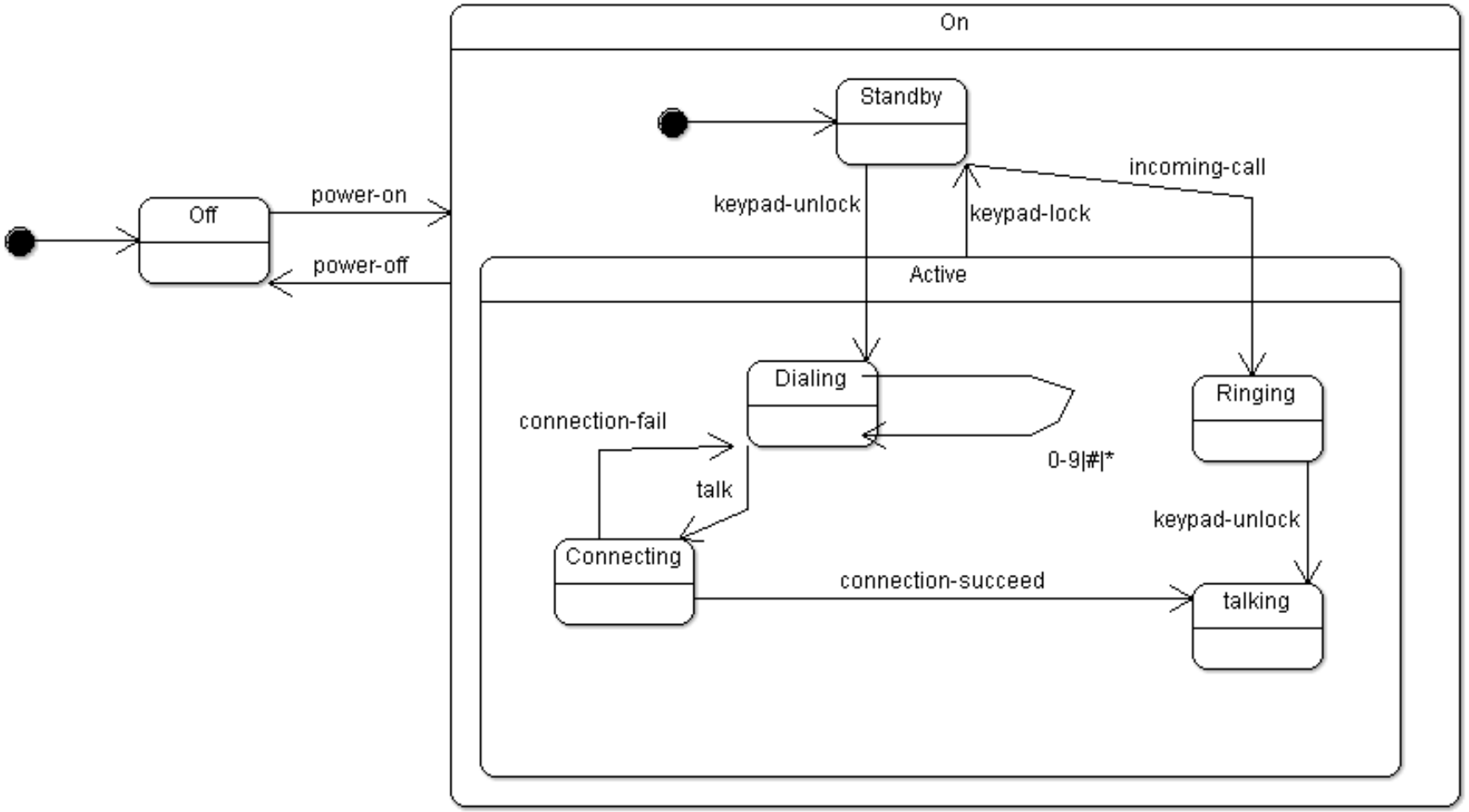
# Create state diagram for checkers

# Checkers state diagram



# Create state diagram for phone

# Example phone state diagram



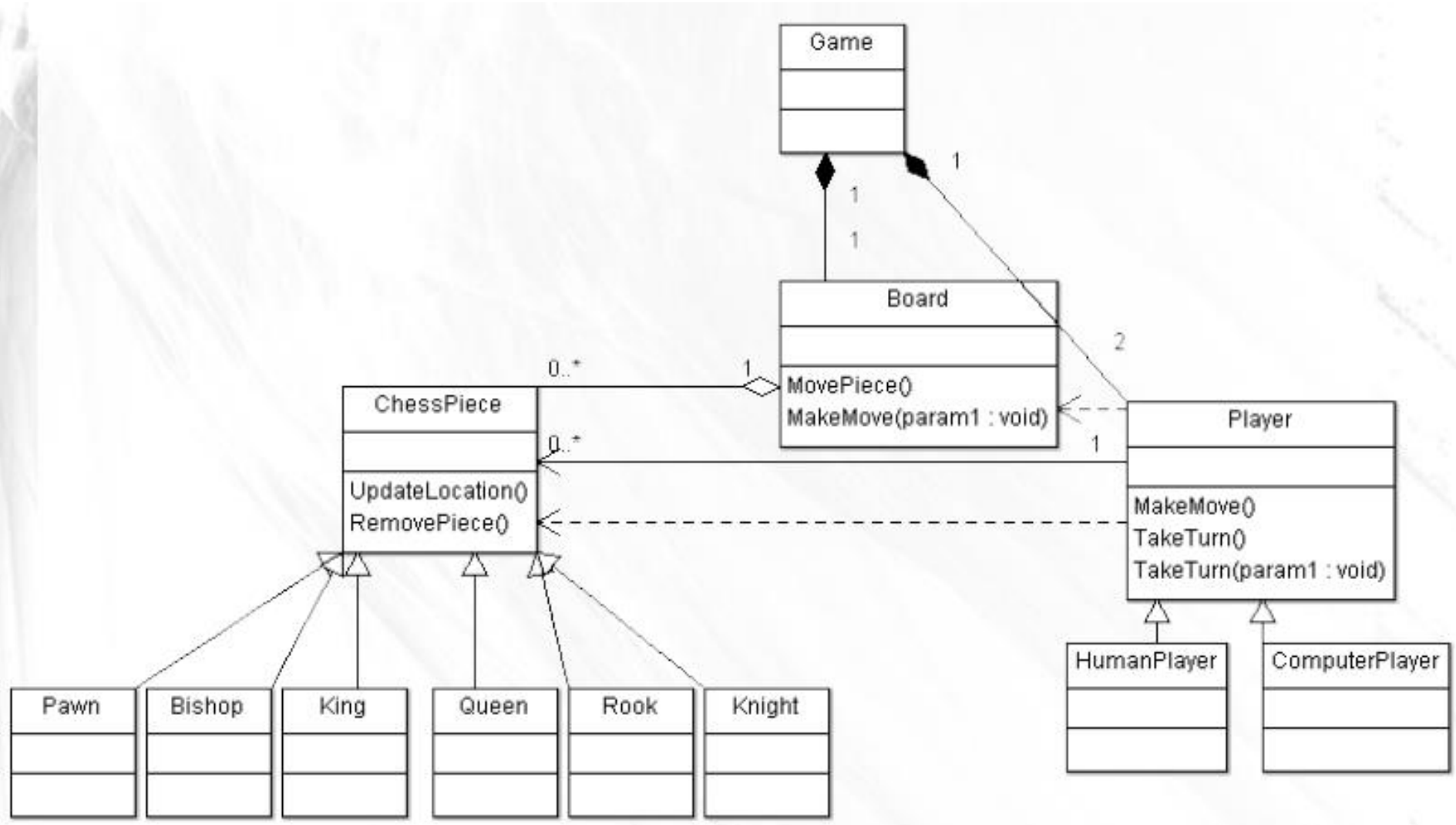
# Group work

- Create state diagram for a solar powered calculator

# Group work

- Create Chess game – Human user can play another Human user or play against computer
  - Create state diagram
  - Create class diagram
  - Create sequence diagram
    - Player takes turn

# One solution



# Sequence diagram