What sorts of things does the Model keep track of?
- All cards and their states
- What state the game is in (READY, GAME_OVER, PICKED_ONE, ...) 

What is meant by “what state the game is in”? The following diagram shows one way of thinking about the states that the game has and the “events” that trigger state changes. Hit and Pick events are sent from the user through the controller. Transient states, shown as diamonds are just entered long enough to decide a branch direction; here “yes” and “no” are the only options. As the model moves between states it also takes actions to update its data. For instance, “shuffle and Init” are the actions of shuffling the cards and initializing the layout for a new game. The state itself is often kept in a variable so changing state requires the state variable to be updated as well.

Q: What about cases where the user hits the Continue button when it’s not allowed or picks a card that shouldn’t be picked?
A: Two basic techniques can be used to deal with this problem. 1) The flow of control is modified to include error checking/exceptions. 2) Prevent the “mistake” from occurring.

The second method, where appropriate can simplify things. As part of updating the state, the model should also determine if various events are even allowable. User interface components have to be enabled in order to generate events so as the interface is updated to reflect the current state of the model, UI components are also enabled or disabled. So for instance, if 2 cards are already picked, it is impossible for a user to pick a 3rd card since cards will no longer be enabled for picking.

Tests create an instance of Model and make sure it has correct game logic by issuing the same calls that the Controller would. To verify proper state, the same query methods that would be used by the View are used and compared against expected results.

Application creates an instance of Model and supplies View and Controller.