

# Detailed Design Models and Design Patterns

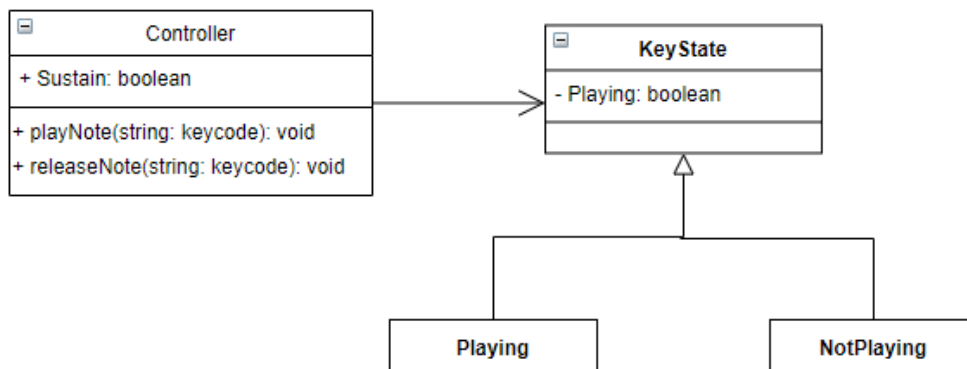
## Design Patterns

### *Structural Pattern: Composite*



The composite pattern was chosen because we have several different objects that need to be treated identically on our webpage. We have 2 different types of keys, white and black keys. The white keys are normal notes, while the black keys are sharps/flats. The keys need to be treated the exact same way when they are played, but they need to have a different value for their octave and note name, which the sound to be played are looked up from. By treating all these keys the same way, the coding will be much simpler and more readable.

## *Behavioral Pattern: State*



The State behavioral pattern is useful for our application because each key has two states, playing and notplaying. The controller will receive the command to play a certain note, and it needs to know if it is already playing or not so it can be played if it isn't, or not start playing it again if it is already playing. If a note is already playing and is played again, it would layer on top, potentially causing discomfort to the user and it might not be able to be stopped.

## **Design Class Diagrams**

## Detailed Design: Playing The Piano

