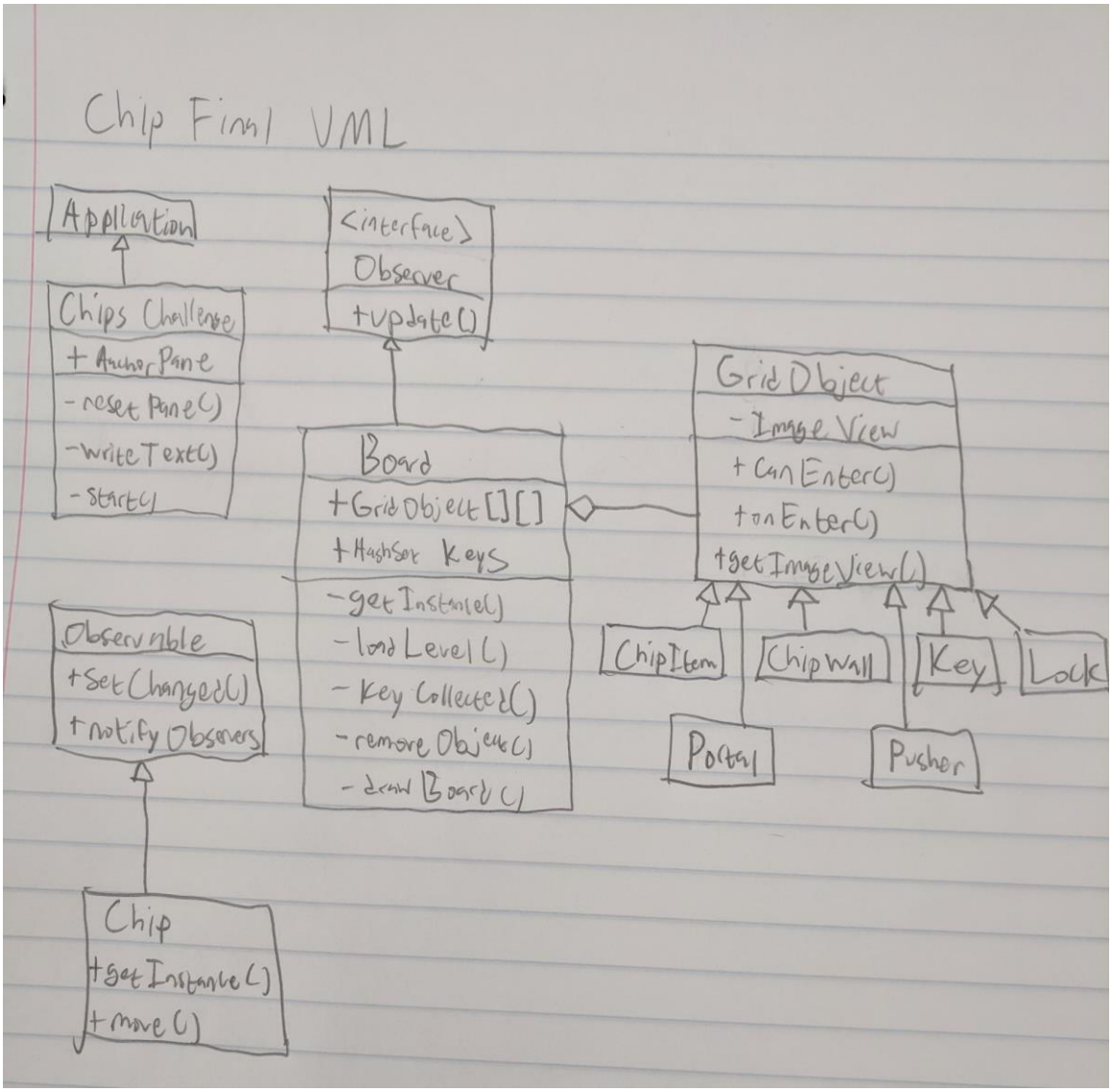
Jacob Beiter

Chip's Challenge

Final Writeup

Final UML Diagram



No considerable structural changes were made to the design shown in the milestone, simply more functionality/methods to the existing classes. Most notably, significantly more GridObjects were created, and an onEnter() method was added to GridObject to handle both collected/opened items disappearing, and the pushers pushing Chip.

Design Patterns

Pattern name: Observer	
Class name	**Role in Pattern**
Chip	Observable
Board	Observer
Purpose: The Board class observes Chip so that it knows when he moves and can redraw the grid to show the new state.	

Pattern name: Singleton	
Class name	**Role in Pattern**
Chip	Singleton
Board	Singleton
Purpose: Both the Board and Chip were good fits for the Singleton pattern for a few different reasons. There should only be one instance of each, and singleton makes sure that no more can be instantiated. Many different classes need to access the single instances too, so instead of passing references to every class, they can simply access via the static getInstance() method.	

Pattern name: Composite	
Class name	**Role in Pattern**
Board	Container
GridObject	Object
Purpose: The board keeps a 2D array of GridObjects to represent the contents of the game grid. Each individual GridObject handles creating its ImageView, and the Board uses their canEnter() and onEnter() methods to control the flow of the game.	

Final Design Discussion

The most important classes in my design are the ChipsChallenge, Board, and Chip classes. The ChipsChallenge class is the main driver of the program; it creates the game window and the other classes, as well as handling the keyboard inputs. If they're arrow keys, the key press is passed to the Chip class, which queries the Board to see if he can move. He updates his image and sets himself as changed; he extends Observable, and is observed by the Board so it knows when to redraw.

The Board handles most of the actual functionality of the level; it keeps a 2D array of GridObjects that, along with Chip's position, represent the level state. It reads the level from a .txt file, where different comma-delimited characters represent different GridObjects. It also handles moving the position of the objects' images, to only view a 9x9 window into the larger 25x25 level.

The GridObjects (e.g. Wall, ChipItem, Key, Lock, etc) are all of the things in the level that aren't Chip or a blank tile. They all inherit from GridObject, and their implementations of the canEnter() and onEnter() functions define their behavior.

If I were to re-do this project from scratch, I wouldn't change too much – overall, I'm pretty happy with my design! I might have to tweak things if I were to add additional features, though. If I were to add different terrain, fire and water would be easy to do (they would simply be their own GridObjects), but ice would be harder. It might be helpful to have a different notion of terrain in addition to the objects – sliding on ice would be handled through the terrain, and there would be a GridObject for the reflectors/bouncers. Additionally, I would need to multithread it if I were to do animations or enemies.