Shane Johnson

10/17/18

Chips Challenge Discussion

In my final design I utilized the singleton, observer, and strategy design patterns. This allowed me to create the most adaptable game given the requirements. The singleton pattern included the ChipMap, FireShoe, and Portal classes. All of these classes were just a single instance of a class which could only run once within the system (ChipGame). These three classes made it easy to call upon the location, imageView, etc. from each class to be used for instructions in the ChipGame system. The observer pattern utilized the Bug and the Chip class. In this pattern, the bug observed chip. This allowed the bug to move in the direction that chip moved (similar to the pirate ship following the pirate). One design decision I made in this pattern was to only allow the bug to move left and right. I found that when the bug could move in all directions it was impossible to win. The bug is only featured on the first level, but the left and right movement of the bug comes from the implementation of the observer patter. The final pattern I used was the strategy pattern. This pattern is how I was able to create the doors and the keys. Each color door and key was just an object of the Key and Door instances. These objects are easily removed and added at any location on the map. If I could make design changes know what I know now, I would have the bug move left and right on its own, rather than moving after observing chip. I would then use the observer pattern to trigger if chip and the bug are at the same location. I would also change how I created the obstacles in the ChipMap class. I would make obstacles an instance (similar to door) and I would then make Shoe and instance. This would allow me to implement the strategy pattern on different obstacles and shoes – making it easier to change where I want to add and remove obstacles. The change I would make would be a small one. I would add a life count for chip and exit the game once all lives were gone. As of now, chip has unlimited lives. I would say that I learned a lot about design patterns and solutions during this mini project. I had a plan for how I would attack the problem, but I did not set hard deadlines for when I wanted to complete specific parts. This allowed me to take a more agile approach and “chip” away at the task at hand. If I found a better solution in the middle of my code, I would take that path if it had the best outcome.