Design patterns:

1. Singleton: We implemented the Singleton design pattern so that there is only ever one SantoriniCLI object created. The constructor of SantoriniCLI is private, and there is a private instance variable of itself. There is also a get_instance method to return the private instance variable of itself.

2. Observer: Each player functions as an observer, and each player's pieces functions as a subject. When a player moves a piece, the location of the piece updates. When a piece moves, it broacasts the level of its new location to the player (subject), which can then notify that that player has won if the piece is on a level 3 building.

3. Command: We implemented a Command class which stores two lists: a history of states and a future of states, and can call the undo() and redo() methods. Each position of the board (incl. worker locations and building heights) is stored in the history list. If undo() is called, the current position is appended to the future list, before it is is updated to the last element of the history list. If redo() is called, the current position is appended to the future list, before it is updated to the last element of the future list. The future list is cleared after every turn. If the history or future lists are empty, nothing happens.

4. State: To handle the different types of AI players, we used the state design pattern. The PlayerContext class acts as the context interface which sets its state as one of the types of players. The CLI holds a PlayerContext object and asks it to change its state to whichever type of AI player needs to make a move. Then, CLI/Game asks the PlayerContext object to make a move, for which the PlayerContext object tells its state to make a move, and it returns the result.

Kassi:

In this assignment I learned how to successfully pair program with GitHub, and how to navigate GitHub through the terminal. I learned how to implement the observer and singleton design patterns. I also learned how to implement the basic functionalities of the CLI, game and random player object.

Robert:

I learned how to throw and catch exceptions, check for invalid moves/builds, and implement
the functionalities of the heuristic player. I also learned how to use design patterns to my advantage
to make organizing a project easier. I had a positive experience completing this project as
I was able to combine much of what I learned from the first 5 psets.