2. Class and method declarations

++-- 2.1 Model

Within the model of the program, various class attributes and methods have been added/removed/changed. However, they are not major design revisions, and instead, are decisions to split out functions, implement getters/setters, and other trivial stuff.
We don’t think it is necessary to avoid these revisions, as we aren’t using a waterfall type development process, a general model to improve upon when programming works well enough.

++-- 2.2 Controller

The controller also has faced some changes in its implementation, such as identifying players by index to identifying by player objects, etc.
These changes also aren’t major, so the prior stance on not needing to avoid these revisions still holds true.

++ 2.2.1 SaveFile

The saving functionality has been separated out from the (main) controller. It was originally included in the main controller because we did not consider multiple classes for anything outside of the model. We then wrongly anticipated the saving mechanism would take a long time to develop and splitted it out for easier concurrent development.
To not anticipate and prepare for difficult development would be stupid, and now we have experience that not only the model, but potentially also the view and controller need multiple classes, we think that we could be more aggressive in the initial class splitting of software requirements.

++ 2.3 View

The view has been created after the fact because we initially thought that the view would be rather simple and was logically tangentially related to the controller enough that we could just handle both in the same class. We were wrong.
The solution is simple, don’t be stupid, conventions are conventions for a reason, follow them.

4. Important design decisions

++ 4.3 Board data structure

> The board is stored as a simple array and the game logic is programmed around that. While it does severely limit the future expandability of map layout and traversability choices without major codebase overhauls, this allows for a vastly quicker turnaround development time for a project with a tight schedule and no plans to further extend upon.
>
> To achieve this, the only thing that is needed to be sidestepped, is the distinction of the In Jail tile and Just Visiting being 2 functional distinct, separate nodes connected to Shek O. The tile unification is easily achieved by giving the player a separate status variable to account for being jailed.

The above design is added after the fact because at the time of writing the API design document initially, we discussed using a directed graph structure for the map layout for advanced expandability, but ended up vetoing this notion as it is labour intensive for the short time we had. We did not think that the inclusion of not doing something is necessary for the design document. However, as this decision is a major time-saver which has major impact on future expandability, it might be wise to also include it for future reference to advocate for schedule-appropriate program complexity.
In the future, decisions no matter small, or to not do something, should still be considered for documentation.