Patterns we have used

1. Creator

Game *contains* Domino objects; hence we give responsibility to it for creating Domino objects when the game begins

2. Low coupling

Where possible, level best effort has been put to keep code minimalist, independent and unrelated to other code.

3. High cohesion

Level best efforts have been put to ensure that classes represent a single well-defined entity and with methods which are responsible for a single well defined task

4. Controller

The Game class acts as the controller for the input system events – the clicks (actionEvent) that allow choosing, dropping and placing a Domino.

5. Protected Variations

Has been used in several classes especially in the Game class.

Patterns we would have used

1. Factory Pattern.

The AI player was not implemented due to time constraints, but we would have used a Factory class to do so. It would create and built the AI player based on the input it receives from other classes such as SetUpPlayers and SetUpMenu.

2. Observer class

We would have used the observer pattern to keep track of current left and right dominos and their details as they constantly change throughout the game. We could also use it to possibly keep track of placing, dropping and choosing Dominos and its association with the current player. The GameUI and Game could be our *possible* observers with the Domino and GridBox classes *possibly* being the observables. The observers would watch for the above-mentioned changes.