Design Pattern Document

We have used the creator design pattern in our code frequently, such as in classes ColourModes, FourHumanPlayer, TwoHumanPlayer, PlayersInformation, and multiple others. The criteria we used to determine what classes had to create what instances of other classes was that one class usually has data that is needed to create an instance of the other class. For example, the class PlayersInformation requires the number of human players and the number of computer players partaking in the game, and classes such as TwoHumanPlayer and FourHumanPlayer take the users input for that information to create an instance of PlayersInformation.

Another design pattern we used was high cohesion. One example where we used this is in the class DominoTile, which sets up the domino tiles for the game and provides methods for rotating and getting information on the tiles such as the terrain type or number of crowns. All methods have a specific functionality, and all are related to each other. This is also used throughout the rest of our code.

Another design pattern we used is low coupling. The classes dealing with game settings like PlayersInformation and ColourModes have very few connections with classes like DominoTile and TileStack. We have tried to make sure that connections we have between classes are only there because they are necessary.

A design pattern we could have used is commands, or a CommandHolder class. We could have created smaller command classes for functionality involving listening to user actions. A CommandHolder class would be able to access these commands and pass them on to where it is needed. For example, we could put the code for user actions separate from the UI code in classes like PlayersInformation and ComputerDifficulty.