DESIGN PATTERNS
Group 8

1. *Information Expert:* We used the information expert pattern to assign responsibilities to the classes which had the most information available to fulfill it. For example, the GameTile class is used to construct one individual panel of the game board. Another example is the class Player that is in charge of creation of players as objects. Each player (the object of this class) has methods that are special to them and these objects are able to keep and provide all the required information by invoking certain methods to fulfill the duty assigned to them.

2. *Creator:* The creator pattern has been implemented in a couple of classes. One example is the GameBoard class where new instances such as the game tiles are created. Second example is the BuildAssets class which is responsible for creating several new instances.

3. *Low coupling*: The ideal example of this pattern is the class GameSettings. We used multiple different classes to reduce the amount that GameSettings class had to do. By doing so we can change it easier without it impacting the rest of our code as much as it would have if we used a lower number of classes. Directing all our connections to one class to avoid high coupling.

4. *High Cohesion:* The group created classes that were individually assigned one task, hence increasing cohesion between classes. For example, the InGameUIPanel that is responsible for showing UI elements for the user. Another example would be GameTile class which is used to construct one individual panel of the board i.e. tiles.

5. *Controller:* In order to handle the input systems more efficiently, we have the InputManager class. This class handles incoming user input and responds accordingly, like handling mouse input.

6. *Polymorphism:* There are two classes that generate objects with the same behaviour. First is the class Player, it can be either human or AI, based on the attributes we can give them polymorphic behaviour. The second example is the BuildAssets class. If the user selects the load game, then the BuildAssets class uses the load game constructor else, it uses save game constructor if save game is chosen.

7. *Indirection:* The best suitable example for this pattern is the InputManager class where we take input from GameTile class, process it through the InputManager and send it back to GameTile.