Our design uses a few different GRASP patterns to achieve its functionality such as creator, information expert, controller, low coupling and high cohesion. Information expert is one of the most widely used patterns in our design and is seen in the classes: SettingsMenu, Player, and Piece. Through this pattern we have made SettingsMenu responsible for setting the various options that players choose for their game as it has the settings information required to perform its functionality. Similarly, Player and Piece each have the necessary information about players and pieces respectively to handle functionality relating to these 2 domain concepts. Both low coupling and high cohesion are also widely used in our design by classes such as: Player, Piece and NumComPlayer. As previously mentioned, Player and Piece perform most of the functionality relating to their respective domain concepts and in turn are very cohesive as their responsibilities are related and focused. As both of these classes have very little dependence on other classes in our project, they have a very low level of coupling. On a similar note, the NumComPlayer class is responsible for solely providing a valid range of options for the number of computer players that the user can add to their game and then displaying these options to the user via a GUI. Since the responsibilities of this class are highly focused and mostly independent of other classes in the project, it is highly cohesive and lowly coupled to other classes. The creator design pattern is used in the class Blokus as that class is used to create instances of the Board class representing the Blokus game board. Since the overall game itself will always contain a Board object, we assigned our Blokus class to create instances of these objects. Finally, both the Board and Blokus classes make use of the controller design pattern. Board is a session controller as it controls many aspects of the actual gameplay such as validating moves, displaying the pieces that have been placed correctly and even placing pieces for computer players based on their difficulty setting. The Blokus class is more like a facade controller as it controls the overall functionality of the game such as: saving and loading, providing hints, counting turns, determining when the game ends, displaying the winner, and providing the main game GUI and interactable buttons to users.

If we had more time to continue work on our project, we would try to use the low coupling and high cohesion design patterns on our Board and Blokus classes as they are quite verbose compared to our other classes and have a wide range of responsibilities that in some cases are not that closely related or focused. Additionally, we would use information expert in order to create a ComputerPlayer class that was responsible for information about a computer player (such as difficulty) and would make moves for it.