The Monopoly simulator has three classes: *Game*, *BoardLocation*, and *Player*. The *Game* class manages most of the actions in the game. It takes one argument: the number of players in the game. When a *Game* object is created, lists of *Player* objects and *BoardLocation* objects are created and stored in attributes of the *Game* object. When a *Game* object is initiated, the initial number of houses and hotels, the Chance and Community Chest cards, and counters used during game play are also stored in *Game* class attributes.

A game is started by using the *play* method in the *Game* class. The *play* procedure shuffles the players and then passes them one a time to the *take\_turn* method (also in the *Game* class), looping until the game has gone on too long or one player looses.

The *take\_turn* method handles all of the gameplay actions, including the following: rolling the dice, moving the player, buying properties, paying rent, and relieving Chance and Community Chest cards. It loops if a player rolls doubles or is sent to a secondary location by a card. When a player is in jail, the procedure has the player attempt to roll doubles. If the player fails on all three attempts, the player will use a “Get Out of Jail Free” card if they have one on hand. Otherwise, they will pay the $50 fee.

Many strategic options that a player has throughout game are decided using certain strategy attributes the *Player* class. When the *Player* objects were instantiated, they required certain strategy parameters, including *buying\_threshold* and *development\_threshold*. The parameter *buying\_threshold* influences a player’s decision to buy a property. If buying the property will cause the player’s amount of money to drop below *buying\_threshold*, they will not buy it. Similarly, if buying a house or hotel for a property will cause a player to drop below their *development\_threshold,* they will not buy the building.

As for times when a player does not have enough money to pay a debt, a player will first sell her buildings and then mortgage properties. The *change\_money* method in the *Game* class handles this scenario. First, the method loops through properties that have been developed by the player with houses and hotels and sells their buildings one by one. If the player has sold all of their buildings and still does not have enough money, *change\_money* will then mortgage the player’s properties one by one. If the player does not have enough money to pay their debt after the mortgaging process, they lose the game.