Controller

* HanabiController
* AIController
  + Testing if bestPlay() return the right PlayValue with the specific information given.
    - Case 1: If any card in the AI Player’s hand is fully known and is the next card in a fireworks pile, then it is the best play
      * Firework pile contains Blue 1, AI has Blue 2 in hand
      * Path: return index of the card Blue 2, playValue of 0
    - Case 2: If a card, from a known rank, could be the next card in a fireworks pile, then it is the next best play
      * Firework pile contains Blue 1, AI has Unknown Colour 2 in hand
      * Path: return index of the card Unknown Colour 2, playValue of 1
    - Case 3: If a card, from a known colour, could be the next card in a fireworks pile, then it is the next best play
      * Firework pile contains Blue 1, AI has Blue Unknown Rank in hand
      * Path: return index of the card Blue Unknown Rank, playValue of 2
    - Case 4: If nothing is known about any of the AI Player’s cards, then a random card is selected as the best play
      * AI has no known card
      * Path: return index of the first card, playValue of 3
  + Testing if bestDiscard() return the right PlayValue with the specific information given.
    - Case 1: If any fully known card in the Player’s hand is a duplicate of a card in a fireworks pile, then it is the best discard.
      * Firework pile contains Blue 2, AI has Blue 2 in hand
      * Path: return index of the card Blue 2, playValue of 0
    - Case 2: If a fully known card is not the last copy of that card, because not all the other copies are in the discard pile, then that is the next best discard
      * Discard pile not contains Blue 1, AI has Blue 1 in hand
      * Path: return index of the card Blue 1, playValue of 1
    - Case 3: If a card, from a know colour or rank, is not the last copy of that card, because not all the other copies are in the discard pile, then that is the next best discard
      * Discard pile contains no card, AI has Blue Unknown Rank in hand
      * Path: return index of the card Blue Unknown Rank, playValue of 3
    - Case 4: If nothing is known about any of the AI Player’s cards, then a random card is selected as the best discard
      * AI has no known card
      * Path: return index of the first card, playValue of 3
* Test if bestInfo() return the right PlayValue with the specific information given.
* ServerComm
* JSONParser
* 4.3.1. bestPlay()
* To test the bestPlay() method from the AIController, we will use four specific tests to not only check the right playValue() but also the right return value. For the first test, the best play possible for the method is when a card in AI Player’s hand is fully known and is the next card needed in a fireworks pile. We set up the firework pile to contain only a Blue 1 card. AI’s hand will include a Blue 2. The test should return the index of Blue 2 card with best playValue of 0.
* The second best play is when an AI’s card has a known rank that can possibly be the next card in the fireworks pile. To test this case, we set up the fireworks pile to contain a Blue 1, and AI’s cards contain a Unknown Colour rank 2. The index of such card should be return along with the playValue of 1. The following test for third best play is when an AI’s card has a known colour that can possibly be the next card in the fireworks pile. For this one, we set up the fireworks pile to contain a Blue 1, and AI’s cards contain a Blue Unknown Rank. The index of such card should be return along with the playValue of 2. The last test is when nothing is known about any of the AI Player’s card. The index of the first card will be returned with the worst playValue of 3.
* 4.3.2. bestDiscard()
* The next method to be tested AIController is bestDiscard(), we will use four another four specific tests to not only check the right playValue() but also the right return value. Firstly, the best card to discard is when a card in AI Player’s hand is a duplicate to one in the fireworks pile. We set up the firework pile to contain only a Blue 1 card. AI’s hand will also include a Blue. The test should return the index this card card with best playValue of 0. The next best card to be removed is a fully known one, and such card is not the last copy. To test this case, we set up the fireworks pile to be empty, and AI’s cards contain a Blue 1. The index of such card should be return along with the playValue of 1. The third best to be discard is when an AI’s card has a known colour or rank is not the last copy of the card since not all other copies are in the discard pile. For this one, we set up the discard pile to contain a no card, and AI’s cards contain a Blue Unknown Rank. The index of such card should be return along with the playValue of 2. The last test is when nothing is known about any of the AI Player’s card. The index of the first card will be returned with the worst playValue of 3.
* 4.3.3. bestInfo()
* The judge for AI’s give information can be determined through bestInfo() method. As mentioned in the DD, there are 3 playValue that this method can give. Thus, we will design 3 specific test cases for such method. The first test case aiming at completing information for the card that is next card in the fireworks pile. We set up a dummy player with a Blue 2 card and he knows that it is a Blue of Unknown Rank. The expected best info to give is the Rank for said card. The index of the player and the card combines with the Rank of 2 should be returned along with PlayValue of 1. Rank 5 card is the most important card possible, and there is only 1 for each color. Therefore, we set up a player with a card of Rank 5 of any colour. The test should return the index and the information of the card with PlayValue of 2. Last test case will aim at giving information to the Player who knows the least about their cards, with the information that would tell them the most. In this case, we would set up one player with known cards in their hands. The other player would have 3 Blue cards and 2 Red cards all of them Rank 1. The information given is expected to be Blue with the PlayValue of 3.
* 4.3.x Send methods
* Each of the send method from ServerComm class make use of the JSONParser to communicate with the server. The following tests are meant to check if the JSON messages are created and received as described in the DD. Thus, we will use a mock server that respond to messages we send at different states. For sendCreate(), the inputs needed for numPlayers, timeout, nsid and force are 5, 60, abc123 and false in this respective order. The message received from the server should be JSON message saying that the game is created along with the game-id and secret token. As for sendJoin(), we will use a game-id 2341 with a secret token “somethingSecret”. The reply can be any of the three messages status: joined, full or no such game found. The sendPlay() will be tested using a input of 1 to point to the index of the first card on the current player’s hand. The received output should be a JSON indicating the move is play a card and position 1. Testing sendDiscard() is very similar with an integer index input and similar JSON message return. sendInfo() will receive integer index of player 1 and a string information of “b” to indicate color blue. The JSON message return will indicate the index of the player along with the information that they are given.