**Testing Plan for Controller**

1. **Constructor Testing:**
   * Condition: Testing if the constructor initializes all required fields properly.
     + Example: Initialize the controller with a non-null **Readable** and **Appendable** object, along with a valid **World** instance.
     + Expected: Verify that the fields are initialized correctly and that the controller is ready to use.
   * Condition: Testing if an **IllegalArgumentException** is thrown when any parameter is null.
     + Example: Initialize the controller with a null **Readable** object and a valid **Appendable** and **World** instance.
     + Expected: Verify that the appropriate exception is thrown.
2. **playGame() Method Testing:**
   * Condition: Testing the method with various inputs, including valid and invalid **World** instances.
     + Example: Initialize the method with both valid and null **World** instances.
     + Expected: Verify that an **IllegalArgumentException** is thrown for the null **World** instance.
   * Condition: Testing the behavior when the maximum number of turns is 0.
     + Example: Set the maximum number of turns to 0 and initialize the game.
     + Expected: Ensure that the game does not start and an appropriate message is displayed.
   * Condition: Testing the behavior when each option is selected.
     + Example: Simulate the selection of each option in the game menu.
     + Expected: Verify that the corresponding action is performed, and the outputs are displayed accurately.
3. **Edge Case Testing:**
   * Condition: Testing the behavior when the maximum number of turns is 1.
     + Example: Set the maximum number of turns to 1 and initialize the game.
     + Expected: Ensure that the game ends after the first turn.
   * Condition: Testing the behavior when the maximum number of turns is negative.
     + Example: Set the maximum number of turns to -1 and initialize the game.
     + Expected: Verify that an **IllegalArgumentException** is thrown.

**Testing Plan for DisplayMap, DisplayPlayerInfo, DisplayRoomInfo, DisplayTargetInfo Command**

1. **execute() Method Testing:**
   * Condition: Testing the method with a valid **World** instance.
     + Example: Initialize the command with a valid **World** instance and execute it.
     + Expected: Verify that the method of the **World** class is executed without errors.
   * Condition: Testing the method with a null **World** instance.
     + Example: Initialize the command with a null **World** instance and execute it.
     + Expected: Verify that the method gracefully handles the null input and does not cause any unexpected errors.

**Testing Plan for PlayNextRound Command**

1. **execute() Method Testing:**
   * Condition: Testing the method with a valid **World** instance and the game not in the final round.
     + Example: Initialize the command with a valid **World** instance and execute it.
     + Expected: Verify that the **playNextRound()** method of the **World** class is executed without errors.
   * Condition: Testing the method with a valid **World** instance and the game in the final round.
     + Example: Simulate a scenario where the game is in the final round.
     + Expected: Verify that the **playNextRound()** method of the **World** class is executed, and the winner is appropriately displayed in the **Appendable** output.

**Testing Plan for AddComputerPlayer and AddHumanPlayer Command**

1. **execute() Method Testing:**
   * Condition: Testing the method with a valid **World** instance and valid user input.
     + Example: Provide valid input to the command during execution.
     + Expected: Verify that the method of the **World** class is executed without errors, and the appropriate confirmation message is printed.
   * Condition: Testing the method with an invalid or null **World** instance.
     + Example: Provide a null **World** instance during execution.
     + Expected: Ensure that the command handles the null **World** instance appropriately, avoiding any potential runtime errors.

**Testing Plan for displayListOfRooms() Method in the Mansion Class**

1. **Testing the Method's Output:**
   * **Condition:** Testing the method's output to ensure the correct display of rooms.
     + **Example:** Provide a pre-defined list of rooms to the **displayListOfRooms()** method.
     + **Expected:** Verify that the method prints the list of rooms along with their corresponding room names and room numbers accurately.

**Testing Plan for getTypeOfPlayer() Method in the Player class**

1. **Functionality Testing:**
   * Condition: Testing the method with a player of type human.
     + Example: Create a player instance with type 0 (human).
     + Expected: Ensure that the method returns 0, indicating the player is of human type.
   * Condition: Testing the method with a player of type computer.
     + Example: Create a player instance with type 1 (computer).
     + Expected: Verify that the method returns 1, indicating the player is of computer type.

**Testing Plan for Newly Added Method in the RoomInfo Class**

1. Testing Plan for **removeWeapon** Method:
   * **Condition**: Testing the method with a valid Weapon instance.
     + **Example**: Provide a valid Weapon instance to the method.
     + **Expected**: Ensure that the specified weapon is removed from the room's list of weapons.
2. Testing Plan for **displayTarget** Method:
   * **Condition**: Testing the method when the target is in the room.
     + **Example**: Set the target to be in the current room.
     + **Expected**: Verify that the method prints "Target is here!" to the console.
   * **Condition**: Testing the method when the target is not in the room.
     + **Example**: Set the target to be in a different room.
     + **Expected**: Verify that the method prints "Target is not here!" to the console.
3. Testing Plan for **displayPlayers** Method:
   * **Condition**: Testing the method when one or more players are in the room.
     + **Example**: Set one or more players to be in the current room.
     + **Expected**: Ensure that the method displays each player's name who is currently in the room.
   * **Condition**: Testing the method when no player is in the room.
     + **Example**: Set no player to be in the current room.
     + **Expected**: Ensure that the method prints "No player is here!" to the console.
4. Testing Plan for **displayWeapons** Method:
   * **Condition**: Testing the method when the room has no weapons.
     + **Example**: Ensure that the room's weapon list is empty.
     + **Expected**: Verify that the method prints "No weapons in this room." to the console.
   * **Condition**: Testing the method when the room has one weapon.
     + **Example**: Set one weapon in the room's weapon list.
     + **Expected**: Ensure that the method displays the name and power of the weapon in the room.
   * **Condition**: Testing the method when the room has multiple weapons.
     + **Example**: Set multiple weapons in the room's weapon list.
     + **Expected**: Ensure that the method displays the count, name, and power of each weapon in the room.
5. Testing Plan for **displayNeighbors** Method:
   * **Condition**: Testing the method when the room has neighboring rooms.
     + **Example**: Set neighboring rooms for the current room.
     + **Expected**: Verify that the method prints the room numbers and names of all neighboring rooms.
   * **Condition**: Testing the method when the room has no neighboring rooms.
     + **Example**: Ensure that the room's neighbor list is empty.
     + **Expected**: Verify that the method prints "This room has no neighboring room." to the console.

**New Testing Plan for the World Class**

1. **addHumanPlayer() Method Testing:**
   * Condition: Testing the method with valid player name input.
   * Example: Provide a valid player name for addition.
   * Expected: Verify that the human-controlled player is added to the list of players with the provided name.
2. **addComputerPlayer() Method Testing:**
   * Condition: Testing the method with valid player name input.
   * Example: Provide a valid player name for addition.
   * Expected: Verify that the computer-controlled player is added to the list of players with the provided name.
3. **ifGameOver() Method Testing:**
   * Condition: Testing the method with different target health values.
   * Example: Set the target's health to different values.
   * Expected: Ensure that the method returns true when the target's health is less than or equal to zero.
4. **getWinner() Method Testing:**
   * Condition: Testing the method when the game is over.
   * Example: Simulate a game over scenario.
   * Expected: Verify that the method returns the correct winner from the list of players.
5. **playNextRound() Method Testing:**
   * Condition: Testing the method with multiple players and a valid game setup.
   * Example: Simulate playing multiple rounds with various actions.
   * Expected: Ensure that the game proceeds as expected with the target and players taking their turns.
6. **displayTargetInformation() Method Testing:**
   * Condition: Testing the method with a valid target in the game.
   * Example: Call the method to display target information.
   * Expected: Ensure that the target information is displayed correctly.
7. **displayPlayerInformation() Method Testing:**
   * Condition: Testing the method with valid player information.
   * Example: Call the method to display a player's information.
   * Expected: Verify that the player's information is displayed correctly.
8. **displayRoomInformation() Method Testing:**
   * Condition: Testing the method with valid room information.
   * Example: Call the method to display a room's information.
   * Expected: Ensure that the room's information is displayed correctly along with the weapons, target, and players present.