Software Developer Attachment Take-Away Assignment: Ayoayo Game Implementation

Objective:

Implement a text-based version of the Ayoayo game in a programming language of your choice, following the rules and specifications provided. DO NOT USE AI

Game Rules: Ayoayo_rules.pdf

Deliverables:

1. A class named `Ayoayo` that implements the game logic.
2. A `reflection.txt` file that captures your thought process and understanding of the project.

```
For this project you will write a class called Ayoayo that allows two
people to play a text-based version of the game (See PDF file for detailed
rules).

For this board game, two players can play. As the figure shows, the player
who choose the bottom red position will be player 1 and the player choose
the top blue position will be player 2.  Each player could only choose the
pit on his side in each round: player 1 can only choose pits in red and
player 2 can only choose pits in blue. The index for each pit is marked in
the figure as well.

No Graphical User Interface (GUI) required for this project and all the
input/output will be in the text format.

Your code for the game must define the class and methods described below,
but you are encouraged to define other methods or classes that may be
useful for the game. All data members must be **private**.

**Ayoayo part**

The Ayoayo object represents the game as played.  The class should contain
information about the players and information about the board, it must
contain those methods (but may have more):
```

* createPlayer: takes one parameter of the player's name as a string and returns the player object. (You can define the player class by yourself. It will be your own design.)
* printBoard: takes no parameter and will print the current board information in this format:

player1:

store: number of seeds in player 1's store

player 1 seeds number from pit 1 to 6 in a list

player2:

store: number of seeds in player 2's store

player 2 seeds number from pit 1 to 6 in a list (check the last part for examples)

* returnWinner: takes no parameter.

If the game is ended, return the winner in this format:

"Winner is player 1(or 2, based on the actual winner): player's name"

If the game is a tie, return "It's a tie";

If the game is not ended yet, return "Game has not ended".

* playGame: takes two parameters, the player index (1 or 2), and the pit index (1 or 2…. or 6), both as integers. This method should follow the rules of this game including the two special rules and update the seeds number in each pit including the store. It should also check the ending state of the game and once the ending state is reached, it should follow the rules and updated the seeds number in the pit and store for both players.

If user input invalid pit index number (>6 or <=0), return "Invalid number for pit index";

If the game is ended at this point, return "Game is ended";

If the player 1 win an extra round following the special rule 1, print out "player 1 take another turn" (similar for player 2)

At the end, the method should return a list of the current seed number in this format:

 [player1 pit1, player1 pit2, player1 pit3, player1 pit4, player1 pit5, player1 pit6, player1 store,
          Player2 pit1, player2 pit2, player2 pit3, player2 pit4, player2 pit5, player2 pit6, player2 store,]

Note: in order to test some methods in fewer steps, we may or may not follow the rules to call the two players in turns for playGame method, so your code should not enforce that. For example, we might keep calling player 1 for four times, then call player 2 twice, and then call player 1 for three times.
Your code file must be named **Ayoayo.java**

* As a simple example, your class could be used as follows:

```
public class Main {
    public static void main(String[] args) {
        Ayoayo game = new Ayoayo();
        Player player1 = game.createPlayer("Jensen");
        Player player2 = game.createPlayer("Brian");

        System.out.println(game.playGame(1, 3));
        game.playGame(1, 1);
        game.playGame(2, 3);
        game.playGame(2, 4);
        game.playGame(1, 2);
        game.playGame(2, 2);
        game.playGame(1, 1);
        game.printBoard();
        System.out.println(game.returnWinner());
    }
}
```

* And the output will be:

```
      player 1 take another turn
      [4, 4, 0, 5, 5, 5, 1, 4, 4, 4, 4, 4, 4, 0]
      player 2 take another turn
      player1:
      store: 10
      [0, 0, 2, 7, 7, 6]
      player2:
      store: 2
      [5, 0, 1, 1, 0, 7]
      Game has not ended


* Another test example could be:
public class Main {
    public static void main(String[] args) {
        Ayoayo game = new Ayoayo();
        Player player1 = game.createPlayer("Jensen");
        Player player2 = game.createPlayer("Brian");

        game.playGame(1, 1);
        game.playGame(1, 2);
        game.playGame(1, 3);
        game.playGame(1, 4);
        game.playGame(1, 5);
        game.playGame(1, 6);
        game.printBoard();
        System.out.println(game.returnWinner());
    }
}

* And the output will be:

      player 1 take another turn
      player1:
      store: 12
      [0, 0, 0, 0, 0, 0]
      player2:
      store: 36
      [0, 0, 0, 0, 0, 0]
```

```
Winner is player 2: Brian
```

Requirements:

1. Implement the `Ayoayo` class with the following methods:
    * `createPlayer`: takes a player's name as a string and returns a `Player` object.
    * `printBoard`: takes no parameters and prints the current board information in the specified format.
    * `returnWinner`: takes no parameters and returns the winner of the game, if any.
    * `playGame`: takes two parameters, the player index (1 or 2) and the pit index (1-6), and updates the game state accordingly.
2. Implement the `Player` class to represent a player in the game.
3. Ensure that all data members are private and follow good coding practices.
4. Write a `reflection.txt` file that captures your thought process and understanding of the project, including:
    * Initial reflection: your first thoughts and approach to the project.
    * Final reflection: your final thoughts, including any significant changes you made to your plan, challenges you encountered, and key takeaways.

Constraints:

1. No graphical user interface (GUI) is required.
2. All input/output should be in text format.
3. Follow the rules of the game, including the two special rules.

Evaluation Criteria:

1. Correctness: Does the implementation follow the rules and specifications of the game?
2. Code quality: Is the code well-organized, readable, and maintainable?
3. Reflection: Does the `reflection.txt` file demonstrate a clear understanding of the project and its challenges?

Submission Instructions:

When you are ready to submit your final version, ensure that your final push (commit) is tagged with “Ayayol".

Note: You can use Java, JavaScript, C++, or Python to implement the game. However, please ensure that you follow the specified requirements and constraints.