ConnectFour Report 1

The original Connect4 code encapsulates everything in one function, which brings about the difficulties of maintenance and readability. The concept of object oriented programming is to design the code based on object instead of functions or logic. Therefore, we can turn different components into objects to decrease the complexity and increase code readability. From the original version, I observed that there are two primary things in the game. One is the players, the other is the board used to play. Therefore, it is sensible to make two independent objects of player and board, and each is responsible for its operations. Besides that, the game control can be treated as another object to operate the game, and this will contain the player and the board objects as its components and other relating operations.

In Java, classes are the implementations of objects, so three classes will be established respectively. The Player class is to include everything about the player, while board is in charge of the board data and operations, like getting the board image and getting row information. In Player class, the attributes should contain the token used by the player and terminal input variable to capture the move of players in each round. The function will include retrieving the token and storing the input from terminal for further processing. In Board class, the two-dimensional array is stored as an attribute since we need to track the game progress. If the external users want to access the board, a function retrieving the board must be available. The most important operation in Board class is to place the token, which is to put the given token into the stated column. In terms of game control class, named ConnectFour in this case, the Player and the Board classes are two significant attributes and the other control processes are wrapped up into functions. There will be one function which controls the progress of the game, and four others to check different directions of lines with four consecutive same tokens. In the main control function, all players are created and a loop is used to control one player's round because the operations are the same for every player. In the loop, the move of one player is caught by using Player class function and the token is placed in the Board. Then, the winning check is incorporated by using if condition and four checking functions that each is for one direction. If the winner occurs, winning message will be printed and game terminates. If the game continues, the system will change the player and lead to the next round of game from the start of the loop. The pseudocode of the main game control is as follows:

```
Player(token1);
Player(token2);
Print(empty Board);
win = false
while (!win) {
    Player.getToken(move);
    Board.placeToken(token, move);
    if (checkVertical() is true or checkHorizontal() is true or checkRightDiagonal() is true or
checkLeftDiagonal() is true){
        win = true;
    }
    if (win == true){
        print("You have won!);
    }
    else{
        switch to the next player;
    }
}
```

By creating the classes, the concept of modularisation is adopted thoroughly in this case. Data and necessary operations are finely encapsulated into variables and functions. These two features of object oriented programming ensure the readability in the game control class and avoid accidental changing of those critical data in the Player and Board. The establishment of Player and Board is useful for reusing in the future and be extended to another class by inheritance, such as making a computer player. The ConnectFour class hides every internal game operations related, which makes it easy to reuse else where.

At this stage, the abstract class, inheritance and interface are not necessary since the structure of the game is rather simple. These three concepts will be more important and worth using if the game involves other forms of players like computer players or minor change of game rules, for instance, five consecutive tokens to win. In that case, Player class is able to be abstract class or be converted to interface for better extension, and the same work can be done to the Board class and ConnectFour class as well.