SUMMARY
In this assignment you will write a program to play the game Connect Four.

DEADLINE
This assignment is due on Friday, April 4 at 11:00 pm.

DESCRIPTION
Connect four is a popular game played by dropping checkers into a vertical grid with six rows and seven columns. The checkers can be dropped in any of the seven columns (permitted they are not full). Once dropped, the checker falls to the lowest unoccupied space. The winner is the first player to occupy four consecutive spaces in a straight line, either horizontal, vertical, or diagonal.

SPECIFICATIONS
In this project you will write a class to represent the game Connect Four. Your class should include an internal representation of the game (to be designed by you) and a Picture object to display the board. You will create the class called ConnectFour.

The class PlayConnectFour (provided) can be used to test your class by allowing a human user to play against a computer opponent. When run, PlayConnectFour will create an instance of your ConnectFour, display the game board and ask the user to specify a column. The user is always player 1 (red checkers) and goes first. After the user enters a column the opponent (player 2, black checkers) will select a column for their piece.

The class WatchConnectFour (provided) can be used to test your class by allowing a the computer to play against itself. Also provided is a simple naive opponent that selects a column at random. The opponent is represented in the class ConnectFourPlayer. You can improve this opponent if you wish.
Create a new class called ConnectFour. Below are the public members that must be present in your class:

Constructors:

• `public ConnectFour()`

Methods:

• `public Picture getDisplay()`
  • This method should return the display picture. (see details below)

• `public String getBoardString()`
  • This method should return a string representation of the board. (see details below)

• `public void dropChecker(int column, int player)`
  • This method should drop a checker for player `player` in column `column`.
    
    player = 1 for red checkers
    player = 2 for black checkers
    columns are numbered from 0 to 6, left to right
    
    • In the event that `dropChecker` is called with column that is already full, the game state should not change, and that player effectively loses their turn.

• `public int getWinner()`
  • This method will determine if the game is over and return the status/winner. You may assume that a game has not progressed beyond an end state. The method should return the following:
    
    1 for player one (red) wins
    2 for player two (black) wins
    3 for a draw
    0 if the game is not finished
SOME DETAILS

Display

The display will be a picture object 700 pixels wide by 600 pixels high. You should begin by painting the entire picture yellow. Then you can use the graphics class to draw circles in the correct places for checkers or missing checkers. Refer to the battleship lab and the methods fillOval and drawOval for how to draw the circles. Each circle should be evenly spaced and have a diameter of 70 pixels. After using fillOval, use drawOval to draw a black edge on the circle. The display Picture object should update whenever a move is made. Do not show() or repaint() the display in ConnectFour. That is done by the client program.

Board String

The board string will be a string containing exactly 42 characters. Each character represents a space in the board and is either a 0, 1, or 2, as follows:

0 → blank space
1 → red checker
2 → black checker

The characters are arranged as follows. The first seven characters represent the first (top) row, left to right. The next seven characters represent the second (from top) row, and so on.

For example, a blank (initial) board would be:

000000000000000000000000000000000000000000

The board shown at the beginning of the assignment would be:

000000000200000200021110021221201112120

The board string should be exactly as described, with no spaces or other characters.

HINTS

Representation

For this assignment, you have to design your own internal representation of the game. While the picture object display is needed as well, it is not the best choice for your internal representation, since it lacks precision. You should use another internal representation and update the display whenever the internal representation is changed.

Colors

Since the same three checker colors are always used, consider using a static final array. For example:

```
private static final Color [] CHECKER_COLORS = {Color.GRAY,
                                              Color.RED,
                                              Color.BLACK}
```

Notice that the indices of the colors conveniently line up with the values used to represent a blank space, player one checker, or player two checker.
EXAMPLE RUN

> run WatchConnectFour

Current Board String: 000000000000000000000000000000000000000000
Player 1 played column 3.
Player 2 played column 4.
Current Board String: 000000000000000000000000000000000000001200
Player 1 played column 1.
Player 2 played column 2.
Current Board String: 000000000000000000000000000000000000121200
Player 1 played column 0.
Player 2 played column 5.
Current Board String: 000000000000000000000000000012000001121220
Player 1 played column 2.
Player 2 played column 0.
Current Board String: 000000000000002000000200000012100001121220
Player 1 played column 5.
Player 2 played column 0.
Current Board String: 000000000000002000000201000012100101121220
Player 1 played column 2.
Player 2 played column 1.
Current Board String: 000000000000002000000201021012102101121220
Player 1 played column 4.
Player 2 played column 4.
Current Board String: 00000000000000200000020000102012102101121220
Player 1 played column 5.
Player 2 played column 4.
Current Board String: 00000000000000200000020000102012102101121220
Player 1 played column 0.
Player 2 played column 0.
Current Board String: 20000001000200020001002012102101121220
Player 1 played column 0.
Player 2 played column 0.
Current Board String: 2000000100020020001002012102101121220
Player 1 played column 5.
Player 2 played column 0.
Current Board String: 2000000100020020001102012102101121220
Player 1 played column 4.
Player 2 played column 1.
Current Board String: 2000100100020001102210210121012101121220
Player 1 played column 5.
Current Board String: 200010010002102000110221021012102101121220
Player 1 Wins!

END DISPLAY
SUBMISSION

Submit the file

    ConnectFour.java

with your method added by the deadline given above. Place all files for submission in a directory in your cs account. Submit the entire directory with the following command:

    submit102 assign2 MY_DIRECTORY

For example, if you named your directory assignment-02, and your current working directory contained the assignment-02 directory, you would submit with the following command:

    submit102 assign2 assignment-02