

Class Connect4

java.lang.Object[✉]
Connect4

All Implemented Interfaces:

Game

```
public final class Connect4
extends Object✉
implements Game
```

An API to handle the game logic of Connect 4. Connect 4 is a two-player game where each player tries to make a straight line (vertical, horizontal, or diagonal) of four of their colored checkers by dropping their checkers into a 6 x 7 grid. The two players take turns playing until one of them wins, or the board is full. See [this video[✉]](#) for a visual on how to play Connect 4. The following is a simple example of how to create a basic ASCII two-player Connect 4 CLI using this API:

```
import java.util.Scanner; 
```

```
public class TextClient {

    public static void main(String[] args) {
        // Create a new game to start.
        Game game = new Connect4();

        // Begin listening to STDIN for game input
        try (Scanner scanner = new Scanner(System.in)) {
            // While the game is not finished:
            while (game.getGameStatus() == GameStatus.IN_PROGRESS) {
                // Print out the board in ASCII
                printBoard(game.getBoard());

                // Get whose turn it is, and ask them to choose a column to put t
                System.out.print(game.getCurrentPlayer() + " choose a column (0-6"

                // Get the column the player chooses
                int column = scanner.nextInt();

                // Drop a checker at that particular column.
                if (!game.dropChecker(column)) {
                    // If unable to drop that checker, print out an error and let
                    System.out.println("Invalid move. Try again.");
                }
            }
        }
    }
}
```

```

        // Print the board state out for the last time
        printBoard(game.getBoard());
        // Say who won the game
        System.out.println("Game over: " + game.getGameStatus());
    }

}

private static void printBoard(Board board) {
    for (int r = 0; r < board.getRows(); r++) {
        for (int c = 0; c < board.getColumns(); c++) {
            Player p = board.getCell(r, c);
            System.out.print(p == null ? ". " : (p == Player.RED ? "R " : "B "
        }
        System.out.println();
    }
    System.out.println("0 1 2 3 4 5 6\n");
}
}

```

Constructor Summary

Constructors

Constructor	Description
<code>Connect4()</code>	Create a new, blank Connect 4 instance.

Method Summary

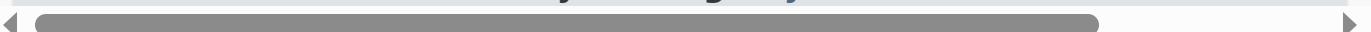
All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method	Description
boolean	<code>dropChecker(int column)</code>	Drop a checker with the selected player, if possible, at the selected column.
Board	<code>getBoard()</code>	Get the game board.
Player	<code>getCurrentPlayer()</code>	Get the current player available.
GameStatus	<code>getGameStatus()</code>	Get the current game status.
void	<code>reset()</code>	Reset the entire board from scratch.

Methods inherited from class `java.lang.Object`



```
clone2, equals2, finalize2, getClass2, hashCode2, notify2, notifyAll2,  
toString2, wait2, wait2, wait2
```

Constructor Details

Connect4

```
public Connect4()
```

Create a new, blank Connect 4 instance. This will be a 6 x 7 grid that starts with the red player playing first.

Method Details

getCurrentPlayer

```
public Player getCurrentPlayer()
```

Description copied from interface: Game

Get the current player available.

Specified by:

getCurrentPlayer in interface Game

Returns:

the player whose current turn it is

getGameStatus

```
public GameStatus getGameStatus()
```

Description copied from interface: Game

Get the current game status.

Specified by:

getGameStatus in interface Game

Returns:

game status (in progress, red or blue win, or draw)

getBoard

```
public Board getBoard()
```

Description copied from interface: Game

Get the game board. See Board for available methods.

Specified by:

getBoard in interface Game

Returns:

game board

dropChecker

```
public boolean dropChecker(int column)
```

Description copied from interface: Game

Drop a checker with the selected player, if possible, at the selected column. The current player can be found with Game.getCurrentPlayer(). If unable to make a move (ex. an invalid column was passed in), the current player remains the same. If they successfully make a move, the board, game status, and current player is updated.

Specified by:

dropChecker in interface Game

Parameters:

column - zero-indexed, the column (by default 0-6).

Returns:

true if making the move was successful, false otherwise

reset

```
public void reset()
```

Description copied from interface: Game

Reset the entire board from scratch. Set the entire Board to be completely empty, for it to be the red checker's turn, and for the game to be started over.

Specified by:

reset in interface Game