CS 251 Intermediate Programming
Project 2: Othello

Brooke Chenoweth

Fall 2025

Please note: This is an individual assignment. It is designed for you to get back even more into programming mode. The idea is that you will have to use all of the skills that you learned in CS152 (or its equivalent) and get further acquainted with Java, while solving some interesting problems.

This assignment will give you some experience implementing an interface, using enums, and interacting with my GUI.

Problem Specification

We are going to implement the game of Othello, sometimes also known as Reversi.[1]

The game is played on an 8x8 board. Each of the pieces are light on one side and dark on the other. Players take turns placing pieces on the board with their assigned color facing up. During a play, any pieces of the opponent's color that are in a straight line and bounded by the piece just placed and another piece of the current player's color are flipped over to the current player's color.

Details of play in our game:

- When the game begins, four pieces (two pieces of each color) are placed in the center. The upper right and lower left of these four will be dark and the other two will be light.
- The dark player will play first.
- The current player must place a piece on the board with their color facing up in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and other piece of the player's color, with one or more contiguous pieces of the opponent's color between them.

  After placing the piece, the current player flips over all opponent pieces lying on a straight line between the new piece and any anchoring pieces of the current player's color.

[1] `https://en.wikipedia.org/wiki/Reversi`

- If the current player has no possible move, the game ends and the player with the most pieces of their color is the winner.

What you're getting

I've prepared the visualization for you in a class called `OthelloGUI`. This class has a static method that takes an object that has the functionality described in the `OthelloInterface` interface.[2] The methods in this interface are the following:

- `int getSize()`

  Specifies the number of rows and columns on the game board. (The board will be square, so we only need one number.) To play the default game, just have this method return the default constant defined in `OthelloInterface`. When testing your code, you may find it easier to experiment with a smaller board. In that case, you can change the value returned by this method and the GUI should be able to adapt.

- `void initGame()`

  This method is called by the GUI whenever a new game is begun. It should set up whatever bookkeeping is needed to start a fresh game.

  At the beginning of a game, the board will be empty except for the initial starting pieces in the center. The dark player always plays first.

- `String getBoardString()`

  This method returns a string representation of the board. The GUI calls this method whenever it needs to know what the board looks like in order to be able to redraw it. The string that is returned must match the specific format described in the interface documentation.

- `Piece getCurrentPlayer()`

  Returns DARK or LIGHT, depending on which is the current player.

- `void configureOpponent(String opponent)`

  This method configures what sort of computer player (if any) will be used.

- `boolean isLegal(int row, int col)`

  Determines if playing at the specified location would be a legal move for the current player.

- `Result handleClickAt(int row, int col)`

  This method is called by the GUI whenever a click occurs on the game board. It's supposed to update the state of the game board, and the return value is a value that describes the current state of the game after the newly entered click was applied.

Please refer to the documentation for `OthelloInterface` for more information.

[2] We'll talk about interfaces in lecture.

What you have to do

To get started, create a file called `Othello.java`. It should be a class that *implements* the interface. I'll show you in lecture how to make this happen. However, the main method should look something like this:

```
public static void main(String[] args) {
    Othello game = new Othello();
    if(args.length > 0) {
        game.configureOpponent(args[0]);
    }
    OthelloGUI.showGUI(game);
}
```

Three things happen in this `main` method:

1. Create a instance of the class you are writing.
2. Possibly configure the Othello object based on a command line argument. (Any command line arguments given to a Java program are available in the String array parameter of the main method.)
3. Pass the Othello object to a static method in the OthelloGUI class that will actually get the GUI started.

There are three parts to the assignment. Each part needs to be completed before the next part is attempted.

1. **Initial Piece Placement** – Make the game work to the point that when clicking on the game board, every click in an empty spot places a piece, alternating dark and light. It should not be possible to place a piece in a square that is already occupied, but otherwise do not yet worry about legality of moves.

   In order to do this, you'll need to implement all the methods, but you don't have to create logic to flip over the pieces or see if someone has won. At this stage you're pretty much playing a large game of tic-tac-toe.

2. **Game Logic and Win Detection** – Everything from the prior part, but add in the logic to detect legal moves and flip the pieces. If the current player has no legal moves, then whoever has the most pieces is the winner.[3] It is possible for the game to end in a tie if both players have the same number of pieces.

3. **Computer Player** – Create a rudimentary computerized player. You'll do this by making the `handleClickAt` method also fill in a value for the player who didn't start. That will now be the computer player.[4] The computer player must make legal moves, but is not required to be particularly intelligent.

[3]The standard game of Othello would have the current player pass and the game would only end when neither player could make a move, but for simplicity's sake, we're going to end it as soon as anyone cannot play.

[4]In other words, the light player will be the computer player.

The computer player should only be enabled if it was configured via the `setComputerPlayer` method. (You may assume that this method will not be called in the middle of a game.) An argument of NONE should disable the computer player, while an argument of COMPUTER should enable it. If you want to explore additional computer player strategies, feel free to recognize additional Strings. If you do this, please be sure to document it so that the grader will know to try them.

Compiling and Running

1. Download `project2.jar` from the course web site and place it in your working directory.

2. Create `Othello.java` in your favorite text editor.

   - Add an import statement for the contents of the jar file.

     ```
     import cs251.project2.*;
     ```

   - Start small. Just make a minimal implementation of the `OthelloInterface` interface to be sure everything is set up before you try something more complicated.

3. Compile using `javac`. If you placed the jar file in the same directory as your source file, you can use the command `javac -cp project2.jar:. Othello.java` to compile your program.

   The `-cp` option specifies the path where the compiler should look for the classes. If no classpath is specified, java looks in the current directory, but not inside the jar file. When you use the `-cp` option to tell the compiler to look inside `project2.jar`, you also need to include the current directory (the dot after the colon) in the path so it can still find your code. (If you are running on a Windows system, you may need to use a semicolon rather than a colon to separate the paths in the classpath argument. Microsoft just had to be different!)

4. Run using `java`.

   If you have set up your main method properly, the command `java -cp project2.jar:. Othello` will open up a `OthelloGUI` panel to play the game. (Not much will happen until you flesh out your implementation, of course.)