

Homework 7: Othello

Homework is due at 11:59 P.M. on the date indicated above. Late homeworks will be penalized by $5 \times 2^{n-1}\%$ where n is the number of days it is late. Please indicate which problems, if any, took extra time.

Total points on this assignment: 200

1. Specifications In the book *Program Development in Java: Abstraction, Specification, and Object-Oriented Design* by Barbara Liskov and John Guttag, you can read the following (on pages 207–208):

10%

As an example, consider the specification

```
static int p (int y)
    // REQUIRES: y>0
    // EFFECTS: Returns x such that x > y.
```

This specification is satisfied by any procedure named `p` that, when called with an argument greater than zero, returns a value greater than its argument. Members of the specificand set [i.e., of the set of procedures that satisfy the specification] include

```
static int p (int y) { return y+1; }
static int p (int y) { return y*2; }
static int p (int y) { return y*3; }
```

Like every specification, this one is satisfied by an infinite number of programs. . . .

Please critique this passage.

2. Othello “Othello” is a popular board game that is normally played on a board with sixty-four squares. One player is “Black,” the other is “White,” and the sixty-four game pieces are black on one side and white on the other. If you do not know the rules for Othello, you can find them here and there on the Internet:

90%

http://www.pressmantoy.com/instructions/instruct_othello.html

appears to be a good site (Pressman Toy is a supplier of Othello game boards).

To begin, copy over the code from:

```
/home/cs2/lab/lab7/src/
```

This code contains a simple graphical user interface to an Othello game board. You should not have to worry about the details of how the user interface is implemented (you should be able to ignore the code in `Board.m3`), but if you are interested, you are welcome to look at it (if you want to know how to do graphics in Modula-3, look at SRC Research Reports 68 and 69,

<http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-069.html>

<http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-068.html>

). To get an idea of what this code does, m3build it and run it. You should see a single black piece in the middle of a board, and if you click on it, it should change colors. If you click anywhere else on the board, the square you click on should cycle through all its possible states: empty, covered with a black piece, and covered with a white piece.

Now look at Board.i3. Board.i3 contains the following:

```
INTERFACE Board;
IMPORT VBT;
IMPORT Point;
TYPE
  T <: Public;
  Public = VBT.Leaf OBJECT METHODS
    (* initialize a board of the given size *)
    init(xsize, ysize : CARDINAL := 8) : T;
    (* set the state of a game square to the given value *)
    setState(p : Point.T; state : Piece);
    (* get the state of a game square *)
    getState(p : Point.T) : Piece;
    (* override this: click will be called whenever a game
       square is clicked on *)
    click(p : Point.T); (* := NIL *)
    (* and this...: quit will be called whenever the "quit square" is clicked
       on *)
    quit() (* := NIL *)
  END;
  Piece = { None, White, Black };
END Board.
```

A VBT.T is a “virtual bitmap terminal,” but you actually don’t have to know that. All you have to know is that a Board.T is an object that you should initialize with a statement like the following:

```
board := NEW(Board.T).init();
```

And then you use it. How? Well, the code inside Board.m3 insulates you from all the details of manipulating graphics. What you have to do is to override the click and quit methods (especially the former). To be perfectly clear: a Board.T object has five methods you should care about. You *call* init, setState, getState when appropriate. You *override* click and quit in order to provide the object with the necessary intelligence for playing Othello and for quitting in an informative way (if you want it to do that). The hidden part of the object will *call your code* when it detects a mouse click. If you are confused by this, *do not* attempt to make sense of it by reading Board.m3 (that is unlikely to help); ask a TA instead.

i. The code in Main.m3 implements a basic version of click. But it doesn’t actually know the rules of Othello (it’s only a single line of code, after all!) Your first task is to make click fancier: it should be able to distinguish between legal moves and illegal moves. Make it print to the terminal “Black’s move” and “White’s move” when it is time for each player to move; it should print “Black passes” and “White

passes” when the players have to pass; and it should print a “game over” message when the game is over, with appropriate statistics (e.g., who won, and by how many points). You can make the program exit once the game is over; see the basic implementation of `quit`.

ii. Now change the code so that the computer plays a reasonably decent game of Othello. Make it use a “greedy” algorithm: make it play as White and always pick the best possible legal move. Make as few changes to the code you wrote for the preceding subpart as possible; make the computer’s own moves appear as much as possible like a human player’s moves to the rest of the code.

iii. Finally, change the code so that you can run the program with the computer playing against itself. If you did the previous section according to our instructions, that should be very easy.

3. (*Extra credit*) Making the computer smarter Add “intelligence” to the computer’s playing: make it pick the best move, not in terms of how many pieces will be of its color after that one move, but in terms of how many pieces will be of its color n moves in the future, assuming that it is playing a perfect adversary. 50%

4. (*Extra credit*) Chandyism 50%

Professor K. Mani Chandy is the Simon Ramo Professor of Computer Science and the current CS20 prof. He is also a really awesome guy. So awesome that we TAs feel the world needs to know about it.

Distribute pamphlets extolling the virtues of Chandyism, a religion/philosophy centered on the man that is Mani Chandy. Your bonus will be based on the overall amount of Chandyism on campus—that means more pamphlets and people extolling Chandyism equals a larger bonus for you. Maximum points will be given for handing out pamphlets in front of Chandler at noon dressed as the Hindu god Ganesh.

For the proper information for your pamphlets extolling the greatness of Chandyism see the following web sites:

<http://www.infospheres.caltech.edu/people/mani.html>

http://www.cs.caltech.edu/cspeople/faculty/chandy_m.html

<http://www.infospheres.caltech.edu/>