THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - INDIVIDUAL ASSESSMENT

Unit of Study: COMP5028 Object-Oriented Design

Assignment name: Othello Game

Tutorial time: Monday night 8:00 -9:00

Tutor name:

DECLARATION

I declare that I have read and understood the *University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy*, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the the *Academic Dishonesty and Plagiarism in Coursework Policy*, can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Student ID: 440042783

Student name: XUN ZHAO

Signed XUN ZHAO **Date** 20/3/2014

SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

ABN 15 211 513 464
CRICOS 00026A

SID: 440042783 STN: XUN ZHAO

COMP5028 - Assignment 1

Question 1:

Analysing the requirements:

How to analyses requirements:

The first step is reading the whole document and find out what are common things, for example (what are ordinary things that may exist in the game of Othello).

- board, white and black stones, and 64 stones, and two players, and different colours, and some rules, and so on.

The send step is discerning the essence of the items. In the Othello, the following items are necessary:

- game board, with 8X8 grids
- two players - each player chooses own unique colour(white/black), and takes 32 stones.
- each player is able to move the stone but has to follow rules.
- when the game starts what the initial configuration is.
- once a player's disc has been placed on the square and the disc is able to outflank the foe's discs, then those discs must be flipped.
- some special rules:
  - black discs move first
  - checking whether a player has chance to move
  - allowing players to move their stones in multiple directions.
  - checking player whether against that he skips own colour disc to outflank enemy discs
  - checking that whether has discs that may only be outflanked as a direct result of a move and must fall in the direct line of the disc placed down
  - checking a player whether flips disc.
  - once disc placed on a square, then it cannot move another square
  - if player still has chance to move the disc even if he out of discs
  - this function is checking that if either player no longer to move disc, then finds out winner who has majority colour discs left.

The step three is modelling, where been displayed on below **modelling section.**

OBJECTS:

board, white/black(circles), player, 32 discs/p, colour, rules, controller

FUNCTIONS:

/******** Common functions *********/
Board(); //implementing the its class function - +InitialPlace(); and +DrawBoard();

InitialPlace(); // initial whole board when the game starts

DrawBoard(); // Drawing the graph of board

PlayerEnd(); // players given up chance because there is no rest of stones for use.

/****** Controller ****************/

move();//movement function that allowed players to move their stone into a exact coordinate.

inputCoordiate(int Ox, int Oy, int Ex, int Ey);/ request player to input cordite

placeInfo();// display the info that after user inputed

winner_check(); //this function is checking that if either player no longer to move disc, then finds out winner who has majority colour discs left.

flipped(String color); //flipped stones

/****** rules**********/

outflank_and_flip_check(); //this function is checking whether the player has chance to move.

multi_outflank();// this function is allowed player to outflank any number of discs in multiple directions, such as in horizontally, vertically or diagonally

skip_own_color_check(); //this function is check whether player has against that he skips own colour disc to outflank enemy discs

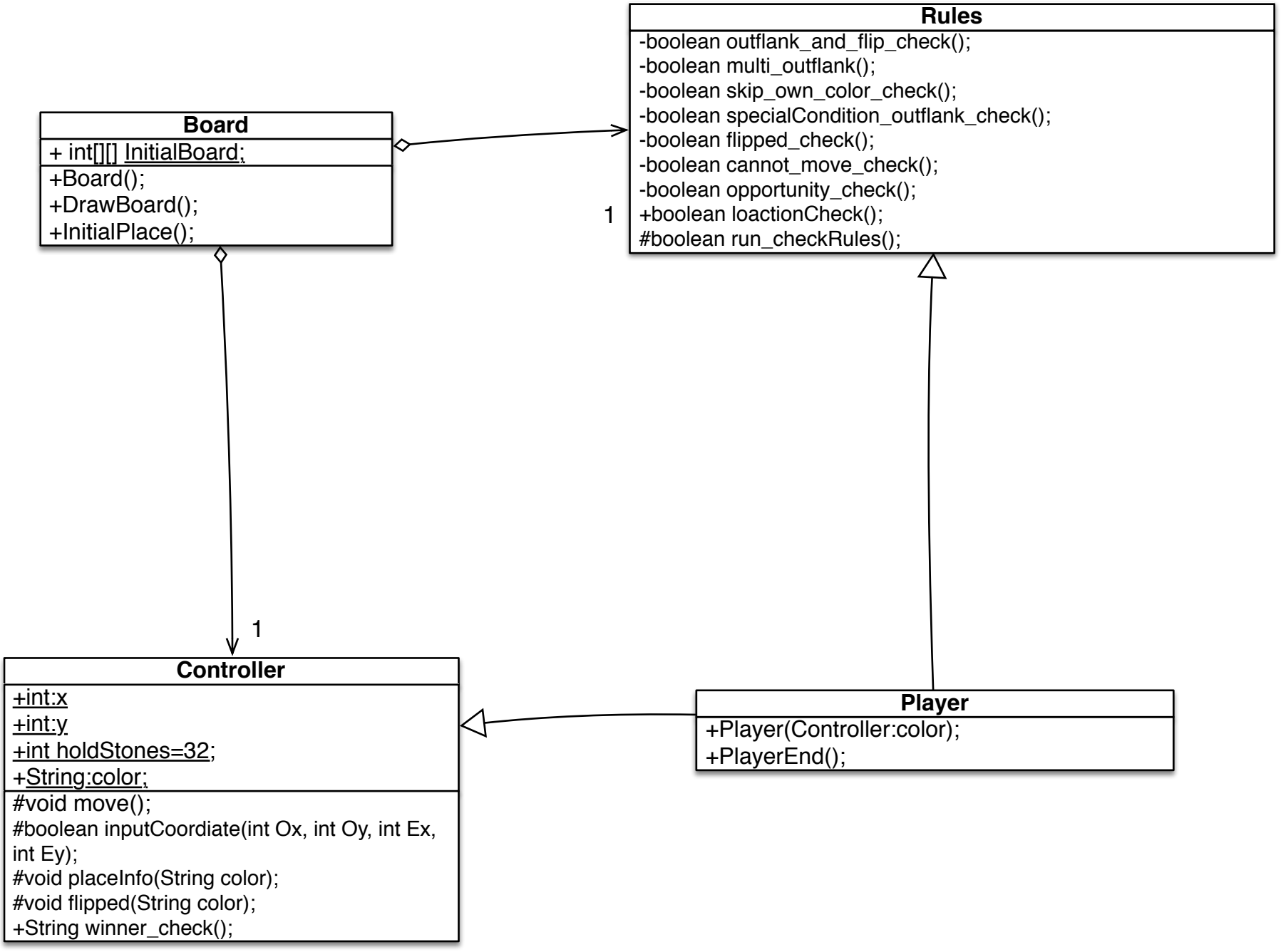
specialCondition_outflank_check(); //this function checks that whether has discs that may only be outflanked as a direct result of a move and must fall in the direct line of the disc placed down

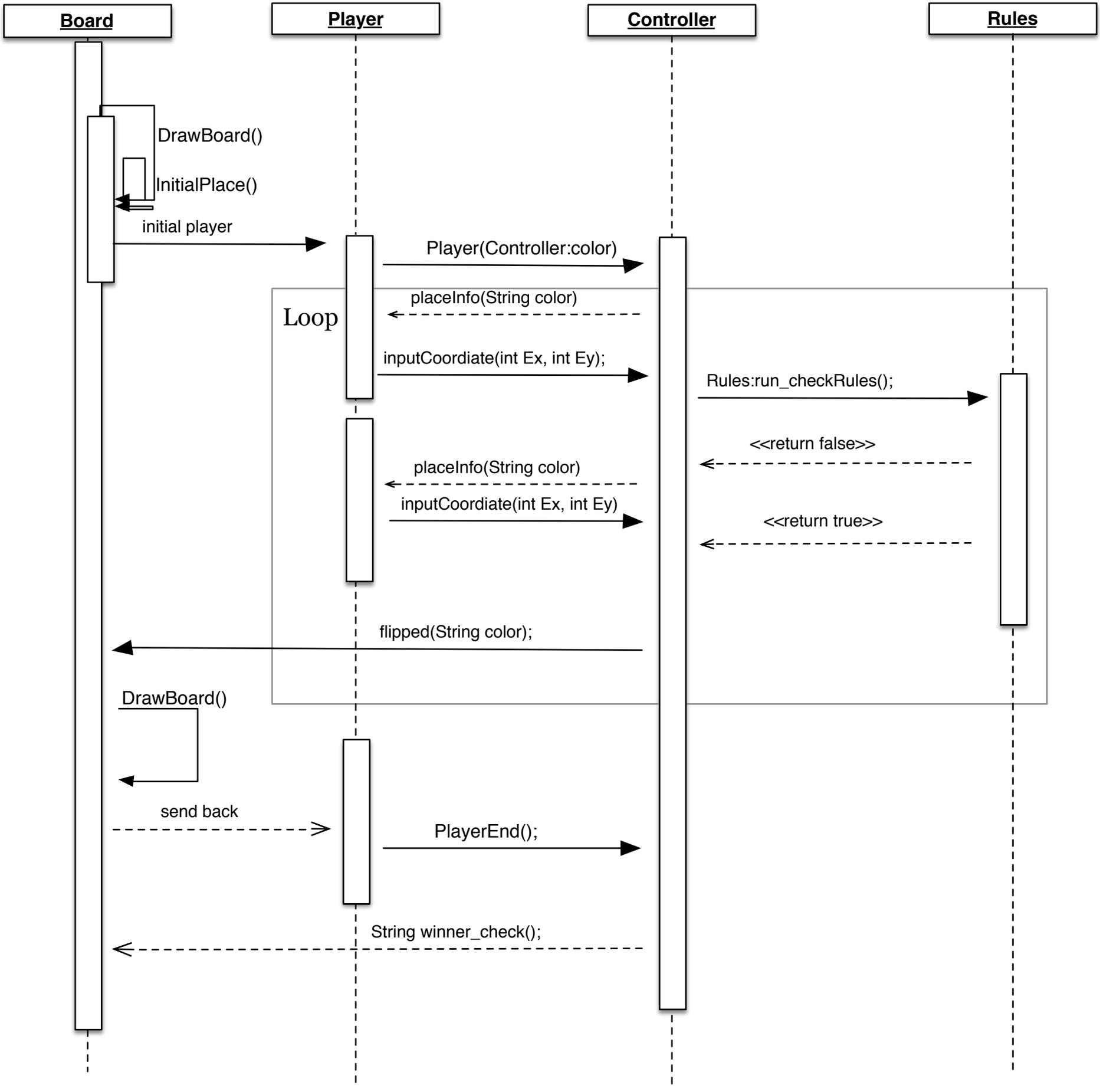
flipped_check(); //this function is checking that player whether flips disc.

cannot_move_check();//once disc placed on a square, then it cannot move another square

opportunity_check(); //if player still has chance to move the disc even if he out of discs

Modelling (Step three)


Rules
-boolean outflank_and_flip_check();
-boolean multi_outflank();
-boolean skip_own_color_check();
-boolean specialCondition_outflank_check();
-boolean flipped_check();
-boolean cannot_move_check();
-boolean opportunity_check();
+boolean loactionCheck();
#boolean run_checkRules();
Board
+ int[][] InitialBoard;
+Board();
+DrawBoard();
+InitialPlace();
1
1
Controller
+int:x
+int:y
+int holdStones=32;
+String:color;
#void move();
#boolean inputCoordiate(int Ox, int Oy, int Ex, int Ey);
#void placeInfo(String color);
#void flipped(String color);
+String winner_check();
Player
+Player(Controller:color);
+PlayerEnd();


Board
Player
Controller
Rules
DrawBoard()
InitialPlace()
initial player
Player(Controller:color)
Loop
placeInfo(String color)
inputCoordiate(int Ex, int Ey);
Rules:run_checkRules();
<<return false>>
placeInfo(String color)
inputCoordiate(int Ex, int Ey)
<<return true>>
flipped(String color);
DrawBoard()
send back
PlayerEnd();
String winner_check();

start

Initialise Board

action on player

instance player

been asked to move

has black stones

yes

no

black moved

yes

no

ask white to move

ask black to move

entry coordinate inputing

entry coordinate inputing

white inputed coordinate

black inputed coordinate

entry rules

entry rules

if white does't match rules then given false

if black does't match rules then given false

check rules

yes

flipped

if white got chance or after black moved

next turn

if black got chance or after white moved

both colour are unable to move

check and display winner

yes

ask for new game

no

End