Implementing the *Best move or first robot* change case

Matthew Eden
Department of Electrical, Computer and Software Engineering
University of Auckland
Auckland, New Zealand
mede607@aucklanduni.ac.nz

***Abstract*—**
***Index Terms*—kalah, mancala, changeability, java**

I. Planning the Change Case

Before any code was actually written, a plan was devised describing the minimal amount of changes that would be required to achieve the desired change case. This plan is as follows.

1) Find the `Kalah` class
2) Find the `Play` method
3) Within the `while` loop, modify the assignment to the variable `selection` to read: `selection = (playerNum == 1) ? GameIO.computeMove(board, playerNum, io) : GameIO.getMove(board, playerNum, io);`
4) Find the `GameIO` class
5) Create a new method: `public static int computeMove(Board board, int playerNum, IO io)`
6) Implement the `computeMove` method

II. Implementing the Change Case

The total amount of time taken to implement the change case was timed as being 15 minutes and 58 seconds. During the execution of the aforementioned plan, there were some issues encountered. Overall, the plan was a success and it was not difficult to implement the change case. A breakdown of the steps carried out follows.

1) No issues in finding the `Kalah` class.

III. Impact of the Change Case

On a scale from *0.0* to *1.0*, the impact of this change case is *0.3*. Although the rules of the *Kalah* are preserved and the overall functionality of the game is preserved, the replacement of the second player with a simple AI changes the game from being multi-player to being single-player.

IV. Changeability Assessment

References

[1] M. Eden "SOFTENG 701 - Assignment 3 - Submission", *Implementing Kalah with a Changeable Object-Oriented Design*
[2] M. Eden "SOFTENG 701 - Assignment 4 - Submission", *Implementing the Change Board Orientation change case*