SOFTENG 701: Assignment 6

Ahmad Umar, auma020
The University of Auckland
auma020@aucklanduni.ac.nz

I. Changeability

Changeability can not be measured instead is used as a relative term. I learned that changeability encompasses many properties. In the lectures, the definition used for changeability referred to the efforts made to implement the changes required for a change case(s). The definition and class discussions emphasized that changeability is related to the necessary modifications to implement a change, and the effort to write new code does not impact changeability. I agree with this definition, but I would not say it is the right one. This is because the financial aspect is one of the crucial factors in industries. I believe the definition of changeability relates well and close to Meyer's description of Reusability. Meyer's reusability definition is a way of writing code that allows efficient maintainability with low costs. In class, the cost factor was omitted, which I believe is very important.

I am going to share an experience to support my claim that cost is an essential factor. I have been working at a pharmaceutical company as a Junior Engineer for the past few months. Part of my work required me to maintain the database. One day the Operations Manager came to me and asked me to look into the database. He then explained a few features that he wanted me to change in the company's database software. I had a look at the software, and it was a mess; there were many code files with minimal to no comments and no documentation for the software. After making a few inquiries, I learned that the person who developed the database passed away a while back, leaving the company with no back-end knowledge of the database. I looked at the database and tried to find the code for the features that needed changes. I had a meeting with the manager and explained to him all my findings. The first question he asked me was: "How long is this going to take?" to which I replied, "Months." The next week he asked me to forget about the changes for now and focus on another task. After asking him why he changed his mind, he said, "it's not worth it." Therefore, I suggested some modern software for them to consider purchasing. From this experience, the discussions with my manager, class discussions, and assignments, I learned two important factors about changeability. First, the code/program needs to be well documented for a new user to be able to understand and bring about changes. Second, the code needs to be written well enough that changes are minimal, and if there are changes, the dependencies should be minimal to allow the code to be separately testable and reusable. I learned that if a code is well written but poorly documented, the new user will be wasting time trying to understand the code. Well written code and minimal costs are related to changeability, but I feel like one aspect that was not discussed enough in class is documentation; this affects changeability and should be included in the definition.

To conclude this section, I would say the definition given by Ewan only partially describes changeability. Changeability can not be described in one sentence. From my university and work experience, I have understood that changeability has to be defined by listing all the factors that can impact changeability. Some of these factors include code quality, dependencies, documentation. High changeability is useful for quick implementation of change cases, but when defining changeability, we need to talk about the importance of changeability in terms of financial costs.

II. Discussion on Implementation of Changeability

From the discussions in lectures and from completing the assignments, I understood Object-Oriented Programming (OOP) could increase changeability. I have not always utilized the changeability feature of OOP. Using features like encapsulation with decreased dependencies allows code to be separately testable and reusable. Furthermore, using generic values in functions allows for different parameters to be passed without any changes. Meyer, in his article v [1], describes the importance of various OOP features that can favor reusability.

After partaking in class discussions and conducting the assignments and tests, I realized I could have written higher quality code for Assignment 3. Looking back at the Assignment 3 code now, I believe it was changeable to some extent.

First, I should have used a generic function to display the board. I used hardcoded values for the print function. Figure 1 shows the code snippet of the diplayBoard function. In the for loop (used to identify spaces), I have hardcoded 14 as the number of total holes. Therefore, doing the change case, NH (number of houses per player) would have resulted in locating this print board function and then changing quite a bit to implement the change case. This could have easily be prevented if I had been a bit more careful in Assignment 3. I should have passed variables to the displayBoard function. These variables can be defined in the Kalah class that sets the number of houses per player at the start.

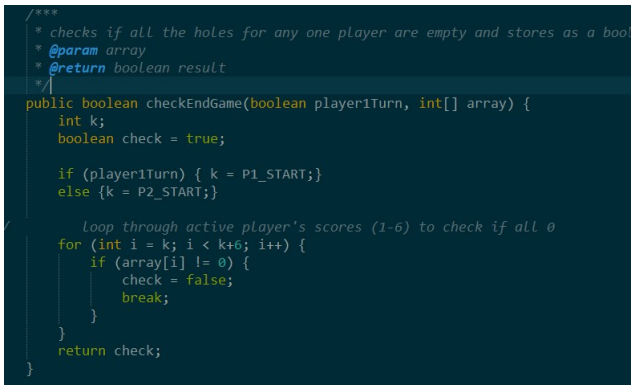
```
public void displayBoard(IO io, int[] array) {


    String[] spaceCheck = new String[14]; //stores " " or "" for each hole

    // iterate through boardArray
    for (int j = 0; j < 14; j++) {
        if (array[j] < 10) {
            spaceCheck[j] = " ";
        } else {
            spaceCheck[j] = "";
        }
    }

    // prints the board
    io.println("+----+-------+-------+-------+-------+-------+-------+----+");
    io.println("| P2 | 6[" + spaceCheck[13] + array[13] + "] | 5[" + spaceCheck[12] + array[12] + "] | 4
    io.println("|    |-------+-------+-------+-------+-------+-------|    |");
    io.println("| " + spaceCheck[0] + array[0] + " | 1[" + spaceCheck[1] + array[1] + "] | 2[" + spaceC
    io.println("+----+-------+-------+-------+-------+-------+-------+----+");
}
```

Fig. 1. Function to print a 14 hole board

Second, I should use a variable to indicate the direction of the game's flow, either clockwise or anticlockwise. A variable could store this, and this could have allowed implementing the change case, DIR, with minimal changes.

Third, when writing Assignment 3, a change case like UD (Undo Move) did not cross my mind. Therefore, in this change case, my design will not handle well at all. However, I believe, to implement this change case, I will need to change a few things in the logic, including storing at least one of the past moves for each player.

Furthermore, in Assignment 3, I could have made more use of OOP features. I should have made separate classes, using encapsulation, for the different objects such as board, player. This would have allowed me to easily implement change cases where the number of players/bots need to be changed or if the number of boards needs to be changed or whether the board itself has to change. One important lesson I have learned is that when producing a design, one does not have to come up with all the possible change cases. The programmer needs to create generic classes/methods, with the least dependencies and appropriate documentation. Also, when designing my assignment 3, I should have completely avoided magic numbers because when implementing a change case such as NH, I would need to find all magic numbers related to number of houses. This reduces the changeability of the design and something I will actively avoid in the future. Figure 2 is a snippet showing the use of a magic number in my *for* loop.



```
/***
 * checks if all the holes for any one player are empty and stores as a boo
 * @param array
 * @return boolean result
 */
public boolean checkEndGame(boolean player1Turn, int[] array) {
    int k;
    boolean check = true;

    if (player1Turn) { k = P1_START;}
    else {k = P2_START;}

    // loop through active player's scores (1-6) to check if all 0
    for (int i = k; i < k+6; i++) {
        if (array[i] != 0) {
            check = false;
            break;
        }
    }
    return check;
}
```

Fig. 2. Usage of magic numbers reduced my design's changeability

Overall, I believe I did not fully understand changeability when it was first discussed in class. However, as I progressed through each assignment and implemented each change case, the concept of changeability became clearer. Even when working through my code, I found a relationship between documentation(comments) and changeability. Therefore, looking back at my first attempt at designing a changeable code, there are many things I will do differently, given my enhanced knowledge and understanding of the concept.

III. Conclusion

In conclusion, Assignment 3 was the first attempt at producing a changeable design. Over time I understood the concept more and found more factors that are important and related to changeability such as financial impact and documentation. With an intensified understanding, I would have done things differently.

References

[1] B. Meyer, "Reusability: The Case for Object-Oriented Design," 1987.