SE701: Assignment 4 | Print Vertical Layout

Shane du Plessis
Department of Electrical, Computer and Software Engineering
The University of Auckland
sdup751@aucklanduni.ac.nz
15 May 2020

1. ASSESS IMPACT

The change case will have an impact of 0.0 on the Kalah functionality. The change case only affects the orientation of the board printed to the console; otherwise the functionality and everything else is the same.

2. PLANNED CHANGES

As the change case only affects the printing to the console, it is sensible to extend the functionality of the existing PrintOutput class (which is responsible for printing to the console). The plan prioritises making minimal changes to the existing code while adding as little new code as possible.

Within Kalah Class

1) Add field private string orientation = "Vertical" to Kalah class

2) Modify the PrintOutput Constructor to take an additional string field (orientation to be passed)

Within PrintOutput Class

3) Add field private string orientation to PrintOutput to store the new variable passed into constructor (orientation).

4) Modify PrintOutput constructor to store new variable value orientation ("Vertical")

5) Add new method 'private void printBankVertical (int Player)', takes argument player ID to determine if it is printing the top or bottom of vertical board

6) Add new method 'private void printHollowVertical()', takes no arguments as getter functions will be used to access data needed (total hollows and seeds per hollows)

7) Modify printBoard method to check value of orientation string with case statement (Horizontal || Vertical) and call appropriate private methods. In this change case the two methods added in steps 5 & 6.

3. EXCECUTION

Start time - 10:15 pm

Steps one through to four were trivial to implement. Adding the new private field to Kalah and modifying the constructor to take the new value was as straight forward as first thought; as well as updating the constructor in the PrintOutput class to accommodate this new variable did not require more detailed steps.

The hardest part about the changes was formatting the output correctly. Writing two new private methods worked well as I did not have to touch any existing code. The steps also enable easy changing between the vertical and horizontal cases. The only thing that needs to be changed in future, is the orientation variable in the Kalah class.

The only addition to the steps outlined, was adding an error message that is printed if the orientation is not initialized to a valid value ("Horizontal" or "Vertical"). Everything went to plan the steps provided enough details to implement the change case.

Completion Time - 10:33 pm
Time Taken – 18 minutes

4. ASSIGNMENT 3 CHANGEABILITY

The modular approach, in assignment three, has allowed me to implement the change case without touching several other classes. My design contains seven classes, where only two needed to be touched. I only had to add two new private methods to the PrintOutput class to add functionality. To support future swapping between horizontal and vertical, I only had to add a variable field to the Kalah class.

Being able to outline a few quick steps, and successfully implement them first try with no problems also indicates the system is very flexible and changeable. The system allows for modules to be modified independently.

Using the rule of thumb as discussed in class (#modified Classes / #total Classes = "changeability"), indicates that my design is changeable. Changes inside classes did not affect the implementation of any other classes, therefor only two of the seven classes needed work. The design supports a highly cohesive and loosely coupled implementation.

After having completed the assignment I wish to stay with my assignment 3 architecture. The design has shown to be accommodating to change cases and follows flexible design patterns (MVC and interface approach).