Sainmont Lucas

Erasmus

ELEC Y 415

SOFTWARE FOR EMBEDDED SYSTEMS

Isola :

The program that I took for this project is my version of the game Isola . I have coded it for a past research project.

This game is played by two players (even more if we want) on a board (used to be 8x6). Each players control a pawn. A player turn is divided in two sub phases.
The first consist to mode the player's pawn on an adjacent square valid, and the second to a blacken a box which will make it invalid. Once a player finish his turn, the other player can play and so on up as a spawn get surrounded by invalid boxes .

This program is a C program and the only hardware requirement is to have a display system for the game board and a keyboard to enter the coordinates.

For this project, the program will be fully re-written in java language in BlueJ. So it will run on any platform with a Java virtual machine.

Test :

As the code will be re written from nothing, I will use the Agile development cycle. Each piece of code implemented will be tested before adding content.

The key to this game is method isValid (), it helps to know if a box is valid for the player (in the array dimensions, not already blackened or no players above). We must therefore ensure that this method is tested as soon as possible.

Some methods do not need to be tested through unit testing as the fill () method and display (). As these methods form the foundation of the game, they are part of the first methods implemented. We can therefore visually check their operation.

Une fois le jeu totalement implémenter, une classe de test sera faite pour faire avancer le jeu suivant un scénario précis. Des pièges seront mis dans ce code et la sortie devra correspondre à la sortie espérée. (exploration test)

For classes Coord and Player, they only exists to store parameters. Only getters are tested and the setter as if it exists.

Estimations v1 :

Comprehension of source code	2
Analyse the code to cuti t into java class	3
Learn about java unit tests	5

User Story	Value
Create broad and display it	2
Enable manual renter coordinates	3
Implement User Interface	3
Create a player and place it on the board	2
Enable to play a turn - Blacken a bloc - Move a player	5
Enable to play several turn - boucle tant que la condition de défaite n'est pas atteinte	2
Final Test of the game	4

Estimations v2 :

Comprehension of source code	2
Analyse the code to cuti t into java class	2
Learn about java unit tests	4

User Story	Value
Create broad and display it	2
Enable manual renter coordinates	2
Implement User Interface	5 (abandonned)
Create a player and place it on the board	2
Enable to play a turn - Blacken a bloc - Move a player	5
Enable to play several turn - **boucle tant que la condition de défaite n'est pas atteinte**	4
Final Test of the game	6

Problem encountered :

- User Interface with JFrame : difficult to implement without changing a lot of thing.
- Problem of writing function and test for adjacent()
- Change the field to scan and test if the field have enough line (rewriting of almost all the function of Main for MainTest

Final Report

The test for the two classes Coord and Player just consist to verify the creation of the object and the getter (plus the setter for the class Player).

The class Parser has a function which take a string in input an extract the two first token of it to create a coordinate. The test verify if the coordinate has been create (if the coordinate is not valid, the null value is returned). And then we get the value X and Y to verify.

In the class Board, the main test method is isValid(). All the other test is based on it. To move/add a player or blacken a box, we verify by testing if the box is valid or not.

Some function such as fill() and display() do not have a test function. The board (char[][]) is private and I do not want to create a getter of it (isValid() can test some precise box of it but not its entirety). I have test this in the early stage of the program. Just create a board and use the function fill() and display() to watch the results.

The class main is not easy to test because of it structure. So I create a class MainTest. For the class MainTest, I have re write the function Main by adjusting it to a reading from a file. I just have to change the initialization of the Scanner.

Moreover each time I want a coordinate, I have to check if the file has a next line.

Therefore, due to the hashmap which contain all the player, I cannot extend the test to multiple player. Indeed, the function keyset() of the HashMap class return a set of player but without knowing the order (sometimes the first player is A and the other times it is B). So we can only test on one player.