Patchwork (intermediate rendering) : dev report

Project in Java language edited by: Massiouane Maibeche et Prashath Sivayanama – Groupe 3

Tables of contents:

I – Project presentation and objectives

Patchwork is a board game that is played by two players whose goal is to have the highest score.

Objectives are implement this game with the language "java", in order to be able to play it at first with two persons.

II – Means implemented

To begin, sources are separated 3 main package.

The first package is named "fr.uge.patchwork" and it contains five java file :

- Table.java (is a class), who represent both the main game board and the game board for players.

As regard the architecture, we chose a ‘’int’’ (1D) array to represent the game board and a two ‘’int’’ to represent the player1 and player2 in the board.
We have an int to represent the distance between two player, a boolean to represent if a player if he passes the other player, and a int (1D) who contains the the coordinates of a player at a time 't'
These fiels are initialisate in the constructor.
We may found several important method as ‘createMap()’ to initialisate the different event and case of the array, ‘pionCase’ to make the move of different player in the game array.
Here are also important functions related to the patch as ‘posePatch’ to allow the player to put their patch.
To remuse, we were chose to represent Tab with a class because some parameter can change during the game like as the (game) array. This integer array (2D) can contain one of several numbers to represent different cases during the game.

- Patch.java (is a record), we decided to represent the patch objects with a record whose fields are an integer 2D array for the patch, two integer who representing height and the length of a patch, and three integers respectively for the number of steps a player takes, the cost of this patch and finally the number of buttons it returns to the player.

This record contains an important method which is 'rotateMatrix' to rotate a patch in the right, the left, and to reverse a patch (in a patch board).

- Money.java, is a class who represents the player's or bank's wallet (formerly represented by a hashmap), it is represented by int fields.
  The constructor initiatize the bank of the game, the other method can be initialise the bank for a player, and manage the button in the game.
  We can find important method such as 'PiecePatch' who retributes button if player buying a patch and the patch contains button, 'pieceCase' which gives buttons to the player according to the number of squares he advances and 'achatPiece' who manage the transaction who player buy a patch.

- Player.java, is a class who groups the parameters of a player, it contains an array to put patch from the Table type, a list of 'Patch' which will contain the patches of the player, its button holder from the 'Money' type and a String who representing his name.
  The constructor initialize all fields. This class contains important method like as 'playerScore', 'patchSpecial' to control if the player is entitled to have special patch.

- Game.java, is a class that contains the elements to launch a game, it contains as fields, a list of Patch, a Money type to manage the game bank, a main game table (Table type), an integer to choose the player mode and an integer that will be the neutral pawn.
  The constructor initialize all fields, and this class contains importants methods like 'patchInitialisation2' to initialize patch from a file to have 33 differents patch, 'patchInitialisation1' to initialize a 2 forms of patch, 'posePatch' to allow the installation of patches in a game, 'eventDo' to display at screen all different patches during an purchase.

Here is the format of a patch in the Patch.txt file :

P A B
0 1 1
C D E -

"P" initialise a new patch, the "A" representing the heigth of the patch, "B" representing the width of the patch, the middle representing the patch, the patch consists of 1 and 0, (1 if it represents part of the patch, and 0 if it represents empty space). "C" representing the button that the patch brings, "D" representing the price of the Patch, "E" representing the case space value the player passes, the patch finish by a "-".

the file format verification analysis is rigorous, the format not respected can lead to a general malfunction.

The second package is named "fr.uge.patchwork.grapchics" and it contains two java file :

- Frame.java, is a class that contains different method to display the element during a graphical game like ‘printArrow’, ‘printTableGeneral’, ‘printTablePlayer’, ‘printPlayerInformation’, ‘printChoicePossible’ (for a patch), ‘printTablePlayer’ and ‘printWinner’.

- ManagementGraphics.java, is a class that contains different method to manage the event during a graphical game, like ‘eventDoGraphics’, ‘patchWantedRotation’, ‘pushPatchGraphics’, ‘buyPatchGraphics’, ‘choicePatchGraphics’ and ‘winnerGraphics’.

The last package is named "fr.uge.patchwork.main" and it contains the main java file :

- Main.java, is a class, there is here to load a "Patch.txt", allows the player to give himself a nickname, to choose a graphic or terminal part (and only in the terminal part to choose between the 2-patch mode and the 33-patch mode).

III – Conclusion

In conclusion, we have in this final phase to implement a version of the game "patchwork".
We thought of building classes and records, which for us seem to be valid.
We represented the patch object with the help of a record and then stored them in a list where it was necessary to use it.

We have represented the game boards in the form of an integer table (1D) in order to manage the player counters, as well as potential events during a game.

Unfortunately we did not finish the part which involves an artificial intelligence against the player.

Petite partie en Français :
En ce qui concerne le fichier build.xml, celui fonctionne comme ceci :

- ant clean : pour nettoyer le dossier classes.
- ant compile : pour recompiler le dossier classes.
- ant javadoc : pour regenerer une nouvelle javadoc
- ant jar : pour obtenir un nouveau jar*

*(Nous avons remarqué que la commande "ant jar" ne fait pas fonctionner le jeu patchwork en version graphique, mais uniquement en version ASCII. Pour pallier à cela, nous avons généré un jar à partir du projet qui quant à lui qui fonctionne, nous sommes désolé pour ce désagrément)