Patchwork (intermediate rendering) : dev report

Project in Java language edited by MAIBECHE Massiouane

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 implements 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 in 3 main packages.

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

- Table.java (is a class), who represents 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 two ‘’int‘’ variables to represent the player1 and the player2 in the board.
We have an int to represent the distance between two players, a boolean to represent which player passes in front of the other, and a int (1D) who contains the coordinates of a player at a time 't'.
These fiels are initialised in the constructor.
We may found several important method as ‘createMap()’ to initialise 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 chosen to represent Tab with a class because some parameter can change during the game like 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 integers which are representing the height and the length of a patch, and three integers respectively for the number of steps of the player, 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 to the right, the left, and to reverse a patch (in a patch board).

- Money.java, is a class which represents the player's or bank's wallet (formerly represented by a hashmap), it is represented by int fields.
  The constructor initialises the bank of the game, the other method can be initialised the bank for a player, and manage the button in the game.
  We can find important method such as 'PiecePatch' which retributes button if the player is 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' which manages the transaction when a player is buying a patch.

- Player.java, is a class which regroups 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 which is representing his name.
  The constructor initializes 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 initializes all fields, and this class contains important methods like 'patchInitialisation2' to initialize patch from a file to have 33 different patches, '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 a purchase.

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

P A B
0 1 1
C D E -

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

The file format verification analysis is rigorous, if the format is not respected it 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, it is here to load a "Patch.txt" and 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 have 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-ci fonctionne comme ceci :

- ant clean : pour nettoyer le dossier classes.
- ant compile : pour recompiler le dossier classes.
- ant javadoc : pour régénérer 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 fonctionne, nous sommes désolés pour ce désagrément).