Scrubble

Scrubble is a game based off of Scrabble. Scrubble is a variation of Scrabble (same rules, similar board layout), except that Scrubble is an online version of Scrabble. Instead of playing against another player, Scrubble players play against the clock. They have exactly two minutes to acquire one hundred points. If they are able to do so, they win the game. If not, they lose.

Just like in Scrabble, the player will compete to form words for points. The Scrubble board consists of 15 rows and 15 columns, where a single tile can fit in one cell. There are an unlimited number of tiles in the game.

The point values for the tiles are as follows:

1 Point - A, E, I, L, N, O, R, S, T and U.

2 Points - D and G.

3 Points - B, C, M and P.

4 Points - F, H, V, W and Y.

5 Points - K.

8 Points - J and X.

10 Points - Q and Z.

Notice: the more “common” the letters, the lesser their point value. The point value of each letter is also displayed on its corresponding tile.

Some squares on the Scrubble board can be used for extra points. If a tile is placed on a light pink square, its point value is doubled. If the square is dark pink/magenta, the point value is tripled. If the square is light purple, the point value of the entire word is doubled. If the square is dark purple, the point value for the entire word is tripled.

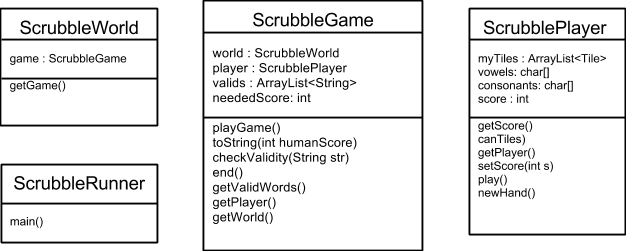
The game of Scrubble starts with the player randomly “picking” 15 tiles. Of course, the player does not actually pick the tiles; the tiles are pseudo-randomly generated, using Math.random(). A player can either place a word from the tiles on his hand, or receive a new random set of 15 tiles.

Placing a word is exactly what it sounds like. The player will be able to place letter tiles onto the Scrubble board at any location as long as the word connects to at least one of the tiles already placed on the board (with the exception of the first word placed). Once a word is placed on the board, it cannot be removed. In order to place a tile on the board, the player must click a location on the board. A drop down menu will then appear with all the tiles in the player’s hand. After the player selects a tile, the corresponding Scrubble tile will appear on the board, in the location that the player clicked. When the player accesses the drop down menu again, he will notice that it is missing an entry. After an entry is selected, it is removed so that the player cannot place a tile that he does not have anymore. This ensures that the player cannot play the same tile twice.The drop down menu also contains two other entries that are not for tiles-- “Use tile” and “Remove tile”. Clicking “Remove tile” will remove the selected tile from the board, and it will reappear as a choice on the drop down menu. Clicking “Use tile” will signal that one of the letters in the player’s placed word has already been placed on the board. It will also aid in verifying that the placed word is connected in some way to the amassing chain of letters on the board. This also helps us determine the actual word the player wants to place. Since we have a string that increases with the letter that has been placed to form the word, it is easily for us to receive the actual word.

Because it is quite possible for the player to get stuck and be unable to create a word from his remaining tiles, the game has been modified to allow the player to exchange his entire hand to contain an all new set of 15 tiles. However, exchanging the hand comes at a price; each exchange subtracts 15 points from the total score. Sometimes, this exchange is worth the loss of points. A new hand bring new possibilities. If the player is skilled enough, the loss of points is easily made up with a string of valuable words.

Because Scrubble is a game of words, a text file of valid English words must be loaded in somehow and have the program check for word validity after each user has placed his word. One way to do this is to load the words into a HashMap. The reasoning behind this is that HashMaps are very efficient at locating something quickly.

The class diagram of Scrubble is provided:



Where is the main() method of the game? The main() is in the class called ScrubbleRunner. ScrubbleRunner is very simple. Its only method is the main(), inside which it creates a new Scrubble Game and calls its playGame() method.

ScrubbleGame is the organizer of the game. It has four instance variables: a world of type ScrubbleWorld, a player of type ScrubblePlayer, a list of words of type HashMap<String>, and a minimum score of type int. The constructor of ScrubbleGame does a few things. First, it initializes the world by passing it itself as the parameter. The ScrubblePlayer is initialized as well. The list of words is initialized, and the file of valid words is read in through a Scanner and put into the HashMap. The minimum score is initialized to be 50. The player must get 50 points or above to win the game. Also, World’s show() method must be called to make the ScrubbleWorld visible. In addition to the constructor, ScrubbleWorld has a few methods. It has two getters; getPlayer() and getWorld() which return the ScrubblePlayer and the ScrubbleWorld, respectively. The boolean method checkValidity(String str) is used to check whether or not the string passed in the parameters is a valid word (if it is contained in the HashMap). The playGame() method is called from ScrubbleGame, and all it does is call the play() method of the ScrubblePlayer, which does nothing. Then, there is the end() method, which is called when the two minutes are up. In the end method, the setMessage() method of world is called, which calls the toString(int humanScore) method in order to display whether or not the player won. The integer passed as a parameter is the player’s score.

A ScrubbleGame contains a ScrubbleWorld, which extends World<Tile>. ScrubbleWorld is relatively simple. Its constructor is passed a ScrubbleGame as a parameter, which is used to initialize its own instance variable of type ScrubbleGame. Its parameter also calls the setMessage() of world to display the following message to the player: “Place a word on the board or receive a new hand. You must rack up at least 100 points to win the game. Keep an eye on the clock-time is short!” Its only method is a getter method named getGame() which returns the ScrubbleGame.

Scrubble uses Gridworld, so it retains the majority of Gridworld’s GUI code. Gridworld’s code is divided into four packages: info.gridworld.actor, info.gridworld.grid, info.gridworld.gui, and info.gridworld.world.

Scrubble requires the use of Scrabble tiles, which are to be placed onto the grid, like an Actor. Therefore, the Tile class and its children and their images are placed in the info.gridworld.actor package. The Tile class has four instance variables. A Tile must have a letter and a value so two instance variables are of type char and int. It also has an array of 26 integer values which match up with their corresponding letters. Lastly, there is a Tile, which will be explained shortly. The constructor of the Tile class takes one parameter of type char. This indicates what letter the tile is. Using this, the point value of the tile can be found with int index = Character.getNumericValue( ch ) - 10, where index can be used as an index (HaHa) into the array of 26 integer values. The instance variable of type char will be initialized to the char in the parameter. Along with instance variables, the Tile class has a couple methods, of which two are getter methods. getChar() and getVal() return the letter of the tile and the point value, respectively. Another pair is putSelfInGrid(Grid<Tile> gr, Location loc, Tile t) and removeSelfFromGrid(Grid<Tile> gr, Location loc), of which both are void. removeSelfFromGrid(Grid<Tile> gr, Location loc) calls the remove method of grid and passes loc as a parameter. The putSelfInGrid() method calls grid’s put method and passes it loc and a Tile as parameters. The Tile is created by calling constructTile(Tile t). The constructTile(Tile t) method checks what the exact type of Tile t is by calling t.getChar() and making a new Tile with that character. For instance, if t.getChar() were ‘A’, then the constructTile method would return a tile of the class TileA, which extends the Tile class. This is necessary so that the GUI can associate the image of the A Scrabble Tile with the class. That said, there are 25 other classes that extend Tile, and they are called TileB, TileC,...,TileZ. There are also images of the same name in the info.gridworld.actor package which correspond to the tiles of the same letter.

The second package info.gridworld.grid is left untouched, as there is nothing that needs to be changed.The third package info.gridworld.world has minor changes. The ScrubbleWorld class is added to that package.

The package with the most changes is info.gridworld.gui. The majority of the new methods are added to this package (ScrubbleRunner, ScrubblePlayer, ScrubbleGame, and JUnit). In order to change the buttons of the existing Gridworld GUI and change what the buttons did, GUIController class must be changed. The original step, run, and stop buttons are changed to “place”, “exchange” and “new hand”. In addition, new methods are added to accommodate all three buttons. The place() method does not do much except disable the use of the place button so that the user will know if he has already clicked the place button or not. The exchange() method calls the newHand() method of ScrubblePlayer to indicate that a new hand of 15 tiles must be dealt. The done() method is the trickiest.

Here is the pseudocode for done():

*if word was placed and word is first word OR word was placed and word is connected*

*if word is valid*

*set message - “valid word”*

*else*

*set message - “invalid word”*

*else if word was placed and word is not connected and word is not first word*

*set message - “use a tile already on the board”*

*else*

*score is score - 15*

Another major section in the game is to calculate the score the player receives. So, in the WorldFrame class in GUIController, there is a calculateScore(ArrayList<Location> locs) method that returns an integer and sets the score of the player to the score it calculated. This method is in WorldFrame because most of the information that needs to be accessed is in the this class. At the beginning of the class, there are 4 arrays that hold the locations of the squares on the board that contains double letter score, triple letter score, etc. In the actual method, an arrayList of the locations of the world placed is placed to the method. So, for every location in the arrayList, you check it against the locations in the 4 arrays at the beginning of the method. To ensure a fair and consist system, the location is first checked against the locations of double letter score and triple letter score. There are two boolean values in the the method, isDoubled and isTripled. If the location is in one of the arrays of DoubleWordScore or TripleWordScore, the respective boolean is set to true. Before the final value is produced, the score will be doubled/tripled depending on which booleans are true.

The GridPanel class contains all the methods of how the Grid is drawn and created. Though most of the class stayed the same, the grid must be modified to be colored. This is done in a very hard-coded way: There is a 15 by 15 array of colors and the colors are accessed in a for-loop that transverse through the array and color the squares.

MENUMAKER!!!!

Here is the class hierarchy:

