Program 5:

Planning/Process Document

Please read the House Keeping stuff it contains important info!

A few house keeping things before the planning doc officially starts:

a.) As far as the ./FileIn/ is using unsafe operations error it turns out that that happens when you use a Vector/List without giving it a data type. i.e. Vector vector = new Vector vs Vector<Integer> vector = new Vector<Integer>() So that has been fixed.

b.) The error you got by placing one tile on top of another none of us could replicate on either Window, OS X, or Linux let us know if it still persists and we will try and do something

c.) We added compiler suppression warnings to a lot of the methods. For example if a certain method uses “rawtypes” or something the warning text should no longer appear when you compile but we will see this could be pointless but we thought we’d try to at least give a cleaner compile.

d.) We changed the wrong move highlight it now just highlights the the tile with a read border. “It’s a bit thick why?” you might ask that’s just so that its more easily visible. To cancel it just click on the tile again. I was thinking of having it cancel if you clicked anywhere on the board but i’m afraid that this will conflict with the action performed especially if you click on another button. If you want us to implement this let us know.

e.) Another thing you might ask is “Why so many pop-ups?” Well It would have been nice to maybe hardcode a few of the options but for stuff like this doing that tends to cause too many unknown problems. So instead we opted for the safer approach by making sure that user is aware of the choices he/she is making.

f.) Paths, paths, and more paths. While working in eclipse all of the paths default to the current workspace so if a file is in downloads typing in any kind of path wont work we changed our eclipse IDE settings to “all of the computer” and it was fine I don’t know if this will be a problem for you when compiling/ choosing but it’s not a coding problem at least. In leu of this we added the your played.mze to our saveFiles folder thus you can simply select it from out drop down menu rather than through the path but both work.

g.) Another question you might have is why do the gridlines look doubled? This is so that when adding the tiles there are no gaps between the images unfortunately this causes the borders to overlap a little.

h.) Tile corners should now be completely readable. If there are issues with different line thicknesses which I believe you mentioned I think this is simply due to the BufferdImage graphics being a little finicky but I don’t know of a way to fix that

i.) You may not like this and if so we will change it but since our default is in a resource we are not worried about not being able to load the default file as such we never need to present a blank slate window. This is doubly good because there were multiple strange error occurring when we had tried to implement that functionality however if you load another maze and that file does not exist a pop-up is shown notifying that the file has not been found and nothing happens.

j.) Right now if you have changed the game state then save and then load the load will prompt for a save we were unsure if this was the way you wanted it so we have kept it this way because it sticks with the instructions we can change it if need be

k.) right know if you load again after having just loaded a default maze it will prompt you for whether or not you want to start a “New Game” using that default maze the new game randomizes the tile and rotation just like the old New Game button used to. I hope this is what you wanted.

l.) Class.getResources() tends to pull from the bin folder we wanted it to pull from the src folder to change this we *had* to mess with the eclipse IDE settings hopefully this is not the case on your end.

m.) We had some issues with monitor bounds for the other people in the group so the tile grids might seem little high but that had to be done the current setting should work for any monitor but its an easy change if there is a problem. Simply go to MiniGame class and scroll down to WindowSetUp() then adjust the first value of the insets for the tgridL and tgridR.

n.) Lastly the program was tested on OS X, Windows, and Linux. We encountered a problem in Linux whereby if your version is too low some stuff does not work but I think you have the latest so it should not be a problem. We did extensive debugging for this but we are only human if we missed anything let us know and we will fix for final.

<———————————>

This week’s collaboration was more restricted to meeting in class and online discussions compared to previous weeks due to rising workloads across the team. That said, while we conducted most of our communication online, we shared and worked together as a group more than previous weeks. This is in part due to the number of challenges that required collaboration and consensus.

Quick rundown of the problems we dealt with:

1. File not found Error pop up
   1. Simple, if FileNotFoundException is thrown, rather than printing stack trace, show a pop-up.
2. Allowing for relative or full paths in Load and Save
   1. This one is a bit of an unknown since there is no real way to tell what the user is typing in, we are assuming that any path can be typed and the file will open i.e. no restrictions just default.
3. Overwriting an existing save
   1. This is also fairly simple: first pop-up that asks whether or not to overwrite assuming that file with same name exists then if “yes” simply save the new file under the same name and OS will do the rest
4. Replacing the current maze with the loaded maze
   1. This one is a little more involved, if they load a played maze, we want to simply replace tile images. If on the other hand they load a completely new maze, we want to read the new file and treat it as essentially a default. That means that randomization and random rotation still need to happen and it needs to happen every time that file is loaded as “new game” so this comes down to the file signature. If it’s a save file simply change icons. However, this requires that Load class know about either TileGrid or MiniGame. Since MiniGame is the glue, Load should really only access static methods from MiniGame therefore making stuff like the tile grids and board static is a must as well as some stuff in tile TileGrid (see code for exact methods if need be). On the other hand if the maze is new then you simply need to pass it to FileIn. Somehow, it turns out we can actually use the “load save” mechanics with a couple of tweaks (see public static void newDefault() in class MiniGame for specifics)
5. If current maze has been changed, prompt for save
   1. This requires that some sort of static boolean flag is set. The best way is to leverage the swap function in MiniGame to set this flag, i.e. if I have swapped a tile then I have changed the game state. Not that you can’t just cycle through the board and grids looking for null icons or something because then if I load a save and I go to quit or load a new one without having actually changed anything the program will prompt for a save when it should not.
6. How to decide operation based on file signature
   1. Easy. Read the first 4 bytes, convert to int and if that int == 0xcafebeef or 0xcafedeed act accordingly. Apparently though this seems to be a problem on Mac so it’s debatable as to how robust it is as an identification method.
7. If file being loaded does not have correct sig, give error and show game window with tile location, board, etc but without tiles on it.
   1. Requires UI adjustments. Mainly that the button grid needs to show up before all other components and also that the TileGrid class should NOT add images to tiles if the file is wrong but it NEEDS to STILL display the empty tiles.
8. Maintaining randomization if new Maze is loaded or if Default is reloaded
   1. See answer four. We are simply leveraging the same process however ONLY if the new file loaded is a brand new maze we don't want the save tiles to be randomized at start.
9. Reset should reset back to last loaded position
   1. Could be complex: current solution——> simply reload the same file
10. Reset does not give prompt to Save file
    1. Reset should NOT change the game state. That means that if we press the reset, “reset()” needs to set the stateChange variable to false.
11. Handling “Cancel” buttons on dialogs
    1. This ended up being quite the annoyance because, if selected, cancel, meaning “I actually don't want to do this, I want to keep playing”, then a null pointer exception came up. So the best way to handle this is by having cancel NOT progress the program. i.e. if cancel is pressed simply return.
12. Program should not ask for save if maze has been solved
    1. This is simple it’s another static boolean flag. In this case, it should be isWon if that is true then if I select quit or load I should not be asked to save the game because otherwise I'm saving the solution! So when the win check triggers if that comes out to true then isWon should be set to true as well
13. Quit should prompt for save if game state has changed
    1. Now that we have our isStateChanged boolean flag this is easily handled.
14. How to tell the game board / tile grids which tiles they need
    1. This revolves around the save and load functions and how they operate. If they operate the way that Phill made them originally then the location data is automatically stored. If it’s the instructed method you need to save the location as a number 1-32 and then when you read it back in, act accordingly.
15. Some classes need to communicate with each other. What’s the best way of doing this while maintaining encapsulation?
    1. Private static variables and public static functions.
16. If I select a Tile for swap and then select say the “Quit” button or something else what should happen?
    1. Make sure to trigger swap ONLY if t1 isInstanceOf Tile and ONLY if t2 isInstanceOf Tile. isInstanceOf is an actual built into Java functionality. See the bottom of the ActionPerformed method for more detail

This week was significantly tougher to meet all the requirements laid out in the instructions and, along the way, multiple ways of doing things were created. For example, early on we had a way of saving that saved the tile icons and identifier and when the game is loaded it just applies those to the tiles. While this worked fairly well, we ran into the problem of whether or not it would be considered a proper solution to the problem. Currently, there are still two save file formats because of an initially different save style, however both still work. In the event that the path method doesn’t work, the played.mze that was given is in the saveFiles folder in the SRC. Another problem that yielded multiple solutions was adjusting the layout to account for some of the new changes that were made, one way was to change the overall dimensions and another was to alter the tiles. Also, the grid on the board looks rather odd, but it ensures that there is no gaps between tile images. In addition, we never show a blank game because we always load from the original default when starting up. This will always be true because instead of a path, we use a resource to get that file. Overall, how to handle saving and loading gave us the most decisions to make, though we believe that we have reached a solution that fulfills the requirements of the prompt rather nicely.