TileManager & EntityManager & GamePanel

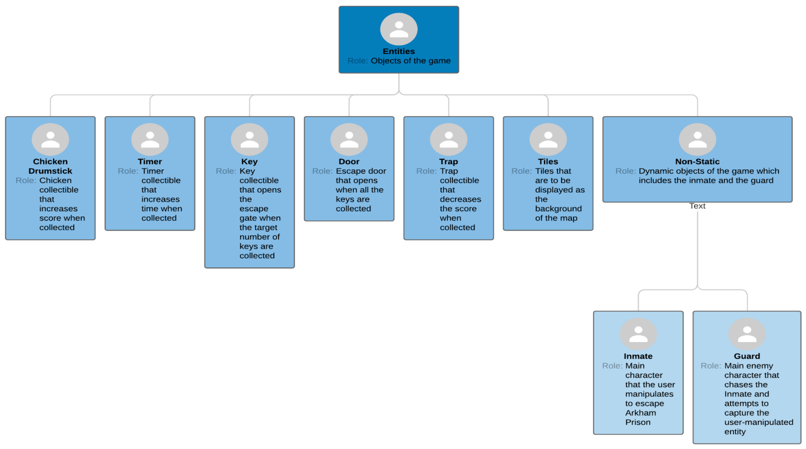
With our tile manager and entity manager classes, we had many different initializations that resulted in **long methods**. For the tile manager class, we had a register image class that consisted of many different calls to the setup function. The method took three different values (index number, name, and a boolean value). We decided to fix this problem by adding all the values to a text file called tileSetUp.txt which contains all the values/booleans associated with each function call. This allowed the method length to be reduced.

Additionally, we did something very similar to this for our EntityManager class where we were containing many createObj() method calls in the setEntityLevel classes. Originally we had three of these classes (setEntityLevel1, setEntityLevel2, setEntityLevel3) where all three of these methods were doing very similar things (creating the objects associated with each of the three maps in our game). Similarly, with what we did with the tileManager class, we decided to move all these significant values needed for the createObj() function call into a text file called EnetityLevel1.txt, EnetityLevel2.txt, and EnetityLevel3.txt. Furthermore, this allowed us to merge all the setEntityLevel functions into 1 more shortened and more cohesive method.

The refactoring we did through the use of textfiles helped a lot in terms as it allowed for easier implementation and reduced unnecessarily long methods. Furthermore, this allowed for better maintenance and future modifications. Refer to commit SHA for `6e0c9197, 3c0ac066`.

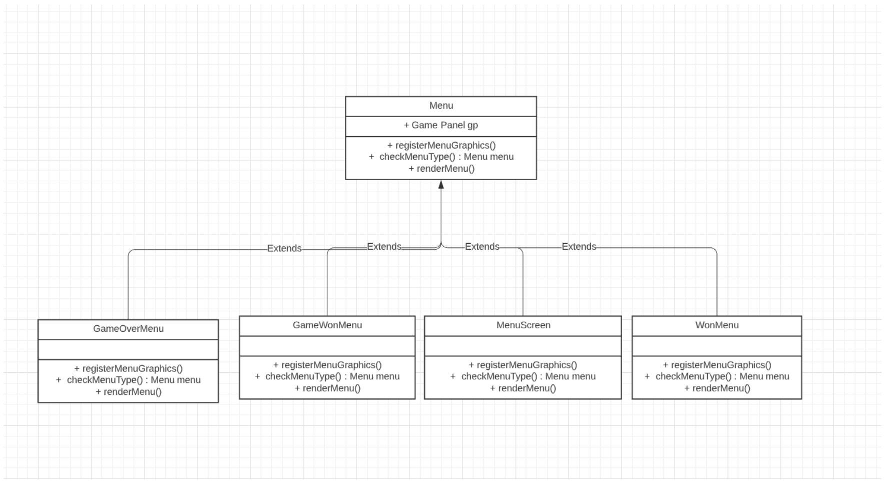
MovingActor superclass / Removing duplicate code

The problem we had with these classes was that both the inmate and the guard class had identical codes for the same methods. The code smell here was that we had **poor hierarchy and duplicate code**. On top of that, the two classes **lack any sort of hierarchy** that could differentiate them from other entities. We decided that it would be best to create a superclass called MovingActor that would act as a superclass for both the inmate and the guard and to have MovingActor contain the duplicate code that is shared between the guard and the inmate classes. Doing such allowed for our program to have a better overall coherence and less duplicate code. Refer to commit SHA for `f30c2646, 580dc0d8, 9889ebd4aa`


Entities
Role: Objects of the game
Chicken Drumstick
Role: Chicken collectible that increases score when collected
Timer
Role: Timer collectible that increases time when collected
Key
Role: Key collectible that opens the escape gate when the target number of keys are collected
Door
Role: Escape door that opens when all the keys are collected
Trap
Role: Trap collectible that decreases the score when collected
Tiles
Role: Tiles that are to be displayed as the background of the map
Non-Static
Role: Dynamic objects of the game which includes the inmate and the guard
Text
Inmate
Role: Main character that the user manipulates to escape Arkham Prison
Guard
Role: Main enemy character that chases the Inmate and attempts to capture the user-manipulated entity

Adding Menu super class & Game Panel implementation / Removing Duplicate code / Removing unnecessary conditionals

Within our game we had 4 different types of menus (GameOverMenu, MenuScreen, GameWonMenu, and PauseMenu) which all had a register and render method within them. Although the implementation for these methods was different for each of the menus, the problem was that when we use them in the GamePanel class, we had to create multiple different conditionals and instances for each type of menu. The code smells involved with the menus were that we had **poor hierarchy and unnecessary conditionals** in the GamePanel. In order to fix this issue, we decided to have a sort of interface implementation for the Menu superclass that contains the method for menu registration and rendering and had the menu subclasses override them with their own implementations. Furthermore, this allowed us to clear up the long paintComponent() method in our GamePanel class to a system that takes advantage of the superclass implementation to know what type of menu is being created. We also created a checkMenuType method to check what type of menu is being created instead of using necessary conditionals in the GamePanel. Refer to commit SHA for `7636ecfd.`



Badly Structured project / Many files were not packages and not organized

With our project folder, we had problems with folders not being in the correct hierarchy and therefore only some of us in our group were able to run the program. Our resources folder was originally in the src directory and had problems running on different IDE or operating softwares.

Since we already had written code, we did not want to lose the progress that our team had made. The way I fixed the problem was by making a copy of what we had and saving it in a different location, then I deleted the original file and created a whole new Maven project from scratch which I had to make sure that everything was in the correct hierarchy and was able to compile and test with maven. After successful testing, we were able to put the previous written code into the new Maven project and run it on everyone's system. Refer to commit SHA for `ebdceaec, 049b722e, df6dda8f.`

Unused / useless methods and Variables

When working on Entity.java, we noticed a number of methods which were never used in the program and they served no purpose. We decided to completely remove them from the Entity class. Refer to commit SHA for `2917622a.`

There was an encounter with a couple of unused variables in the GamePanel.java. The variables were of type string and served as the file location for each level of the game which was removed completely. Refer to commit SHA for `8786f78f.`

Lack of Documentation/Comments

We encountered problems with understanding the logic of some methods which caused a lot of confusion in trying to fix bugs due to the lack of comments and documentation. There is a method called pickUpObject in the file inmate.java which takes an integer as a parameter. We had a hard time understanding what the parameter of the method is used for, which brought a lot of confusion in trying to edit the code.

The way we approached this problem was getting the people who are responsible for each method that didn't have proper documentation, and encouraging them to add proper documentation to their code. Refer to commit SHA for `f456c0aa, f077effa.`