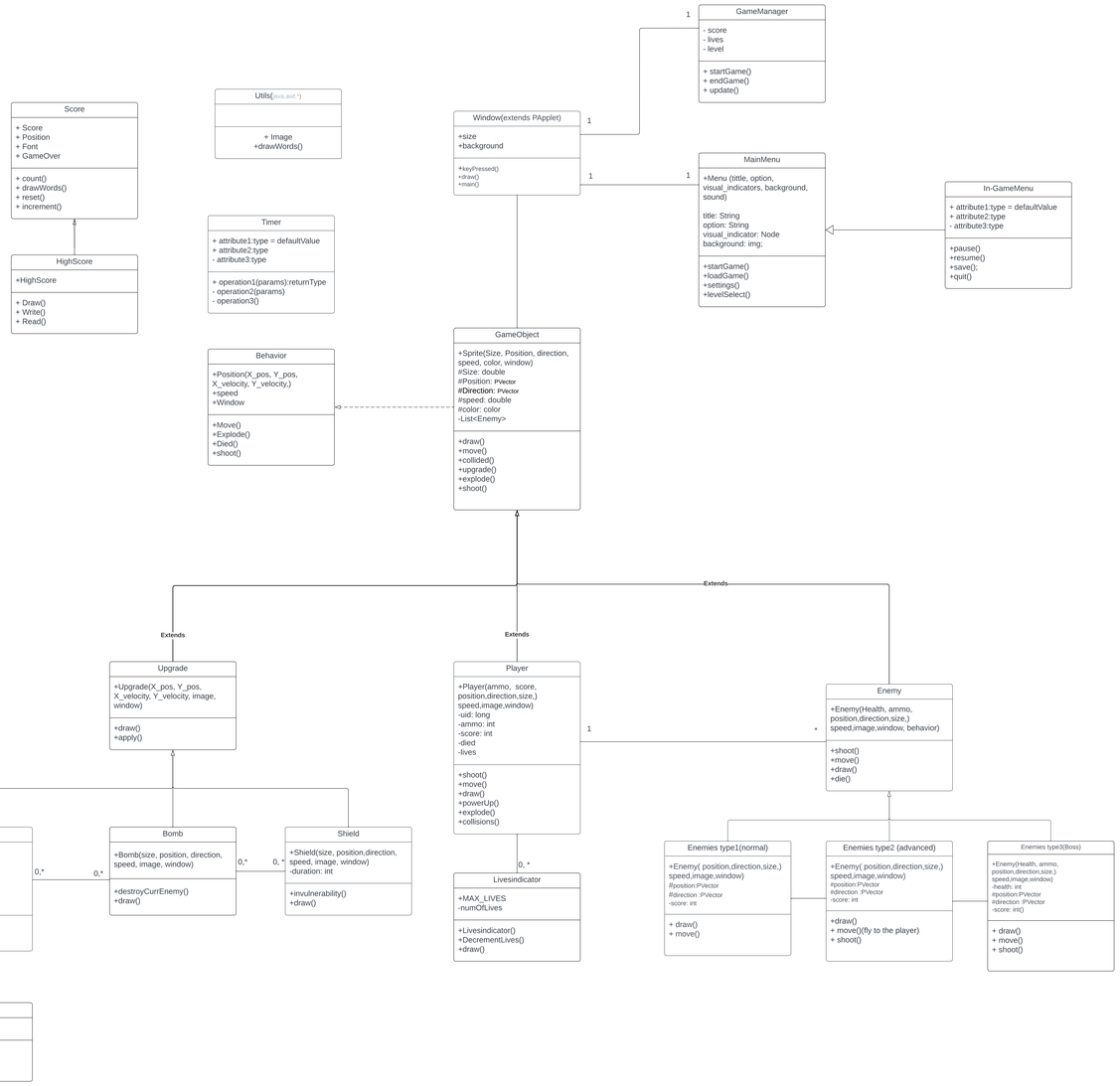
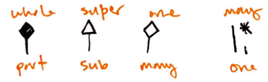


the window class might be responsible for displaying the game on the screen and handling user input, while the game manager would be responsible for controlling the game logic, updating the game state, and controlling the behavior of other classes in the game.
In this case, the window class and the game manager would have a one-to-one relationship, with the window class having a reference to the game manager, and the game manager having control over the window class. The game manager would use the window class to display the game and receive user input, and the window class would use the game manager to access the game state and update the display accordingly



For the game, some potential areas to test could include:
Gameplay mechanics: Test whether the player's ship moves, shoots, and captures enemy ships as expected.
Level progression: Test whether the game advances to the next level as expected when all enemies have been defeated.
Scorekeeping: Test whether the player's score increases correctly for shooting down enemy ships and bonus items.
Difficulty: Test whether the game's difficulty increases as the player advances to higher levels.
Audio and Visuals: Test whether the game's sound effects and graphics work as expected and are synchronized with gameplay.
User interface: Test whether the game's menus and controls are easy to use and provide the expected functionality.
Performance: Test the game's performance under various conditions to ensure that it runs smoothly on a wide range of devices.
Stability: Test for any crash or bug in the game.

DATA structure
Here are some possible data structures that could be used for a game like Galaga:
1. Array or List: An array or list can be used to represent the game board, which is a two-dimensional grid of cells. Each cell can represent either empty space or a game entity such as a player ship or an enemy ship. The array or list can be updated each frame to reflect the current state of the game.
2. Queue: A queue can be used to manage the movement of enemy ships. Each enemy ship can be added to a queue and moved one step at a time. This ensures that enemy ships move in a coordinated and predictable manner.
3. Stack: A stack can be used to manage the bullets fired by the player. Each bullet can be added to a stack and moved one step at a time. This ensures that bullets are fired and move in a predictable and consistent manner.
4. Linked List: A linked list can be used to manage power-ups that appear randomly on the game board. Each power-up can be represented by a node in the linked list, and new power-ups can be added to the list as they appear.
5. Hash Table: A hash table can be used to store information about the game entities such as their position and state. This information can be accessed quickly and efficiently using a hash function.
6. Tree: A tree can be used to manage the different levels of the game. Each level can be represented by a node in the tree, and the player can progress through the levels by moving from one node to another.

These are just a few examples of the data structures that could be used for a game like Galaga. The choice of data structures will depend on the specific requirements of the game and the programming language used to implement it.

Now you have enough to create a rudimentry `JSON`-like object. If a `MyDictionary` can contain either `MyDictionary` or an array of `KVPair`, it's effectively just a `JSON` structure in Java memory. Now try writing out your `MyDictionary` as a `JSON` file. You can essentially just modify the `print` method to instead build a string of key-value pairs. If you want to challenge yourself (this is not required) you can try to (1) deal with multiple levels of nesting; and (2) parse from a JSON file into the `MyDictionary` structure.

TODO 3 (Project): Plan your JSON data structure for your project

Along with your team, start to design the data structure that you'll need for your project. It's OK for it to be a draft, but have something that can be written in and out to a real file to show Paul for next week. Make sure to include everything that is important for your program state. For example, the position of every enemy, or the tiles on your game board (if you have those kinds of projects).