GREED

**## Program Structure**

---

The project files and folders are organized as follows:

root (project root folder)

+-- greed (source code for game)

+-- game (specific game classes)

+-- \_\_main\_\_.py (entry point for program)

+-- README.md (general info)

Following the same pattern as in the rfk game, we will have three directories:

**>Casting**

* Actor: shared logic for changing the displayed text and screen position
* Cast: stores the actors in a dictionary for later access
* Gems: Actors with a points value

**>Directing**

* Director: Controls the game flow and updates the actors as needed

**>Services**

* Keyboard Service: listens to the keyboard for any inputs
* Video Service: Creates the program window and draws to the screen

**Main()**

* Creates the necessary objects and starts the game loop

**Casting**

The classes needed are the nearly same as rfk, with the following changes:

**Gems**

* Inherits the Actor class
* **\_points**: an integer that holds the number of points the gem is worth

Rocks will be anti-gems that have negative points

* **set\_points()**: a method that sets the **\_points** attribute to a specified value
* **get\_points()**: retrieves the value of the **\_points** attribute

**Directing**

**Director**

The classes needed are the nearly same as rfk, with the following changes:

* **\_do\_updates():** additional code will move all the falling gems and rocks, while also updating the players score when colliding with objects

**Services**

These classes needed are the nearly same as rfk, with the following changes:

* **keyboard\_service()** receiving modification to only check for right and left arrow presses.

**Shared**

* These classes should be the same as in the rfk game, no changes are needed.

**main()**

* **Main()** will be changed to create gem and rock objects from the Gem class. Additional adjustments need to be made to ensure the player is placed on the correct row and objects are created with the necessary attributes.