Custom Program OOP 2017 Semester 1 Design Document
Andrew Davis

A list of the primary objects that make my game tick:

Game Manager:
Game Manager is the the primary object. It handles organising other objects to perform their functions. From prompting the player to do something based on input, instructing the Physics Manager to handle collisions, or the Graphics Manager to render the the game.

Memory Manager:
Memory Manager is a singleton that has references to most game objects. It provides a point of access for other objects to interract with game objects, for example the physics manager to access and all collidable objects to check collisions. And as a point of access for the factory to give an object to the memory manager and the memory manager will handle where it is then stored.

Graphcs Manager:
Graphics Manager is a singleton whose purpose is to coordinate all objects that are renderable to render themselves, as well as handling rendering of the background and the HUD. Along with standard Swingame calls to clear the screen and refresh the screen.

Factory:
Factory is as singleton builds objects that appear on the screen. From bullets to the player. It uses memory manager methods to pass the new objects to the memory manager.

Physics Manager:
Physics manager is a singleton that has a primary function of checking collisions and ensuring that the objects perform their required methods to handle them in a safe manner.

TimeManager:
TimeManager is a singleton that handles timing of different events. Timemanager creates a list of Phase objects and instructs them to tick their counter and perform their duty then removes them from the list.

Phase:
Phase objects handle what spawns and when in coordination with the TimeManager. The child classes of Phase offer different types of game phases with different enemies. Standard phase offers a more dynamic way of spawning enemies by relying not on code, but data stored in a CSV file to handle what drone enemies are spawned. Whereas boss phases are boutique classes that handle creation of a very specific enemy. A wait phase also exists that provides a way to prevent the next phase from starting till the current phase is over. For example you may want to defeat a boss before a new phase of enemies are spawned.

UpdateableObject:
Updateable objects forms as the parent class of most objects displayed on the screen. It is an abtract class. It also implements the Renderable and Collidable interfaces. It's children typically rely heavily on it's Update() method using its fields _x _y and deltaX and deltaY to handle movement, while the

Renderable and Collidable methods tend to be more specific to the child.

Ship:
Ship is a child of Updateable Object and has the definitive Render() implemention for objects that render sprites. It serves as a parent class to both enemies and the Player.

Enemy:
Enemy is a child of Ship. It like UpdateableObject is an abstract class. Drones and Bosses are children of enemy. It serves primarily as a way to group all enemies together for collision detection purposes and to provide enemies a shared base to handle collisions.

Player:
The object controlled by the player. It is a child of moveable object, it is manipulated by Game Manager based on user input.