Firstly, for homework1, my function was quite simple with 1 playable and 2 non playable object classes, all of which was extended from the SFMLI rectangle class, functions in these classes included movement function within the playable class that detects wasd input and then move accordingly, at this point there wasn't any collision detection with in the playableChar class. As for the other 2 classes, one of them is a stationary platform class that is just drawn with a different color and layout, differ from the regular rectangle class that it detects collision, fluctuating up and down by 20 unit whenever it collide with a moving platform. As for the moving platform, at this point, it only moves horizontally, bouncing back and forth from the edge of the screen and then aforementioned stationary platform, causing the stationary platform to fluctuate up and down upon contact. Also, currently, the player class only have the x and y speed field, same goes for the moving platform class.

Next, for homework 2, since there will be alot of events that will happen to both the playableChar and the platforms soon, including moving, shooting, speeding up, bouncing off of walls, moving vertically continuously either upwards or downwards (this goes for multiple moving platforms, as seen in the most recent homework). Because of all this events, it makes sense to make the engine multi-threaded, I decided to make 2 threads, one that handles anything that has to do with platforms and the other dealing with player interactions. Then, I implemented a timeline class, this class included getTime which gets the real time, pausing, unpausing, changeTic rate (makes the game faster/slow down) and a boolean isPaused(), I have written about the specific implementations of this class before so I will not go into it here, so I will be brief, this timeline is integrated into the movements of everything that moves in the game, from the playableChar to the movingPlatform pausing by pressing P will stop all actions or movements in the game, and then you can press P again to unpause, regaining normal function. Next, for my first implementation of the server client system, I decided to hard code the system to only be able to handle 3 clients, at this point,, each client will send a hello message to the server and then as the clients come, the server will increment up the client count, if it's equal to 1, the server send out the message with of client1 back to the first client, so on and so fourth with client 2 and client 3 (for 2 and 3, when there are 2, the server will a message to both Client 1 and 2, same concept apply for 3 clients). My implementation at that point used 5557, and 5558. Also, the server to keep a map of each client and their position with the clientID as the key and their position being the value. The idea is for each client that connects to the server, the client will send a req rep Hello message and the server will send a message back, assigning an ID to this client. Then , when the client sees the ID (In this case, it can only be ID0, ID1 or ID2, this is the same way I implemented part 3 where I am only making use of 3 clients, future version of the game will allow more clients) it will either draw the character for client0 and establish it's movement or draw the character for client0 and client1 and establish client1 movement or draw client0, client1 and client2 and establish client2 movement. Next, the client sends the updated position of these characters to the server and the server broadcast this new positions through pub sub to the rest of the clients. This system was very faulty and was massively improved upon in homework 3.

For homework 3, the big change was the complete overhaul of the server and client system, a second moving platform, collision detection with the the playableChar as well as a moving camera and a deadzone. I added a horizontal moving pattern to the moving object, shown by a second movingPlatform that moves up and down, bouncing between the edges of the screen and a new stationary platform I made that was obscure by side edges of the screen (this stationary platform can be revealed more by moving sideway, i Implemented a view functionality to make it like a 2d platformer). At this point however, the playableChars still doesn't have collision with this movingPlatforms. However, I made a new stationaryPlatform that when collides with a player, resets them to their spawn decision. In retrospect, I should have made this collision be with the movingPlatform. Also, for the deathzone, I made it so that if the player crosses a y point of 600, they die, which means they get reseted to their spawn point and lose a life (I set a new field in the player class that's called life, set to 4, when it hits 0, all movements of the player is suspended.) In retrospect, I should have sent this field across server and clients too, so that when each window checks their list of playableChars can check if the player is alive and not draw them if they are dead (lives = 0). Lastly is the refractor of the server client interactions. I did multithreading for each clients, reading and sending date to and from the server, I decided to keep a list of connected clients/playable characters on both the server and on each clients, as each clients connects, the server will publish a signal letting every other clients know to establish a new character in their list, publishing their location works similarly. The GameObject I decided to send is a playableChar object, to send this, I decided to pack all if it's information into a string and the receiver can unpack that string and use setters to edit their own copy of that playable object for something like updating it's position. This is where alot of the refracting of code from HW2 came in. for section 2 and section 3, all clients are controllable by WASD input and their position is updated in real time on each window of each client. In reference to this part of the code, I still haven't had the heartbeat model working.

For Homework 4, I changed alot of interactions in the game into events, to be precise, I changed the playermovent, new player spawning, player dying and collision. Without going into too indepth what I did as I already did in HW4 write up, the biggest changes include moving the playerMovement/ wasd key detection from inside the playableCharacter class into the main class in Client, now, the playableMovement function return a boolean instead, being called whenever wasd is pressed, then if it's true, the movement event is raised, and the actual moving is done in the event implementation, passing in the wasd key as well. The next big change is in the death event, with player's lives decreasing in the even implementation.

For home work 5 and the rest of what was done, I have include all details in the homework 5 write up. To reflect on it all on how well my engine works for a second game, I honestly think it works quite well as my playableCharacter already has most of the functionality it needed, I just needed to add a new bullet, shooting that bullet as well as player interaction with enemy entity.