`public void testinitSettings()`

Tests to ensure the game settings configurations are correct. Evaluates the graphical settings, title of application, and scene correspondents. All of it is evaluated through the Gamesettings class variable.

`public void testinitInput()`

Checks the input for the keyboard. Specifically, ensuring that the key down presses are correct and have the intended resulting effect. Function being tested inherits initinput() for parent class. Edge cases are tried thoroughly.

`public void testinitGame()`

Checks for correct setup of game configurations. Variables, starting values, and points on the grid are checked for correct setup. The evaluation is configured to test the two dimensional grid for exactly correct algorithmic configuration.

`public void testinitPhysics()`

Ensures accuracy for physics in initialization in the application. Make sure that the collision handler operands are correctly executed. Allows for testing that the instantiations of abstract classes produce correctly.

`public void testinitUI()`

Test the setup for configuration variables for user interface. Makes sure that local variables pertaining to the class are correctly set up. Cross checks with children of class to ensure that the objects are sufficient to standards.

`public void testspawnPath()`

Check the thickness of the path to ensure correctness. Cross check math of width and height so it can be objectively validated. Tests the spawndata object and corresponding spawn function to ensure accuracy.

`public void testspawnEnemy()`

Check the initial variables. Ensures that the spawn function is correctly setup for the corresponding function. Makes sure that the spawn data initialization is correct and accurately portrays the math backed algo.

`public void testspawnMonument()`

Check the initial variables. Ensures that the spawn function is correctly setup for the corresponding function. Makes sure that the spawn data initialization is correct and accurately portrays the math backed algo.

`public void testonEnemyKilled()`

Ensures that the levelEnemes variable decrements properly. Ensures that the xMark variable correctly configures the user interface to act accordingly. Also, checks that the entity enemy correctly validates the event.

```
public void testreduceHp()
```

Checks the conditional for the player.getHP() function returns less than a decreased amount of variable condition. Ensures gameOver() function returns properly and is completely functional. Checks, hplabel local variable for correct test being set from Player.getHp() function.