TESTING PLANS

<u>Collectable Entities Tests:</u>

Arrow Tests:

ArrowContructor (unit test):
- Tests the constructor to make sure an arrow can be made with the given parameters

ArrowsConsumedByPlayer (integration test):
- Test that the arrows can be collected by the Player, added to the inventory
- After, check that the when the arrow is consumed by the Player that the arrow is no longer in the inventory

Bow Tests:

BowConstructor (unit test):
- Tests the constructor to make sure an bow can be made with the given parameters

BuildBowThenUsedInBattle (integration test):
- Test that the bow can be created, materials have been used and is now in the Player inventory
- Test the bow can be used in battle and then after durability has been depleted then check the bow is removed from the inventory

Key Tests:

KeyConstructor (unit test):
- Tests the constructor to make sure key can be made with the given parameters

KeyConsumedByPlayer (integration test):
- Test that the key is added to the dungeon
- Test that they key can be collected by the Player, added to inventory and then removed from the dungeon
- Test that when the key is consumed and used that it is no longer in the Players inventory

Shield Tests:

ShieldConstructor (unit test):
- Tests the constructor to make sure shield can be made with the given parameters

BuildShieldThenUsedInBattle (integration test):
- Test that the shield can be built with the correct materials, then check it is in the Player’s inventory
- Test the shield in battle simulation until it breaks, check that the shield is indeed broken and no longer able to be used

Sword Tests:

SwordConstructor (unit test):

- Tests the constructor to make sure sword can be made with the given parameters

SwordConsumedByPlayer (integration test):

- Tests the sword can be added into the dungeon, and can be collected by Player
- Test the sword in battle simulation and then check that the durability is depleted afterwards and no longer in the Player’s inventory

Treasure Tests:

TreasureConstructor (unit test):

- Tests the constructor to make sure treasure can be made with the given parameters

TreasureConsumedByPlayer (integration test):

- Test that the treasure can be added into the dungeon, and can be collected by the Player
- Test that the Player inventory now contains the treasure and then when consumed, is no longer in the inventory

Wood Tests:

WoodConstructor (unit test):

- Tests the constructor to make sure wood can be made with the given parameters

WoodConsumedByPlayer (integration test):

- Test that the wood can be added into the dungeon, and can be collected by the Player
- Test that the Player inventory now contains the wood and then when consumed, is no longer in the inventory

Potion Tests:

PlayerUsePotionTest (integration test):

- Test if the player picks up an invincible potion, the inventory will increase by one.
- Test if ths player consumes the potion, its state will change to InvincibleState and the potion is removed from inventory.
- Test if the potion only last for the given limited time
- Test if the potion effect is queued when another potion is consumed.

Key Tests:

PlayerUseKeyTest (unit test):

- Test Player’s inventory has increased by one after pick up the key

- Test if the player has the right key to the door he/she is at, the door is opened, and the key is removed from the player's inventory.
- Test if the player has the wrong key, the door state would remain closed

Goals Tests:

BasicExistGoalTest (unit test):

- "goal-condition": { "goal": "exit" }
- Test if the player successfully exits and achieves the exitGoal, the goal string should be empty after exit.

BasicEnemiesGoalTest (unit test):

- "goal-condition": { "goal": "enemies” }
- Test if the player successfully destroy all enemies.

BasicTreasureGoalTest (unit test):

- "goal-condition": { "goal": "treasure” }
- Test if the player successfully picks up certain number of treasures.

BasicBouldersGoalTest (unit test):

- "goal-condition": { "goal": "boulders” }
- Test if the player successfully push boulders on to switch

GoalCompleteInOrderTest (unit test):

- "goal-condition": { "goal": "AND”, “subgoals”: [{“goal”: “enemies”}, {“goal”: “exist”}] }
- Test the player must destroy all enemies and then get to the exit, not the other way around.

Player Tests:

testPlayerMovementUp/Down/Right/Left (unit test):

- Simple tests to check if the player is able to move in the given direction without any other entities obstructing it
- Tests moving once in the given direction

testPlayerMovementRandom (unit test):

    - Test to see if the player is moving to the correct position based on multiple directional inputs

Spider Tests:

testSpiderBasicMovement (unit test):

- Test to see if the spider is moving in a circular pattern around the given spawn point
- No obstacles

firstTickMovementWithBoulder (unit test):

- Testing spider remaining still if there is a boulder above its spawnpoint testSpiderBoulderInPath (unit test):

- Tests to see if the spider reverses direction when coming into contact with a boulder
- One Boulder in the way at Position(x + 1, y - 1)

testSpiderBetweenBoulders (unit test):

- Tests to see if the spider remains still after moving in between two boulders

spawningSpiderWithSpawnRateX (unit test):

- Test to see if spiders spawn on the map at the correct tick

Zombie Tests:

BasicMovement (unit test):

- Testing random movement of the zombie

SpawningZombieWithSpawnRateX (unit test):

- Testing spawning zombies with different spawn rates

ZombieSpawnWithWallsAroundIt (unit test):

- Testing spawning zombies around a spawner that has 0..* many walls around it

Static Entities Tests:

BasicStaticEntities:

- Test that the static entities can be created and placed in the dungeon that are within the dungeon map file