Test plans

Unit and integration tests

Moving Entity:

- Zombie
    - Check random movement
    - Check same movement constraint as player
    - Check movement under effect of potion
- Spider
    - Check clockwise movement in normal circumstance
    - Check anticlockwise movement when hit a boulder
    - Check movement under effect of potion
- Mercenary
    - Check follow player
    - Bribe
    - Check movement after bribed
    - Check movement under effect of potion
    - Check same movement constraint as player
- Player
    - Check player movement by just give the position and dungeon information
    - Check the behaviour of player move in to different static entities
    - Check if player start a battle if move in to a moving entity(i.e. enemy)
    - Check if the player return the correct Dungeon response.
- Assassin
    - Test assassin may cannot be bribe when the chance to fail is 1
    - Test assassin may can be bribe when the chance to fail is 0
    - Test assassin will still follow player within a certain range even if the player is invisible

Item:

- Collectable
    - Check EntityResponse
    - Check collection
- InventoryItem
    - Check ItemResponse
- InventoryItem > Sword
    - Check end of duration
- InventoryItem > Bomb
    - Check using (explode)
    - Check using (place without explode)
- Potion
    - Use two potion at the same time, make sure the effect work like a queue

Static Entity:

- PlacedBomb
    - Check EntityResponse
- Boulder
    - Check EntityResponse
    - Check the position of boulder
    - Check if boulder moved when player push
    - Check if boulder moved when player push and there is another boulder in the way.
- Door
    - Check EntityResponse
    - Check the position of door
    - Check if door can be opened with a correct key
    - Check if door can not be opened with a wrong key.
- Exit
    - Check EntityResponse
    - Check the position of exit
- FloorSwitch
    - Check EntityResponse
    - Check the position of FloorSwitch
    - Check floorSwitch turned on when boulder's on it.
    - Check floorSwitch turned off when boulder's not on it.

- Portal
    - Check EntityResponse
    - Check the position of Portal
    - Check the player can teleport to the portal.
    - Check the player can’t teleport to the portal if portal is surrounded by walls.
- Exit
    - Check EntityResponse
    - Check the position of Wall
- Zombie spawner
    - Check EntityResponse
    - Check the position of spawner
    - Check if zombie can be spawned in the given tick
    - Check if zombie can not be spawned when it’s surrounded by walls.
- Swamp_tile
    - Test movement factor on all type of moveing entity

Goals:

basicGoal:

- enemyGoal
    - Check if the goal achieved(removed from the goal string) if all spawners are destroyed and has killed certain number of enemies.
- BoulderGoal
    - Check if the goal achieved(removed from the goal string) if all boulders are on the floorswitches.
- treasureGoal
    - Check if the goal achieved(removed from the goal string) if the player has picked up a certain amount of treasure.
- ExitGoal
    - Check if the goal achieved(removed from the goal string) if player got to the exit.

ComplexGoal:

- andGoal
    - Check if the goal achieved(removed from the goal string) if both the subgoals are achieved

- orGoal
    - Check if the goal achieved(removed from the goal string) if one of the subgoals is achieved.

Buildable Entities:

- Have unit tests testing the getters/setters of each Buildable Entity (Bow, Shield, Sceptre, Midnight Armour)
- Use Integration Tests to make sure that crafting is working properly for each Buildable Entity
- Use Integration Tests to test that the mind control function of the sceptre works as intended on both the Mercenary and Assassin Moving Entities

Time Travel:

- Trigger exception when tick is less than 0 for rewind function
- Test one tick back
- Test 5 tick back
- Test portal30tick back
- Test portal less than 30tick back to origin
- Test older_player exist when time travel
- Test older_player move in the pattern before time travel

System tests

- Player walk, mercenary approach, player collect invincible potion, player consume the potion, mercenary run away from player.
- Player walk, mercenary approach, player collect invisible potion, player consume the potion, mercenary walk randomly
- Player walk, mercenary approach, player collect invisible potion, player consume the potion, no battle occur when there is a collision between them.
- Player walk, assassin approach, player collect invisible potion, player consume the potion, assassin still approach the player
- Check the AND logic for all the logic entities(logic switch, logic bomb, logic light bulb)
- Wires can be used to test logic tests
- Test persistence by storing all the class info into a file, then check if the file exist

- Test persistence by loading the file back to the project, to see if all the entities remain in the map,.
- Test persistence by loading the file back to project, to see if ticks can work.