Recommendation #1

What problems does this solve?

A Player can harvest a Dirt with grass or Tree only if the Player is standing on the relevant Ground object. Overriding the allowableActions method (from the Ground class) in the Dirt and Tree classes was therefore not an option; this is because a Player would be allowed to harvest a Dirt with Grass or Tree if they were adjacent to the relevant Ground object.
Our current implementation works around this issue. However, it introduces a **dependency from Player to HarvestAction** (which would otherwise not exist if we used the aforementioned implementation instead). Furthermore, though the playTurn method in Player is currently “manageable” in terms of size and complexity, its design is **not particularly extensible**. If we wanted to extend the system with another Action specific to the Ground object a Player is standing on (e.g. teleportation pad), then we would need to account for this in the Player’s playTurn method (as we did for harvesting). Eventually, adding more such actions would **cause the Player’s playTurn method to become bloated**.

Proposed design to address the issue

We recommend that the following changes be adopted:

- In Ground, rename the existing method allowableActions to allowableActionsIfAdjacent
    - Used in processActorTurn method in World, for objects of Ground child classes that can only be acted upon if the Player is adjacent to it
- In Ground, add a new method called allowableActionsIfOn
    - Implemented in the exact same way as allowableActionsIf
    - Used in processActorTurn method in World, for objects of Ground child classes that can only be acted upon if the Player is standing on top of it
- Add another line in processActorTurn which calls allowableActionsIfOn on the Ground object linked to the Player’s current Location

Advantages and disadvantages of proposed design

Advantages

- Removes dependencies: Player class does not need to be aware of any specific Action relating to the Ground the Player is currently standing on

- Increases modularity of the system: adding another Action relevant to a Ground child class object (e.g. Teleporter) the Player is standing on will simply involve creating a new class for the Action and overriding allowableActionsIfOn in the relevant Ground child class
- Implementation of actions that the Player performs on his/her surroundings becomes standardised (e.g. consistent with how Player attacking a Dinosaur is implemented)

Disadvantages

- Introduces another method into the Ground class (which is similar to an existing method) - a game designer using this engine needs to be aware of these differences

New functionality available

This change will provide more flexibility to future game designers in terms of how the Player can interact with the object of a Ground child class (based on the Player's position relative to the object).

Recommendation #2

What problems does this solve?

At the moment, the game engine does not provide a simple solution for one Actor to get the Location of another Actor. The Actor would need access to the GameMap, which is passed as a parameter in the Actor's playTurn() method. Currently, we are using the Scan class which checks locations within three spaces using for loops. However, this is very inefficient because it runs in O(N^2) time. If we wanted to extend the "line of sight" of an Actor, then the time complexity would increase further.

This is a problem because an **unnecessary dependency** is needed to be created through the playTurn() method when passed as an argument. Another issue with this is that other methods would not be able to access the GameMap as easily as playTurn().

Proposed design to address the issue

We recommend creating a HashMap attribute in Actor which contains each Actor's Location in real-time. This allows the child classes of Actor to have access to the Location of every other Actor throughout execution of the game.

Advantages and disadvantages of proposed design

Advantages

- Reduces dependencies in the playTurn() method for having to pass the GameMap as an argument
- Other methods attempting to access the GameMap will be less complicated

Disadvantages

- Trying to keep track of every Actor in real-time may be complex

New functionality available

The only effect with this proposed design is improved efficiency when searching for an Actor.