**Recommendation for extensions to the game engine**

In my opinion, there is nothing needed to be changed as:

Don’t Repeat Yourself(DRY) principle is followed in the game engine. For example, in GameMap class, the at(int x, int y) method is used in add method and tick method without duplication of the same code.

Beside that, the PickUpItemAction, MoveActorAction, DropItemAction, DoNothingAction which is extended to Action class have followed the Liskov Substitution Principle(LSP) as in those classes, execute method and menuDescription method are overridden from the Action class as we could use each of them in different condition as derived classes should be able to substitute the Action classes without the behavior of the code changing.

Reduce dependencies(ReD) principle is followed in the game engine as there is not creation of instance of other classes in a class via the new operator like for example, in Exit class, there is no line of code like GameMap gamemap = new Gamemap(xxx);. This allows us to test the class in isolation

The WeaponItem class is extended to Item class and is separated from the other items which are used to increase hitpoints of the actor and this follows the Single Responsibility Principle (SRP) where each class only has one job.

**Recommendation**

I would suggest that more actions could be added and the actions will be shown at the application in a form of menu like we could choose to move in a direction every turn. This actually solves the problem as we cannot straight execute the DropItemAction. For example of new actions which are recommended, we could implement ridingAction for the player by extending the ridingAction class to Action class and we could use the method inherited from the Action class in ridingAction and this is followed to the Don’t Repeat Yourself principle. However for this new functionality, the action the users choose to execute might return null as for example, when the users choose to execute PickUpItemAction, there might be nothing to let users to pick up and it is the disadvantages of the proposed change.

The problem I found is if there is enough food for the dinosaurs, they would survive easily and their populations would be a great number after many turns. Therefore, I suggest that age could be implemented on the Actor class as protected attributes like hitPoints and we could use the playTurn method which overrides the Actor class in each type of dinosaur and every turn, the age will increase by 1 and if function is used to set the maximum age that that type of dinosaur will die naturally like if (this.age==80).

Other than these, we would suggest making different maps like some maps would reduce the actor hit points faster or reduce their water points faster. For example, a snowy as well as cold map could be added and heat points could be implemented as attributes onto the actor and if their heat points are below a certain number, the actor will become unconscious.