Recommendations for change to the game engine

Problem 1: The actions that can be taken an actor every turn is determined in world

Every turn, World's processActorTurn will iterate through items in the actor's inventory, the grounds and actors adjacent to the actor, and the items on the actor's current location to collect the allowable actions that can be taken by the actor. The collection of actions is put into an Actions object and it is passed to the actor playTurn(), the actor will then decide what action to take. Looking back into our game, we've realised that there's no need to collect the list of actions for every single actor because the only actor that requires the parameter actions is the Player whereas other actors will simply ignore it and return their own action according to their own behaviours. In the current game engine, the processActorTurn will generate an Actions for every actor regardless if they need it or not, this subsequently increases unnecessary runtime especially when there's a lot of actors in the game.

Solution: Place the functionality of processActorTurn in Actor class

It is common for a game to have different characters that behave differently and be able to do different things. Therefore, instead of deciding the actions of the actor in World, it makes more sense to put the functionality in the playTurn() method of Actor class. Moreover, by giving the actor the responsibility to check for their own actions makes the code easier to be understood as the readers can just look up to the playTurn() and understand how the actor will behave and does not need to trace the actions parameter in playTurn() all the way to World. Lastly and most importantly, this allows actors such as zombie and human to override the playTurn() according to their own needs and functionality, hence, no redundant code has to be run.

Problem 2: GameMap's locationOf() returns the location of the actor even though the player is not on the map

The method locationOf() in GameMap will return the location of the actor despite the absence of that actor in that particular map that this method is called on. For example, when town.locationOf(player).map().contains(player) is called, it will return true even if the player is not in the town map. This creates confusion as we would expect this method to raise an exception or return null when we try to get the location of an actor that is not present on the map. Besides, removeActor(), moveActor() and isAnActorAt() also faces this similar issue, meaning that the instance of GameMap is disregarded when using these methods, and this will create the same problem as stated above.

Solution: Declare the methods and actorLocations as static to avoid confusion

The reason why those methods can work even regardless of the GameMap the actor is on is because all GameMap shares the same actorLocations and this actorLocations does not differentiate between two actors from different maps. This implies that the methods above do not act on one particular instance, by declaring those methods as static methods, we can invoke the method on GameMap instead of its instance to avoid confusion.

Since static methods can only access static variables, the attribute actorLocations has to be modified to static. By having both static methods and static variables, we can avoid confusion and the reader would know that there is only one instance of ActorLocations and it doesn't matter which GameMap the actor is on to invoke those methods. However, the downside of doing so is that the variable cannot be overridden. This means when a subclass of GameMap is created, those implementations of static methods cannot be modified and changed according to needs.