Design Rationale REQ4

Implementation & Why:

A new BOSS_MAP has been added to the maps class. Maps class stores all the maps of the game to easily keep track and edit the game maps. It also keeps the application code clean and maintainable.

A new Boss Actor has been introduced to the game. It spawns at the BOSS_MAP

The two new weapons that has been introduced to the game extends from the TradeableWeaponItem class as they are weapon items that can be traded. By inheriting a parent class, it would have all the attributes of all trade-able weapon items to reduce code redundancy when many different weapon items that can be traded are created to adhere to DRY (Do Not Repeat Yourself) principle. Every Tradeable weapon item would have setprice method that is able to set a new selling price depending on which actor's inventory currently holds it.

StabStepAction is an Action of a GreatKnife that has a parameter of the instance of the GreatKnife that implements the action, and the Actor the action 'steps' around. A step method is implemented to allow extendibility to the game where a step Location around a target actor is returned once called.

GreatSlamAction has similar implementation logic to StabStepAction but it is an action of a Giant Hammer. The slam() method loop through all the exits of the otherActor (in this case is the enemy) and deals half of the originalDamage to all the actors surrounding it.

A new attack() method has been added to the AttackAction class that allows the usage of the atack logic previously implemented in the execute method. This modification allows other actions in the game, such as StabStepAction and GreatSlamAction, to efficiently leverage the attack functionality provided by the AttackAction. This design enhancement adheres to the DRY (Don't Repeat Yourself) principle by enabling these actions to utilise the attack logic without the need for redundant implementation in every weapon item's execute() method.

The implementation where the player is prompted to purchase the GreatKnife before leaving the Building is by adding a new Great Knife instance to the inventory of the Traveller so that the any Playable actor can purchase the Great Knife from the Traveller. In order to not make the Giant Hammer purchasable, the Giant Hammer is not added to the inventory of the Traveller, making it only available to be picked up from the floor.

Pros:

The addition of the attack() method in the AttackAction class enhances code reusability which adheres to DRY principle. This enables actions like StabStepAction and GreatSlamAction to efficiently leverage the attack logic without redundant implementations.

Cons:

Depending on the frequency of actions like GreatSlamAction, which affects multiple actors, performance considerations may arise. Iterating through all exits of an actor and applying damage to surrounding actors could be resource-intensive.

Extendability:

The slam() method in GreatSlamAction is coded as a class method that is called in execute method instead of directly coding into the execute method as this allows for future extendability where there might be a new weapon action that has the AOE slam functionality as well. The same cane be said for the step action in StabStepAction, other actions can access the step action in future implementations so that the code logic does not have to be repeated (DRY principle).

A TradeableWeaponItem and TradeableItem classes were created to unify all the weaponItems and Items that can be traded to easily get and set the price through getter and setter methods. An interface implements the characteristics of a tradeable Item/WeaponItem which shows the use of Interface Segregation method as the interface is implemented in the wrapper class for all tradeable Item/WeaponItem.