DESIGN RATIONALE

Task 1: Estus Flask

- In order to display the current health, the approach would be to use an Estus flask class. The player class would have an attribute of type Estus flask and this would represent the flask itself. The flask would be inheriting an item other than having the property of being portable as we cant drop or pick this item up. In terms of consuming the item, this action will be appended to the Arraylist which is in the Action class. This array list will get printed out on the menu and of the user was to select the hotkey, it would perform the action by increase user's HP.

Task 2: Bonfire

- In order to create the bonfire, the creation of a class which inherits Ground was needed. As the bonfire needs to be interacted with, a new RestAction was created where if the actor has the capability *REST* they are able to do so. This RestAction would then run the ResetManager class so that all instances of Resettable are resetted. After running the ResetManager class it would also run cleanUp() to remove all flagged Resettables to be deleted.

Task 3

- The soul is the currency if the game and the there will be a class called playerSoul which will hold all the methods regarding the operations to be performed relating to souls. The player class will have an attribute of the playerSoul and this is where the players money will be kept. This class will also have methods that rewards the player upon killing an enemy depending on the enemy type. The class will implement the soul Interfave so we can access methods like transferSoul.

Task 4: Enemies

- For enemies, use an abstract class which all enemies inherit from. This reduces repeat code as enemies will share many behaviours and opens the possibility of easily creating similar enemies.
- All actions of enemies will be dictated by behaviours that implement a common behaviour interface. When it comes time to get action for enemies it will loop through a behaviour list in which has a hierarchy based on importance eg. AttackBehaviour > WanderBehaviour
- The skeletons resurrect feature will be within the abilities enum, and handled within isConscious method of the skeleton class.

Task 5: Terrains (Vallery and Cemetery)

- Valleys will be changed to allow players to step on them as the current version does not allow any Actor to step on them. MovePlayerAction is a new class which inherits MoveActorAction that detects if a player’s move action will land on a valley. If this action is executed, the player will die.
- Cemetery will be a new class which inherits the Actor class and utilises the SpawnAction class for a chance to spawn an Undead. The reason for inheriting the Actor class is because every turn, the cemetery performs an action.

Task 6: Soft reset/Dying in the game

- The AttackAction will be modified to ensure the Player class is not removed when the healthPoints drop below zero. Instead, when the player dies ResetManager will run and clean up. Following this, a TokenOfSouls instance will be created. As ResetManager will perform its tasks first, any previous TokenOfSouls instances will be deleted before a new one is placed.
- TokenOfSouls will be a class that inherits PortableItem because the TokenOfSouls needs to be picked up by the player. TokenOfSouls will also implement Souls as when the player dies, all of their souls must be transferred to something that implements Souls.
- The new MovePlayerAction class will contain the current location and location where the player will move next. If the next location is a valley, when the action is executed the player will die and the reset will run as if the player died by an attack except the TokenOfSouls will be placed at the current location.

Task 7: Weapons

- Each weapon will have its own class. This way several instances of one particular weapon can exist, especially important considering that there may be several instances of broadsword and great axe with many different skeletons existing on the map.
- Active weapon abilities will also have their own class, as weapon abilities are varying and too broad to define within a single class.
- Passive weapon abilities will be created through the enum abilities. On attackAction.execute weapon abilities will be checked and effects will be designated appropriately

Task 8

- The vendor will be the trading spot in the game called fire keeper. This location will be a ground type. Further Modifications will be made to this spot to make sure no enemies or player can step on the locaty itself, very much like valley. Upon coming in range with the store, the player will be able to see a menu of the items that they can purchase. There will be a class for this store called fire keeper. This will handle all transactions as it will implement the soul Interface. This class will have attributes of the weapon class representing the items that are sold in the store. The weapons will be transferred to the player using the swapWeaponAction class and it's methods. The class will access the person's attributes when the user buys an upgrade on max HP.