# Assignment 3 Design Rationale: CL_AppliedSession10_Group1

## Requirement 2 – The Static Factory; Sellable & Humanoid figure

This design rationale considers implementation of the humanoid figure. This Actor can purchase certain items that are considered as 'sellable' from the player. Core logic is abstracted into an NPC base class, representing characters with no pre-defined behaviours or attacking behaviour. This abstraction allows for easy extension of new NPC characters with unique characteristics and behaviours. The Sellable interface is implemented by items that can be sold in the game. It defines methods like SellFrom(Actor actor), allowableActions(Actor otherActor, Location location), and getSellValue(). These methods respectively handle the selling process, determining the actions that can be performed with the item, and getting the sell value of the item. The VENDOR and SELLABLE enums are capabilities that actors and items can have. An actor with the VENDOR capability can buy items, and an item with the SELLABLE capability can be sold.

By abstracting the humanoid figure and vendor logic into an abstract base class 'NPC', this approach considers the singular responsibility principle. That way the 'Humanoid Figure' class has specified methods unique to that form of NPC within its class. The open closed principle has been considered as by using a sellable interface without default methods we are allowing for extension without modification. That is each new item with sellable functionality will depend on when it is implemented, with no factorisation necessary for it to become workable. This in turn will retain code reusability and robustness. Another principle evidenced in this requirement is Liskovs substitution principle. That is anywhere Sellable assumes Item, it may also assume WeaponItem, Fruit, or Scrap. One example may be Item and PotofGold are both Sellable. Anywhere a sellable item is expected, PotofGold or item can be used as the both implement these sellable interfaces (where sellable).

Our approach of using a sellable interface, and having it interact with the humanoid figure adheres to code futureproofing and dependency inversion. The code follows the dependency inversion principle by depending on abstractions instead of concrete implementations. In relation to the sellable interface, this means that The SellAction depends on the Sellable interface, not the specific item that is declared as sellable. This means that the SellAction can work with any class that implements sellable. This will improve scalability and futureproof our code for further considerations and developments.

An alternative approach may consider abstracting the consumption into an abstract class as the logic for consuming an item along with the logic for selling an item depends on the actors around them and if they have some certain condition as True. This would be rewarding as the current implementation may be more adverse to Open Closed Principle violation as each new item must implement functionality. Moreover, an alternative approach may use the default method such that some item that may be sold can sell just by calling the remove item and add sellvalue logic. However, it may also be argued that the default method of an interface defeats the whole purpose of implementing an interface!

Another alternative approach may have considered extending HumanoidFigure from ground. This is the obvious implementation idea, however intuitive is not necessarily the most applicable. This is for a few reasons. Firstly, we cannot step on the entity, which is default for actors, but not for ground. In future game developments we can have travelling sellers, or kill sellers to rob their items, which is easily implemented by adding behaviours/capability to the NPCS, mainly Wander Behaviour, HostileToPlayer Capability.