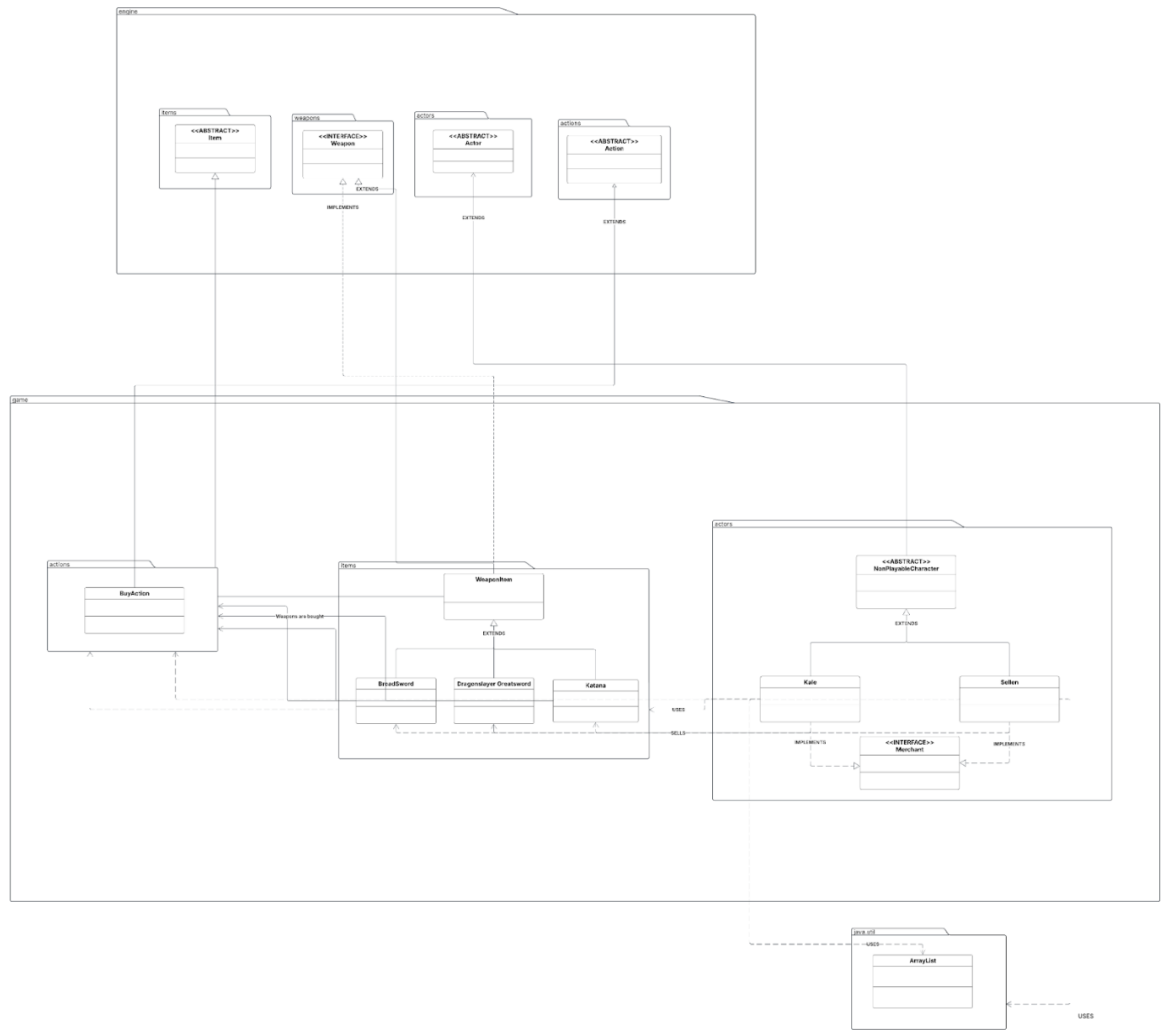
Req 4 UML Diagram



Design 1: Specific logic for Kale and Sellen using Merchant interface

Both Kale and Sellen implement the Merchant interface, which initialises the merchant specific logic for weapons bought by the Player. Each merchant overrides the getCost(), applyPurchaseEffects() and getName() methods from this interface.

PROS	CONS
Clear way to implement this task, each specific behaviour is separately handled for each merchant	Code is duplicated for each merchant

Merchant’s decisions are independent, can be helpful to trace for debugging	Adding new merchant and their specific effects might require code change
Adheres to the SRP	Violates OCP because not easily extensible

Design 2: Inheritance for Weapons that can be bought

Each weapon type extends WeaponItem class and implements a void function of applying the purchase effects. Because each weapon is sold by only 2 vendors, Hard coded the logic so that if Player buys from Kale, their max stamina increases and cost is different.

PROS	Cons
There's clear encapsulation of the effects for each weapon	Hard-coded values means that it requires code change if more Merchants or weapons are added
Able to test each weapon effect due to it being independent	Effect logic has the ability to be duplicated
Adheres to LSP	Code repetition vulnerability for merchant specific variations

Design 3: Centralising buying using BuyAction

Buy action class encapsulates the entire buying a weapon functionality. It checks the Player’s balance upon purchase, deducts balance if sufficient funds are available, adds the item to Player’s inventory. Merchants only provide the item and essentially start this process during interaction.

PROS	CONS

All of the logic is centralised and reusable	There's more indirection, meaning that the buying isn't visibly evident in kale and sellen classes
Kale and Sellen classes are kept organised and focused just on the purchase effects	
It also helps with testing and extending the code as this is the centralised class that is responsible for the handling	

Referring to the Final UML diagram for REQ-04, the Merchant interface is implemented by Kale and Sellen, showing the completion of Design 1. Both Kale and Sellen extend from the NonPlayableCharacter abstract class, highlighting polymorphism and the code adhering to the SOLID principles. Furthermore, the WeaponItem superclass serves as the base of the three different weapons introduced in this requirement, this demonstrates Inheritance even though there's evidence of code-reuse. Finally, the BuyAction class is seen to be tightly coupled with Kale and Sellen as well as the different weapons, to help coordinate the interactions between these entities and our Player.