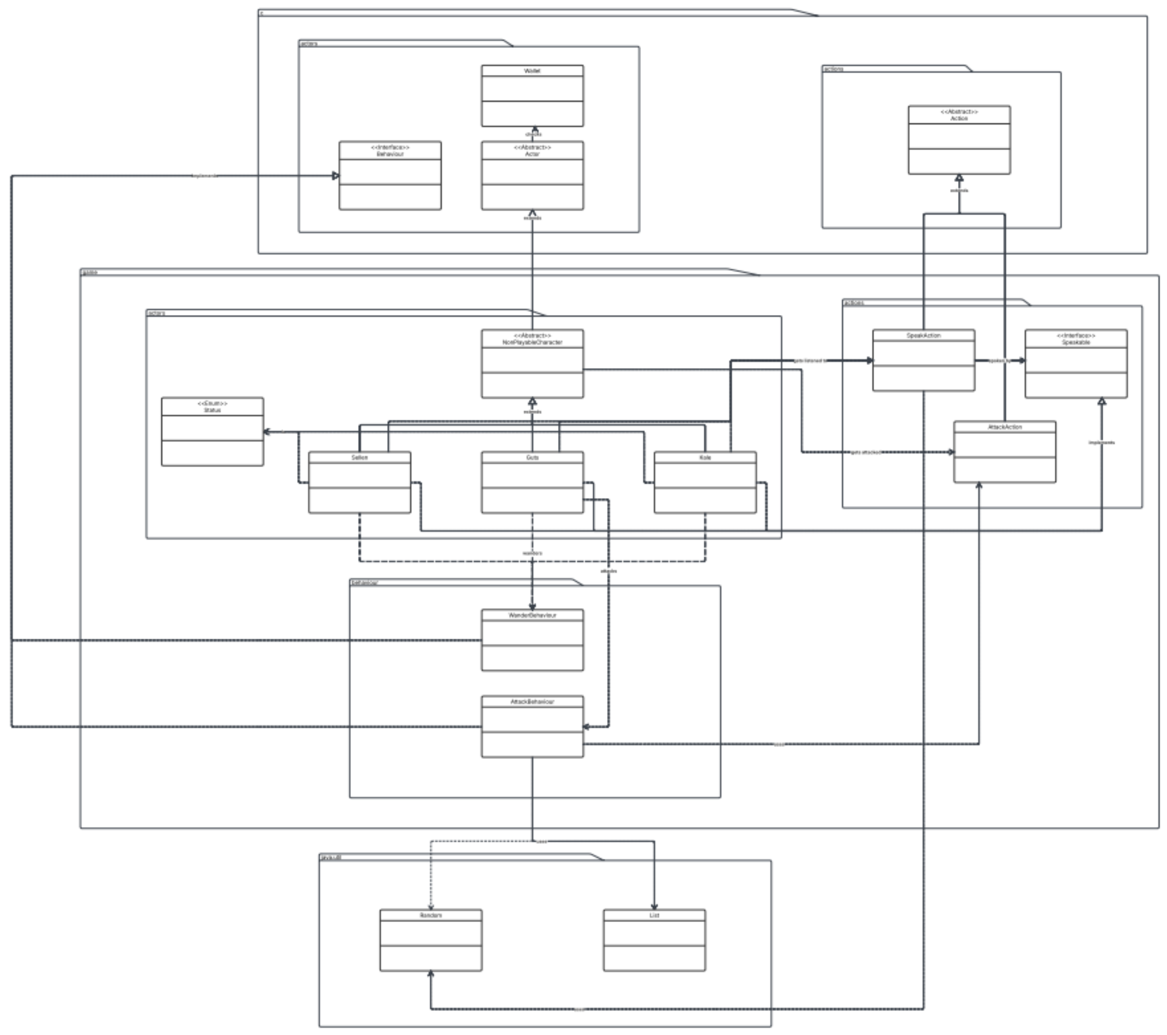
Req 3 UML Diagram



A: Creation of the new talking NPCs - Sellen, Kale and Guts

Design 1 (Chosen Option): NPC Inheritance and Speakable Interface for NPCs that speak

Advantage	Disadvantage
Open closed principle (OCP) - all speakable NPCs can guarantee support for monologue interactions. Classes relying on Speakable can use polymorphism.	Many new classes at first, makes the system seem quite complicated

Non-speaking NPCs like animals are not forced to implement unused method like getMonologues()	If there is shared dialogues between several npcs, you would need to repeat them for all those npcs implementing the interface
Can easily add in new NPS and choose to implement Speakable if necessary on that NPC child class without changing the existing code since different NPCs will speak differently or in a different way, depending on conditions.	

Design 2 (alternative - insufficient design): Instead of separating dialogue using a Speakable interface, all talking NPCS inherit from a single MonologueNPC base abstract class, which inherits from the NPC abstract class

Advantage	Disadvantage
Can manage shared dialogues/methods between all talking NPCS and have the child classes add on to them	Monologue logic is mixed into the core NPC behaviour for all character, even those that should not speak (example: animals and future cases npc)
One class to look at for dialogue behaviour	Hard to maintain. Extending an abstract class creates a tight coupling between the child classes and their superclass. Doing multiple layers of this, then making a change to the top most superclass could affect lower layers.
	Not easy to extend functionalities. MonologueNPC has dialogue, and (for example) MerchantNPC has wares to sell. What if an NPC can do both?

In the end, design A was chosen as using a separate interface on a NPC-inheriting class allows for better extensibility as each interface is only responsible for a specific purpose/function (SRP), and an entity that is a subclass of NPC (due to sharing common methods/variables) can add further functionality specific to them with each interface that have their own defined purpose (ISP).