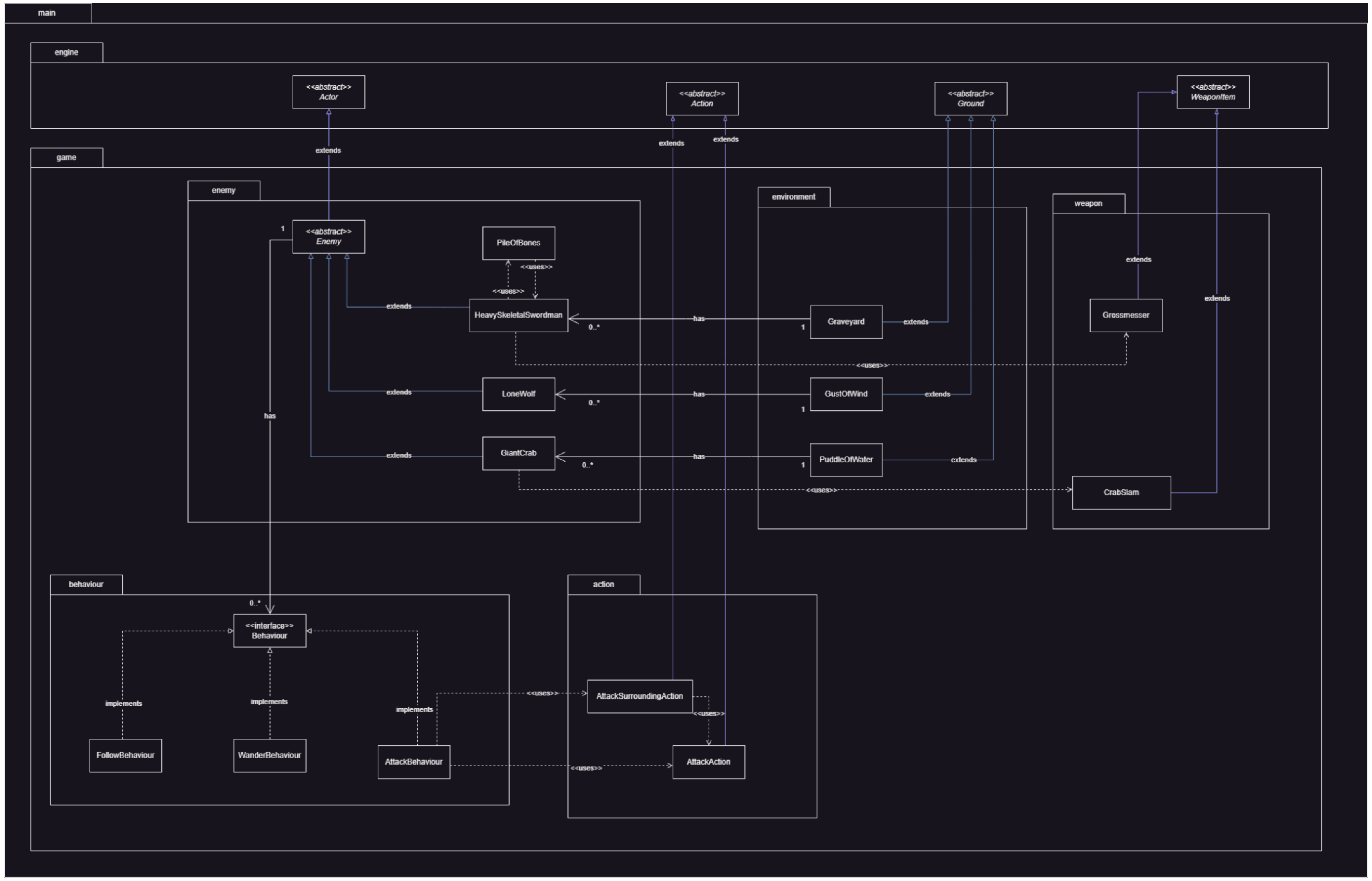
FIT 2099 group 3 assignment 1

Lee Sing Yuan
Loo Li Shen
Yeoh Ming Wei

REQ 1:UML



main
engine
<<abstract>> Actor
<<abstract>> Action
<<abstract>> Ground
<<abstract>> WeaponItem
game
enemy
<<abstract>> Enemy
PileOfBones
HeavySkeletalSwordman
LoneWolf
GiantCrab
environment
Graveyard
GustOfWind
PuddleOfWater
weapon
Grossmesser
CrabSlam
behaviour
<<interface>> Behaviour
FollowBehaviour
WanderBehaviour
AttackBehaviour
action
AttackSurroundingAction
AttackAction
extends
has
implements
<<uses>>
1
0..*

Rationale

The graveyard, puddle and wind are all similar to other ground types with the addition of spawning enemies. Therefore, it makes sense that we used a parent class (DRY). Then, the respective environments will be associated with their respective enemies(for example: Graveyard to Skeleton) . By using this method we have reduced the number of dependencies on the Graveyard class. When adding a new type of skeleton, we just make it inherit from the Skeleton class.(open close principle).
^
To improve the modularity and maintainability of the code, we have applied several principles in our design. Firstly, to avoid code duplication, we have created a parent class for the similar ground types, such as the graveyard, puddle, and wind (DRY). Additionally, we have implemented the open-close principle by making it easy to add new types of skeletons. For example, when adding a new type of skeleton, we only need to make it inherit from the Skeleton class.

Besides that, for the enemies with skills, it will be assumed that the enemies will have a weapon for that skill in their inventory. This means that the enemy class handles existence, the skill will be handled through an action class by checking if the weapon exists (single responsibility principle).
^
Furthermore, we have applied the single responsibility principle by having the enemy class handle only the existence of the enemy, while the skill,which is assumed that the enemies will have a weapon for that skill in their inventory, is handled through an action class by checking if the required weapon exists.

Moreover, for the skeleton skill, it will be assumed that it is another enemy. So when the skeleton dies, it creates this other enemy to replace it and when the count is completed,it creates a skeleton again. It is done this way, because for future enemies that have the same skill (not limited to skeleton) they just have to instantiate this class without touching any other already existing code. (open close principle)

The trader uses the interfaces to handle inventory and selling of weapons. This reduces the number of dependencies to the trader and if there are new weapons added in the future, they have to implement the interface without any changes done to the trader. (open close principle). For the weapons, they have many similar properties. Hence it inherits from an abstract class (DRY).

REQ 3 and runes REQ1 write it @ming wei runes for enemies also
For runes, they use classes to handle the dropping and collection of it. This allows enemies to drop but not collect runes and the player to perform both. This methods are

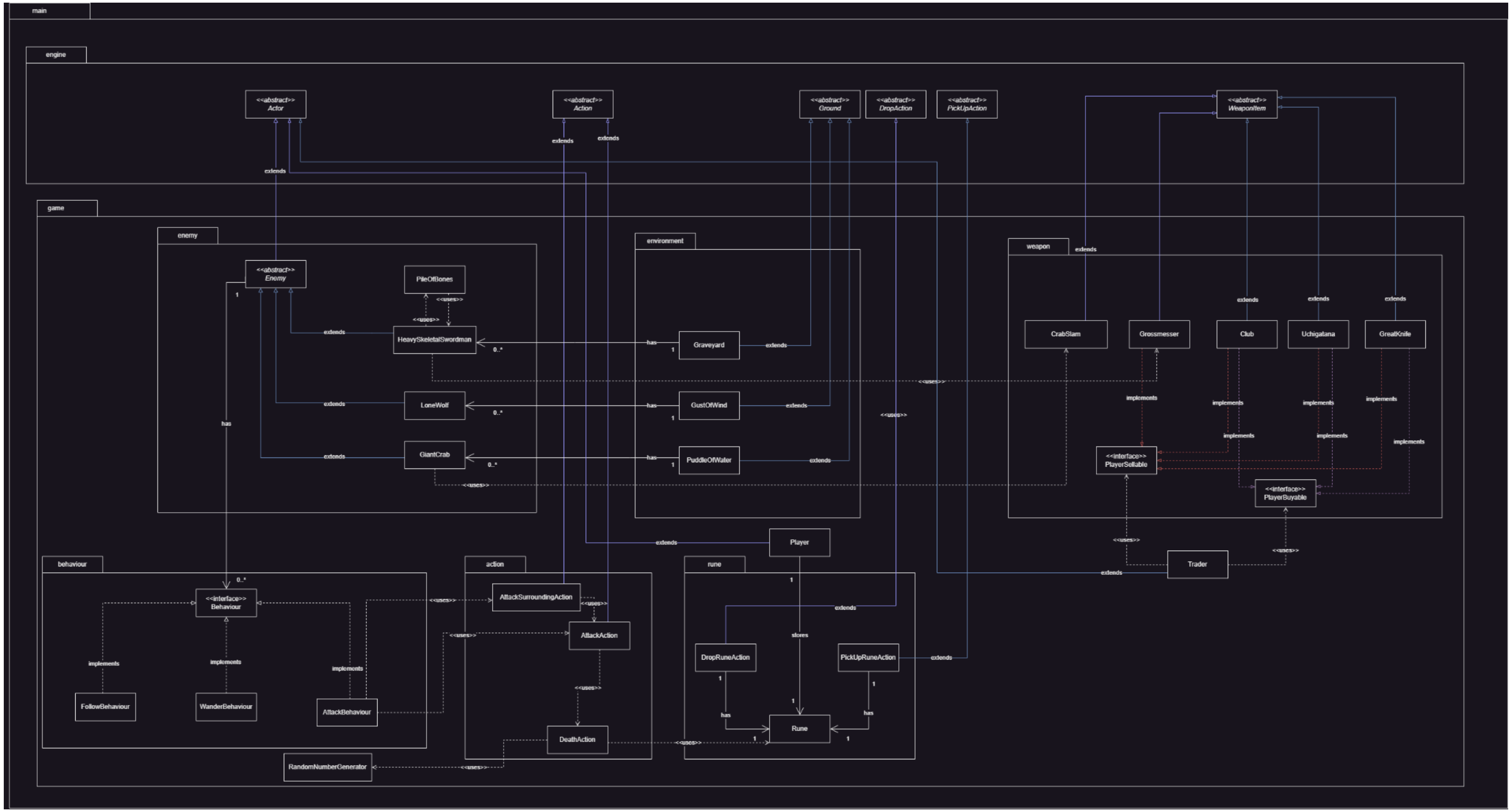
For the flask of crimson tears, it inherits a potion abstract class so that if in the future, any new potions are added, they just have to inherit from the potion class without changing any code (open close principle)

For the Site of Lost Grace, it inherits ground abstract class. The SiteOfLostGrace class should have only one responsibility, which is to manage the player's progress by resetting and allowing them to respawn at the last visited site. The "SiteOfLostGrace" class will call upon the "ResetManager" when the player dies or rests at a site of lost grace. The "ResetManager" class is responsible for managing the reset functionality of the game, while the "Enemy", "Player", and "PotionItem" classes implement the "resettable" interface, which has a single responsibility of providing the reset functionality. By separating the reset functionality into its own class and interface, the design follows the SRP, which helps to make the code more maintainable, testable, and flexible.

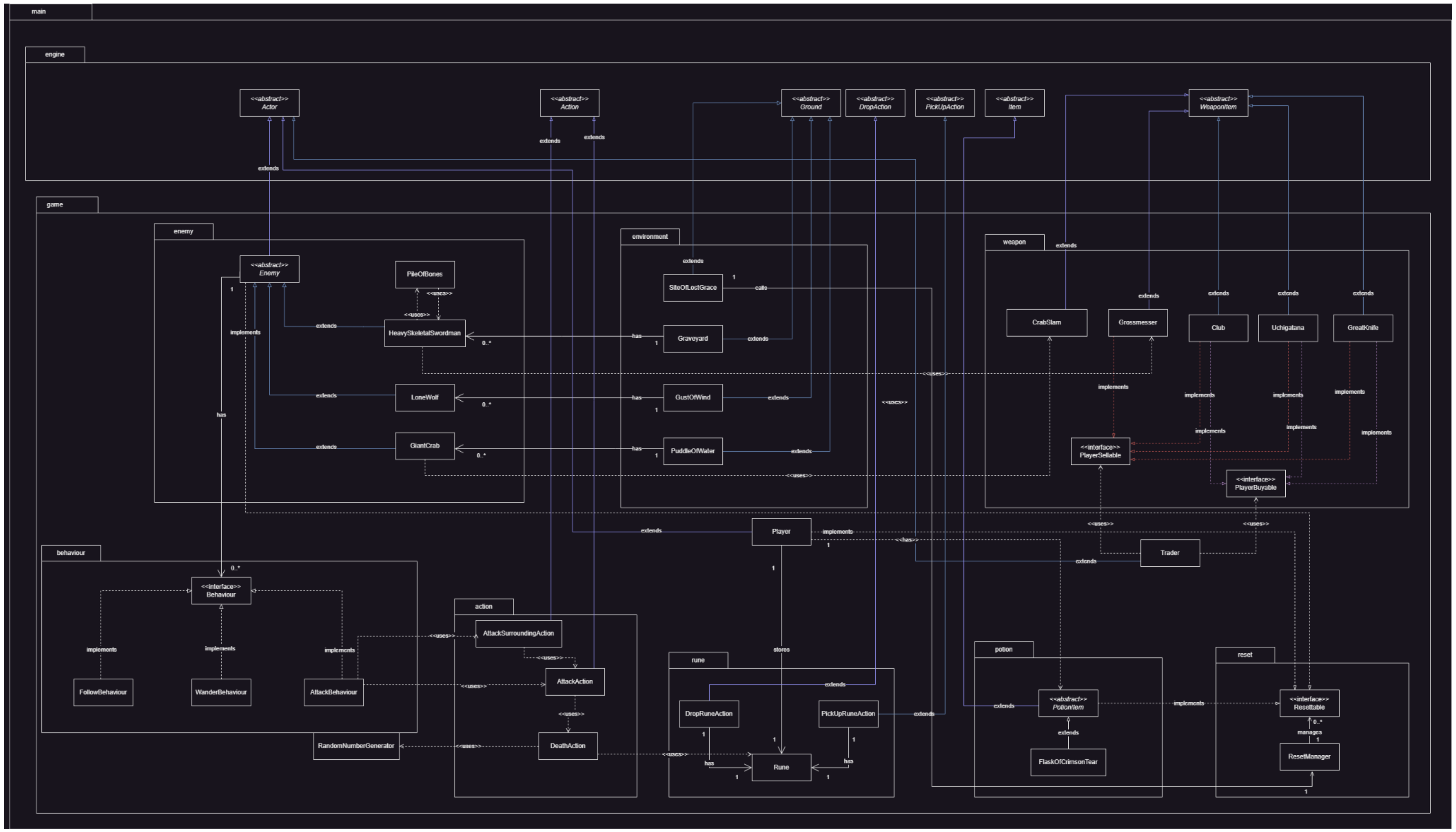
Upon resetting, the reset manager calls all who implemented the resettable interface. This allows addition of new resettable entities without modifying any existing code (open close principle).

The roles or classes for the player, is done by an abstract player class because of the similarities (DRY).

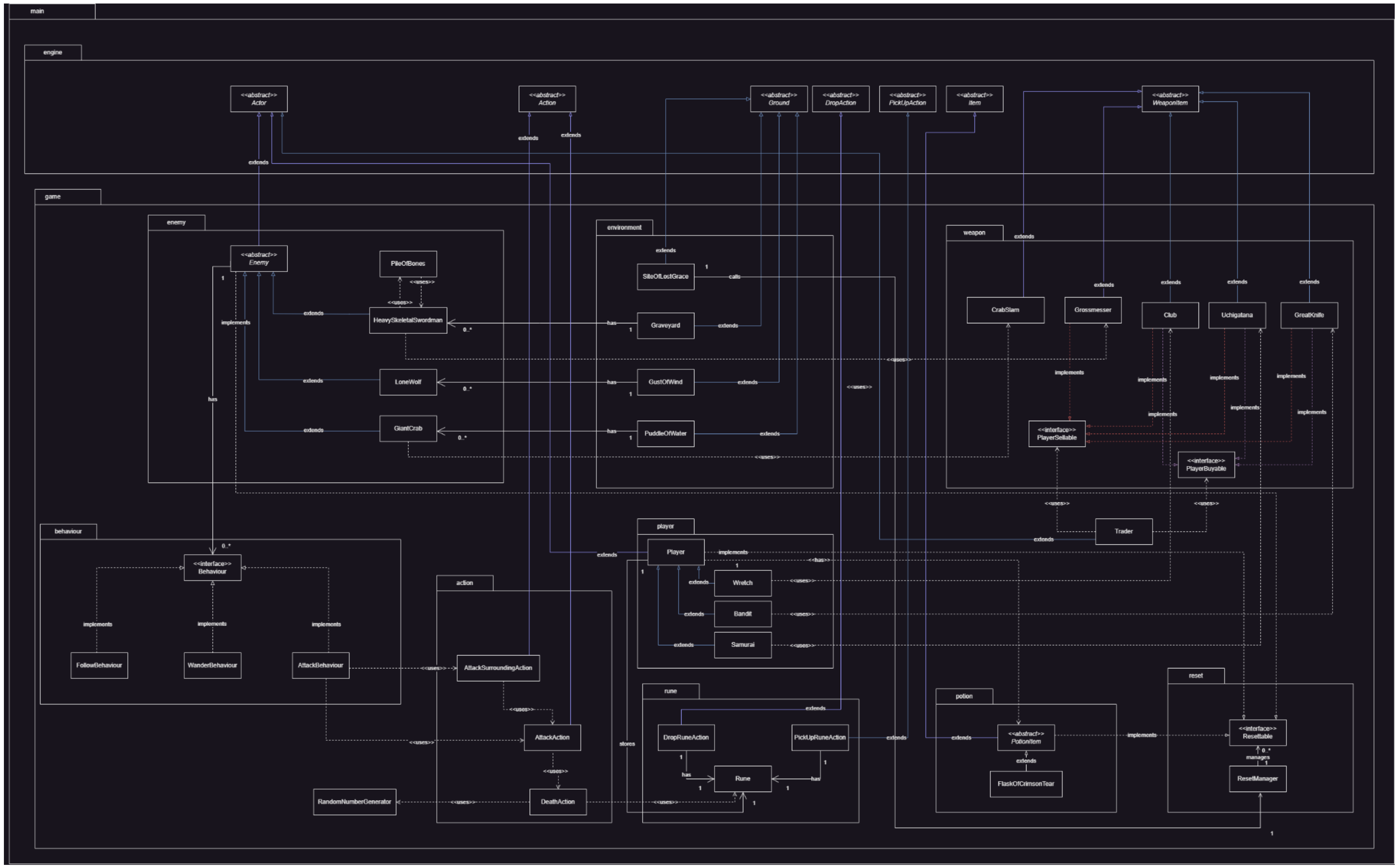
REQ 2: UML


main
engine
<<abstract>> Actor
<<abstract>> Action
<<abstract>> Ground
<<abstract>> DropAction
<<abstract>> PickUpAction
<<abstract>> WeaponItem
game
enemy
<<abstract>> Enemy
PileOfBones
HeavySkeletalSwordman
LoneWolf
GiantCrab
environment
Graveyard
GustOfWind
PuddleOfWater
weapon
CrabSlam
Grossmesser
Club
Uchigatana
GreatKnife
<<interface>> PlayerSellable
<<interface>> PlayerBuyable
Player
Trader
behaviour
<<interface>> Behaviour
FollowBehaviour
WanderBehaviour
AttackBehaviour
action
AttackSurroundingAction
AttackAction
DeathAction
rune
DropRuneAction
PickUpRuneAction
Rune
RandomNumberGenerator
extends
implements
has
stores
<<uses>>
0..*
1

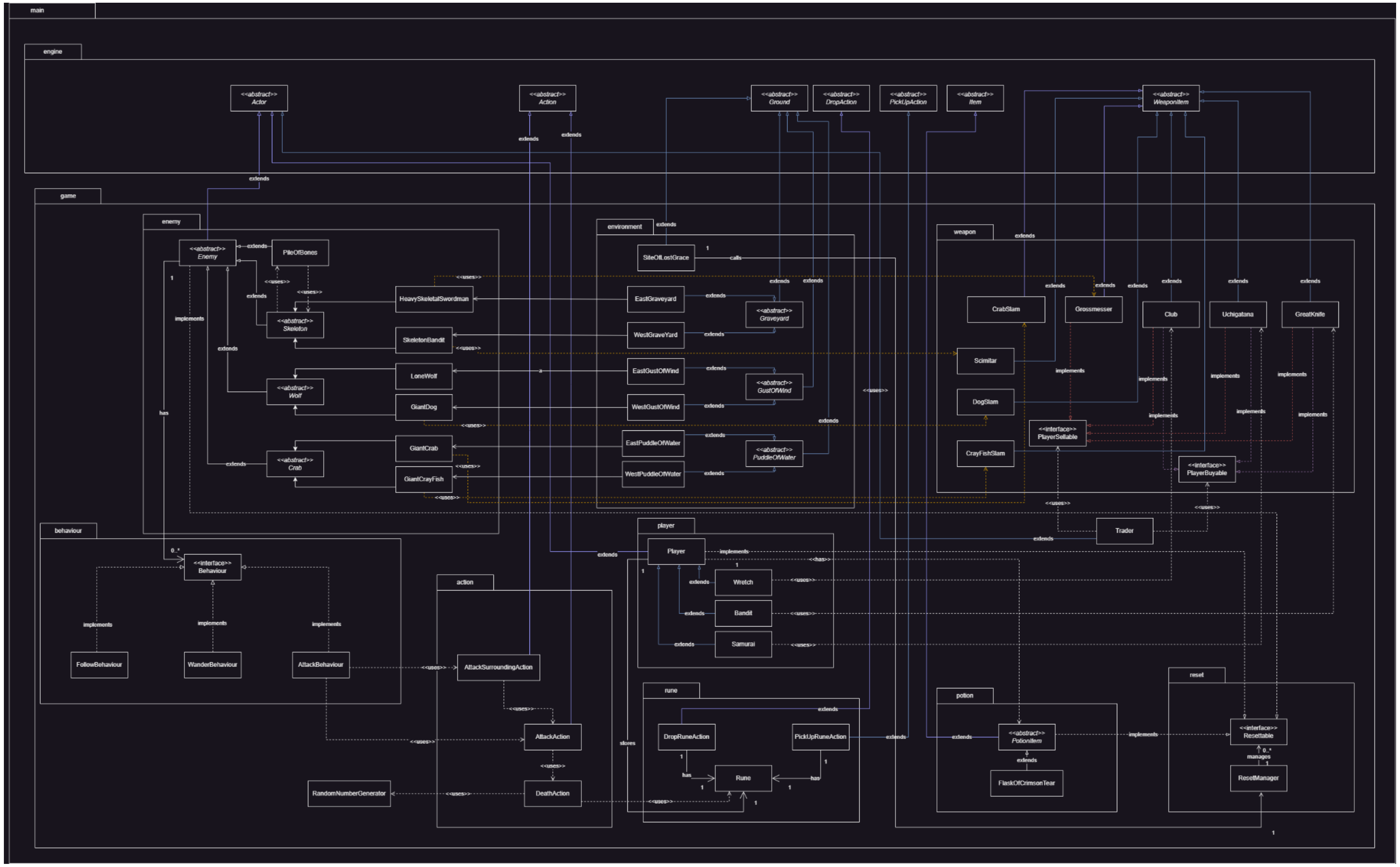
REQ 3: UML


main
engine
<<abstract>> Actor
<<abstract>> Action
<<abstract>> Ground
<<abstract>> DropAction
<<abstract>> PickUpAction
<<abstract>> Item
<<abstract>> WeaponItem
game
enemy
<<abstract>> Enemy
PileOfBones
HeavySkeletalSwordman
LoneWolf
GiantCrab
environment
SiteOfLostGrace
Graveyard
GustOfWind
PuddleOfWater
weapon
CrabSlam
Grossmesser
Club
Uchigatana
GreatKnife
<<interface>> PlayerSellable
<<interface>> PlayerBuyable
Player
Trader
behaviour
<<interface>> Behaviour
FollowBehaviour
WanderBehaviour
AttackBehaviour
RandomNumberGenerator
action
AttackSurroundingAction
AttackAction
DeathAction
rune
DropRuneAction
PickUpRuneAction
Rune
potion
<<abstract>> PotionItem
FlaskOfCrimsonTear
reset
<<interface>> Resettable
ResetManager
extends
implements
has
<<uses>>
<<has>>
calls
stores
manages
1
0..*

REQ 4: UML


main
engine
<<abstract>> Actor
<<abstract>> Action
<<abstract>> Ground
<<abstract>> DropAction
<<abstract>> PickUpAction
<<abstract>> Item
<<abstract>> WeaponItem
game
enemy
<<abstract>> Enemy
PileOfBones
HeavySkeletalSwordman
LoneWolf
GiantCrab
environment
SiteOfLostGrace
Graveyard
GustOfWind
PuddleOfWater
weapon
CrabSlam
Grossmesser
Club
Uchigatana
GreatKnife
<<interface>> PlayerSellable
<<interface>> PlayerBuyable
Trader
behaviour
<<interface>> Behaviour
FollowBehaviour
WanderBehaviour
AttackBehaviour
action
AttackSurroundingAction
AttackAction
DeathAction
RandomNumberGenerator
player
Player
Wretch
Bandit
Samurai
rune
DropRuneAction
PickUpRuneAction
Rune
potion
<<abstract>> PotionItem
FlaskOfCrimsonTear
reset
<<interface>> Resettable
ResetManager
extends
implements
<<uses>>
<<has>>
has
calls
stores
manages
1
0..*

REQ 5: UML


main
engine
<<abstract>> Actor
<<abstract>> Action
<<abstract>> Ground
<<abstract>> DropAction
<<abstract>> PickUpAction
<<abstract>> Item
<<abstract>> WeaponItem
game
enemy
<<abstract>> Enemy
PileOfBones
<<abstract>> Skeleton
HeavySkeletalSwordsman
SkeletonBandit
<<abstract>> Wolf
LoneWolf
GiantDog
<<abstract>> Crab
GiantCrab
GiantCrayFish
environment
SiteOfLostGrace
EastGraveyard
WestGraveYard
<<abstract>> Graveyard
EastGustOfWind
WestGustOfWind
<<abstract>> GustOfWind
EastPuddleOfWater
WestPuddleOfWater
<<abstract>> PuddleOfWater
weapon
CrabSlam
Grossmesser
Club
Uchigatana
GreatKnife
Scimitar
DogSlam
CrayFishSlam
<<interface>> PlayerSellable
<<interface>> PlayerBuyable
Trader
behaviour
<<interface>> Behaviour
FollowBehaviour
WanderBehaviour
AttackBehaviour
action
AttackSurroundingAction
AttackAction
DeathAction
RandomNumberGenerator
player
Player
Wretch
Bandit
Samurai
rune
DropRuneAction
PickUpRuneAction
Rune
potion
<<abstract>> PotionItem
FlaskOfCrimsonTear
reset
<<interface>> Resettable
ResetManager
extends
implements
<<uses>>
<<has>>
has
calls
stores
manages
1
0..*