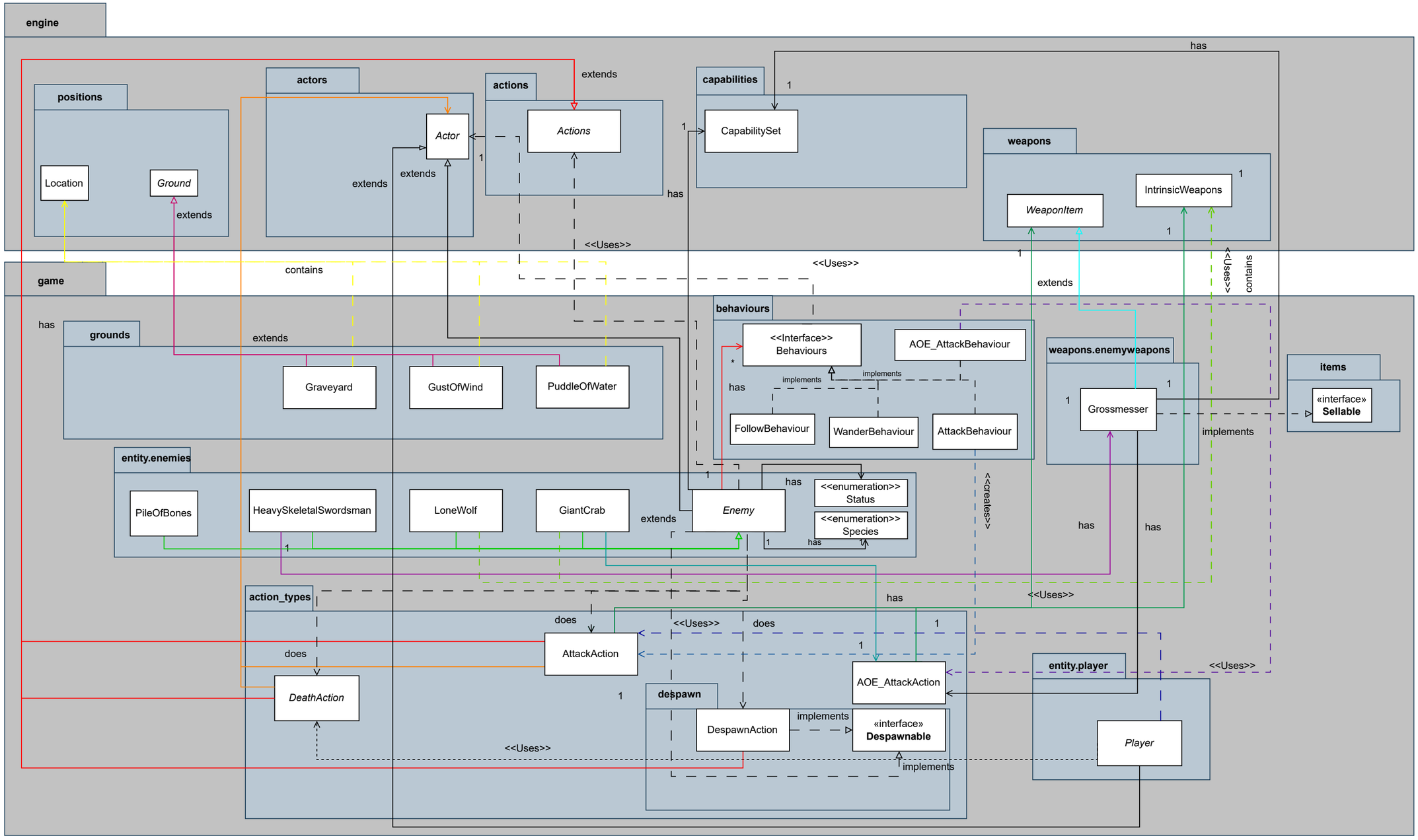
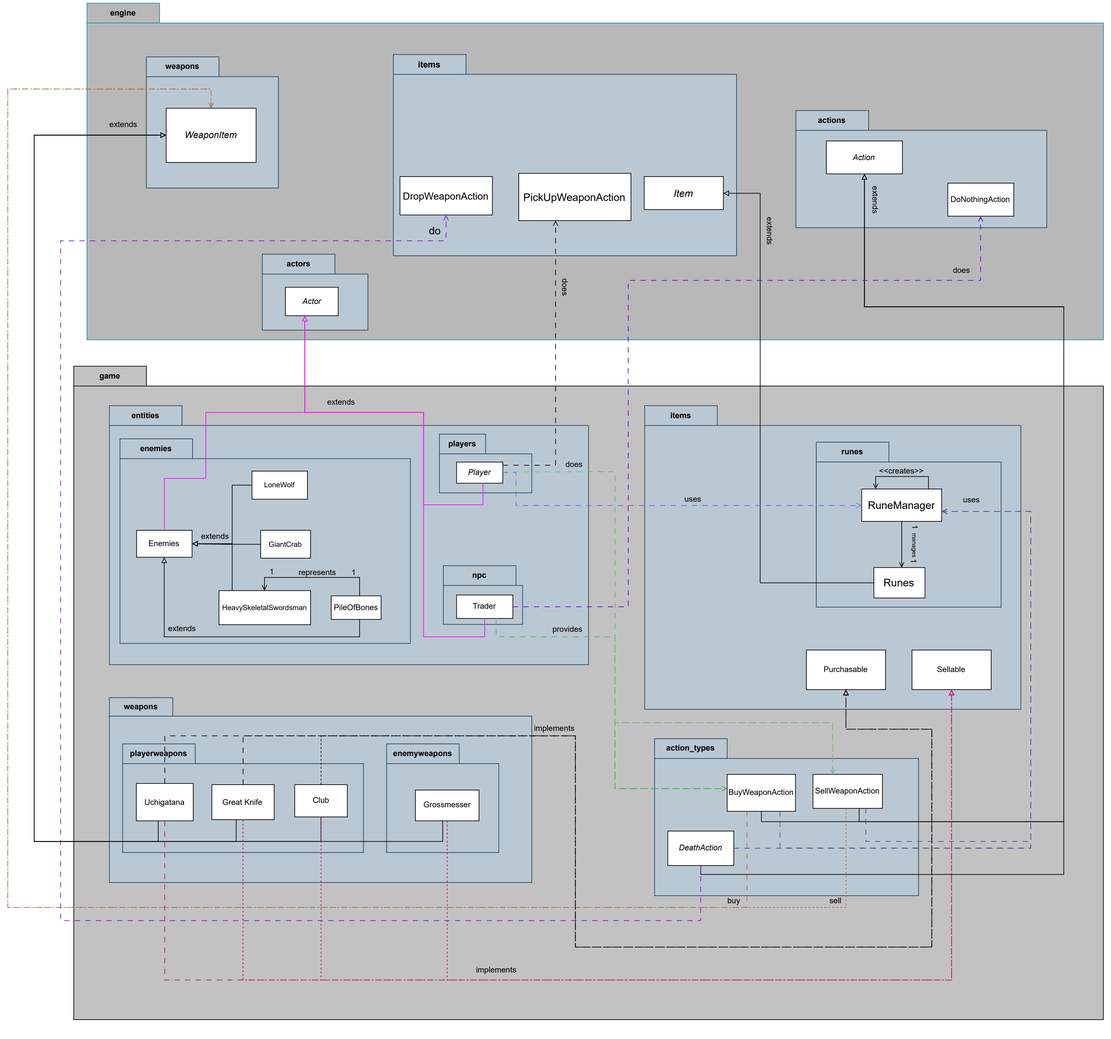
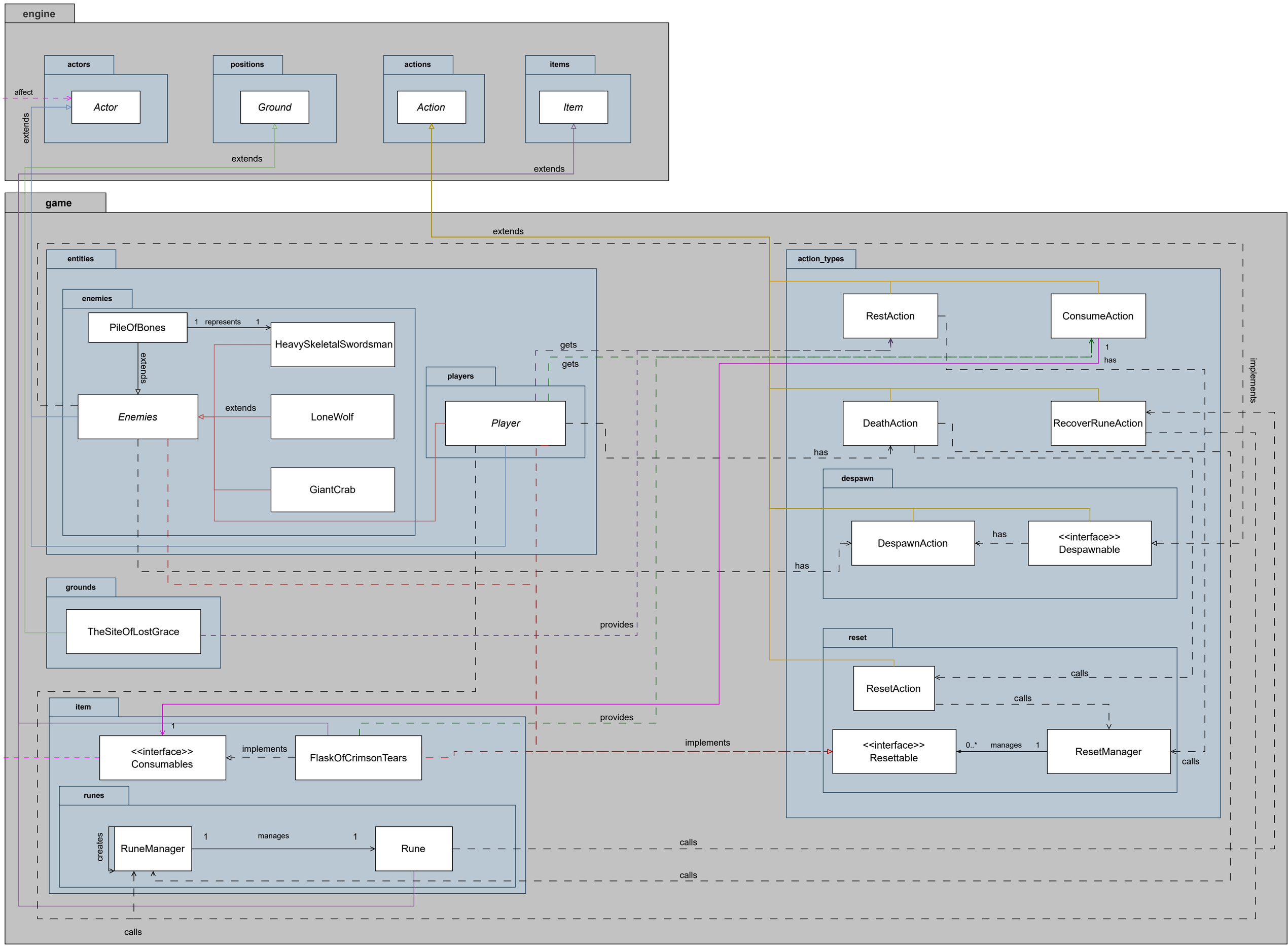
Req 1


engine
positions
Location
Ground
extends
actors
Actor
extends
extends
actions
Actions
1
extends
has
capabilities
1
1
CapabilitySet
has
weapons
WeaponItem
IntrinsicWeapons
1
1
1
<<Uses>>
<<Uses>>
<<Uses>>
contains
extends
<<Uses>>
contains
game
has
grounds
extends
Graveyard
GustOfWind
PuddleOfWater
behaviours
<<Interface>>
Behaviours
*
AOE_AttackBehaviour
implements
implements
has
FollowBehaviour
WanderBehaviour
AttackBehaviour
weapons.enemyweapons
1
Grossmesser
1
items
«interface»
Sellable
implements
entity.enemies
PileOfBones
HeavySkeletalSwordsman
LoneWolf
GiantCrab
extends
Enemy
1
has
<<enumeration>>
Status
<<enumeration>>
Species
1
has
<<creates>>
has
has
1
has
<<Uses>>
action_types
does
does
<<Uses>>
does
1
AttackAction
1
does
DeathAction
despawn
1
DespawnAction
implements
«interface»
Despawnable
AOE_AttackAction
entity.player
<<Uses>>
Player
<<Uses>>
implements

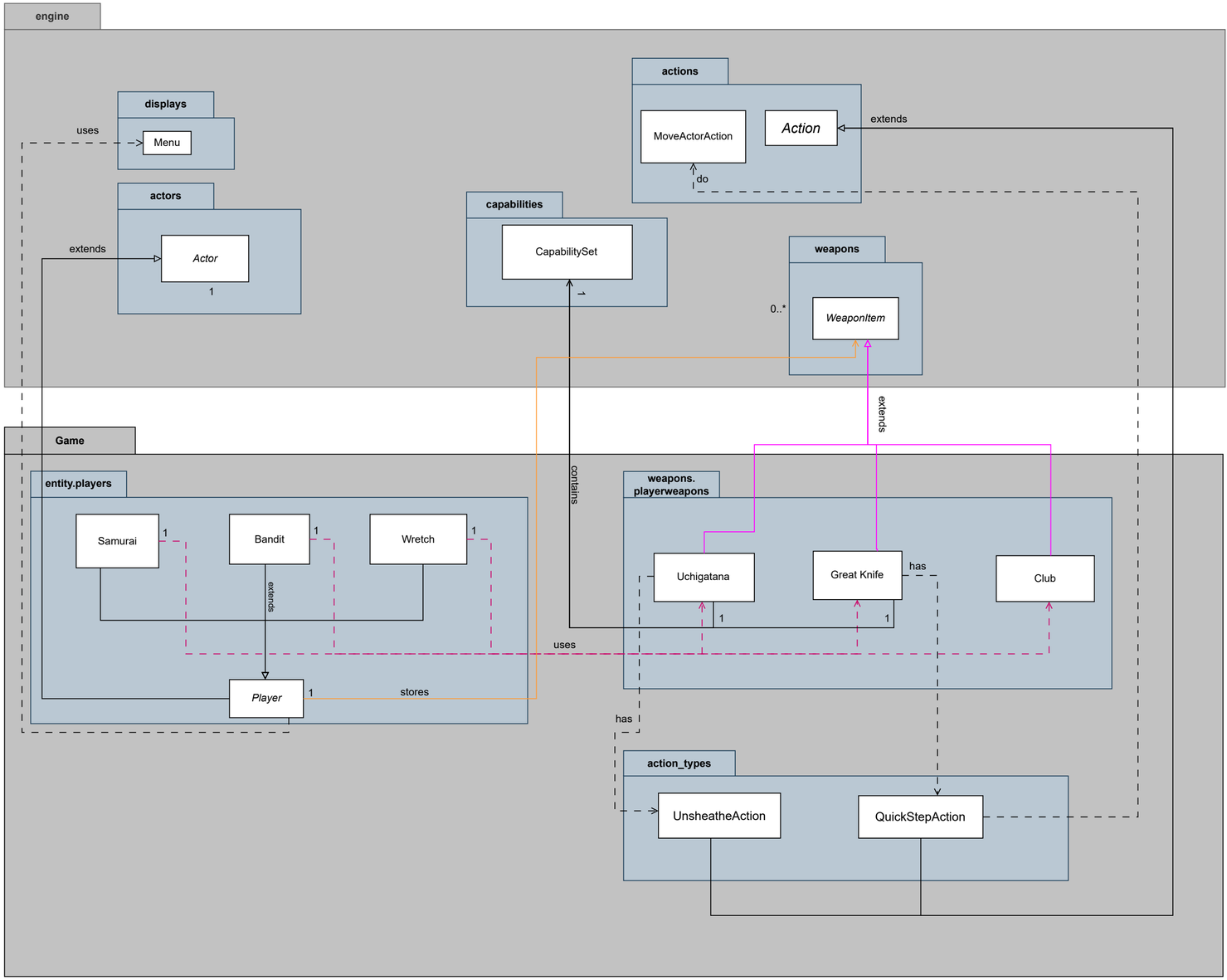
Req 2


engine
weapons
extends
WeaponItem
items
DropWeaponAction
do
PickUpWeaponAction
Item
actions
Action
extends
DoNothingAction
does
actors
Actor
does
extends
game
extends
entities
enemies
LoneWolf
Enemies
extends
GiantCrab
1
represents
1
HeavySkeletalSwordsman
PileOfBones
extends
players
Player
does
npc
Trader
provides
items
runes
<<creates>>
RuneManager
uses
uses
1
requires
1
Runes
Purchasable
Sellable
weapons
implements
playerweapons
Uchigatana
Great Knife
Club
enemyweapons
Grossmesser
action_types
BuyWeaponAction
SellWeaponAction
DeathAction
buy
sell
implements

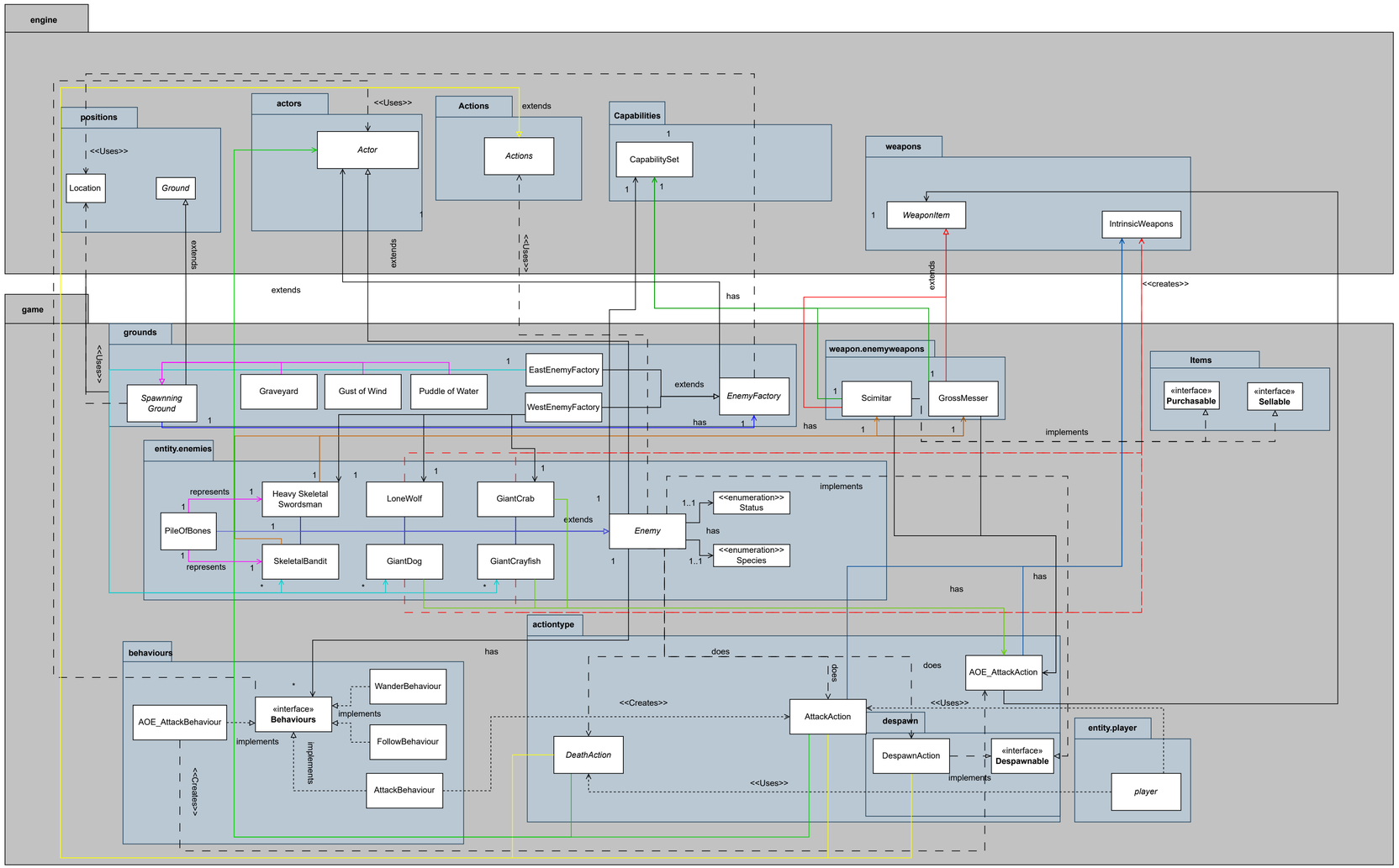
Req 3


engine
actors
Actor
positions
Ground
actions
Action
items
Item
affect
extends
extends
extends
game
entities
enemies
PileOfBones
1
represents
1
HeavySkeletalSwordsman
extends
Enemies
extends
LoneWolf
GiantCrab
players
Player
gets
gets
grounds
TheSiteOfLostGrace
provides
item
1
<<interface>>
Consumables
implements
FlaskOfCrimsonTears
provides
implements
runes
creates
RuneManager
1
manages
1
Rune
calls
calls
calls
extends
action_types
RestAction
ConsumeAction
1
has
implements
DeathAction
RecoverRuneAction
has
despawn
DespawnAction
has
<<interface>>
Despawnable
has
reset
ResetAction
calls
calls
<<interface>>
Resettable
0..*
manages
1
ResetManager
calls

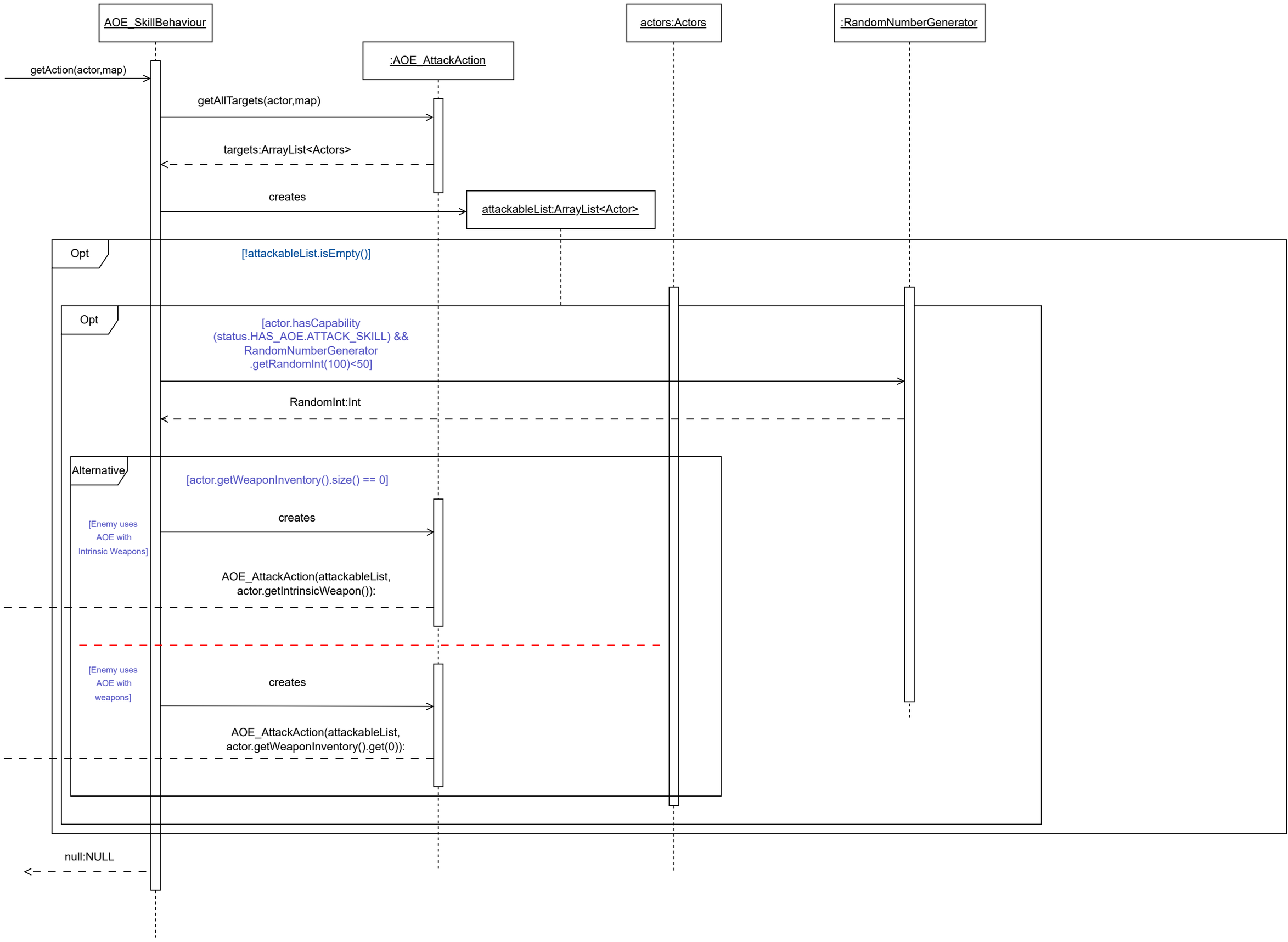
Req 4


engine
displays
uses
Menu
actors
extends
Actor
1
actions
MoveActorAction
Action
extends
do
capabilities
CapabilitySet
1
weapons
0..*
WeaponItem
extends
Game
entity.players
Samurai
1
Bandit
1
Wretch
1
extends
Player
1
stores
contains
weapons.
playerweapons
Uchigatana
Great Knife
has
Club
1
1
uses
has
action_types
UnsheatheAction
QuickStepAction

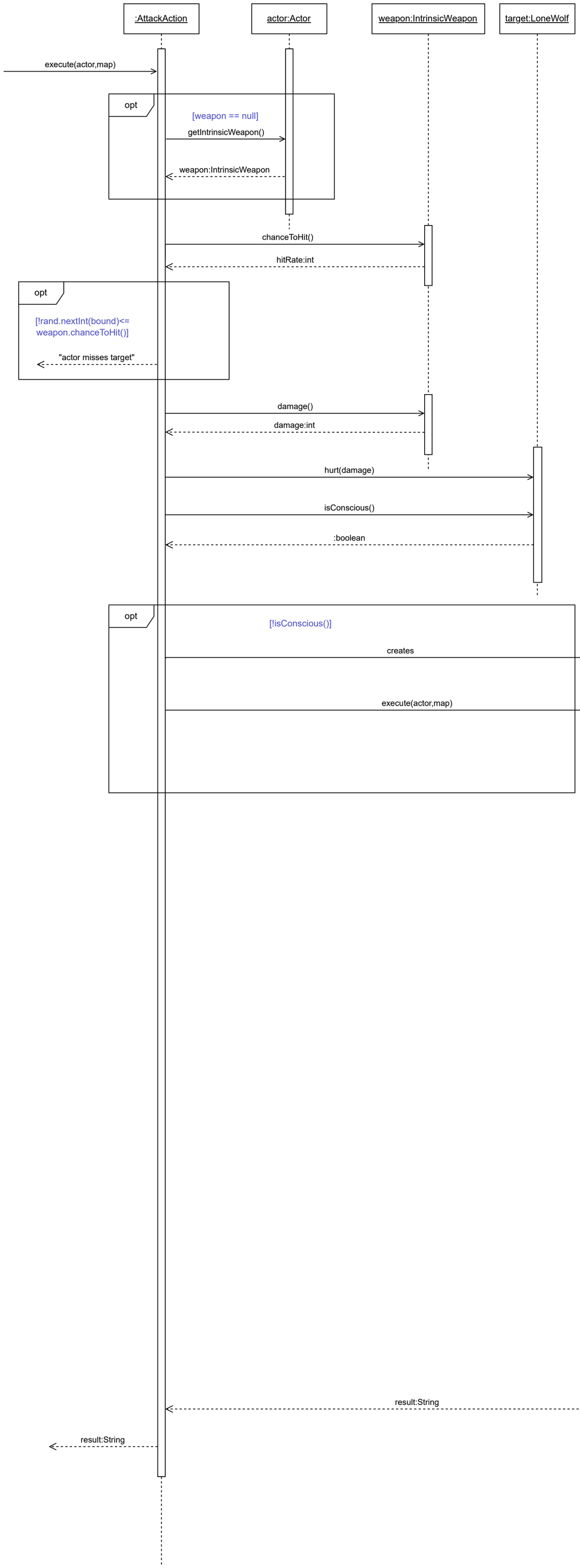
Req 5


engine
positions
<<Uses>>
Location
Ground
actors
<<Uses>>
Actor
Actions
extends
Actions
Capabilities
1
CapabilitySet
1
1
weapons
1
WeaponItem
IntrinsicWeapons
extends
<<creates>>
extends
has
game
grounds
<<Uses>>
Spawnning Ground
Graveyard
Gust of Wind
Puddle of Water
1
EastEnemyFactory
WestEnemyFactory
extends
EnemyFactory
has
1
has
1
weapon.enemyweapons
1
Scimitar
GrossMesser
1
1
implements
Items
«interface» Purchasable
«interface» Sellable
entity.enemies
represents
1
Heavy Skeletal Swordsman
LoneWolf
GiantCrab
1
PileOfBones
SkeletalBandit
GiantDog
GiantCrayfish
represents
extends
Enemy
1..1
<<enumeration>> Status
has
1..1
<<enumeration>> Species
implements
has
has
actiontype
behaviours
has
does
does
does
AOE_AttackAction
<<Creates>>
AttackAction
<<Uses>>
despawn
entity.player
AOE_AttackBehaviour
«interface» Behaviours
WanderBehaviour
FollowBehaviour
AttackBehaviour
implements
implements
implements
<<Creates>>
DeathAction
DespawnAction
«interface» Despawnable
implements
<<Uses>>
player

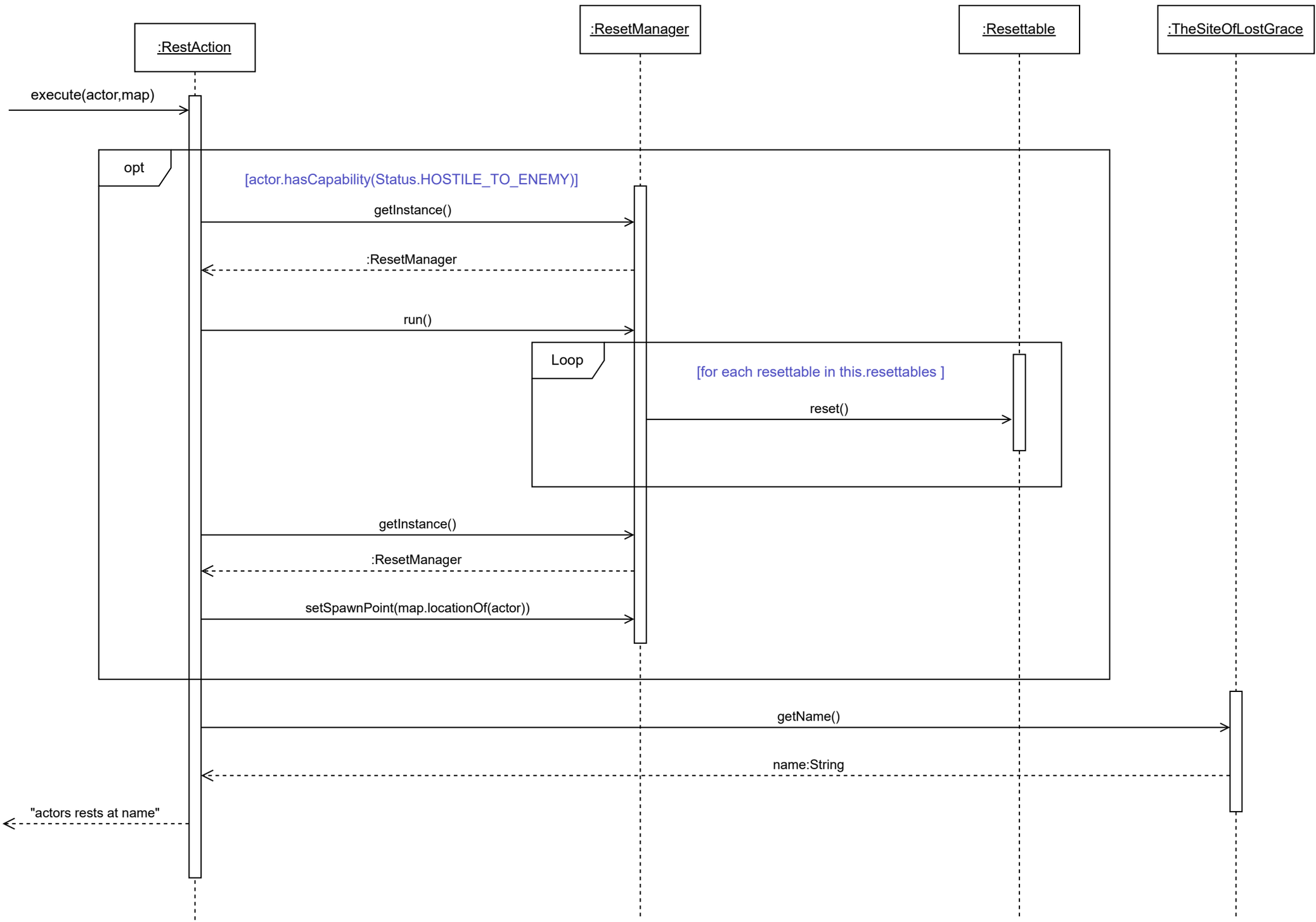
Scenario: A Giant Crab with AOE_SkillBehaviour



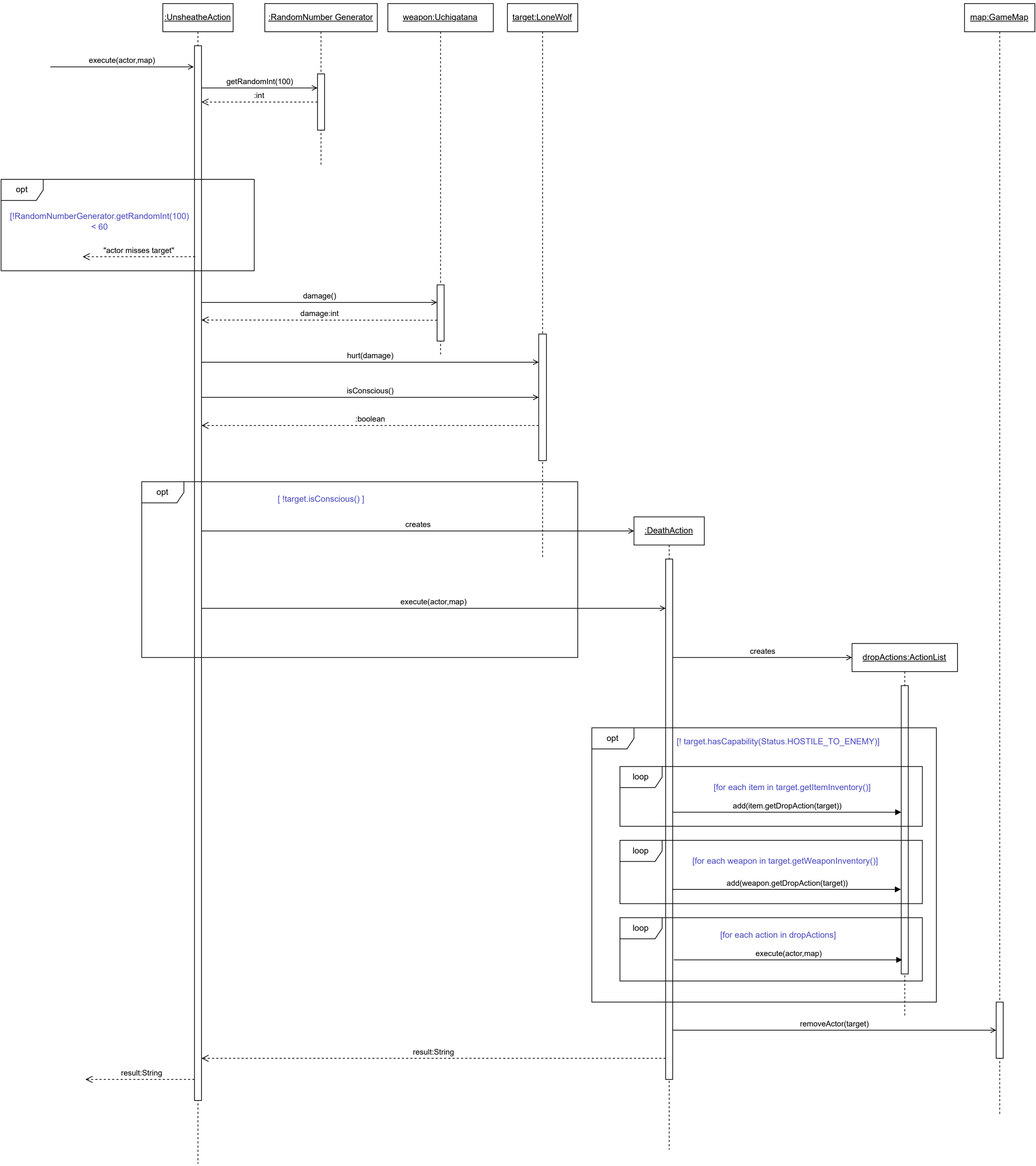
Scenario:Player attacks a Lone Wolf with an intrinsic weapon and obtain runes after killing it

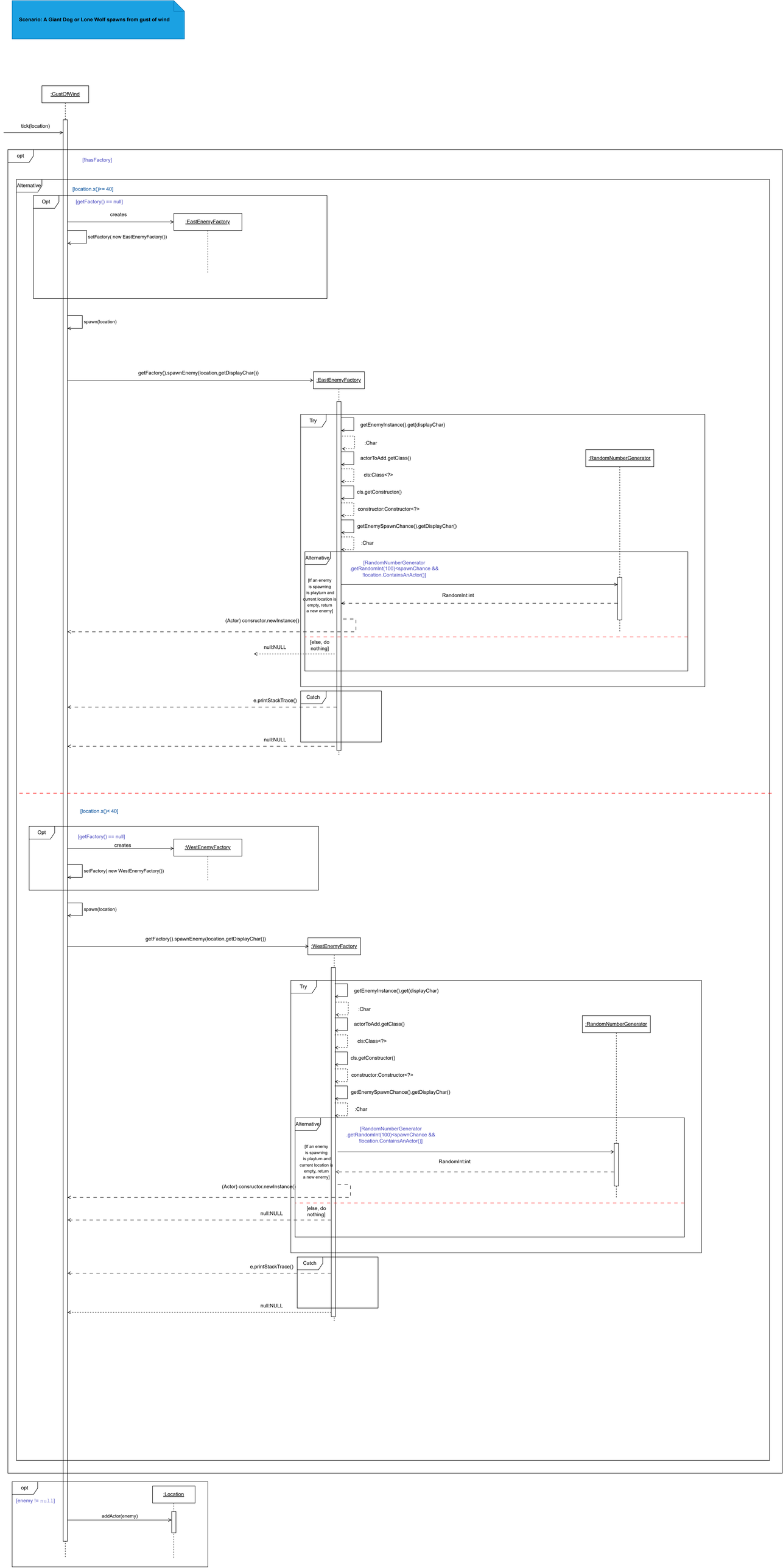


Scenario : Player will rest at TheSiteOfLostGrace which causes a game reset



Scenario:
Samurai attacks a Lone Wolf using its unsheathe skill




Scenario: A Giant Dog or Lone Wolf spawns from gust of wind
:GustOfWind
tick(location)
opt
[!hasFactory]
Alternative
[location.x()>= 40]
Opt
[getFactory() == null]
creates
:EastEnemyFactory
setFactory(new EastEnemyFactory())
spawn(location)
getFactory().spawnEnemy(location.getDisplayChar())
:EastEnemyFactory
Try
getEnemyInstance().get(displayChar)
:Char
actorToAdd.getClass()
:RandomNumberGenerator
cls:Class<?>
cls.getConstructor()
constructor:Constructor<?>
getEnemySpawnChance().getDisplayChar()
:Char
Alternative
[RandomNumberGenerator
.getRandomInt(100)<spawnChance &&
!location.ContainsAnActor()]
[If an enemy
is spawning
is playturn and
current location is
empty, return
a new enemy]
RandomInt:int
(Actor) consructor.newInstance()
[else, do
nothing]
null:NULL
Catch
e.printStackTrace()
null:NULL
[location.x()< 40]
Opt
[getFactory() == null]
creates
:WestEnemyFactory
setFactory(new WestEnemyFactory())
spawn(location)
getFactory().spawnEnemy(location.getDisplayChar())
:WestEnemyFactory
Try
getEnemyInstance().get(displayChar)
:Char
actorToAdd.getClass()
:RandomNumberGenerator
cls:Class<?>
cls.getConstructor()
constructor:Constructor<?>
getEnemySpawnChance().getDisplayChar()
:Char
Alternative
[RandomNumberGenerator
.getRandomInt(100)<spawnChance &&
!location.ContainsAnActor()]
[If an enemy
is spawning
is playturn and
current location is
empty, return
a new enemy]
RandomInt:int
(Actor) consructor.newInstance()
[else, do
nothing]
null:NULL
Catch
e.printStackTrace()
null:NULL
opt
[enemy != null]
:Location
addActor(enemy)