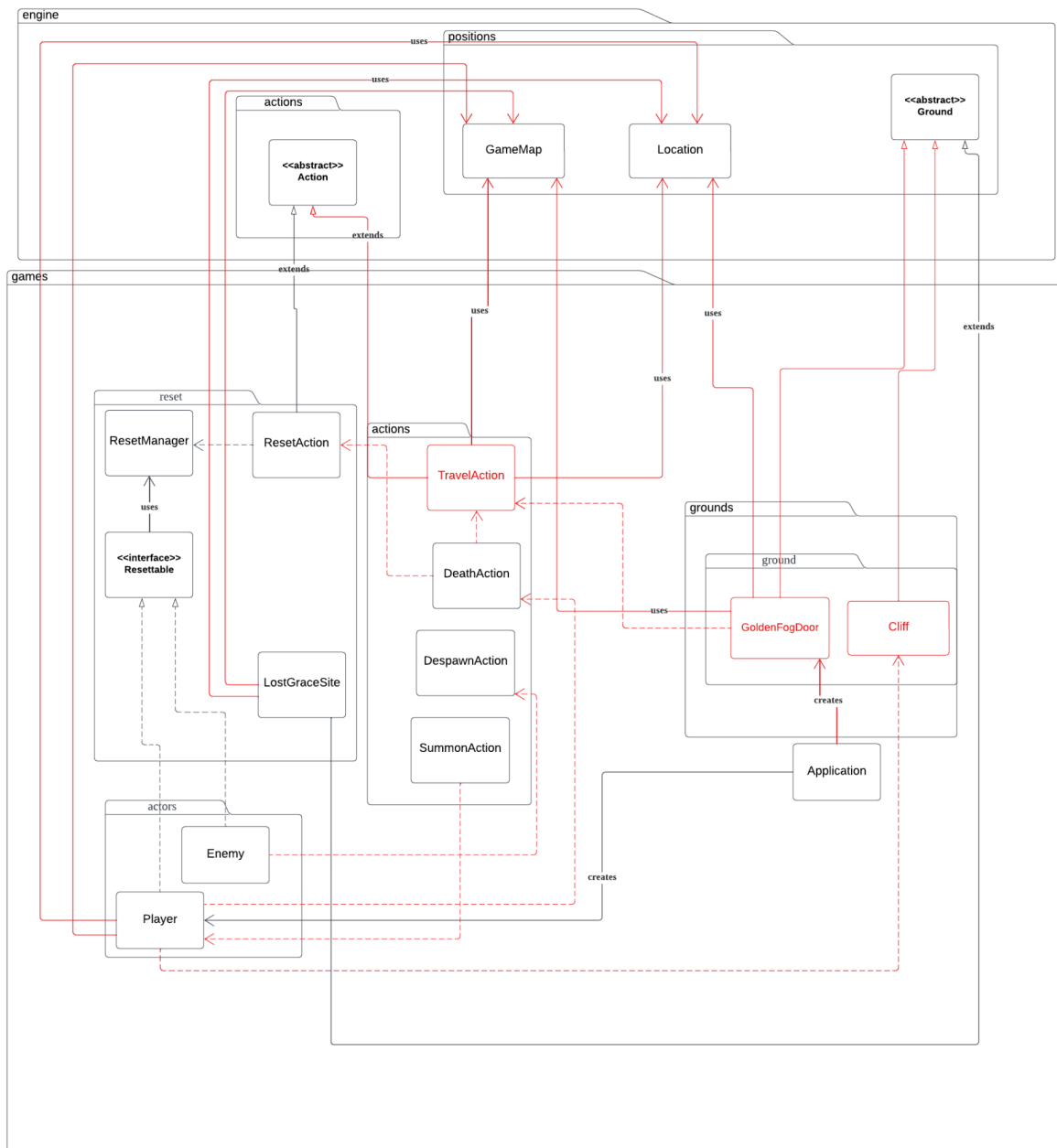


Lab06Team07's UML Diagram

Team Members:

Chew Xin Ning, Foo Kai Yan, Ng Yu Mei

REQUIREMENT 1



The diagram illustrates the architecture of a game engine, organized into two main sections: **games** (top) and **Engine** (bottom).

games Section:

- utils:** Contains `RandomNumberGenerator` and `Status`.
- actors:** A central package containing:
 - skeletal:** `HeavySkeletalSwordman`, `SkeletalBandit`, `BonePiles`, and `<<abstract>> SkeletalEnemy`.
 - dog:** `GiantDog`, `LoneWolf`, and `<<abstract>> DogEnemy`.
 - ocean:** `GiantCrab`, `GiantCrayFish`, and `<<abstract>> OceanEnemy`.
 - demigod:** `Dog`, `GodrickSoldier`, `<<abstract>> DemigodProtector`, and `<<abstract>> Enemy`.
- enemy_factory:** Contains `WestMapEnemyFactory`, `<<interface>> EnemyFactory`, and `EastMapEnemyFactory`.
- actions:** Contains `AttackAction` and `DeathAction`.
- behaviours:** Contains `AttackBehaviour` and `<<interface>> Behaviour`.
- capable:** Contains `Killable`.
- ground:** Contains `Cage` and `Barrack`.
- groundspawning:** Contains `Graveyard`, `GustOfWind`, and `PuddleOfWater`.
- Environment:** A `<<abstract>> Environment` class that interacts with the spawning and ground packages.
- ActorType:** A class that interacts with the `utils` and `actors` packages.
- MapLocation:** A class that interacts with the `utils` package.

Engine Section:

- positions:** Contains `<<abstract>> Ground`, `Location`, and `GameMap`.
- actors:** Contains `<<abstract>> Actor`.
- actions:** Contains `<<abstract>> Action`.

Relationships (Solid and Dashed Lines):

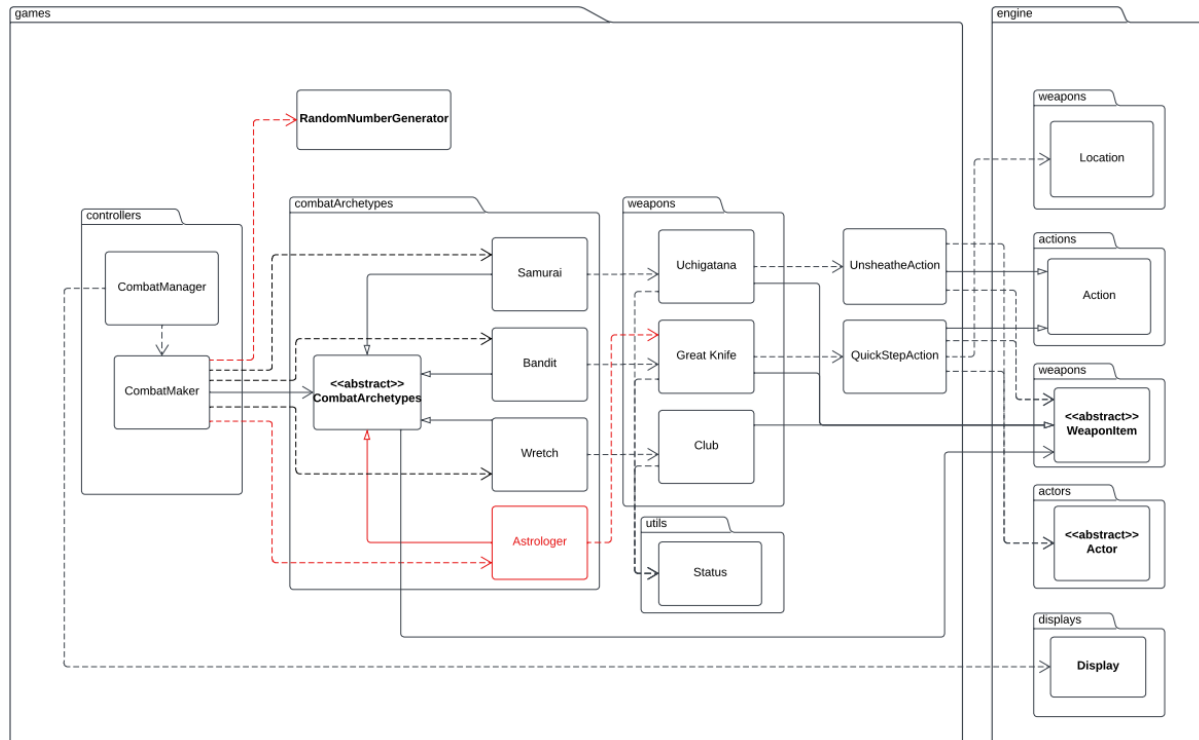
- Solid Lines:** Represent generalization or composition. For example, `<<abstract>> Enemy` is a base class for `Dog`, `GodrickSoldier`, and `DemigodProtector`. `<<abstract>> Environment` is a base class for `Graveyard`, `GustOfWind`, and `PuddleOfWater`.
- Dashed Lines:** Represent associations or dependencies. For example, `AttackAction` depends on `AttackBehaviour`, and `AttackBehaviour` depends on `<<interface>> Behaviour`. `WestMapEnemyFactory` and `EastMapEnemyFactory` implement the `<<interface>> EnemyFactory`.

REQUIREMENT 3

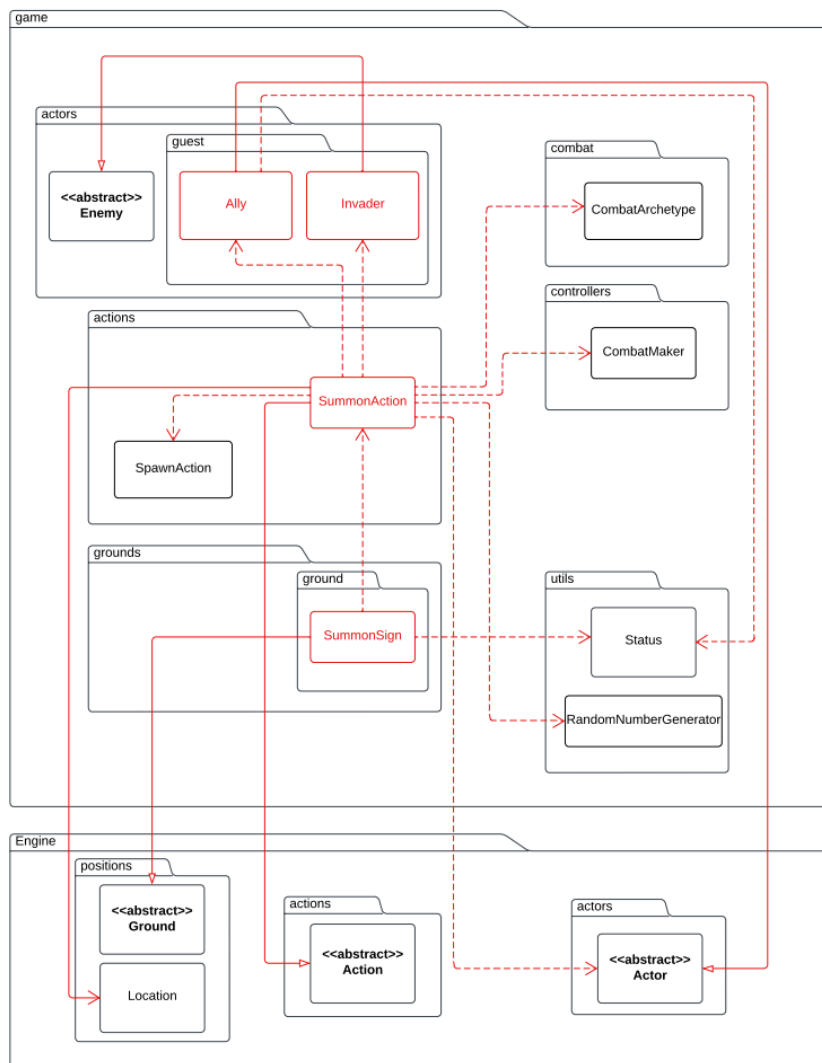


REQUIREMENT 4

Part A



Part B



REQUIREMENT 5

