# Ant Farm Simulation Code Documentation

## Introduction

## This document provides detailed documentation for the Ant Farm Simulation code. The program simulates the lifecycle and combat between ant colonies (ant farms) within a shared environment (meadow). Each colony comprises workers and warrior ants, with various functionalities like combat simulation, tracking colony statistics, and managing ant populations.

## Ant Base Class

The `Ant` class serves as the base class for all types of ants in the simulation.   
It contains the following members:  
role`: A string representing the role of the ant .

## WorkerAnt Class

The `WorkerAnt` class inherits from the `Ant` class and represents worker ants with the role "Worker".

## WarriorAnt Class

The `WarriorAnt` class inherits from the `Ant` class and represents warrior ants with the role "Warrior".

## AntFarm Class

The `AntFarm` class models a colony of ants.   
Each ant farm tracks its ants, kills, and state (alive or dead) and participates in the simulation.  
  
Key members:  
- `id`: A unique identifier for the colony.  
- `species`: The species of ants in the colony.  
- `ants`: A vector of ants (worker and warrior).  
- `workers`: Count of worker ants.  
- `warriors`: Count of warrior ants.  
- `antKills`: Total number of ants killed by this colony.  
- `colonyKills`: A map storing the number of enemy colonies killed by species.  
- `ticksAlive`: The number of simulation ticks the colony has survived.  
- `alive`: A boolean indicating if the colony is still active.

Key methods include:

## addAnt(shared\_ptr<Ant> ant)`: Adds an ant to the colony. recordKill(string species)`: Records a kill of an enemy colony. removeAnts(int count)`: Removes a specified number of ants from the colony. tick()`: Increments the colony's age by one simulation tick. displaySummary()`: Outputs a summary of the colony's status.

## Meadow Singleton Class

The `Meadow` class acts as the shared environment where all ant farms coexist and interact.   
It follows the Singleton design pattern, ensuring only one instance exists.  
  
Key members:  
- `antFarms`: A vector of all ant farms in the meadow.  
  
Key methods:  
- `addAntFarm(shared\_ptr<AntFarm> antFarm)`: Adds a new ant farm to the meadow.  
- `simulateCombat()`: Simulates combat between ant farms.  
- `removeAntFarm(int id)`: Removes an ant farm by ID if it is no longer alive.  
- `displaySummary()`: Displays summaries of all ant farms.

## Command Functions

The program defines several command functions to manage the simulation:  
- `executeSpawn(int id, string species, int workers, int warriors)`: Creates a new ant farm with the specified parameters.  
- `executeTick(int ticks)`: Advances the simulation by a specified number of ticks, during which combat occurs.  
- `executeSummary()`: Displays a summary of all ant farms in the simulation.

## Main Function

The `main()` function demonstrates the usage of the program. It spawns two ant farms, runs the simulation for 25 ticks, and displays a summary of the results.

## Sample Output

The program generates a console-based output detailing the lifecycle of the ant farms.   
Each tick displays the combat results, and the final summary shows the status of each colony.

## Conclusion

This simulation effectively models the lifecycle and interactions of ant colonies within a shared environment.   
The modular design allows for extensibility, such as adding more ant roles or complex combat mechanics.