|  |  |
| --- | --- |
|  | **Namal University Mianwali**  **Department of Computer Science** |

**Assignment 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Course | Software Engineering | | |
| Instructor | **Miss. Asiya Batool** | **Session / Semester** | 2023-2027 (3rd) |

|  |  |
| --- | --- |
| Topic | Ant Farm Simulation System with Design Patterns |
| Name | **Rayan Badar 18** |

**Ant Farm Simulation**

**Task Overview (1 & 2)**

This task is an implementation of an **Ant Farm Simulation** using C++ to demonstrate the application of **Design Patterns**. The simulation incorporates:

* **Creational Patterns**.
* **Structural Patterns**.
* **Behavioral Patterns**.

The simulation models a meadow with multiple ant colonies that interact through different resource management tasks. It uses a **command-line interface (CLI)** for user interaction.

**Key Features**

1. **Meadow**:
   * Ensures only one meadow exists in the simulation.
   * Manages all ant colonies.
2. **Ant Creation**:
   * Dynamically creates different types of ants.
3. **Ant Farm Construction:**
   * Systematically constructs ant farms with rooms and ants.
4. **Ant Attributes:**
   * Dynamically modifies ant attributes, such as strength and speed.
5. **Tick-Based Simulation:**
   * Orchestrates the actions of ants in a tick-based system.
6. **Rooms and Resting**:
   * Rooms allow ants to rest, with limited capacity, consuming food resources.
7. **Battles**:
   * Warriors engage in battles, and the winner gains strength while the loser is removed.
8. **Colony Takeovers**:
   * If a queen is defeated, the victorious colony takes control of the losing colony.

**System Requirements**

* **Language**: C++
* **Compiler**: Any C++ compiler

**Usage Instructions**

The program provides a **menu-driven interface**. Below are the available commands:

1. **Display All Colonies**:
   * View a list of all colonies in the meadow.
   * Command: 1
2. **Spawn a New Colony**:
   * Creates a new colony with a queen ant.
   * Input: 2
   * Required: Coordinates (x, y) and species name.
3. **Give Resources to a Colony**:
   * Add resources (ants or food) to a specific colony.
   * Input: 3
   * Required: Colony ID, ant type, and count.
4. **Simulate Ticks**:
   * Simulate actions of ants for a specified number of ticks.
   * Input: 4
   * Required: Number of ticks.
5. **Display Colony Summary**:
   * View the status and details of a specific colony.
   * Input: 5
   * Required: Colony ID.
6. **Exit Simulation**:
   * End the simulation.
   * Command: 6

**Code Structure**

* **main.cpp**: Contains the core logic and menu-driven interface.
* **Meadow Class**: Singleton class managing all colonies.
* **Ant Class**: Base class for ants with derived classes.
* **Room Class**: Represents rooms in ant farms for resting ants.
* **Ant Farm Class**: Manages ants and rooms using the Builder and Factory patterns.
* **Utility Functions**: Handles other commands and tick-based simulation.

**Design Patterns Used**

1. **Singleton Pattern**: Ensures only one meadow instance exists.
2. **Factory Pattern**: Dynamically creates ants based on type.
3. **Builder Pattern**: Constructs ant farms with rooms and ants.
4. **Decorator Pattern**: Adds dynamic attributes to ants.
5. **Mediator Pattern**: Coordinates actions in a tick-based system.

**Acknowledgments**

This project is inspired by the need to understand and apply software design patterns in a practical scenario. Special thanks to the **Software Engineering Course** instructors for their guidance.