Concurrent and Distributed Systems Homework Assignment

Rădulescu (Poenariu) Ionela-Alexandra

CEN3.2B

2024

Computers and Information Technology
Department Faculty of Automatics, Computers and Electronics

Technical Report

14 of January 2024

1 Introduction

The objective of this project is to simulate an environment where wild animals, such as bears and foxes, are monitored in multiple zones to identify potential diseases that might be spread by wild animals. The simulation is implemented in Java, leveraging concurrency mechanisms to model the collaborative actions of wild animals, authorities' employees, and a laboratory manager.

2 The decisions taken in creating the project architecture.The motivation behind it.

The conceptualization and development of this project architecture were motivated by a keen interest in simulating the complex interplay between wildlife and the systems designed to monitor and manage it.

3 Project Architecture and Design Decisions

The system delineates distinct classes, each encapsulating specific functionalities. The MonitoredZone class represents a zone with responsibilities spanning animal management, movement control, and trouble reporting. Threaded behavior is simulated by the WildAnimal class, encapsulating individual animals' actions, including spawning, movements, and trouble production, while synchronized methods ensure orderly execution. The AuthoritiesEmployee class embodies personnel collecting samples, utilizing semaphores for controlled access to monitoring information. The LaboratoryManager class oversees sample processing through TCP/IP communication. Synchronization mechanisms include semaphores, exemplified by sampleSemaphore, and locks, such as moveLock for animal movements. Thread collaboration is evident in the independent actions of wild animals and synchronized reporting of troubles. The system implementation comprises interrelated classes, exemplifying the dynamic collaboration of threads and effective synchronization mechanisms.

3.1 Architecture Overview

Monitored Zones:

- Each monitored zone is represented as an $N \times N$ matrix.
- The zones are created and managed by the Headquarter class.
- Wild animals are assigned to random zones, with restrictions on the number of animals in a zone.

Wild Animals:

- Modeled as separate threads to simulate concurrent movements and trouble production.
- Spawn randomly in zones, move in different directions, and produce trouble periodically.
- Synchronized methods ensure proper coordination, preventing concurrent access to critical sections.

Authorities' Employees:

- Authorities await to collect samples from devastated zones.
- Controlled access to monitoring information using semaphores.
- Multiple employees can read from the monitoring system concurrently, respecting maximum limits.

Laboratory Manager:

- Listens for samples from employees and processes them through TCP/IP communication.

3.2 Synchronization Mechanisms

Semaphores:

- Used to control access to monitoring information when collecting samples.
- Semaphore named sampleSemaphore ensures a maximum of 10 authorities' employees can read concurrently.

Locks:

- ReentrantLock named "moveLock" ensures synchronization of animal movements within a zone.

3.3 Thread Collaboration

Wild Animal Threads:

- Independently initiates move and trouble-producing actions.
- Concurrent execution of these actions is managed to simulate dynamic animal behavior.

Reporting Troubles:

- Proper synchronization ensures that both report and move methods are not called concurrently for a given object.
- The monitoring system reports trouble to authorities based on animals' actions.

4 Implementation Details

4.1 Implemented Classes

WildAnimal: Representing individual wild animals as separate threads, the WildAnimal class encapsulates functionalities such as random spawning, movements, trouble production, and resting.

AuthoritiesEmployee: This class represents authorities' employees responsible for collecting samples. Leveraging semaphores, it controls access to monitoring information, facilitating efficient collaboration between the field and authorities.

LaboratoryManager: The LaboratoryManager class plays a pivotal role in the system by listening for samples through TCP/IP communication and processing the collected samples.

MonitoredZone: Representing a monitored zone with its resident wild animals, the MonitoredZone class manages various aspects, including animal movements, rest periods, time simulation, and reporting troubles.

MonitoringSystem: Functioning as the centralized system for reporting troubles and managing information, the MonitoringSystem class provides essential methods for authorities to collect samples and report new devastated places.

Headquarter: The Headquarter class assumes the responsibility of centralized management for monitored zones and the allocation of wild animals.

5 Observations after Implementation

The implemented simulation demonstrates effective concurrency in modeling the interactions between wild animals, authorities, and the laboratory manager. The chosen architecture and synchronization mechanisms provide a foundation for a realistic and dynamic simulation. Further testing and potential enhancements can contribute to the system's robustness and scalability.

5.1 Key Requirements Met

The project meets key requirements specified in the "Context" material:

1. **Number of Monitored Zones:**
   - The system supports multiple monitored zones.
   - The number of zones is randomly determined between 2 and 5.
2. **Zone Characteristics:**
   - Each zone is designed as a matrix with $N \times N$ dimensions, where $100 \leq N \leq 500$.
   - The zones are created and managed by the Headquarter class.
3. **Wild Animals in Zones:**
   - Each zone has a list of wild animals.
   - Wild animals are created and assigned to random zones by the Headquarter class.
4. **Animal Spawn and Movement:**
   - Wild animals spawn randomly in each zone at a random place.
   - Spawn times are random, where $500 \leq t \leq 1000$ milliseconds.
   - Two animals cannot occupy the same place.
5. **Trouble Production:**
   - Wild animals produce trouble by moving in different directions.
   - When reaching zone margins, they move in any other direction.
   - With each move, they produce trouble, devastating the place.
   - If surrounded by other animals, they stop for a random rest time ($10 \leq t \leq 50$ milliseconds).
   - The monitoring system informs authorities about the trouble's position.
6. **Resting Time:**
   - After producing trouble, an animal rests for 30 milliseconds.
   - Animals produce trouble while alive.
7. **Authorities' Employees:**
   - Authorities await to collect samples from devastated zones.
   - They can read information from the monitoring system regarding the number of devastated places.

- A maximum of 10 authorities' employees can read from the monitoring system concurrently.
- There's a random time delay between two consecutive monitoring system readings.
- All employees deliver the collected samples to the laboratory.

8. **Headquarter Functionality:**
   - The Headquarter class contains all monitored zones.
   - It creates the zones and assigns wild animals to random zones.
   - Each wild animal registers itself in the corresponding zone.

The project successfully captures the essence of the specified requirements, providing a simulation of a dynamic environment with multiple zones, wild animals, and collaborative interactions between authorities, monitoring systems, and the laboratory manager.

This technical report outlines the decisions made in creating the project architecture, motivations behind them, details of the implementation, synchronization mechanisms, and observations after running the program.