# Concurrent and Distributed Systems

# Practical 3 – Synchronising Java Threads

**This lab contains a checkpoint at the end of Task 2. Please contact any lecturing staff when you reach this point to receive your credit. You will be expected to explain your program design and to demo the implementation.**

Useful material:

- Lecture on Threads
- Lecture on Java Synchronisation
- Solution from Lab 1

This lab is focussed around synchronising concurrent Java threads. The experiments are based on the 'Ornamental Garden Problem' as known from Practical 1.

As a brief reminder, the 'Ornamental Garden Problem' can be described as: A large ornamental garden is open to members of the public who can enter through either one of two gates at the lower and upper part of the garden. The management want to determine how many people are in the garden at any one time. They design a computer system to do this.

**Task 1:** Design a solution to the Garden Gate Problem using **threads** (rather than processes as in lab 1) as a means to model the concurrent activities. In other words, each of the two counters at the gates should be represented by a thread. The shared counter should again be represented **as a single location in a file.** Do not code immediately! – Think about your solution first! Then, experiment with your implementation. Verify that, using unsynchronised threads, you get the same fault as in lab 1

**Task 2:** Identify the critical sections in your program. Resolve the synchronisation problems using Java synchronisation mechanisms. Which of the mechanisms covered in the lecture on Java synchronisation is the most appropriate? Remember that the lock is associated with a Java object *not* a class!

**Checkpoint.**