Lab5 : Multithreading

Exercise 1 – Runnable and synchronized

1. Why use runnable ?

Runnable is used to separate the code that has to be run asynchronously by threads, from the rest of the code that is being synchronously executed. A runnable class has one single method which is run(). Therefore, each thread instantiated with this runnable will execute the code from this run() method.

2. What do you observe ? Do you think the behaviour of the program is correct ?

We can see that the 4 threads are executed concurrently. They all count from 0 to 1000 than stop.

5. Explain what happens ?

At first, no problem was observable. This was because of the “system.out” that were in the method run(). Printing count in the console allows the threads to naturally synchronize because it takes time to print out values. However, if we comment the lines that print the count, we see that the ArrayList has a size that is less than expected (4000). So we can infer that there is a synchronisation problem between threads when accessing to the list. We should synchronise the access to the list.

Exercise 2 – Dining philosophers problem

1. What is the problem with the given code ?

The program is deadlocked after a few loops because the philosophers are all trying to eat at the same time so they each have a fork but it is not sufficient to eat, which means nobody is able to eat.
We have to modify the code in order to ensure that philosophers take forks only if the forks are available. To do so, we use the function tryLock().

Exercise 3 – Rendezvous

1. What does the following code do ?

This program runs 2 threads, the main is one thread, and another thread is created in the main. The thread created prints “ping”, and then the main prints “pong” only if “ping” has been printed, otherwise it sleeps for 0.1 second). When we run this program, “ping” and “pong” are printed.

2. Comment the line Thread.sleep(100); in the method pong(). What happens and why?

The program now prints only “ping” and keeps running. This is because the main thread executes the pong method before the other thread executes the ping method and therefore remains stuck in the while loop without doing anything. Meanwhile, the other thread has printed “ping” but since the main thread is still stuck in the while loop, it doesn’t do anything.