Threads in Java
Lab exercises

EPL222 – Lab12

Semaphores in Java

- In addition to locks the Java Concurrency API also supports counting semaphores
- Semaphores in Java works like in C

`public Semaphore(int permits)`

- Creates a Semaphore with the given number of permits and non-fair fairness setting

`public Semaphore(int permits, boolean fair)`

- Creates a Semaphore with the given number of permits and the given fairness setting

- *permits* – the initial number of permits available (the value of the semaphore)
  - This value may be negative, in which case releases must occur before any acquires will be granted
- *fair* – true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

Semaphores in Java

`public void acquire() throws InterruptedException`

- Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one
- If no permit is available then the current thread becomes blocked

`public boolean tryAcquire()`

- Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one
- If no permit is available then this method will return immediately with the value false

`public void release()`

- Releases a permit, increasing the number of available permits by one
- If any threads are trying to acquire a permit, then one is selected and given the permit that was just released
- That thread is unblocked
- There is no requirement that a thread that releases a permit must have acquired that permit by calling acquire()
- Correct usage of a semaphore is established by programming convention in the application

Lab exercise 1

- Solve the dining philosophers problem in Java using
  - Simple synchronization
  - Semaphores
  - Monitors
- The dining philosophers problem:
  - The dining philosophers problem states that there are 5 philosophers sharing a circular table and they eat and think alternatively
  - There is a plate of food for each of the philosophers and 5 forks
  - A philosopher needs both their right and a left forks to eat
  - A hungry philosopher may only eat if there are both forks available
    - Blocking solution: otherwise a philosopher waits for their forks to become available
    - Non blocking solution: otherwise a philosopher puts down the fork he got and begin thinking again