UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Practical 0

- **Date issued:** 28 July 2022
- **Deadline:** None

1 Introduction

1.1 Objectives and Outcomes

This practical aims to introduce the topic of Java threads. It is already assumed you are able to create programs within the Java language.

1.2 Provided Code

Some example code to demonstrate the functionality of threads within Java as well as two different implementations of threads have been provided. Please download these examples from either ClickUp or the Discord server before continuing.

2 Practical

Java supports multiprocessing via a wide array of methods, however for the purposes of this course, we will be focusing on threads specifically.
Java encompasses the idea of a thread into a suitably named class, the **Thread** class. There are a couple of methods we can use to implement this class, which will be introduced to you in this course.

No matter which method you choose to use, the end result is the creation of an explicit **Thread** object in Java.

1. The first way (and perhaps the easiest way) to create this object is by extending the **java.lang.Thread** class. In the code provided to you, the **TDemo** class is an example of this method.
2. The second way to create a Thread object is by creating a class that implements the **java.lang.Runnable** interface, and have an instance of this class passed to the constructor of a new **Thread** object. The **RDemo** class is an example of this implementation.

The **Main** class in the example given covers creation of the **Thread** objects and the process of starting the threads.

Please study the example carefully!

start calls run ... start is in main and run in the class

Notes:

- No matter which method you use, your class will need to contain a **run()** method. This method serves as the 'action' of the thread that will occur when you call the thread's **start()** method.
- Try playing around with the example to gain a better understanding of threads.
- See what happens if you comment out lines 18-22 in the **Main** class of the demo.
- Some useful functions for the Thread class:
  - **void sleep(long)** will cause the thread to cease execution for the number of milliseconds given. (Note that this method throws an **InterruptedException** which will need to be caught)
  - **String getName()** will return the name of the thread. (This is very useful for output).
  - **Thread currentThread()** will return a reference to the currently executing thread. (This can be replaced with the 'this' keyword if you are extending the Thread class)
- Further information about the **Thread** class and **Runnable** interface can be obtained via the online Java Documentation.

then you don't wait for the threads to finish

Tut 1:

```
try {
        for (int i = 4; i > 0; i--) {
            System.out.println("Thread: " + threadName + ", " + i);
            //Let the thread sleep for a while
            Thread.sleep(random());
        }
    } catch (InterruptedException e) {
        System.out.println("Thread: " + threadName + " interrupted");
    }
```