CO661 Class 9 - Data parallelism (in Java)

The **Madhava-Leibniz** series can be used to calculate π as follows:

$$\pi \;=\; 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}$$

An approximation of π can be calculated with a finite rather than infinite series here, i.e., with $4\sum_{k=0}^{l}\ldots$ for some $l \in \mathbb{N}$.

Based on this definition, create a Java program that performs a data-parallel computation to approximate π using the notion of futures, e.g., with `FutureTask`, which provides the following key methods:

```
public class FutureTask<V> implements Future<V>, RunnableFuture<V> {
  public FutureTask(Callable<V> callable);
  public V get();
  public void run();  // means it is Runnable
}
```

- A good starting point is to have a future running a worker for every k from 0 to l, creating a `Thread` for each future to run in parallel (see lecture examples), storing the futures in a list, and then `get`ing the results of each future in sequence;
- Next, adapt your workers to sequentially compute a subsequence (i.e., chunk the series), rather than computing just one iteration for a single k;
- Finally, if you have time, explore the use of an `ExecutorService` rather than using `Thread`. You can create an executor service with a fixed pool of threads (which can map onto cores), e.g. as:

  ```
  ExecutorService pool = Executors.newFixedThreadPool(8);
  ```

  (see the Java docs online).

  Runnable tasks (e.g., futures) can be "submitted" to the pool via `pool.submit(future);` and the pool can be forced to execute any remaining tasks with `pool.shutdown();`