UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems Date Issued: 25 October 2021

Practical Assignment 3

- **Due Date:** 28 October 2021
- **Assessment:** The practical will be assessed offline.
- This practical consists of 2 tasks. Read each task **carefully**!

1 Information

1.1 Objective

This practical aims to explore wait-free methods of concurrency. Specifically, consensus protocols.

1.2 Provided Code

You will be provided with some skeleton code to aid in the assignment, consisting of the following Java classes:

- Consensus.java
- ConsensusThread.java
- ConsensusProtocol.java

1.3 Mark Allocation

For each task in this practical, in order to achieve any marks, the following must hold:

- Your code must produce console output. (As this is not marked by fitchfork, formatting is not that strict)
- Your code must not contain any errors. (No exceptions must be thrown)
- Your code may not use any external libraries.
- Your name and student number MUST appear in EVERY file you upload.

The mark allocation is as follows:

Task Number	**Marks**
Task 1	10
Task 2	10
Total	20

2 Assignment

This assignment involves exploring two consensus protocols, one that has a consensus number of 2 and one that has infinite consensus number.

2.1 Task 1 - Read-Modify-Write Consensus

For this task, two friends are deciding how much to spend on a night out, you must simulate their decision by performing a RMWConsensus protocol:

2.1.1 Implementation

You must implement the following:

- ConsensusProtocol.java
  - This class is given.
  - Implement the **propose()** method.
- ConsensusThread.java
  - This class is given.
  - Implement the **run()** method
- RMWConsensus.java
  - You must create this class to extend ConsensusProtocol
  - Implement the **decide()** method.

2.1.2 Notes

- There should be two threads for this task, each must do the following:
  - Each thread must propose an amount to spend between 100 and 200.
  - The threads must then wait for a random amount of time between 50 and 100 ms.
  - Each thread must then decided on the same chosen amount.
  - This must be repeated 5 times.
- Be sure to reset your consensus protocol between each run.
- Be sure that the value decided is the same for both threads.

2.1.3 Output

The following output needs to occur:

- Output the value that the thread proposes to spend when **propose()** is called.
- Output the value of the register when **decide()** is called.
- Output the value each thread decided on.

2.2 Task 2 - Compare and Swap Consensus

For this task, the social circle has grown and there are now 5 friends trying to decide how much to spend on a night out. Extend Task one by using a CASConsensus to now have an infinite consensus number:

2.2.1 Implementation

You must implement the following:

- ConsensusProtocol.java
  - This class is given.
  - Implement the **propose()** method.
- ConsensusThread.java
  - This class is given.
  - Implement the **run()** method
- CASConsensus.java
  - You must create this class to extend ConsensusProtocol
  - Implement the **decide()** method.

2.2.2 Output

The following output needs to occur:

- Output the value that a thread proposes to spend when **propose()** is called.
- Output the value of the atomic value when **decide()** is called.
- Output the value each thread decided on.

2.3 Submission

Submission of this assignment will be via click-up, please zip all of your source code from each task SEPERATELY and upload each zip folder to the click-up submission.
Ensure your name and student number are present in all of your classes!

2.4 Notes

- You must implement your own Main.java file to test your code.