COS226 - Concurrent Systems Assignment 1 Specification

Release Date: 25/08/2025

Due Date: 22/09/2025 at 23:59

Total Marks: 100

1 General Instructions

- Read the entire assignment thoroughly before you start coding.
- This assignment should be completed individually; no group effort is allowed.
- To prevent plagiarism, every submission will be inspected with the help of dedicated software.
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Ensure your code compiles with Java 9
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent), and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). If you have any form of question regarding this, please ask one of the lecturers to avoid any misunderstanding. Also note that the OOP principle of code reuse does not mean that you should copy and adapt code to suit your solution.

3 Overview

In this assignment, you will design and implement a classic concurrency problem using multi-threading. The focus is on correctness, synchronization, and demonstrating understanding of concurrency control.

4 Treasure Chest Game

For this assignment, you are expected to implement the treasure chest game.

In this game there is a treasure chest that starts with totalCoins (2000). Each player is a thread which tries to grab a random number of coins (in the range of 1 to 3) from the chest. The chest is protected by a **lock**. The game ends when there are no more coins in the chest. When the game ends then you should be able to see a tally of how many coins each player has.

In this game, the **lock** corresponds to a chosen lock of the following options:

- TTAS Lock with Backoff
- CLH Lock

You should implement all the lock options and test them to see how they compare in terms of fairness. i.e. how fairly are the coins distributed after the game ends?

In your report you should briefly explain the two different locks as well as compare the locks in terms of fairness.

You should do multiple tests in your main, specifically for the following:

- Different numbers of players (2, 8 and 16 players)
  ** Ensure you record the execution time vs number of players.
- Differing contention (low, medium, high).
  To simulate different contentions, you should use different "think time" in ms for the players as well as "work time" in iterations to simulate "work" in the chest when a player is taking coins.

Present the results in your report as follows:

- Explain how you simulated your low, medium and high contention.
- Graph showing the execution time vs number of players for each lock.
- Discuss the differences in execution time vs number of players for each lock and why you think this occurs.
- Discuss the time difference for each lock in different contention and why you think this occurs.
- Discuss which lock is the fairest and why.
- Discuss which lock scales the best as the number of players increase.

4 Marking

The marking for this practical will work as follows:

- You will implement the tasks and upload your solution to Fitchfork and Clickup.
- You will receive marks as follows:
  - Correctness of implementation (20%) – From Fitchfork
  - Ability to configure experiments (40%)
  - Quality of results presentation (20%)

- Depth of analysis/discussion (10%)
- Clarity of writing and structure (10%)

5 Upload Checklist

The following files should be in the root of your archive.

- Main.java (will be overridden by Fitchfork)
- Player.java
- TreasureChest.java
- Lock.java
- TTASLock.java
- CLHLock.java
- Report.pdf (Only required on Clickup)

6 Submission

You need to submit your source files on Fitchfork and Clickup. All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 9 standard.

For this practical, you will have 5 upload opportunities on Clickup and 20 upload opportunities on Fitchfork. On fitchfork, your best mark will be your final mark and your last attempt on Clickup will be your final. Upload your archive to the appropriate slot on the Fitchfork and Clickup website well before the deadline. **No late submissions will be accepted!**