UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS226 - Concurrent Systems

Practical 3 Specification

Release Date: 11/08/2025

Due Date: 18/08/2025 at 23:59

Total Marks: 60

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 8
- 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 Task (60)

For this task you are required to implement a Boba.java, Straw.java and a main program where you can do testing. For this task, only one boba can fit in the straw at any given time but any number of boba can be in the cup. The boba wait patiently at the bottom of the cup in rows

leading to the straw. Assuming the drinker does not backwash into the cup, write the program to ensure only one boba enters the straw at a time.

When a boba acquires the **Filter lock**, it should output “Boba x waiting patiently to be sipped” with x being the number of the thread. Once it has gained access then it should output “Boba x is being sipped up the straw” before sleeping for a random number of seconds. This is to emulate the sipping and should be done using the code provided. The boba will need to be in the straw for three sips as this is how long it takes to reach the end of the straw. Once it has reached the end of the straw it should output “Boba x is being consumed”.

Each boba will sleep for different times in between trying to get sipped up which must be simulated in your code.

From here you will notice that the straw constantly gets fed with boba but there is an issue with how the boba move through the straw (check your output to determine what the problem is).

To fix the problem, you will need to make a second lock called the **Improved_Filter** lock which can then be passed to the straw using the “V2” lock type.

Your system will be correct when it is deadlock and starvation free and there is mutual exclusion in the access to the straw.

4 Marking

The marking for this practical will work as follows:

- You will implement the tasks and upload your solution to Fitchfork.
- The test classes will be overridden by FitchFork and will then be used to mark the output that is produced by your code.
- You will receive a mark on Fitchfork.
- A mark on Fitchfork of greater than 0 will allow you to go to a practical session to be marked by a tutor.
- In the practical session a tutor will ensure that your implementation has kept within the restrictions (i.e no copy-pasting, not using libraries to do the whole task). The tutor will then assess your understanding of your implementation and the practical content.
- You will receive a final mark which will be some weighted combination of your Fitchfork mark and your "understanding" mark.

5 Upload Checklist

The following files should be in the root of your archive

- Main.java will be overridden

- Straw.java
- Boba.java
- Filter.java
- Improved_Filter.java

6 Submission

You need to submit your source files on the FitchFork website (https://ff.cs.up.ac.za/). 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 8 standard.

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