Homework 1
Due Date: 11/10/2023 11.59PM

Students

We're using GitHub Classroom, which basically means you do all your work in a pre-prepared GitHub repository.
The history of the repository must show contributions from all.

Instructions

Modify the provided dining philosophers code from the textbook so that it includes a more complete `main` method.
This should allow your program to be executed from the command line with five arguments:

`java driver` *np nc tt et rl*

np, *nc*, *tt*, *et*, and *rl* are five non-negative integers that represent the following:

- *np* is the number of philosophers (and forks).
- *nc* is the number of cycles that each philosopher will go through. If this parameter is 0, they run forever.
- *tt* is the maximum thinking time, in units of milliseconds.
- *et* is the maximum eating time, in units of milliseconds.
- If *rl* is 0, then all philosophers are right-handed and try to grab their right chopstick first. If *rl* is 1, then even-numbered philosophers are right-handed and odd-numbered philosophers are left-handed.

Obviously you will also have to modify the philosopher class to implement these changes.
Add print statements to trace the flow of execution.
A philosopher should go through the following cycle - I have included sample id numbers and times:

```
    Philosopher 0 thinks for 123 units
    Philosopher 0 wants right chopstick
    Philosopher 0 has right chopstick
    Philosopher 0 wants left chopstick
    Philosopher 0 has left chopstick
    Philosopher 0 eats for 456 units
    Philosopher 0 releases left chopstick
    Philosopher 0 releases right chopstick
```

You can extend this format if you want to include other useful information, such as the specific chopstick id, the handedness of a philosopher, or anything else that you think is interesting.

Execution

Run your completed program at least three times with different sets of parameters, to show different interesting behaviors.
Capture all of the output, and create one long file called `Trace.txt` to show your results.
Separate the different traces in `Trace.txt` to make them easy to distinguish, and add comments to the ends of lines that illustrate interesting states of the system.

In particular, you should have at least one trace with all right-handed philosophers that results in a deadlock.
`java Driver 2 0 0 0 0` may be a good set of parameters to obtain this.

If you have difficulty achieving deadlock, you can try adding `Thread.yield()` statements after a philosopher acquires a chopstick. However, I did not find this necessary with either replit.com or a local JDK.

For any given deadlock, you should only show the part of the trace from the first chopstick acquisition that causes the deadlock to the deadlock itself.
Do not include pages and pages of uninteresting trace output.

Rubric

2/3 of the points will be based on whether you successfully followed the instructions to arrive at correct code, that generates an interesting set of traces, and everything is present in your GitHub repository.

1/3 of the points will be based on the formatting, clarity, and appropriateness of your code.
I am not holding you to a specific code style guide, nor am I demanding professional-level code quality, but to get this 1/3 of your points your code must be clear, well-formatted, and have made appropriate design decisions.