Dining Philosophers Problem

Sashank Silwal

ss13163

Assignment 6

Note: Ensure that ZooKeeper is running and accessible at 'localhost:2181' before executing the code.

Execution

```
zkServer start
python3 main.py
```

Description:

This code demonstrates a solution to the Dining Philosophers Problem using the Kazoo library for distributed coordination and synchronization.

Classes:

- Philosopher: Represents a philosopher and their behavior. Each philosopher thinks for a random amount of time, tries to acquire two forks (one on the left and one on the right), eats for a random amount of time, and then releases the forks.
    - Methods:
        - **init**(philosopher_id, zk_client): Initializes a Philosopher object with the given philosopher ID and ZooKeeper client instance.

- think(): Represents the thinking behavior of the philosopher, where they randomly sleep for a certain amount of time.
- eat(): Represents the eating behavior of the philosopher, where they randomly sleep for a certain amount of time.
- run(): Runs the main loop for the philosopher, alternating between thinking, acquiring forks, eating, and releasing forks.
- acquire_forks(): Acquires locks for the left and right forks from ZooKeeper. Returns True if locks are acquired successfully, False otherwise.
- release_forks(): Releases the locks for the left and right forks from ZooKeeper.

Functions:

- main(): The main function that sets up the environment, creates philosopher objects, starts philosopher threads, and waits for them to complete.

Control Flow

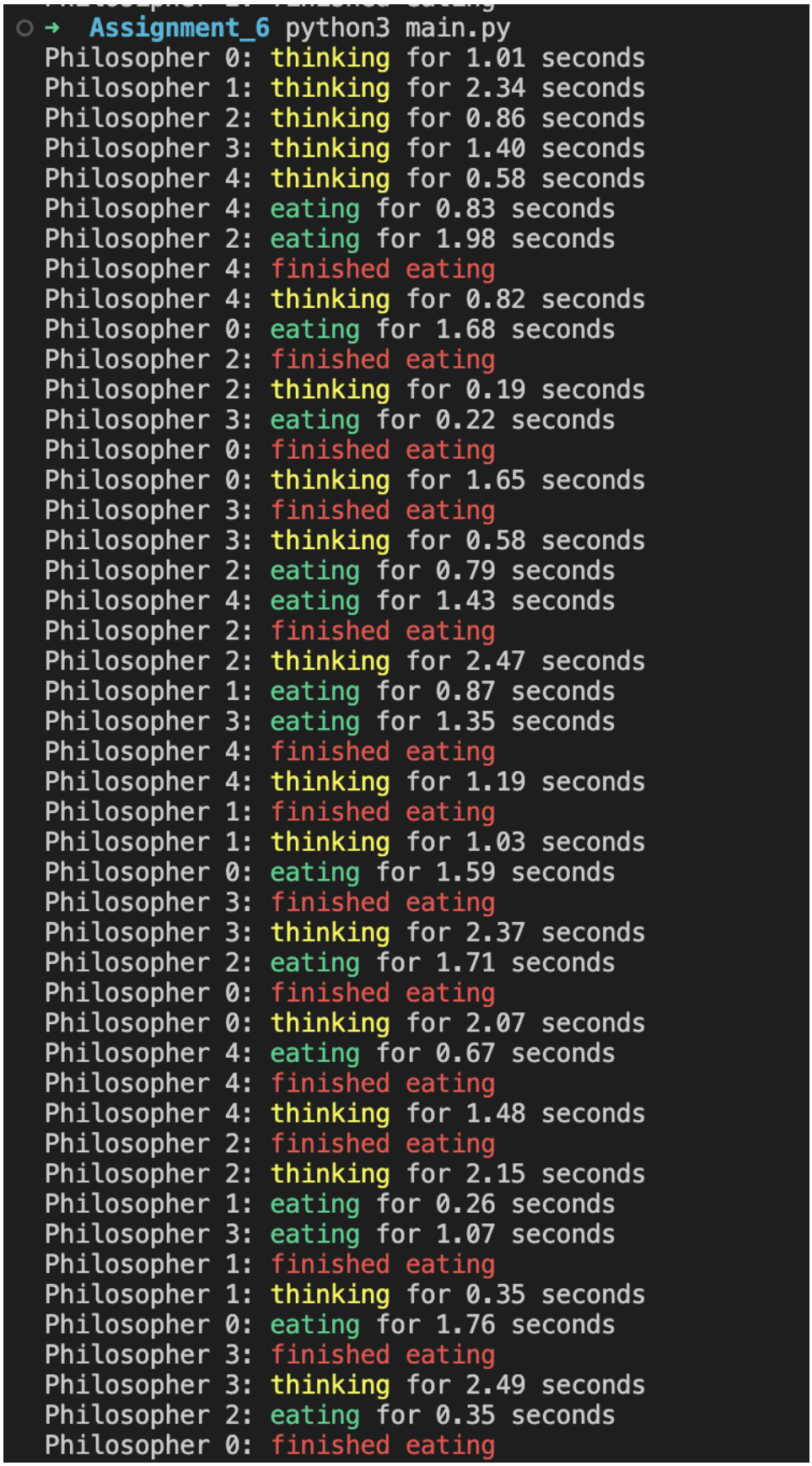
a. Setup Control Flow

1. A ZooKeeper client (`zk_client`) is created and started by connecting to the ZooKeeper server.
2. The program ensures that the "/fork_resource" path exists in ZooKeeper. If it doesn't exist, it creates it.
3. Any existing forks (children of "/fork_resource") are deleted.
4. A list of philosophers is created, where each philosopher is initialized with an ID and the ZooKeeper client.
5. For each philosopher, a philosopher thread is created. Each thread's target is set to the philosopher's `run` method.
6. The philosopher threads are started.

b. Philosophers Class control flow

1. The philosopher enters a loop that runs for the maximum number of meals.
2. The philosopher thinks by calling the `think` method, which simulates thinking for a random time.
3. The philosopher enters a loop to acquire forks by calling the `acquire_forks` method. This loop continues until the philosopher successfully acquires both forks.
4. Inside the `acquire_forks` method, the philosopher creates locks for the left and right forks by creating znodes in ZooKeeper. If any exception occurs during the creation of locks, the created locks are deleted, and the method returns `False`. Otherwise, it returns `True` to indicate successful acquisition of the locks.
5. If the philosopher successfully acquires the forks, it proceeds to eat by calling the `eat` method, which simulates eating for a random time.
6. After eating, the philosopher releases the forks by calling the `release_forks` method, which deletes the respective fork znodes from ZooKeeper.
7. The philosopher prints a message indicating that it has finished eating.
8. The loop continues until the maximum number of meals is reached.
9. Once the philosopher's thread completes the loop, the thread exits.

Output


➜ Assignment_6 python3 main.py
Philosopher 0: thinking for 1.01 seconds
Philosopher 1: thinking for 2.34 seconds
Philosopher 2: thinking for 0.86 seconds
Philosopher 3: thinking for 1.40 seconds
Philosopher 4: thinking for 0.58 seconds
Philosopher 4: eating for 0.83 seconds
Philosopher 2: eating for 1.98 seconds
Philosopher 4: finished eating
Philosopher 4: thinking for 0.82 seconds
Philosopher 0: eating for 1.68 seconds
Philosopher 2: finished eating
Philosopher 2: thinking for 0.19 seconds
Philosopher 3: eating for 0.22 seconds
Philosopher 0: finished eating
Philosopher 0: thinking for 1.65 seconds
Philosopher 3: finished eating
Philosopher 3: thinking for 0.58 seconds
Philosopher 2: eating for 0.79 seconds
Philosopher 4: eating for 1.43 seconds
Philosopher 2: finished eating
Philosopher 2: thinking for 2.47 seconds
Philosopher 1: eating for 0.87 seconds
Philosopher 3: eating for 1.35 seconds
Philosopher 4: finished eating
Philosopher 4: thinking for 1.19 seconds
Philosopher 1: finished eating
Philosopher 1: thinking for 1.03 seconds
Philosopher 0: eating for 1.59 seconds
Philosopher 3: finished eating
Philosopher 3: thinking for 2.37 seconds
Philosopher 2: eating for 1.71 seconds
Philosopher 0: finished eating
Philosopher 0: thinking for 2.07 seconds
Philosopher 4: eating for 0.67 seconds
Philosopher 4: finished eating
Philosopher 4: thinking for 1.48 seconds
Philosopher 2: finished eating
Philosopher 2: thinking for 2.15 seconds
Philosopher 1: eating for 0.26 seconds
Philosopher 3: eating for 1.07 seconds
Philosopher 1: finished eating
Philosopher 1: thinking for 0.35 seconds
Philosopher 0: eating for 1.76 seconds
Philosopher 3: finished eating
Philosopher 3: thinking for 2.49 seconds
Philosopher 2: eating for 0.35 seconds
Philosopher 0: finished eating