Concurrency

Franck van Breugel

March 18, 2018

1 The dining philosophers problem

In the dining philosophers problem, due to Dijkstra, five philosophers are seated around a round table. Each philosopher has a plate of spaghetti. A philosopher needs two forks to eat it. The layout of the table is as follows.

The life of a philosopher consists of alternative periods of eating and thinking. When philosophers get hungry, they try to pick up their left and right fork, one at a time, in either order. If successful in picking up both forks, the philosopher eats for a while, then puts down the forks and continues to think.

```
public class Philosopher extends Thread {
  private int id;
  private Table table;

  public Philosopher(int id, Table table) {
    this.id = id;
    this.table = table;
  }
```

```
  public void run() {
    while (true) {
      this.table.pickUp(id);
      this.table.pickUp((id + 1) % 5);
      // eat
     this.table.putDown(id);
     this.table.putDown((id + 1) % 5);
  }
}


public class Table {
  ...
  public Table() { ... }
  public void pickUp(int id) { ... }
  public void putDown(int id) { ... }
}


public class Philosophers {
  public static void main(String[] args) {
    Table table = new Table();
    for (int p = 0; p < 5; p++) {
      (new Philosopher(p, table)).start();
    }
  }
}
```

1. Of what information about the table and its forks should we keep track?

2. How do we represent this information?

3. Where and how do we initialize the attribute?

4. Implement the method **`pickUp(int id)`**.

   - When does a **`Philosopher`** have to wait?
   - How does the array **`pickedUp`** need to be updated?

```
public synchronized void pickUp(int id) {




}
```

5. Implement the method **`putDown(int id)`**.

   - How does the array **`pickedUp`** need to be updated?
   - Do **`Philosopher`**s need to be notified?

```
public synchronized void putDown(int id) {




}
```