Dining Philosophers

A Java Implementation

Definitions

- Starvation - a condition where a thread is unable to obtain regular access to a shared resource. The thread is blocked and unable to make progress.

- Livelock - a condition where a thread is not blocked but is not making progress. Sometimes occurs when two threads are busy responding to the other thread.

Fork.java (1/3)

```
package hw2;

/**
 * A Fork object. A fork can alternate between two states: being picked up
 * and being put down. If it is currently picked up, it must be put down
 * before it can be picked up again. Similarly, if a fork is put down, it
 * must be picked up before it can be put down again.
 *
 * @author david
 */
public class Fork {
     /** id for this fork */
     private int id;

     /** flag to indicate if this fork can be picked up */
     private boolean availableFlag = true;

     /** part of an exception message */
     private final String pickUpMsg = " is not available";

     /** part of an exception message */
     private final String putDownMsg = " is not being held";

     /**
      * Creates a Fork with a specified id.
      *
      * @param id the specified id
      */
     public Fork(int id) {
          this.id = id;
     }
```

Fork.java (2/3)

```
/**
 * Tests whether this Fork is available.
 *
 * @return true if this Fork is available
 */
public boolean isAvailable() {
      return availableFlag;
}

/**
 * Picks up this Fork.
 *
 * @throws IllegalStateException if the Fork is not available
 */
public void pickUp() {
      if (!availableFlag) {
            throw new IllegalStateException(toString() + pickUpMsg);
      }
      availableFlag = false;
}

/**
 * Puts down (releases) this Fork.
 *
 * @throws IllegalStateException if the Fork is available
 */
public void putDown() {
      if (availableFlag) {
            throw new IllegalStateException(toString() + putDownMsg);
      }
      availableFlag = true;
}
```

Fork.java (3/3)

```
    /*
     * (non-Javadoc)
     * @see java.lang.Object#toString()
     */
    @Override
    public String toString() {
        return "Fork " + id;
    }
}
```

Main.java

```
// instantiate 5 forks

Object waiter = new Object();

Philosopher[] phil = new Philosopher[5];
phil[0] = new Philosopher(names[0], waiter, forks[0], forks[4]);
for (int i = 1; i < 5; i++) {
      phil[i] = new Philosopher(names[i], waiter, forks[i], forks[i-1]);
}

Thread[] th = new Thread[5];
for (int i = 0; i < 5; i++) {
      th[i] = new Thread(phil[i]);
      System.out.println("Philosopher Id: " + phil[i].getId());
}

for (int i = 0; i < 5; i++) {
      th[i].start();
}

try {
    Thread.sleep(sleepTime);

     for (int i = 0; i < 5; i++) {
        phil[i].setStopFlag(true);
        th[i].join();
    }
} catch (InterruptedException ie) {
    ie.printStackTrace();
}

// print data from each philosopher
```

Philosopher.java (1/2)

```
@Override
public void run() {
    long startTotalTime = System.nanoTime();
    boolean haveForks = false;
    while (!isStopping()) {
        // think

        // get forks                    << critical section

        // eat

        // put down forks
    }
    estTotalTime = System.nanoTime() - startTotalTime;
}
```

Philosopher.java (2/2)

```
public class Philosopher implements Runnable {
    …
    boolean stopFlag = false;

    …
    boolean isStopping() {
    …
    }

    void setStopFlag() {
    …
    }
}
```

// think

```
thoughts++;
long thinkTime = (long)(random.nextFloat() * 10.0);
try {
    Thread.sleep(thinkTime);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

Output

```
Davids-Air:cs364-2019 david$ java -cp build hw2.Main 2000
Welcome to the diner
Philosopher Id: Plato
Philosopher Id: Socrates
Philosopher Id: Kant
Philosopher Id: Confucius
Philosopher Id: Hadot

The diner is closed.
Philosopher: Plato
    Thoughts: 288,  Meals: 288
    Fork access time (ns): 1150314265
    Total time (ns): 4010230326
    Ratio -- Fork access time / Total time: 0.2868
Philosopher: Socrates
    Thoughts: 301,  Meals: 301
    Fork access time (ns): 1014381878
    Total time (ns): 4015142104
    Ratio -- Fork access time / Total time: 0.2526
Philosopher: Kant
    Thoughts: 294,  Meals: 294
    Fork access time (ns): 1044022170
    Total time (ns): 4017603354
    Ratio -- Fork access time / Total time: 0.2599
Philosopher: Confucius
    Thoughts: 286,  Meals: 286
    Fork access time (ns): 1108982649
    Total time (ns): 4020899733
    Ratio -- Fork access time / Total time: 0.2758
Philosopher: Hadot
    Thoughts: 276,  Meals: 276
    Fork access time (ns): 1178863007
    Total time (ns): 4034216899
    Ratio -- Fork access time / Total time: 0.2922
Davids-Air:cs364-2019 david$
```