Exclusion mutuelle

```java
class LockOne implements Lock {
  private boolean[] flag = new boolean[2];
  // thread-local index, 0 or 1
  public void lock() {
    int i = ThreadID.get();
    int j = 1 - i;
    flag[i] = true;
    while (flag[j]) {}          // wait
  }
  public void unlock() {
    int i = ThreadID.get();
    flag[i] = false;
  }
}
```

```java
class LockTwo implements Lock {
  private int victim;
  public void lock() {
    int i = ThreadID.get();
    victim = i;                 // let the other
    while (victim == i) {}      // wait
  }
  public void unlock() {}
}
```

```java
class Peterson implements Lock {
  // thread-local index, 0 or 1
  private boolean[] flag = new boolean[2];
  private int victim;
  public void lock() {
    int i = ThreadID.get();
    int j = 1 - i;
    flag[i] = true;               // I'm interested
    victim = i;                   // you go first
    while (flag[j] && victim == i) {}; // wait
  }
  public void unlock() {
    int i = ThreadID.get();
    flag[i] = false;              // I'm not interested
  }
}
```

```
class Filter implements Lock {
  int[] level;
  int[] victim;
  public Filter(int n) {
    level  = new int[n];
    victim = new int[n]; // use 1..n-1
    for (int i = 0; i < n; i++) {
       level[i] = 0;
    }
  }
  public void lock() {
    int me = ThreadID.get();
    for (int i = 1; i < n; i++) { // attempt level i
      level[me] = i;
      victim[i] = me;
      // spin while conflicts exist
      while ((∃k != me) (level[k] >= i && victim[i] == me)) {};
    }
  }
  public void unlock() {
    int me = ThreadID.get();
    level[me] = 0;
  }
}
```

```
class Bakery implements Lock {
  boolean[] flag;
  Label[] label;
  public Bakery (int n) {
    flag  = new boolean[n];
    label = new Label[n];
    for (int i = 0; i < n; i++) {
       flag[i] = false; label[i] = 0;
    }
  }
  public void lock() {
    int i = ThreadID.get();
    flag[i] = true;
    label[i] = max(label[0], ...,label[n-1]) + 1;
    while ((∃k != i)(flag[k] && (label[k],k) << (label[i],i))) {};
  }
  public void unlock() {
    flag[ThreadID.get()] = false;
  }
}
```