Assignments for week 12

Anders Degn Lapiki, Jacob Kjærulff Furberg, Jonas Ishøj Nielsen

19/11/2020

Contents

1 12.1 - green

1.1 12.1.1 - green

See function `testBankParallel` in file `PicoBankTest`.
The test shows the class is not thread safe, by starting x threads that, once the threads are ready they can open the barrier, and do 10.000/x bank transfers. The sum is guaranteed to only be computed once the cyclicBarrier once more has enough threads waiting including the main thread. This test achieves race conditions on only 2 simultaneous threads, and always succeeds on a single test.

1.2 12.1.2 - green

See class `PicoBankSynchronised` in file `PicoBankTest`.

The only method in the `PicoBankSynchronized` that needs to be synchronized is the `transfer` method. Balance doesn't need synchronization as it is only used sequentially and the calling thread doesn't have account in cache before that.

1.3 12.1.3 - green

Run each test in isolation, meaning with other tests commented out.
12.1.2 is slowest, 12.1.1 is fastest as expected.

1.4 12.1.4 - green

See class `PicoBankLocks` in file `PicoBankTest`.

1.5 12.1.5 - green

See class `PicoBankAtomic` in file `PicoBankTest`.
Using AtomicLong is a valid solution to prevent race conditions, since current balance never changes workflow.
It would fail in cases where: if ballance $>$ 0: transfer 200 else transfer -200.

1.6 12.1.6 - Yellow

See class `PicoBankAtomicAlt` in file `PicoBankTest`.

We created an implementation with only get and then compareAndSet, this always worked. Even though it seemed like it shouldn't (what if the value changed between the operations).

Therefore we added a do while, to get the value again if it was changed between the get and the compareAndSet.

2 12.2

2.1 12.2.1 - green

Only tests for sequential correctness.Problems:

```
Only test one scenario(no property based testing), which leads to low coverage.
  Not easy to read and understand
```

Fix:
See function `testAllMapsGood` in `TestStripedMap.java`.

```
Put adds elements
Remove removes elements
PutIfAbsent doesn't overwrite
```

2.2 12.2.2 - yellow

See function `testAllMapsConcurrent` in `TestStripedMap.java`.
StripedMap succeeds all tests.
StripedWriteMap fails on put succeeds on rest.

2.3 12.2.3 - yellow

See function `testAllMapsConcurrent` in `TestStripedMap.java`.
WrapConcurrentHashMap succeeds all tests.

2.4 12.2.4 - red

See function `testAllMapsConcurrent` in `TestStripedMap.java`.

2.5 12.2.5 - red

See function `testMapCounting` in `TestStripedMap.java`.

Assume error happens because of:

```
Thread1: contains(v) == true
Thread2: contains(v) == true
Thread1: remove(v)
Thread2: remove(v)      // returns u
```

Could get it to work by synchronizing on ourmap, but that would remove concurrency test.

2.6 12.2.6 - red

We appear to have tested adequately. Perhaps there could be a concurrent test for making sure the reallocation of buckets functions correctly.

3 12.3

3.1 12.3.1 - red

The test did not discover the lack of synchronization when removed from both `containsKey` and `get`. When removed from `put` and `remove` the tests found the lack of synchronization.

3.2 12.3.2 - red

It fails sometimes on `remove` and `put`, but succeeds on rest.

3.3 12.3.3 - red

Where we supposed to test this for StripedWriteMap?

3.4 12.3.4 - red

Where we supposed to test this for StripedWriteMap?