Concurrent Object Oriented Languages

Testing Concurrent Code

wiki.eecs.yorku.ca/course/6490A

Testing sequential code

If possible, first test your code in the same way you test sequential code.

Testing sequential code

Problem

Test the class

```
public class RedBlackTree<T> {
  public RedBlackTree() { ... }
  public boolean contains(T key) { ... }
  public boolean add(T key) { ... }
}
```

Testing sequential code

Question

What “input” does the constructor need?

Answer

Nothing.

Testing sequential code

Question

What is the “input” for the methods `contains` and `add`?

Answer

A red-black tree and an element.

Testing sequential code

Question

Which red-black tree and which element?

Answer

A random red-black tree and a random element.

Testing sequential code

Question

How do we create a random red-black tree?

Answer

By adding random elements, starting from an empty red-black tree.

Testing sequential code

```
RedBlackTree<Integer> tree =
  new RedBlackTree<Integer>();
for (...) {
   int element = random integer;
   tree.add(element);
}
```

Testing sequential code

Question

How do we test the `contains` method?

Answer

Compare it against the `contains` method of the `HashSet` class.

Testing sequential code

Question

How do we test the `add` method?

Answer

Compare it against the `add` method of the `HashSet` class.

Testing sequential code

Question

Is there anything else we want to check?

Answer

We may want to check that the tree is a red-black tree.

Testing sequential code

To the class `RedBlackTree` add the method

```
/**
 * ...
 * @throws RuntimeException if this tree
 * is not a red-black tree.
 */
public void isValid() throws RuntimeException
{
  ...
}
```

Testing sequential code

Problem

Check that each test terminates.

Testing concurrent code

Facts

- Writing concurrent code is harder than sequential code.
- Anything that can go wrong in sequential code can also go wrong in concurrent code.
- But concurrent code may also contains deadlocks, livelocks, data races, etc.

Testing concurrent code

Question

Why is testing concurrent code more difficult than testing sequential code?

Answer

- The test suite consists of concurrent code (which is more difficult to write than sequential code).
- The failures in concurrent code are nondeterministic (as a consequence, reproducing failures can be maddingly difficult).

Testing

Caution

Test code can introduce timing or synchronization artifacts that can mask bugs.

Bugs that disappear when you add test code are sometimes called Heisenbugs. The term is a pun on the name of Werner Heisenberg, the physicist who first asserted the observer effect of quantum mechanics, which states that the act of observing a system inevitably alters its state.

source: German Federal Archives

Randomization

Tests should be random so that the compiler cannot precompute the results.

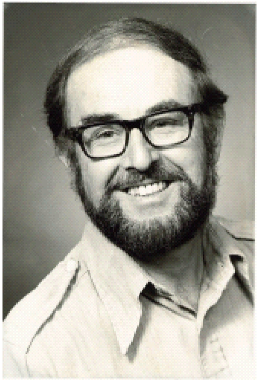
Random number generators can create couplings between classes and timing artifacts, because most random number generator classes are thread safe.

Random number generator

```
static int xorShift(int random)
{
    random ^= (random << 6);
    random ^= (random >>> 21);
    random ^= (random << 7);
    return random;
}
```

George Marsaglia

- American mathematician and computer scientist.
- Professor at Washington State University and Florida State University.
- Known for developing some of the most commonly used methods for generating random numbers.



George Marsaglia

(1924–2011)

source: Journal of Modern Applied Statistical Methods

Testing concurrent code

Factors to consider when testing concurrent programs

- Run tests more than once since they are nondeterministic.
- Explore different interleavings:
  - Ensure that all threads start at the “same time.” Use for example a CyclicBarrier.
  - Ensure that each thread runs “long enough.”
- Match the number of threads to the platform:
  - as many threads as cores, thereby reducing potential interactions between threads, and
  - many more threads than cores, thereby reducing the potential interactions between threads.