CSE 331: Testing

How can we *know* that a method satisfies its spec?

- We can think hard about it and try to convince ourselves. (*Reasoning*)
- We can ask the Java compiler what it thinks. (*Type checking*)
- We can try it on examples. (*Testing*)

All three are useful for different reasons. For now, we focus on testing. There are various forms of testing:

- *Unit tests* test a small piece of the program, usually a single method.
- *Integration tests* test larger pieces of the program, perhaps the way two modules work together.
- *End-to-end tests* run the whole program.

We'll focus on unit testing for now. Steps of writing a test:

1. Come up with some input.
2. Figure out the expected behavior on that input.
3. Run the method on the input and observe its behavior.
4. Compare the observed behavior to the expected behavior. The test passes if they match and fails otherwise.

How to come up with good inputs? Using heuristics.

- *Opaque (black-box) testing*: look at the specification and try to exercise all described behaviors.
- *Transparent (clear-box) testing*: look at the code and try to exercise all paths.
- *Boundary cases*: look for inputs that are on the boundary between described behaviors, or corner cases like `null`.
- *Exhaustive testing*: if there are only a few inputs, test them all.

How to figure out the expected behavior for an input?

- *Specification-based testing*: Look at the spec. Only test for behavior guaranteed by the spec.
  - A specification-based test should pass on *any* correct implementation of the spec.
- *Implementation-based testing*: Look at the code. Test for behavior that the code appears to provide, even if it is not guaranteed by the spec.
  - An implementation-based test may fail on other implementations if they have different behavior.

```
/**
 * @requires 0 <= lo <= hi <= a.length
 * @returns *some* index i with lo <= i < hi and a[i] == x,
 *     or -1 if no such index exists
 */
public static int indexOf(int[] a, int lo, int hi, int x) { ... }

// Usually in a different file:
@Test
public void testIndexOf() {
  int[] a = {3, 0, 7};
  assertEquals(2, indexOf(-1, 0, 3, 7));
  assertEquals(-1, indexOf(-1, 0, 3, 42));
}
```