JUNIT

- A library for unit testing in java
- Create a class for testing (may have more than one depending on scope of project)
- Add a decorator `@Test` before each method that is a test
- Tests run in arbitrary order

ASSERTIONS

assertEquals(x, 4)
assertEquals("not 4", x, 4)

- tests typically return void
- asserts are ways of saying something has to be true (if not the test will fail)
- General Listing: `assertEquals`, `assertTrue`, `assertFalse`, `assertNull`, `assertNotNull`, `assertSame`, `assertNotSame`, `assertArrayEquals`, ...
- Really convoluted test: `fail()`
- Most asserts can either have message first or not

TESTS+EXCEPTIONS

- what happens when you want to make sure that an exception is thrown?

```
@Test(expected = IndexOutOfBoundsException.class)
public void shouldBeOOB() {
  int[] myints = new int[10];
  int oobint = myints[20];
}
```

- replace with the appropriate exception

```
@Test(expected = InvalidArgument...)
public void testConstructorInvalid() {
    Lift l = new Lift(102);
}
```

TEST+TIMEOUTS

- what happens if the test calls something that is infintely looping?
- how can you guarantee the test will fail?

```
@Test(timeout=5000)
public void loopForever() {
    int x = 0;
    while(x <= 0) {
        x--;
    }
}
```

OTHER ANNOTATIONS

- sometimes you need other methods that "setup" things or "tear down" before/after tests
- Decorators:
  - `@Before`: executed before every test
  - `@After`: executed after every test
  - `@BeforeClass`: executed once before start of all tests
  - `@AfterClass`: executed once after end of all tests
- disable a test: `@Ignore`