BLOCK NONCE #4: Don't make your tests too DRY

by guy.f@

What?

The DRY principle ("Don't Repeat Yourself") encourages code reuse rather than duplication. But is it also the best thing for tests?

Is it clear what this test does just from reading its code?

```
#[test]
fn verify_balance_on_transfer() {
  let mut account = create_default_account();  // Test helper method.
  make_transfers(&mut account);                // Test helper method.
  assert_eq!(1500, account.balance());
}
```

Why?

While the test body above is concise, the reader cannot understand what it does without context switching to another place. Since tests don't have tests, it should be easy for humans to manually inspect them for correctness, even at the expense of greater code duplication.

How?

Make your tests **DAMP**! (Descriptive and Meaningful Phrases).
In the context of unit tests, this test would be easier to read and verify for correctness:

```
#[test]
fn verify_balance_on_transfer() {
  const INITIAL_BALANCE : u32 = 1000;
  const TRANSFER_AMOUNT : u32 = 500;

  let mut new_account =
      Account::new(AccountType::NO_OUTGOING_TRANSFERS(INITIAL_BALANCE));
  let mut account_to_transfer_from =
      Account::new(AccountType::SUPPORTS_TRANSFERS(TRANSFER_AMOUNT));

  assert_eq!(INITIAL_BALANCE, new_account.balance());
  account_to_transfer_from.transfer_to(&mut new_account, TRANSFER_AMOUNT);
  assert_eq!(INITIAL_BALANCE + TRANSFER_AMOUNT, new_account.balance());
}
```

TIP💡 The DRY principle is still relevant in tests. Ideally, test code should be both readable and unique, but sometimes there's a trade-off. When writing unit tests and faced with a choice between the DRY and DAMP principles, lean more heavily toward DAMP.