Block By Block.

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

What?

The DRY principle ("Don't Repeat Yourself"), is a best practice that encourages code reuse rather than duplication, e.g., by extracting helper methods or by using loops. But is it also the best thing for tests?
Is it clear what this test does just from reading its code?

```
#[test]
fn verify_positive_balance_on_transfer() {
  let mut account = create_default_account();  // Test helper method.
  make_transfers(&mut account);                // Test helper method.
  assert!(account.has_positive_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_positive_balance_on_transfer() {
  let mut new_account = Account::new(AccountType::NO_OUTGOING_TRANSFERS);
  let mut account_with_balance = Account::new(AccountType::SUPPORTS_TRANSFERS);
  account_with_balance.deposit(50);

  assert!(!new_account.has_positive_balance());
  account_with_balance.transfer_to(&mut new_account, 50);
  assert!(new_account.has_positive_balance());
}
```

TIP💡 The DRY principle is still relevant in tests; for example, using a helper function for creating value objects can increase clarity by removing redundant details from the test body. 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.