Testing

1) Consumer
2) Supplier
3) Predicate
4) Function
5) Runnable

1. Unit Testing
2. Integration Testing
3. Component Testing
4. Contract Testing
5. End-End Testing
6. Blackbox Testing
7. Whitebox Testing
8. Performance Testing
9. Security Testing

Three things for the Test:

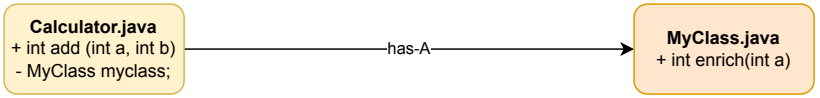
1. Fixture (Precondition/Given)
2. What are we testing?
3. What are the changes that took place?

```
public int add(int a, int b) {
  return a + b;
}
```

Unit Testing

1. Sociable Unit Test
2. Solitary Unit Test

Inheritance -> is-A
Composition -> has-A



Class Under Test

CalculatorTest.java

```
public class Calculator {
private MyClass myClass; // has-A

      public int add(int a, int b) {

      int c = myClass.enrich(a);
      int d = myClass.enrich(b);
        return c + d;
      }

}
```

Characteristics of Tests:

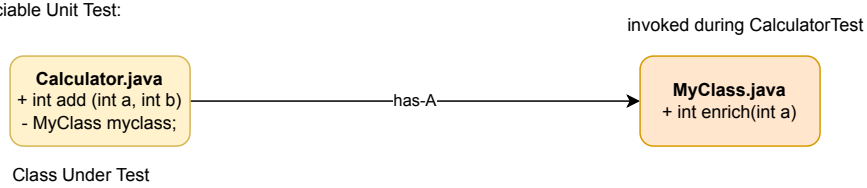
1. Automated
2. Repeatable
3. Fast
4. Easy To Run
5. Complete

Unit Testing

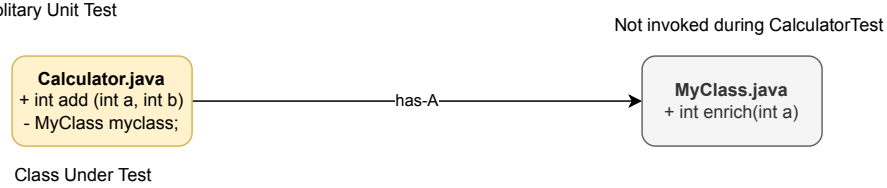
1. Sociable Unit Test
2. Solitary Unit Test

```
public class Calculator {
private MyClass myClass; // has-A

      public int add(int a, int b) {

      int c = myClass.enrich(a);
      int d = myClass.enrich(b);
        return c + d;
      }

}
```

1. Sociable Unit Test:



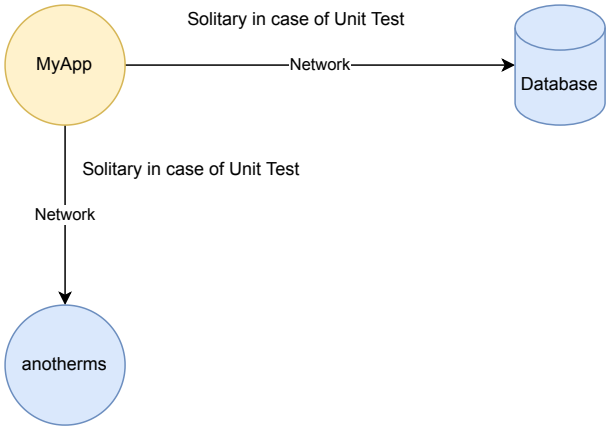
2. Solitary Unit Test



CalculatorTest.java

```
public class CalculatorTest {

@Test
void sociableTest() {
      Calculator calc = new Calculator(new MyClass());
      calc.add(2,3);
}

@Test
void solitaryTest() {
      MyClass myClossMock = Mockito.mock(MyClass.class);
      Calculator calc = new Calculator(myClassMock);
      calc.add(2,3);
}


}
```

Integration Testing

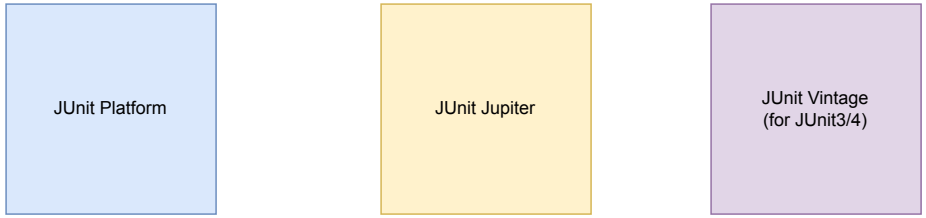


JUnit 5 Architecture

Logical Separation of Concerns:

1. An API to write tests for developers
2. A mechanism to discover and run tests
3. An API to allow easy interaction with IDEs and tools and to run tests from them

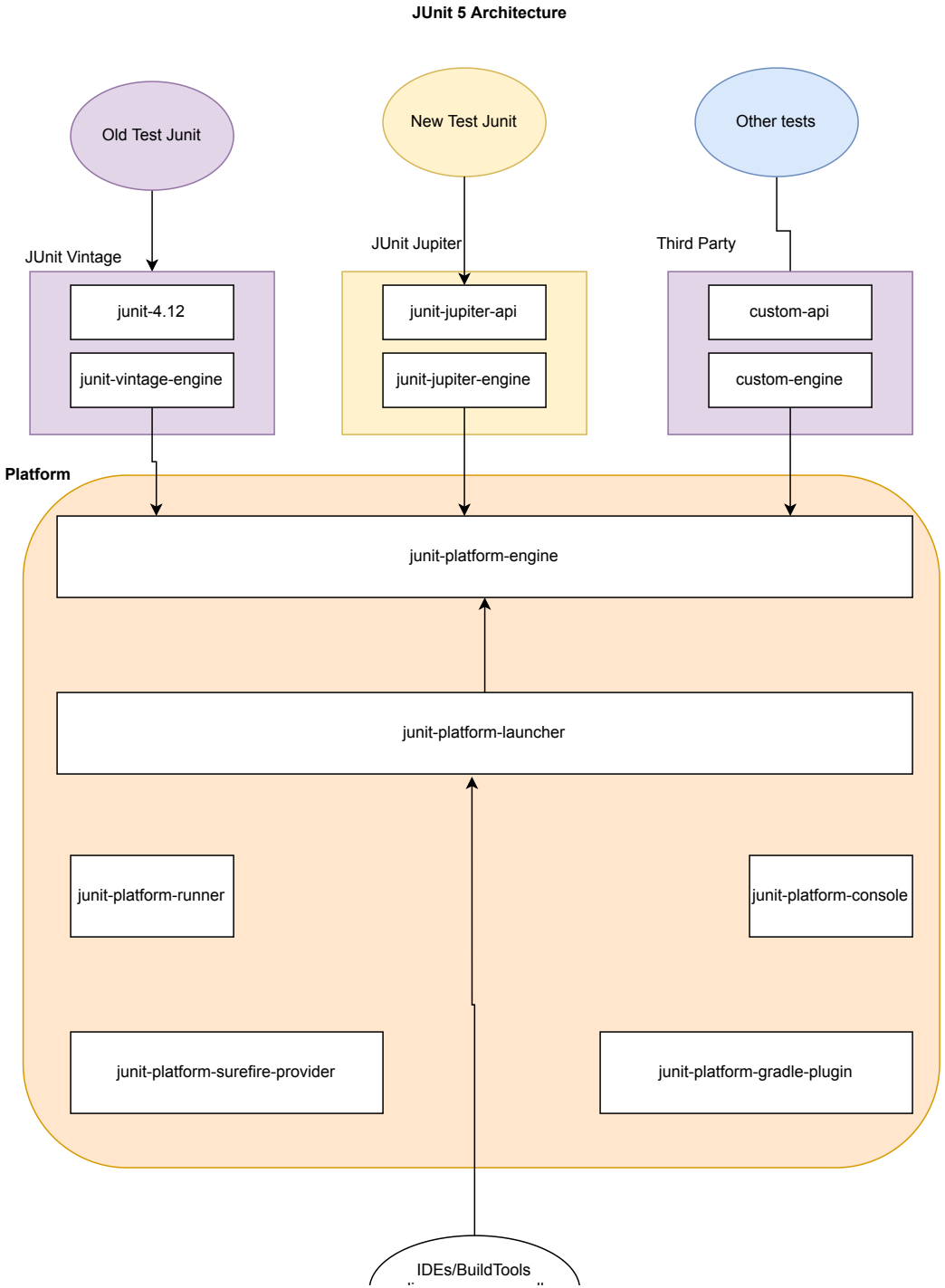
Three modules of JUnit 5:

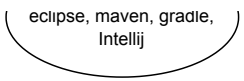


- junit-platform-commons
- junit-platform-console
- junit-platform-engine
- junit-platform-launcher
- junit-platform-surefire-provider
- junit-platform-gradle-plugi

- junit-jupiter-api
- junit-jupiter-engine
- junit-jupiter-params
- junit-jupiter-migrationsupport

- junit-vintage-engine

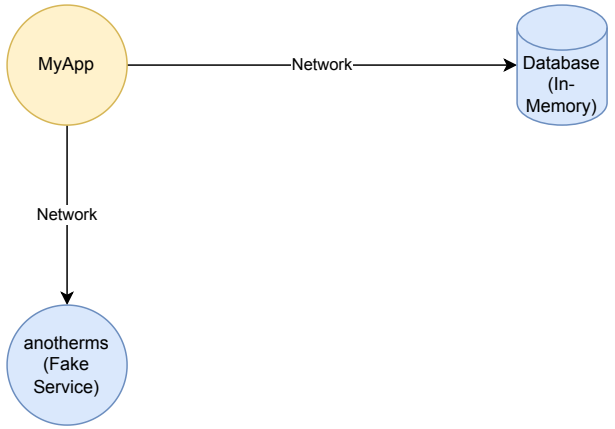

JUnit 5 Architecture
Old Test Junit
New Test Junit
Other tests
JUnit Vintage
JUnit Jupiter
Third Party
junit-4.12
junit-vintage-engine
junit-jupiter-api
junit-jupiter-engine
custom-api
custom-engine
Platform
junit-platform-engine
junit-platform-launcher
junit-platform-runner
junit-platform-console
junit-platform-surefire-provider
junit-platform-gradle-plugin
IDEs/BuildTools


eclipse, maven, gradle,
Intellij

Test Double

1. Dummy: dummy object, not very important, keeps the compiler happy
2. Fake: Working Implementation (in-memory Database, Fake Web Service)
3. Stub: Ready-made(canned) answer to method calls(only what is required for a test)
4. Spy: Partial Mock
5. Mock: Tests the interaction with the object. May stub or return values. Focus is on verification without making actual call to the mocked calss

Fake



Stub