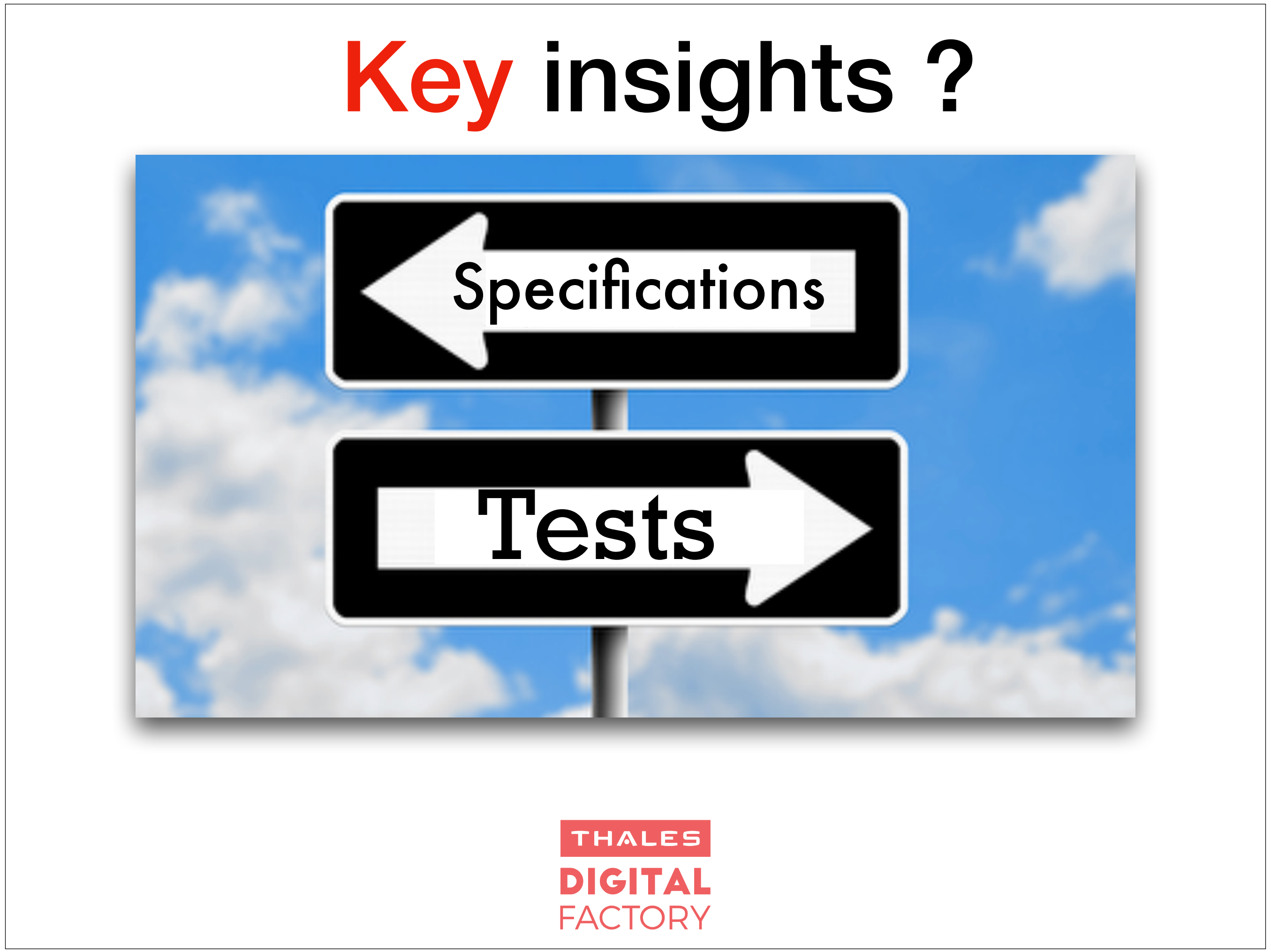
Property Based Testing

Hint: It's not unit-testing

THALES
DIGITAL
FACTORY

What is this about?

Property-based testing allows you to write properties that describe your code’s behaviour, and leave the task of test case generation and property evaluation to the testing tool

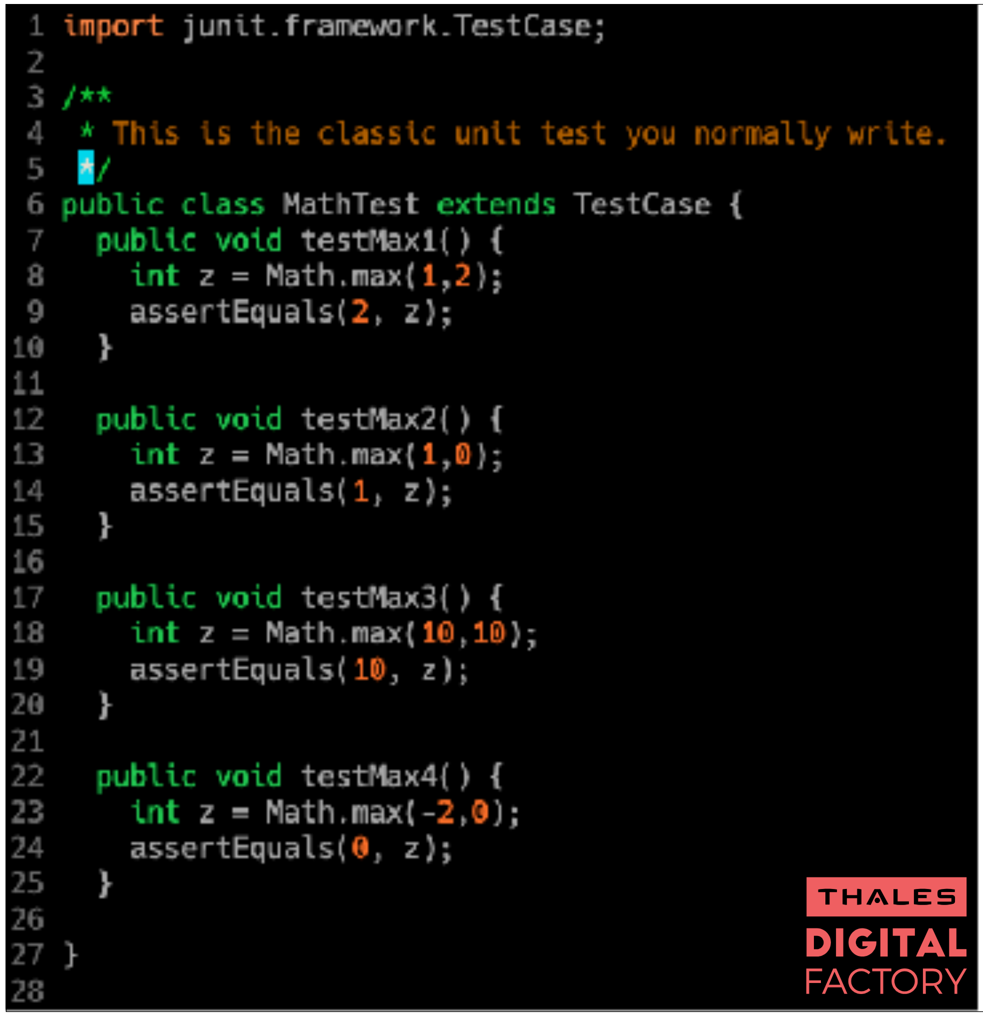


Key insights ?

THALES
DIGITAL
FACTORY

=> Specification is a definition of a program’s behaviour in the general case
=> Test are concrete examples of how a program should behave in specific scenarios

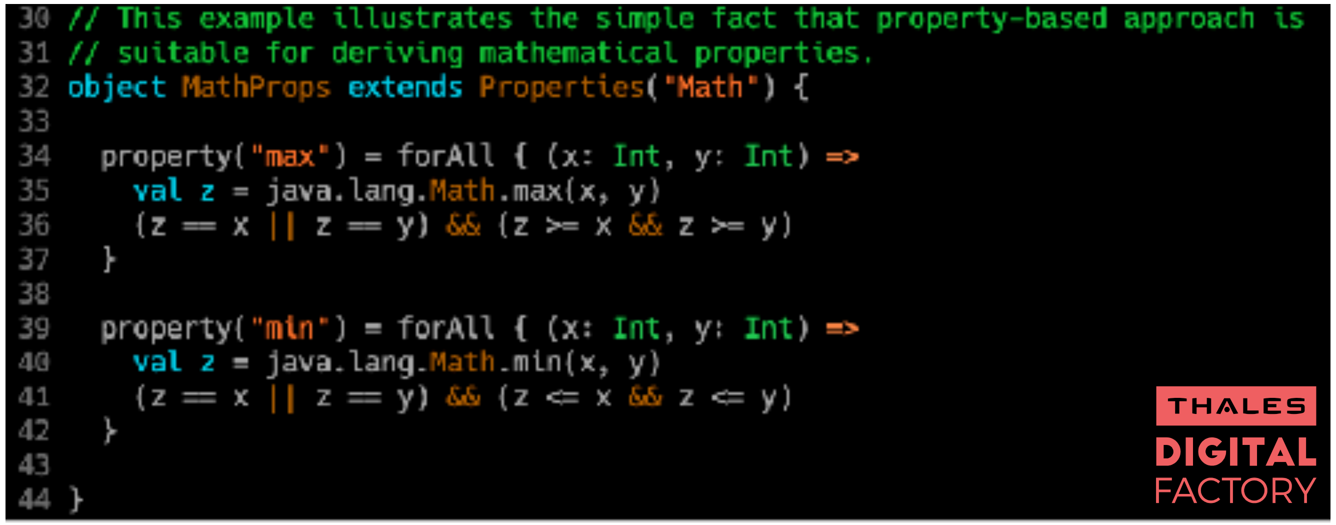
Test



```
import junit.framework.TestCase;

/**
 * This is the classic unit test you normally write.
 */
public class MathTest extends TestCase {
  public void testMax1() {
    int z = Math.max(1,2);
    assertEquals(2, z);
  }

  public void testMax2() {
    int z = Math.max(1,0);
    assertEquals(1, z);
  }

  public void testMax3() {
    int z = Math.max(10,10);
    assertEquals(10, z);
  }

  public void testMax4() {
    int z = Math.max(-2,0);
    assertEquals(0, z);
  }

}
```

Link: [1]

Specification



```
// This example illustrates the simple fact that property-based approach is
// suitable for deriving mathematical properties.
object MathProps extends Properties("Math") {

  property("max") = forAll { (x: Int, y: Int) =>
    val z = java.lang.Math.max(x, y)
    (z == x || z == y) && (z >= x && z >= y)
  }

  property("min") = forAll { (x: Int, y: Int) =>
    val z = java.lang.Math.min(x, y)
    (z == x || z == y) && (z <= x && z <= y)
  }

}
```

Link: [1]

Demo

With TS

- Several libraries inspired by **Haskell QuickCheck**
  - fast-check (compatible with jest, mocha, jasmine…)
  - JSVerify (mocha, jasmine)
  - testcheck

References

- [1] Property Based Testing @ TDF Gitlab
- [2] Introduction to Property-based Testing (fast-check)
- [3] jqwik
- [4] scalacheck
- [5] junit-quickcheck
- [6] jest
- [7] fast-check