psc_ TRANSITION TECHNOLOGIES

Testowanie jednostkowe, TDD

BITECODELAB

Testowanie jednostkowe

Czym jest i dlaczego jest dobre?

Dlaczego testujemy?

@ (a] a]]] a =

Pa]]]]]] _ ab]] _ i utrzymanie kodu

I ac] a]]] =]] +] a _d]
a _]

Pa] _ ac = a_ b] _ =

Pa] a] a] _d a

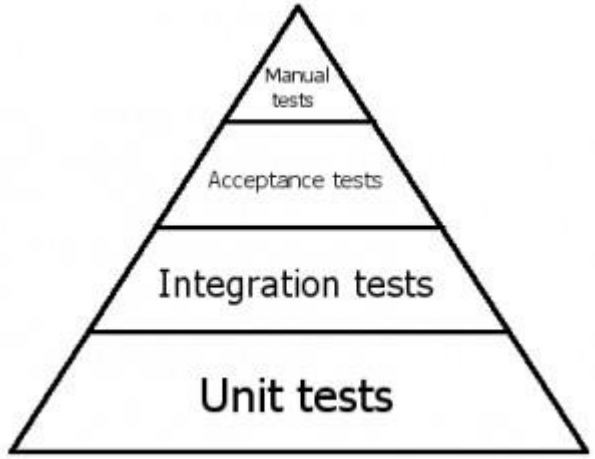
! Rozwiązanie bez testu NIE DZIAŁA

F] a]] a a

Jednostkowe
podstawowe testy obejmujące pojedynczą odpowiedzialność (klasa)

Integracyjne
weryfikują poprawność działania kilku modułów

Akceptacyjne
najczęściej tworzone i wykonywane na zlecenie i przy udziale klienta; potwierdzają poprawność z punktu widzenia wymagań biznesowych



S] _ a
służą do potwierdzenia możliwości zastosowania danej technologii lub modułu

Manualne
najczęściej spotykane
szybkie i… nic więcej
czasami wykorzystywane jako end-to-end

Test jednostkowy

Unit test

P _d]] _ b] c a
b _] _ ac] _d
warunkach

Testowanie jednostkowe

Jest to technika testowania oprogramowania

@] _ ac] = a a = a;

@ a (a =]]

V a] _d a _ _d b _] _] _d

O] a]] a] _

V a]] a

V a] a] _ a a =

Q]]] ab]] _

> a programistami

L] a a a przyjemne

L a] development

V a]] _d

?]]] a a =

Ea _] _d]]] a=a ; =

Odp: to] a

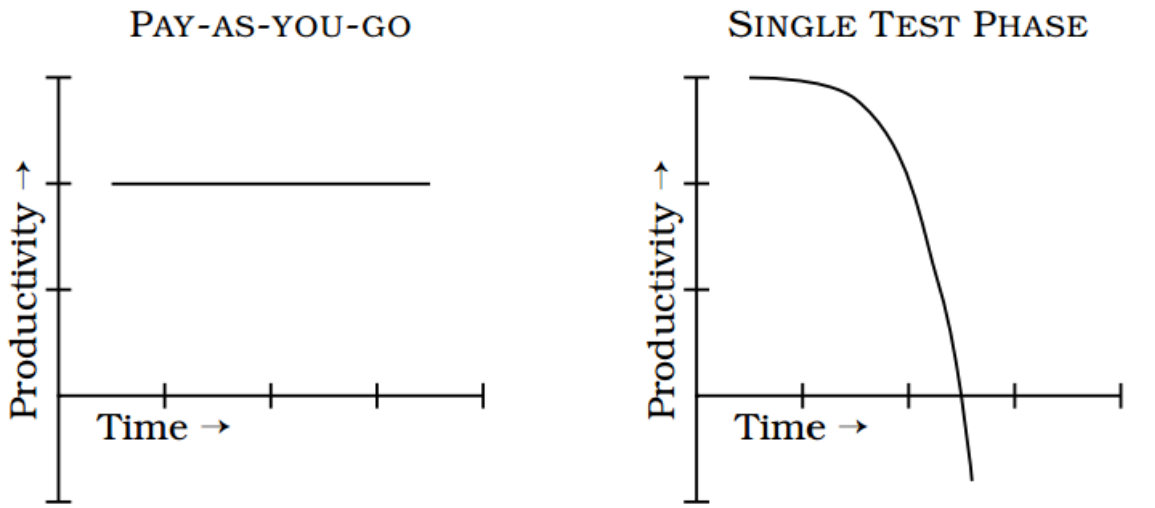


Figure 1.1: Comparison of Paying-as-you-go vs. Having a Single Testing Phase

Zasada FIRST

F] a a =a a;

F fast

I isolated

R repeatable

S self-verifying

T timely

JUnit

J]] a a b _ a

JUnit

B] a]] _ = _d]]]] a a =a a]] a F]

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>4.12</version>
   <scope>test</scope>
</dependency>
```

JUnit wsparcie w eclipse

MoreUnit

- Ctrl + J a _]] =a + a a a a] a
- Ctrl + R uruchom testy do danej klasy

EclEmma Java Code Coverage 2.3

- Bazuje na bibliotece jacoco

JUnit absolutne podstawy

```
@Test
public void testOrderOfList()
```

K]_](a]] a]] a]]] a _ a =

```
@Ignore
@Test
public void testOrderOfList()
```

Fa a _d_a _ _]] a a _ ac a = _] ac] = =
testowego (klasy).

JUnit absolutne podstawy

assertEquals([String message], expected value, actual value)

Zapewnienie(a a] _ a

assertArrayEquals([String message], expectedArray, resultArray)

Zapewnienie(a a] _a] _ a

assertTrue([String message], boolean condition)

assertFalse([String message], boolean condition)

Zapewnienie(a] a a a = a a a =

JUnit absolutne podstawy

assertNull([String message], java.lang.Object object)

assertNotNull([String message], java.lang.Object object)

Zapewnienie(a a a a = a a null.

assertSame([String message], java.lang.Object expected, java.lang.Object actual)

assertNotSame([String message], java.lang.Object unexpected, java.lang.Object actual)

Zapewnienie(a] a ac] ac _d a

Junit

JUnit c.d.

@BeforeClass
public static void setUpTestCase()

K]_](a]] a]] a]]] a] a a =] a]

@AfterClass
public static void tearDownTestCase()

K]_](a]] a]] a]]]] = _d a] a]

JUnit c.d.

@Before
public void setUpTest ()

K] _] (a]] a]] a]] a] a] =ac a] a]

@After
public void tearDownTest()

Oznacza(a]] a]] a]]]] =ac a] a]

JUnit

Pa] a (a]] =

```
@Test (expected = Exception.class)
void testThrowingExceptoin() {
}
```

lub

```
void testThrowingExceptoin() {
        try {
                methodThrowingException();
                fail();
        } catch (Exception e) {
                // weryfikacja wyjątku
        }
}
```

JUnit fixtures

L c] _ a a a]] _d(a a a a] _ a < Before.

```
public class JunitTest {
  private MyFixture fixture = new MyFixture();

  @Test
  public void testRule() {
        assertNotNull(fixture);
  }

 @Test
  public void testRule2() {
        assertNotNull(fixture);
  }

}
```

JUnit Rule

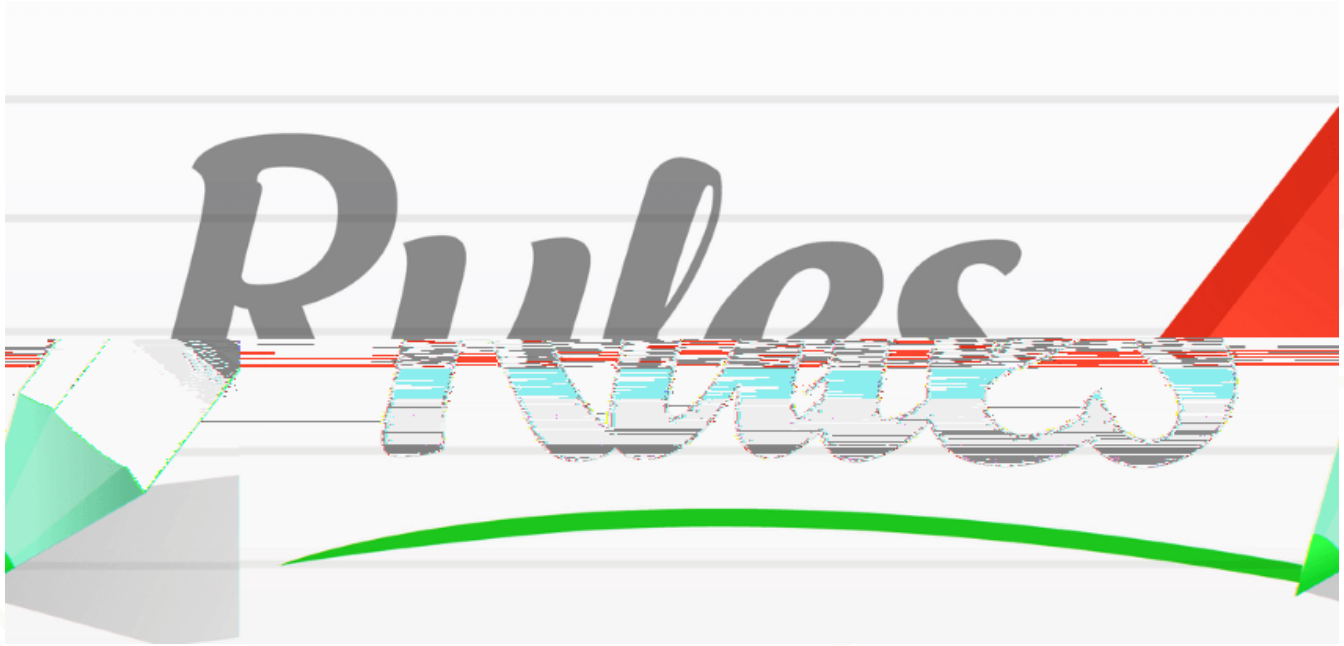
L a ab] a _ a a _a = a =a a] _ a(c=] a= = _ a katalogu tymczasowego.

```
public class JunitRuleTest {

  @Rule
  public TemporaryFolder tempFolder = new TemporaryFolder();

  @Test
  public void testRule() throws IOException {
    File newFolder = tempFolder.newFolder("Temp Folder");
    assertTrue(newFolder.exists());
  }
}
```

JUnit Rule

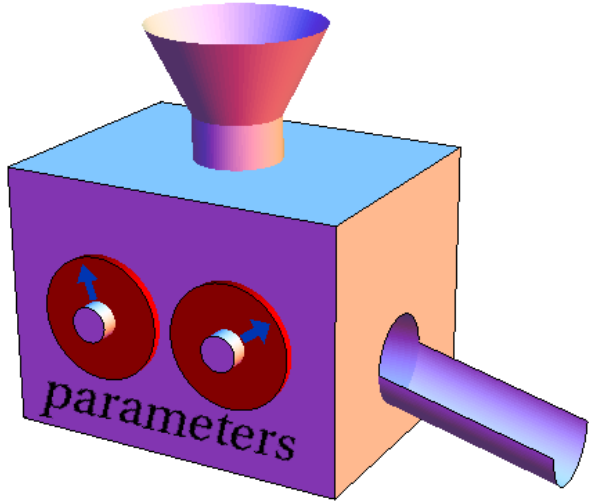
F]] ac ;

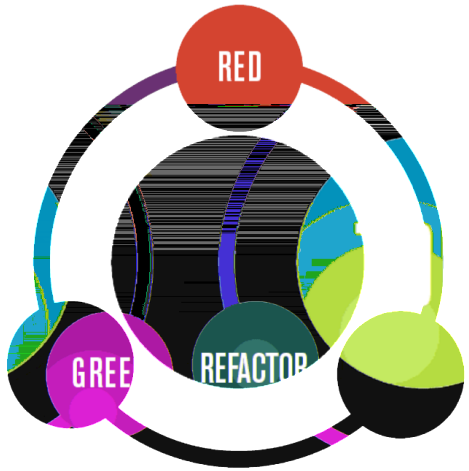
http://www.codeaffine.com/2012/09/24/junit-rules/



JUnit testy parametryzowane

```
@RunWith(Parameterized.class)
public class OperationTest {
   @Parameters
   public static Collection<Object[]> data() {
      return Arrays.asList(new Object[][] {
            { 0, 1 }, { 1, 4 }, { 2, 3 }
        });
   }

   private int input;   private int expected;
   public OperationTest(int input, int expected) {
      this.input = input; this.expected = expected;
   }
   private Operation service = new Operation();
   @Test
   public void shouldIncrement() {
      int result = service.inc(input);
      assertEquals(expected, result);
   }
}
```





TDD

Czym jest TDD?
Code kata

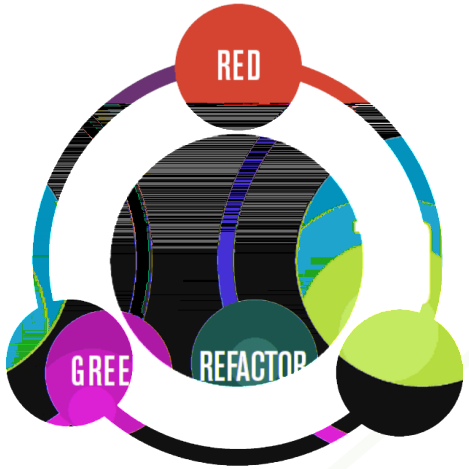
Co to jest TDD?

Test Driven Development, czyli TDD, to technika tworzenia oprogramowania sterowana przez testy. Tworzenie kodu
]] a a] _d a_d c _d

Stworzenie testu jednostkowego

Implementacja funkcjonalności

Refaktoryzacja



TDD - historia

Twórcą metody jest **Kent Beck**, twórca programowania ekstremalnego i jeden z sygnatariuszy Manifestu Agile. Oświadczył on w 2003, że TDD zachęca do stosowania prostych projektów i budzi w ludziach pewność siebie.

Test-Driven Development jest związany z pierwszymi koncepcjami programowania ekstremalnego rozpoczętych w 1999, ale ostatnio stał się samodzielną metodą.

TDD

Zalety TDD.

- O a _d] a
- L a]
- I a a]] b = _] _ a _d]]] _] ac c]]] =

Wady TDD

- wymaga dodatkowego czasu
-] c] _]]] a a

TDD] a] =a]

Testy a automatyczne

- Uruchamiane a a_a a ā] IDE)
- Sprawdzenie] _ a = =] _ a]
- S] _a] _ a = a] =
- Bardzo szybki

TDD] a] =a] c.d.

Testy tworzone przed kodem

- Najpierw piszemy test
- Potem a a]_ =

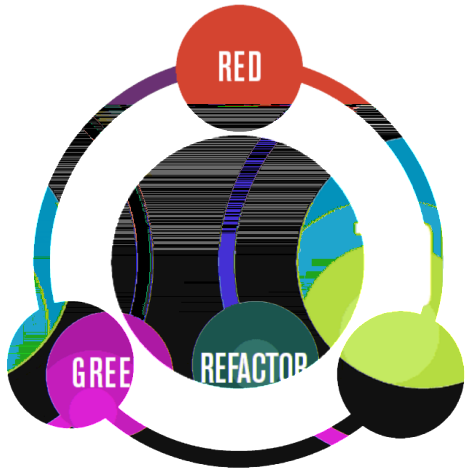
TDD] a] a] c.d.

Zasada] _d

- **Piszemy a (a a a a a _] a a] ac testu**
- **Szybka informacja zwrotna**
- **] a znalezienie przyczyny problemu**

TDD struktura dobrego testu

- Given-When-Then
- Arrange-Act-Assert



Code kata

Implementacja stosu w technice TDD

Code kata

Zaimplementujmy stos.

Wymagania podstawowe:

1. Operacja wrzucenia elementu na stos.
2. Operacja _] a a a a

Dodatkowe wymagania:

1. Ograniczenie rozmiaru
2. S a przy przekroczeniu limitu

www.tt.psc.pl/en
www.25lat.tt.com.pl/en
www.iot.tt.com.pl/en

Looking forward to build longterm partnership with You

Paweł Łagan pawel.lagan@ttpsc.pl