TECHNISCHE UNIVERSITÄT DRESDEN

OOSE4

Testen mit BlueJ/JUnit 4

Lehrstuhl Softwaretechnologie, Dr. Birgit Demuth
Sommersemester 2016

Vorgehen beim Unit-Test allgemein

1. Testfälle ausdenken / Testfalltabellen erstellen
2. Testfälle nach gemeinsamen Fixtures (Halterungen) in Klassen gruppieren
3. i. Allg. pro Testklasse eine Halterung (Fixture)
   - Methode **setUp()**
   - Methode **tearDown()**
   - pro Testfall eine Testmethode schreiben
4. Testfälle (nach Änderungen im Programm wiederholt) ausführen
   → Regressionstest

Wie wähle ich Testdaten für Testfälle aus (Vorlesung!)

- Bestimme die Extremwerte der Parameter der zu testenden Methode (**Grenzwertanalyse**)
  - Nullwerte immer testen, z.B. 0 oder null
  - Randwerte z.B. 1.1. und 31.12.
- Bestimme Bereichseinschränkungen
  - Werte außerhalb eines Zahlenbereiches
  - Negative Werte, wenn natürliche Zahlen im Spiel sind
- Bestimme **Äquivalenzklassen** von Testdaten und teste nur die Repräsentanten
- Bestimme **Zustände**, in denen sich ein Objekt nach einer Anweisung befinden muss
- Bestimme alle Werte aller **booleschen Bedingungen** in der Methode (Steuerfluss)

Äquivalenzklassenbildung (1)

Anforderungen genau definieren

Beispiel

- Basispreis für Flug von A nach B kostet 100$.
- Buchung im Reservierungssystem für Reihe 1-10 kostet 20$ mehr , Reihe 11-21 kosten 10$ weniger. Das Flugzeug hat nur 21 Sitzreihen.
- Essen kostet 5$ extra.
- Wenn Frühbucher (bis 4 Wochen vor Flugtermin), dann 10%Rabatt auf Gesamtpreis, bis zwei Wochen vor Flugtermin 5 %, sonst kein Rabatt auf Buchung.

Äquivalenzklassenbildung (2)

Bildung der Äquivalenzklasse durch Klassifizierung der Wertebereiche für Ein- und Ausgabedaten

Beispiel

- **Parameter 1: Basispreis**
  ÄK11={100€}
- **Parameter 2: Reihe // Das Flugzeug hat nur 21 Reihen.**
  ÄK21={1-10}
  ÄK22={11-21}
- **Parameter 3: Verpflegung // Ja - Nein**
  ÄK31={keine Verpflegung}
  ÄK32={Verpflegung}
- **Parameter 4: Rabatt // 3 Rabattierungstypen**
  ÄK41={bis 4 Wochen vor Termin}
  ÄK42={bis 2 Wochen vor Termin}
  ÄK43={kein Rabatt}

Äquivalenzklassenbildung (3)

Definition von Testfällen für

1. gültige Äquivalenzklassen

Beispiel

- 100% Äquivalenzklassenabdeckung (Vollständige Abdeckung)

→ 1*2*2*3 = 12 Testfälle

2. ungültige Äquivalenzklassen

Beispiel

- Negativer Basispreis
- Ungültige Sitzreihen
- Ungültiger Rabattierungstyp

Schreiben von Unit-Tests mit BlueJ (verwendet JUnit 4)

Verwendung von Annotationen um

- Methoden als Testmethoden zu kennzeichnen
- die Testabarbeitung zu konfigurieren

Empfehlenswertes Tutorials

- Junit 4 allgemein

http://www.frankwestphal.de/JUnit4.0.html

- BlueJ / JUnit4

http://www.bluej.org/tutorial/testing-tutorial.pdf

Annotation	Description
@Test public void method()	The @Test annotation identifies a method as a test method.
@Test (expected = Exception.class)	Fails if the method does not throw the named exception.
@Test(timeout=100)	Fails if the method takes longer than 100 milliseconds.
@Before public void method()	This method is executed before each test. It is used to prepare the test environment (e.g., read input data, initialize the class).
@After public void method()	This method is executed after each test. It is used to cleanup the test environment (e.g., delete temporary data, restore defaults). It can also save memory by cleaning up expensive memory structures.
@BeforeClass public static void method()	This method is executed once, before the start of all tests. It is used to perform time intensive activities, for example, to connect to a database. Methods marked with this annotation need to be defined as static to work with JUnit.
@AfterClass public static void method()	This method is executed once, after all tests have been finished. It is used to perform clean-up activities, for example, to disconnect from a database. Methods annotated with this annotation need to be defined as static to work with JUnit.
@Ignore	Ignores the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted. Or if the execution time of this test is too long to be included.

TECHNISCHE UNIVERSITÄT DRESDEN

BlueJ / Junit 4

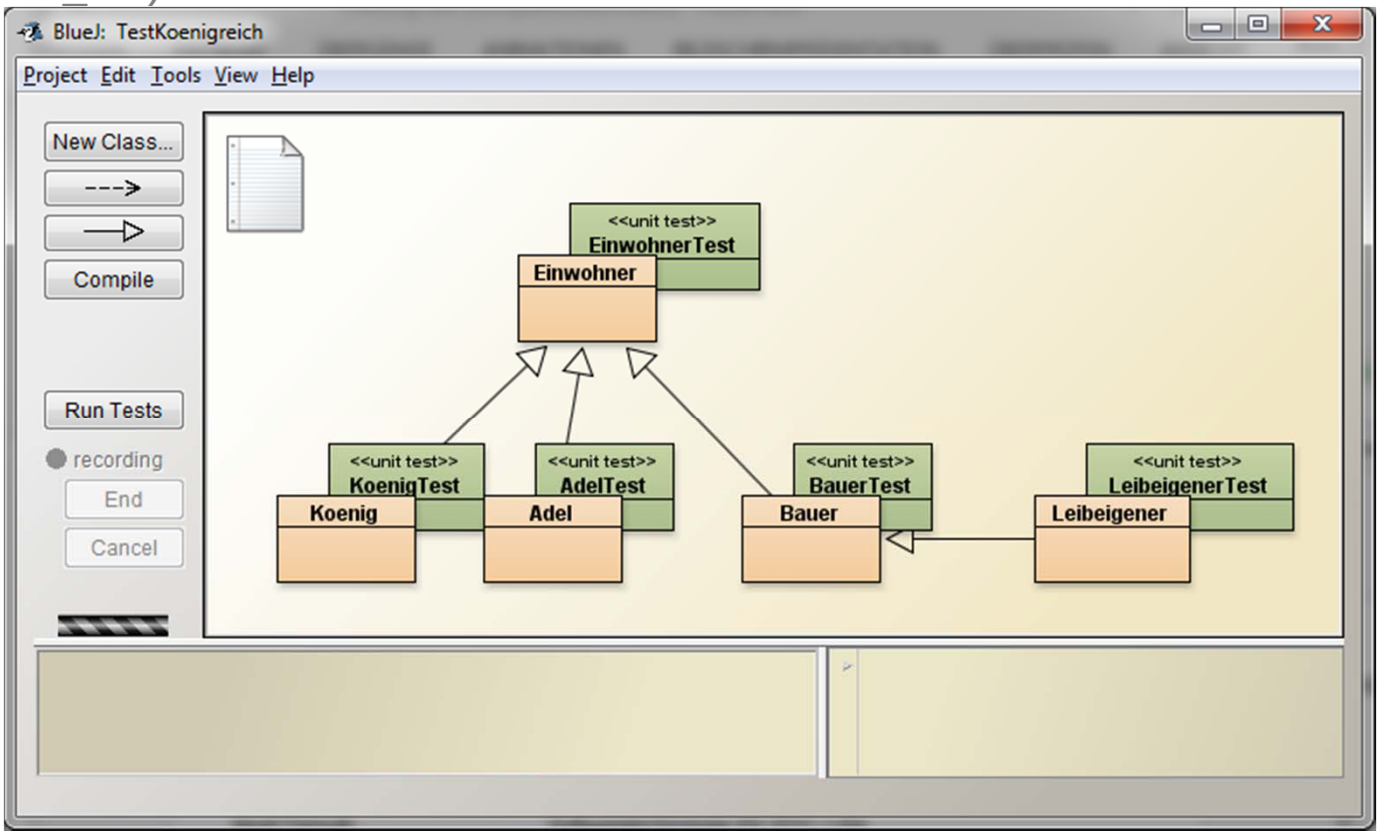
TESTVARIANTE 1

Testfalltabelle für die Methode steuer()

Test-fall	Erwarteter Status	Klasse	int einkommen	Ausgabe
E1	Exception	Einwohner	-1	Einkommen darf nicht negativ sein
E2	ok	Einwohner	0	1
E3	ok	Einwohner	20	2
E4	ok	Einwohner	25	2

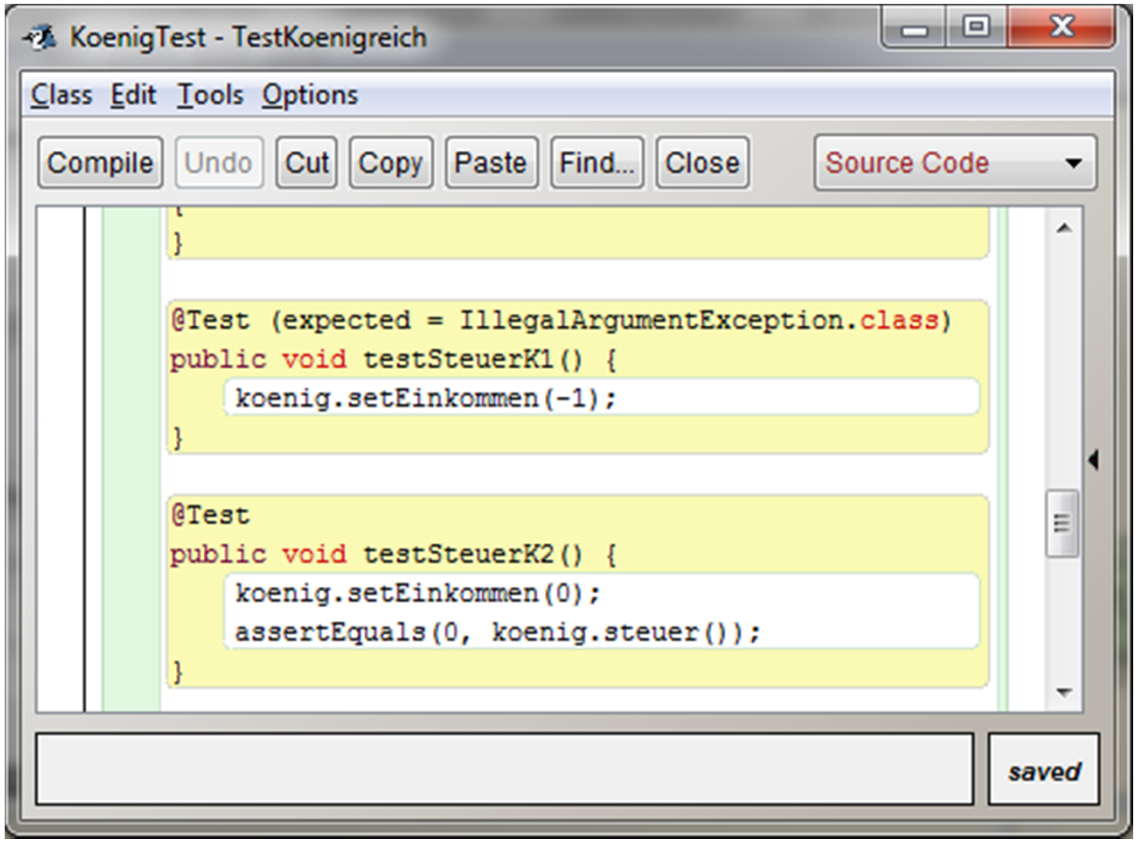
Test mit BlueJ/JUnit 4

(TestKoenigreich_V1)

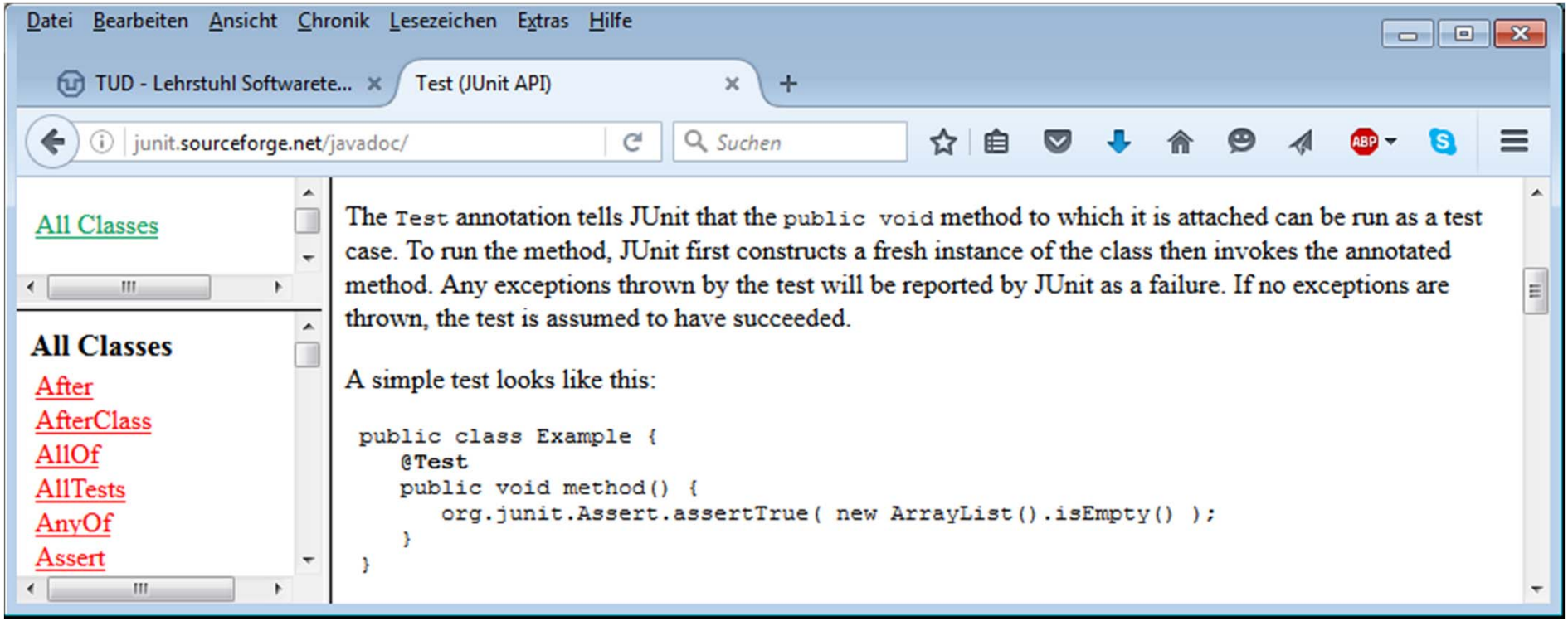


Test mit BlueJ/JUnit 4

(TestKoenigreich_V1)

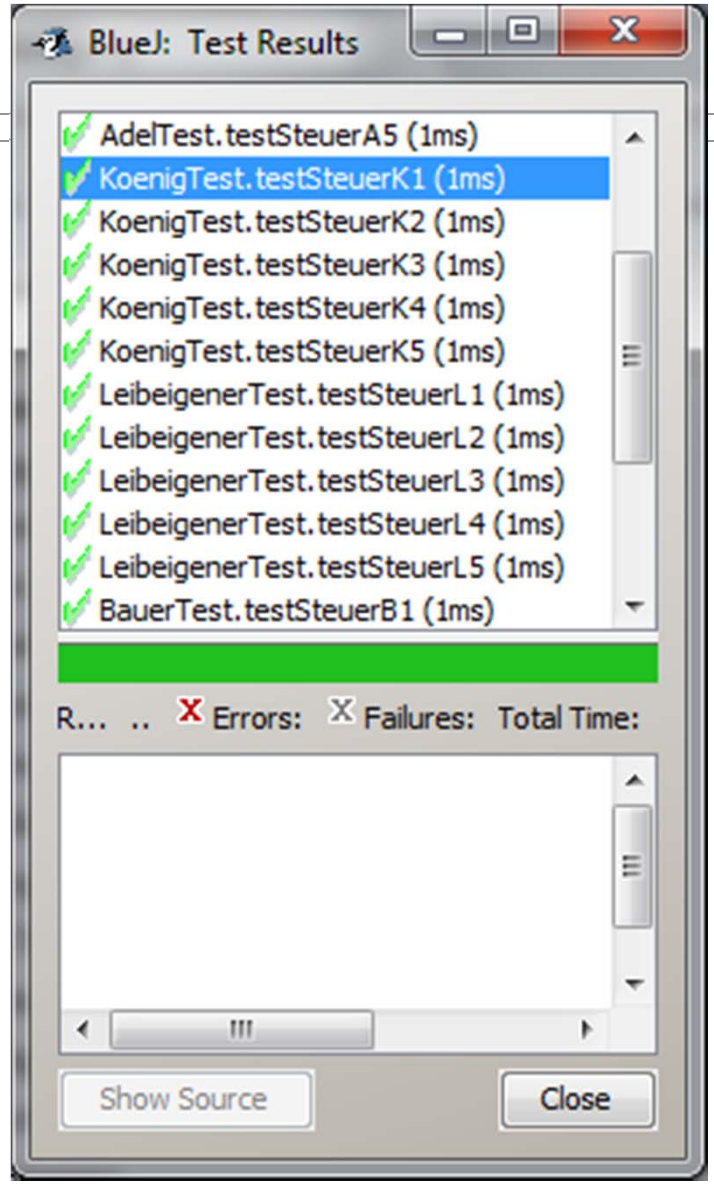


javadoc von Junit 4

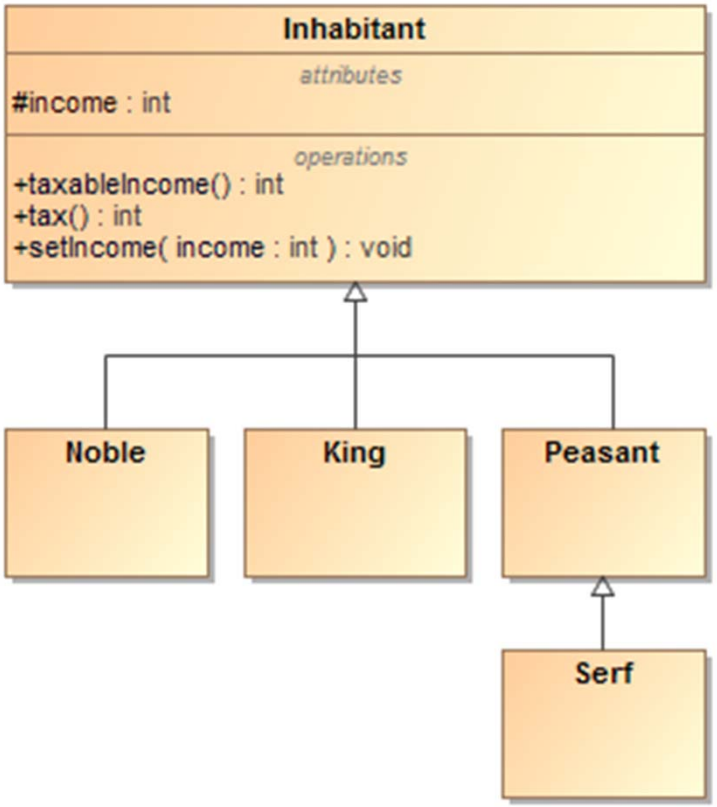


Test mit BlueJ/JUnit 4

(TestKoenigreich_V1)

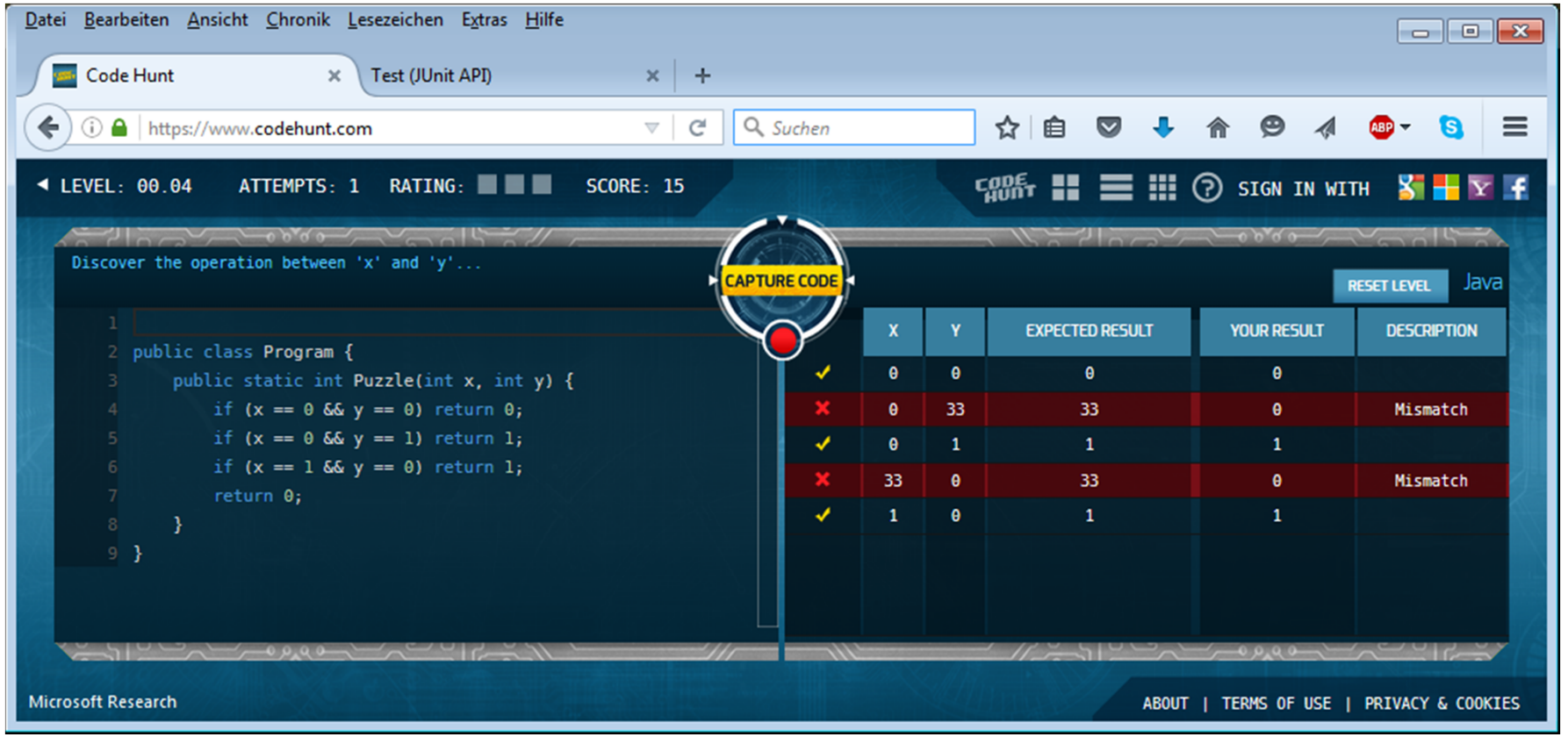


INLOOP Tests reverse?



- JUnit-Tests werden in INLOOP bereitgestellt

Java Training mit Code Hunt



Kleiner erweiterter Selbsttest

http://bit.do/OOSE_Test

- Fragen der letzten Übung (OOSE 3)
- Zusätzliche Fragen zum Testen
- Bei Angabe einer Emailadresse wird das Ergebnis des Selbsttestes an diese Emailadresse zugeschickt