Proseminar Softwareentwicklung in der Wissenschaft

Testen

BENEDIKT BLUME

Gliederung

1. Motivation

- Erster Todesfall durch autonomes Testauto
- Im Mai 2018 im US-Bundesstaat Arizona
- Fahrzeug des Mobilitätsdienstes Uber überfährt Fußgängerin, die ihr Fahrrad über die Straße schiebt
- Auto von Uber besaß gravierende Programmierfehler
- Das Protokoll der Katastrophe zeigt eine Aneinanderkettung von Fehleinschätzungen

Uber © Creative Lab / Shutterstock.com

[1]

1.1 Was ist Testen?

- Es wird getestet ob Fehler in einer Software sind
- Fehlerquellen in neuen aber auch in weiterentwickelten Programmen
- Teams die sich nur auf das Testen konzentrieren
- Testen zeitaufwendigster Part der Softwareentwicklung
- Test-First-Ansatz
- Verschiedene Arten von Tests

[2]

1.1 Was ist Testen?

Statische Software-Testverfahren:

- Verfahren ohne Programmausführung
- Menschen analysieren den Code
- Ausreichende Dokumentation
- Doppelte Codes vorhanden
- Stimmen die Funktionen
- Statische Analyzer (z.B. Clang Static Analyzer)

[3] [9] [10]

1.1 Was ist Testen?

Dynamische Software-Testverfahren:

- Verfahren für lauffähige Software
- Testobjekte werden stichprobenartig überprüft
- Die Ist-Ergebnisse werden mit Soll-Ergebnissen verglichen
- Prüfmethoden, um mit Softwaretests Fehler in der Software aufzuspüren
- Testmetriken (z.B. Code Coverage)

[4] [8]

1.2 Warum testen?

- Software steckt voller Fehler
- Software soll funktionieren
- Folgen von Fehlern
- Faustformel: *"Je später Fehler entdeckt werden, desto aufwändiger ist ihre Behebung“*

[6]

1.2 Warum testen?

Ein Beispiel aus der Realität:

- Fehler in der Produktion von Automobil und Flugzeug-Branche in den USA
- Allein für das Jahr 2000 ein Schaden von 1,8 Milliarden aufgrund fehlerhafter Software
- Die Hochrechnung der einzelnen Brachen auf die gesamte amerikanische Volkswirtschaft ergibt die Zahl von 59,5 Milliarden US-Dollar pro Jahr als Folge fehlerhafter Software
- Die rechtzeitige Beseitigung würde 70% der Fehlerbehebungskosten und das Schadenpotenzial um 50% verringern

[5]

1.3 Was soll erreicht werden?

Ziele:

- So wenig Fehler wie möglich
- Vertrauen schaffen
- Folgefehler verringern
- Qualität bestimmen
- Kosten sparen

[7]

2. Test-Framework

Test-Framework:

- Framework = Programmiergerüst
- Framework selbst noch kein fertiges Programm
- Framework gibt Anwendungsarchitektur vor
- Blackbox-Test
- Whitebox-Test

[11]

2. Test-Framework (JUnit)

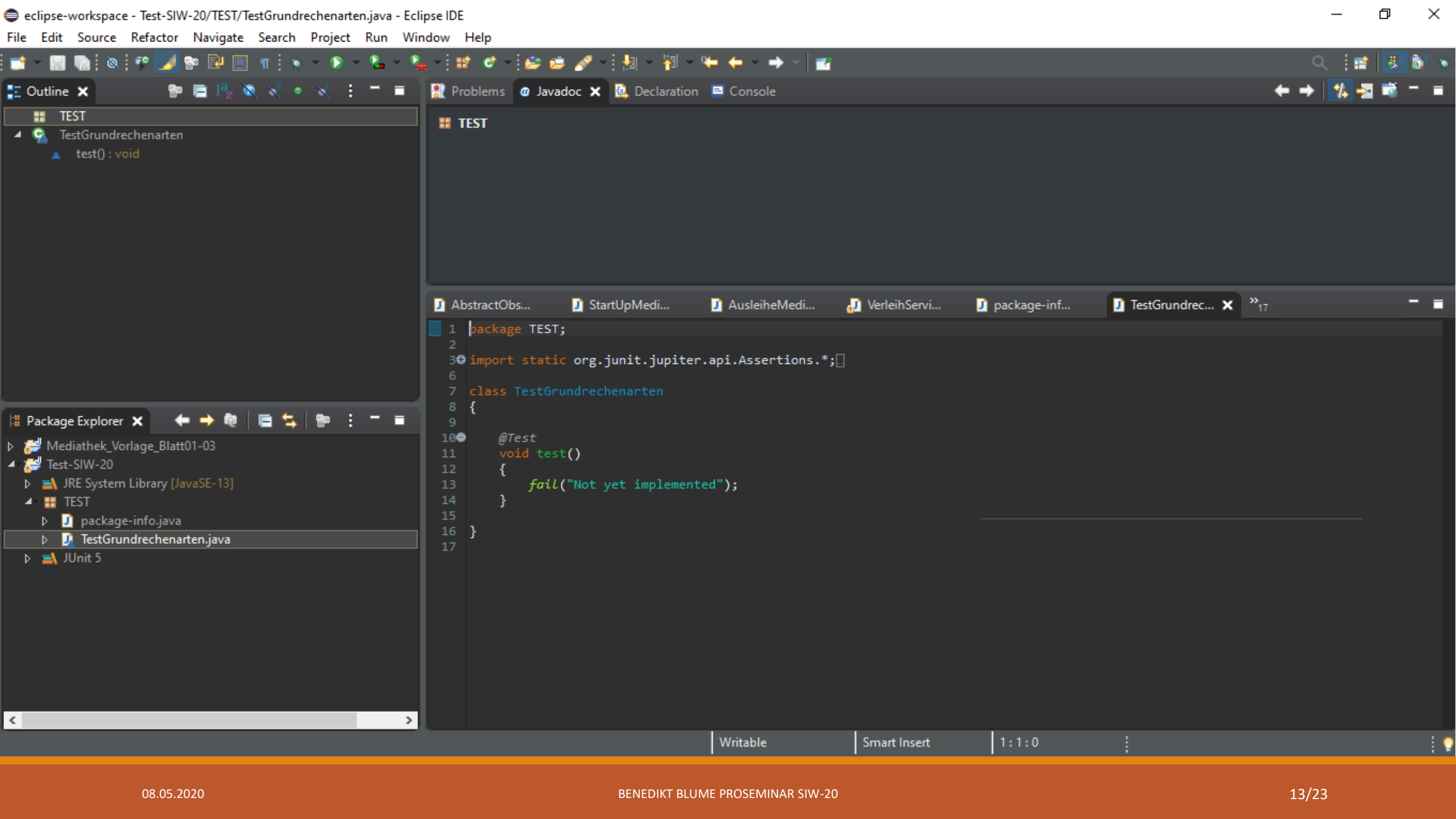
JUnit:

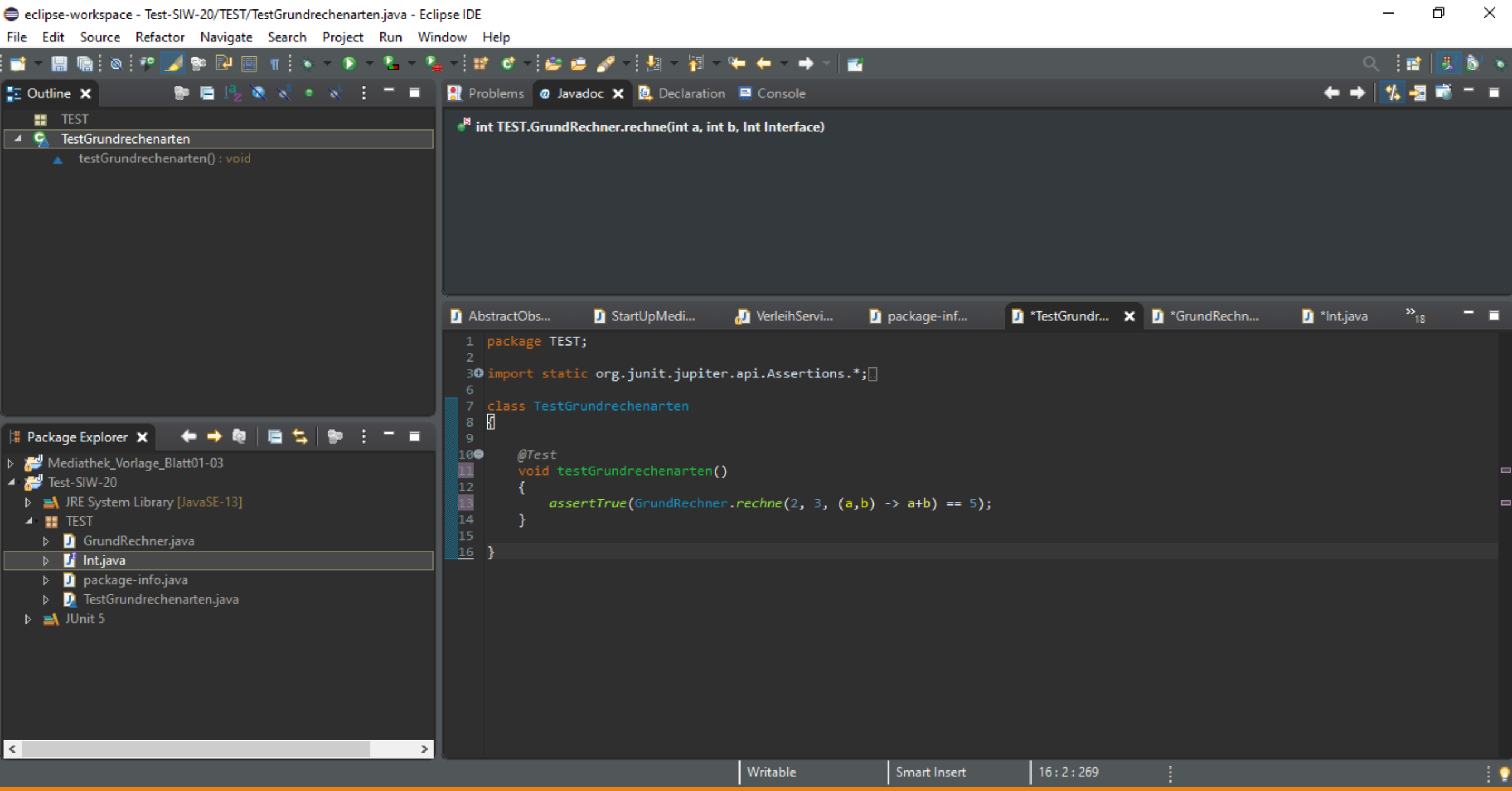
- für Java Programme (Aktuellste Version JUnit 5)
- Unit-Tests (Klassen oder Methoden)
- Klassen um Quelltext zu prüfen
- Nur zwei Ergebnisse: gelungen oder misslungen
- Misslungen unterteilt in Failure oder Error

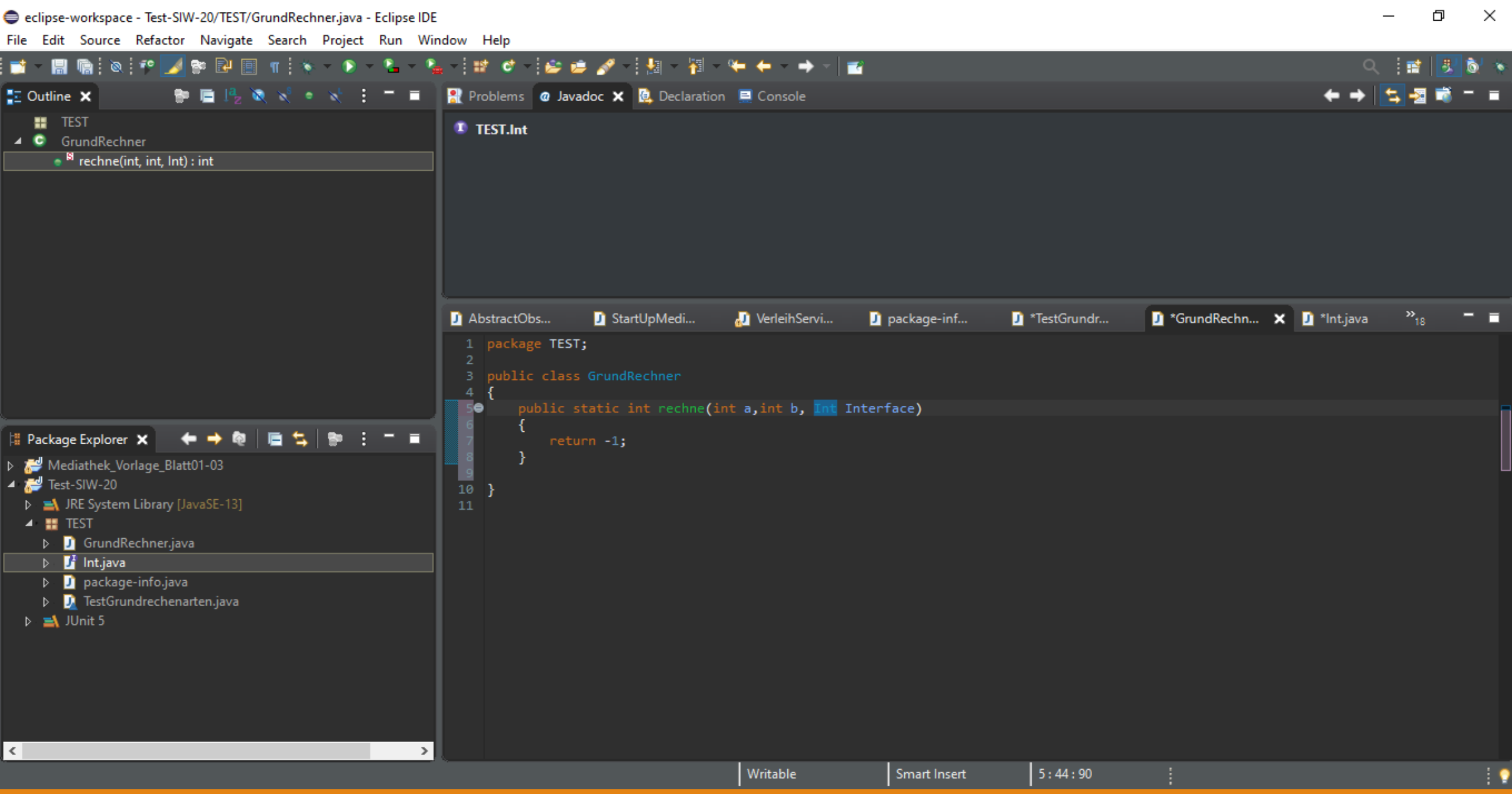
[12]

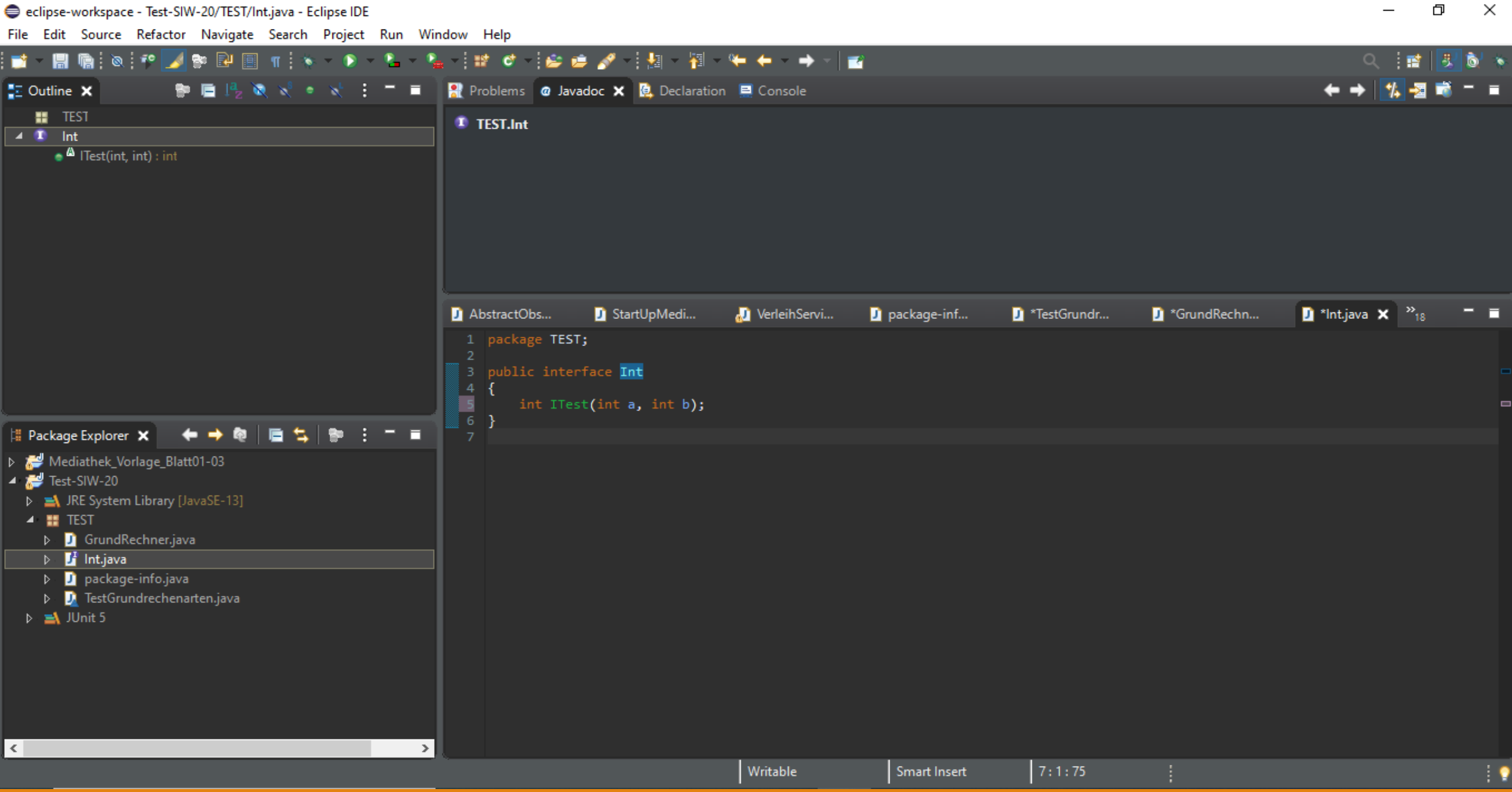
3.1 Wie wird ein Test geschrieben?

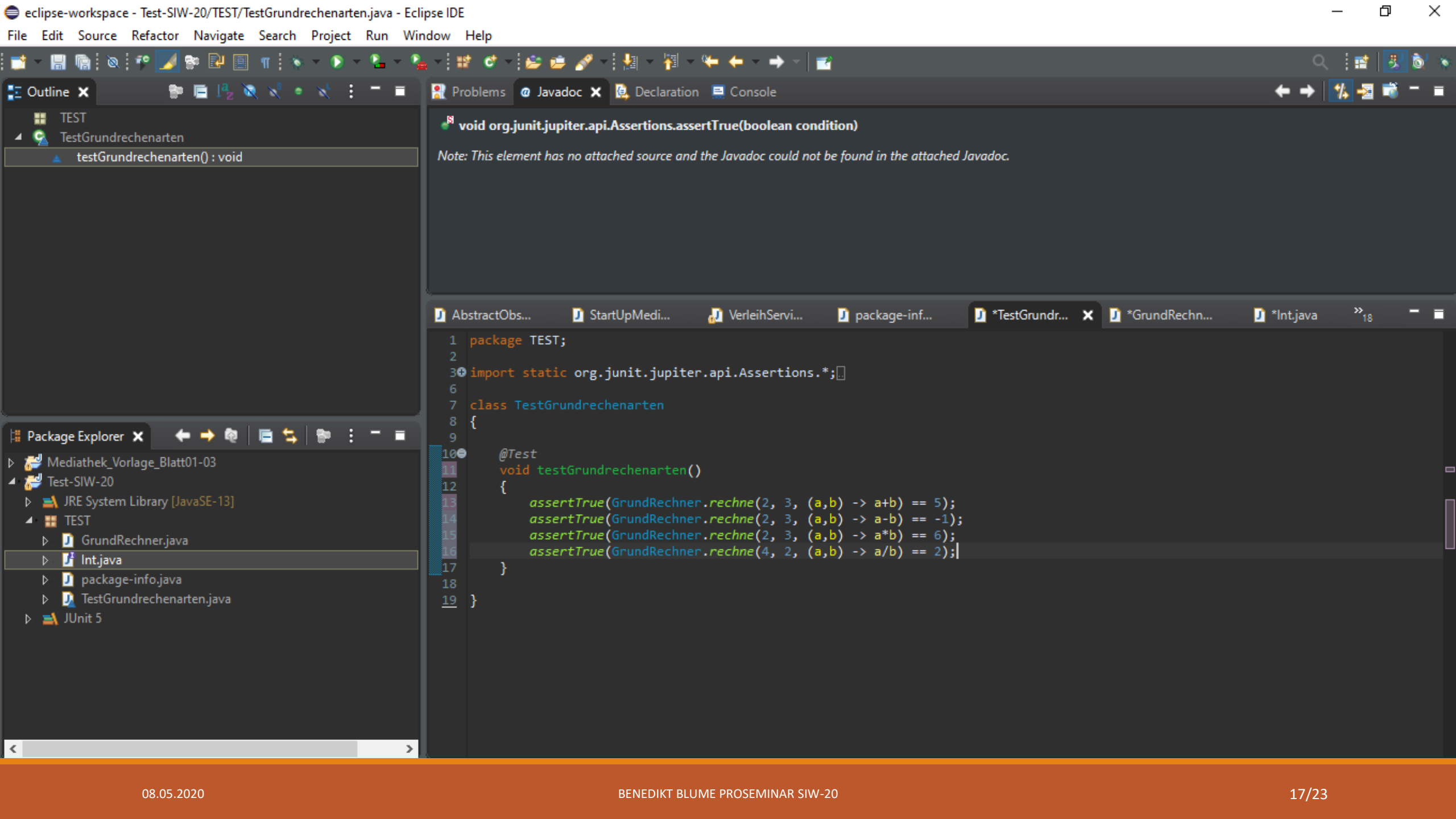
- Testart: Dynamischer Software-Test
- Systemoberfläche: Eclipse
- Programmiersprache: Java
- Hilfsmittel: JUnit
- Beispieltest: Grundrechenarten

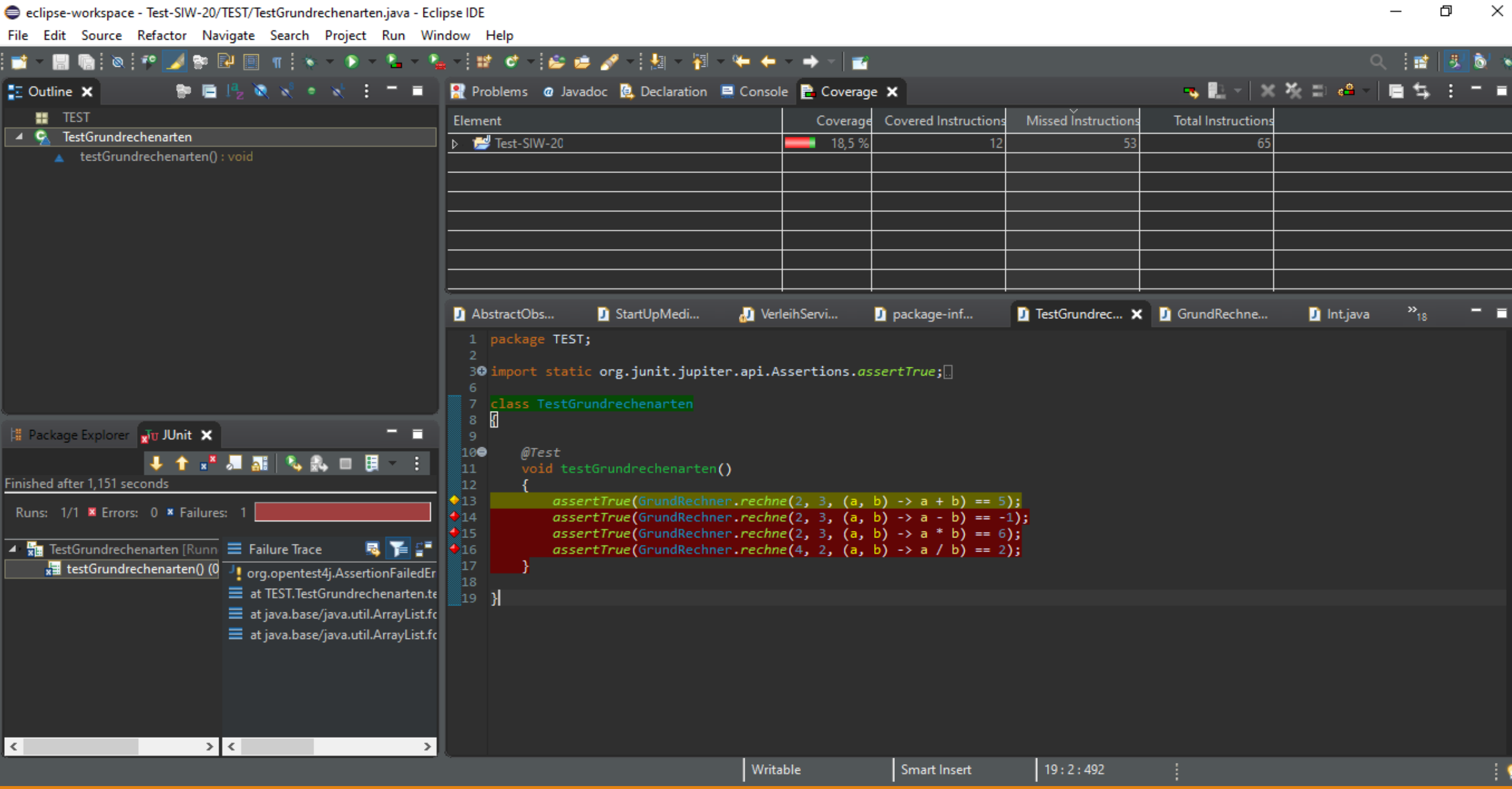

eclipse-workspace - Test-SIW-20/TEST/TestGrundrechenarten.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
TestGrundrechenarten
test() : void
Problems
Javadoc
Declaration
Console
TEST
AbstractObs...
StartUpMedi...
AusleiheMedi...
VerleihServi...
package-inf...
TestGrundrec...
1 package TEST;
2
3 import static org.junit.jupiter.api.Assertions.*;
6
7 class TestGrundrechenarten
8 {
9
10 @Test
11 void test()
12 {
13 fail("Not yet implemented");
14 }
15
16 }
17
Package Explorer
Mediathek_Vorlage_Blatt01-03
Test-SIW-20
JRE System Library [JavaSE-13]
TEST
package-info.java
TestGrundrechenarten.java
JUnit 5
Writable
Smart Insert
1 : 1 : 0

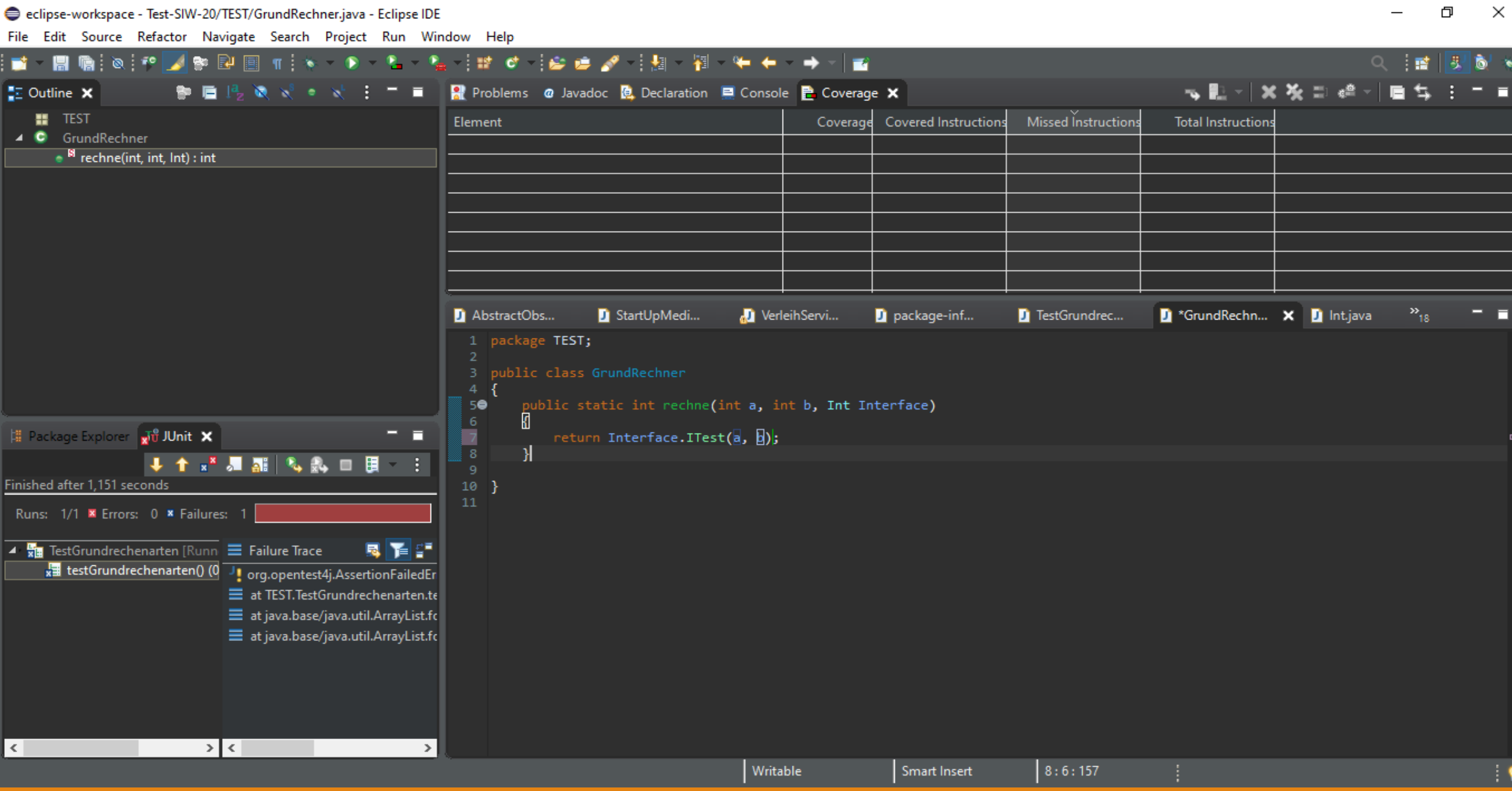

eclipse-workspace - Test-SIW-20/TEST/TestGrundrechenarten.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
TestGrundrechenarten
testGrundrechenarten() : void
Problems
Javadoc
Declaration
Console
int TEST.GrundRechner.rechne(int a, int b, Int Interface)
AbstractObs...
StartUpMedi...
VerleihServi...
package-inf...
*TestGrundr...
*GrundRechn...
*Int.java
1 package TEST;
2
3 import static org.junit.jupiter.api.Assertions.*;
6
7 class TestGrundrechenarten
8 {
9
10 @Test
11 void testGrundrechenarten()
12 {
13 assertTrue(GrundRechner.rechne(2, 3, (a,b) -> a+b) == 5);
14 }
15
16 }
Package Explorer
Mediathek_Vorlage_Blatt01-03
Test-SIW-20
JRE System Library [JavaSE-13]
TEST
GrundRechner.java
Int.java
package-info.java
TestGrundrechenarten.java
JUnit 5
Writable
Smart Insert
16 : 2 : 269

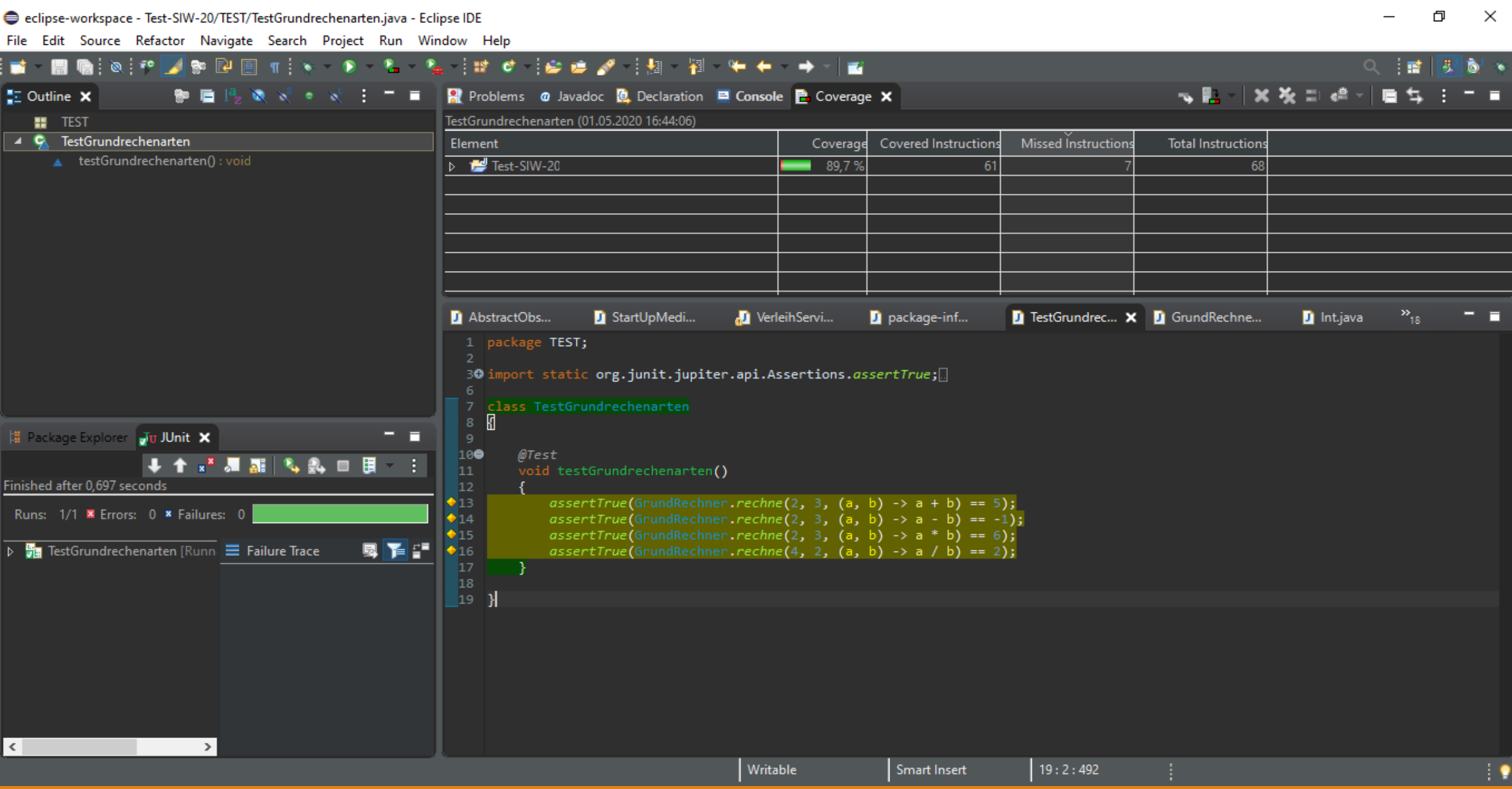

eclipse-workspace - Test-SIW-20/TEST/GrundRechner.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
GrundRechner
rechne(int, int, Int) : int
Problems
Javadoc
Declaration
Console
TEST.Int
AbstractObs...
StartUpMedi...
VerleihServi...
package-inf...
*TestGrundr...
*GrundRechn...
*Int.java
1 package TEST;
2
3 public class GrundRechner
4 {
5 public static int rechne(int a,int b, Int Interface)
6 {
7 return -1;
8 }
9
10 }
11
Package Explorer
Mediathek_Vorlage_Blatt01-03
Test-SIW-20
JRE System Library [JavaSE-13]
TEST
GrundRechner.java
Int.java
package-info.java
TestGrundrechenarten.java
JUnit 5
Writable
Smart Insert
5 : 44 : 90


eclipse-workspace - Test-SIW-20/TEST/Int.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
Int
ITest(int, int) : int
Problems
Javadoc
Declaration
Console
TEST.Int
AbstractObs...
StartUpMedi...
VerleihServi...
package-inf...
*TestGrundr...
*GrundRechn...
*Int.java
1 package TEST;
2
3 public interface Int
4 {
5 int ITest(int a, int b);
6 }
7
Package Explorer
Mediathek_Vorlage_Blatt01-03
Test-SIW-20
JRE System Library [JavaSE-13]
TEST
GrundRechner.java
Int.java
package-info.java
TestGrundrechenarten.java
JUnit 5
Writable
Smart Insert
7 : 1 : 75


eclipse-workspace - Test-SIW-20/TEST/TestGrundrechenarten.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
TestGrundrechenarten
testGrundrechenarten() : void
Problems
Javadoc
Declaration
Console
void org.junit.jupiter.api.Assertions.assertTrue(boolean condition)
Note: This element has no attached source and the Javadoc could not be found in the attached Javadoc.
AbstractObs...
StartUpMedi...
VerleihServi...
package-inf...
*TestGrundr...
*GrundRechn...
*Int.java
1 package TEST;
2
3 import static org.junit.jupiter.api.Assertions.*;
6
7 class TestGrundrechenarten
8 {
9
10 @Test
11 void testGrundrechenarten()
12 {
13 assertTrue(GrundRechner.rechne(2, 3, (a,b) -> a+b) == 5);
14 assertTrue(GrundRechner.rechne(2, 3, (a,b) -> a-b) == -1);
15 assertTrue(GrundRechner.rechne(2, 3, (a,b) -> a*b) == 6);
16 assertTrue(GrundRechner.rechne(4, 2, (a,b) -> a/b) == 2);
17 }
18
19 }
Package Explorer
Mediathek_Vorlage_Blatt01-03
Test-SIW-20
JRE System Library [JavaSE-13]
TEST
GrundRechner.java
Int.java
package-info.java
TestGrundrechenarten.java
JUnit 5


eclipse-workspace - Test-SIW-20/TEST/TestGrundrechenarten.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
TestGrundrechenarten
testGrundrechenarten() : void
Problems Javadoc Declaration Console Coverage
Element Coverage Covered Instructions Missed Instructions Total Instructions
Test-SIW-20 18,5 % 12 53 65
AbstractObs... StartUpMedi... VerleihServi... package-inf... TestGrundrec... GrundRechne... Int.java
1 package TEST;
2
3 import static org.junit.jupiter.api.Assertions.assertTrue;
6
7 class TestGrundrechenarten
8 {
9
10 @Test
11 void testGrundrechenarten()
12 {
13 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a + b) == 5);
14 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a - b) == -1);
15 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a * b) == 6);
16 assertTrue(GrundRechner.rechne(4, 2, (a, b) -> a / b) == 2);
17 }
18
19 }
Package Explorer JUnit
Finished after 1,151 seconds
Runs: 1/1 Errors: 0 Failures: 1
TestGrundrechenarten [Runn
testGrundrechenarten() (0
Failure Trace
org.opentest4j.AssertionFailedEr
at TEST.TestGrundrechenarten.te
at java.base/java.util.ArrayList.fo
at java.base/java.util.ArrayList.fo
Writable Smart Insert 19 : 2 : 492


eclipse-workspace - Test-SIW-20/TEST/GrundRechner.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
GrundRechner
rechne(int, int, Int) : int
Problems Javadoc Declaration Console Coverage
Element Coverage Covered Instructions Missed Instructions Total Instructions
AbstractObs... StartUpMedi... VerleihServi... package-inf... TestGrundrec... *GrundRechn... Int.java
1 package TEST;
2
3 public class GrundRechner
4 {
5 public static int rechne(int a, int b, Int Interface)
6 {
7 return Interface.ITest(a, b);
8 }
9
10 }
11
Package Explorer JUnit
Finished after 1,151 seconds
Runs: 1/1 Errors: 0 Failures: 1
TestGrundrechenarten [Runn
testGrundrechenarten() (0
Failure Trace
org.opentest4j.AssertionFailedEr
at TEST.TestGrundrechenarten.te
at java.base/java.util.ArrayList.fc
at java.base/java.util.ArrayList.fc
Writable
Smart Insert
8 : 6 : 157


eclipse-workspace - Test-SIW-20/TEST/TestGrundrechenarten.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Outline
TEST
TestGrundrechenarten
testGrundrechenarten() : void
Problems Javadoc Declaration Console Coverage
TestGrundrechenarten (01.05.2020 16:44:06)
Element Coverage Covered Instructions Missed Instructions Total Instructions
Test-SIW-20 89,7 % 61 7 68
AbstractObs... StartUpMedi... VerleihServi... package-inf... TestGrundrec... GrundRechne... Int.java
1 package TEST;
2
3 import static org.junit.jupiter.api.Assertions.assertTrue;
6
7 class TestGrundrechenarten
8 {
9
10 @Test
11 void testGrundrechenarten()
12 {
13 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a + b) == 5);
14 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a - b) == -1);
15 assertTrue(GrundRechner.rechne(2, 3, (a, b) -> a * b) == 6);
16 assertTrue(GrundRechner.rechne(4, 2, (a, b) -> a / b) == 2);
17 }
18
19 }
Package Explorer JUnit
Finished after 0,697 seconds
Runs: 1/1 Errors: 0 Failures: 0
TestGrundrechenarten [Runn
Failure Trace
Writable Smart Insert 19 : 2 : 492

4. Zusammenfassung

- Statische Software-Testverfahren
- Dynamische Software-Testverfahren
- Faustformel: *"Je später Fehler entdeckt werden, desto aufwändiger ist ihre Behebung“*
- Software ist nie fehlerfrei
- verschiedene Test-Ziele
- Test-Framework = Programmiergerüst
- Testaufbau in Java mit JUnit

5. Quellennachweis

[1] https://www.welt.de/wirtschaft/article203121966/Autonomes-Fahren-0-2-Sekunden- vor- dem-toedlichen-Aufprall-reagierte-das-Uber-Auto.html

[2] Der Systemtest : anforderungsbasiertes Testen von Software-Systemen / Harry M. Sneed; Manfred Baumgartner; Richard Seidl. 2011 Seite 1

[3] https://de.wikipedia.org/wiki/Statisches_Software-Testverfahren

[4] https://de.wikipedia.org/wiki/Dynamisches_Software-Testverfahren

[5] Der Systemtest : anforderungsbasiertes Testen von Software-Systemen / Harry M. Sneed; Manfred Baumgartner; Richard Seidl. 2011 Seite 7

[6] Pol, Koomen, Spillner: Management und Optimierung des Testprozesses

[7] Andreas Spillner, Tilo Linz, Basiswissen Softwaretest, Dpunkt Verlag 2005

5. Quellennachweis

[8] https://www.qa-systems.com/tools/cantata/code-coverage/

[9] https://clang-analyzer.llvm.org/

[10] https://clang.llvm.org/

[11] https://de.wikipedia.org/wiki/Framework

[12] https://de.wikipedia.org/wiki/JUnit