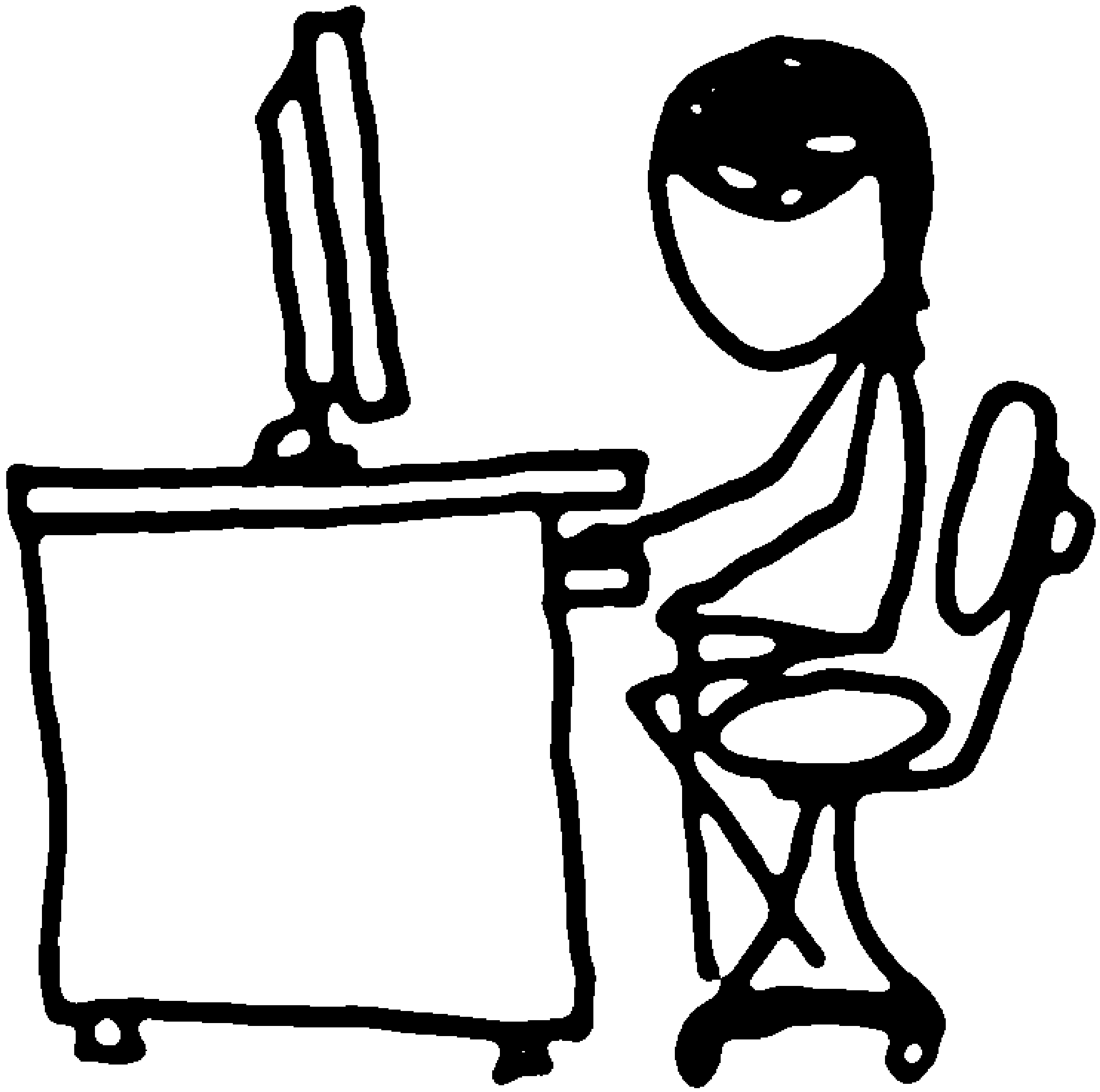
ReAssert: Suggesting Repairs for Broken Unit Tests

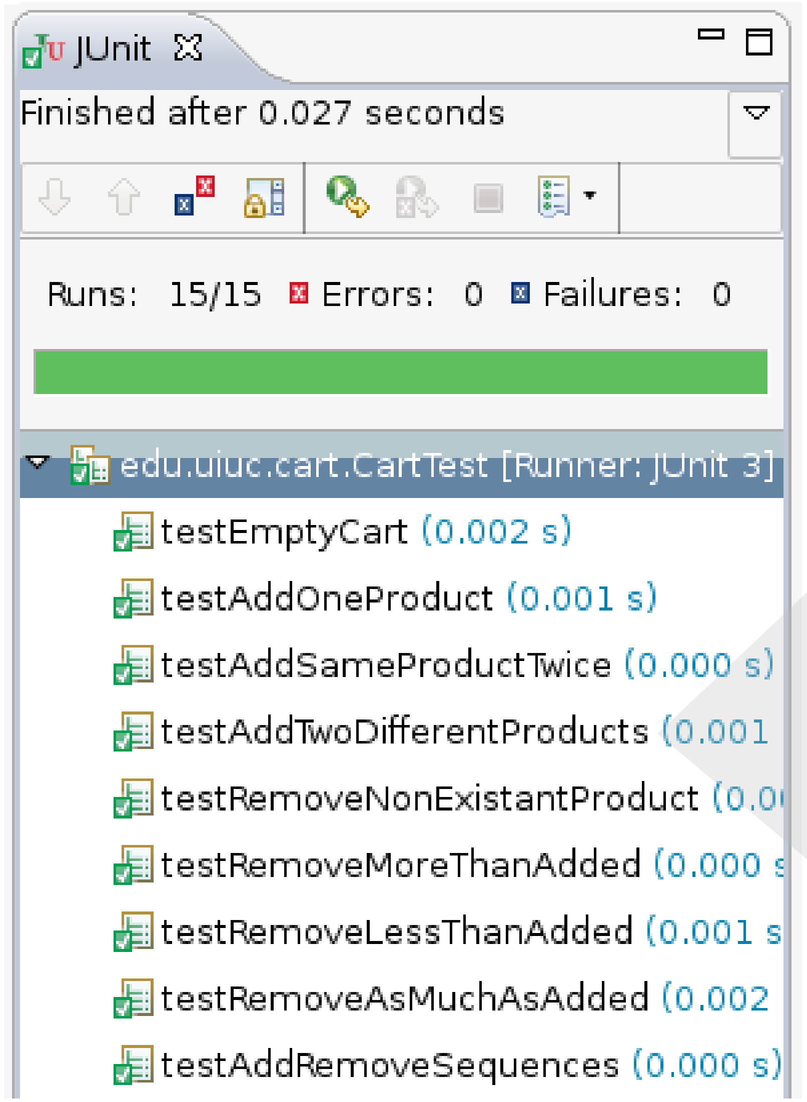
Brett Daniel
Vilas Jagannath
Danny Dig
Darko Marinov

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

This is Alice



Her unit tests pass



JUnit

Finished after 0.027 seconds

Runs: 15/15 Errors: 0 Failures: 0

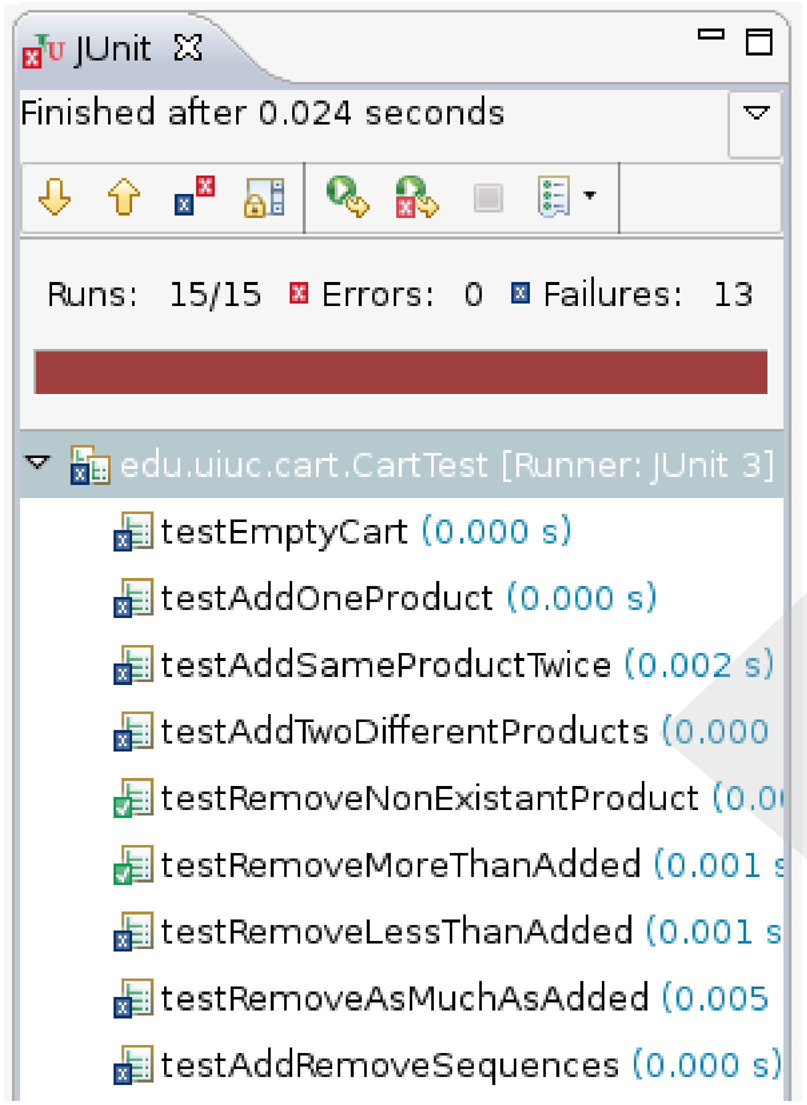
edu.uiuc.cart.CartTest [Runner: JUnit 3]

```
testEmptyCart (0.002 s)
testAddOneProduct (0.001 s)
testAddSameProductTwice (0.000 s)
testAddTwoDifferentProducts (0.001
testRemoveNonExistantProduct (0.0
testRemoveMoreThanAdded (0.000 s
testRemoveLessThanAdded (0.001 s
testRemoveAsMuchAsAdded (0.002
testAddRemoveSequences (0.000 s)
```

```
public class Cart {
    ...
    public double getTotalPrice() {...}
    public String getPrintedBill() {...}
    ...
}
```

```
public void testAddTwoDifferentProducts() {
    Cart cart = ...
    assertEquals(3.0, cart.getTotalPrice());
    assertEquals(
        "Discount: -$1.00, Total: $3.00",
        cart.getPrintedBill());
}
```

But requirements change



JUnit

Finished after 0.024 seconds

Runs: 15/15 Errors: 0 Failures: 13

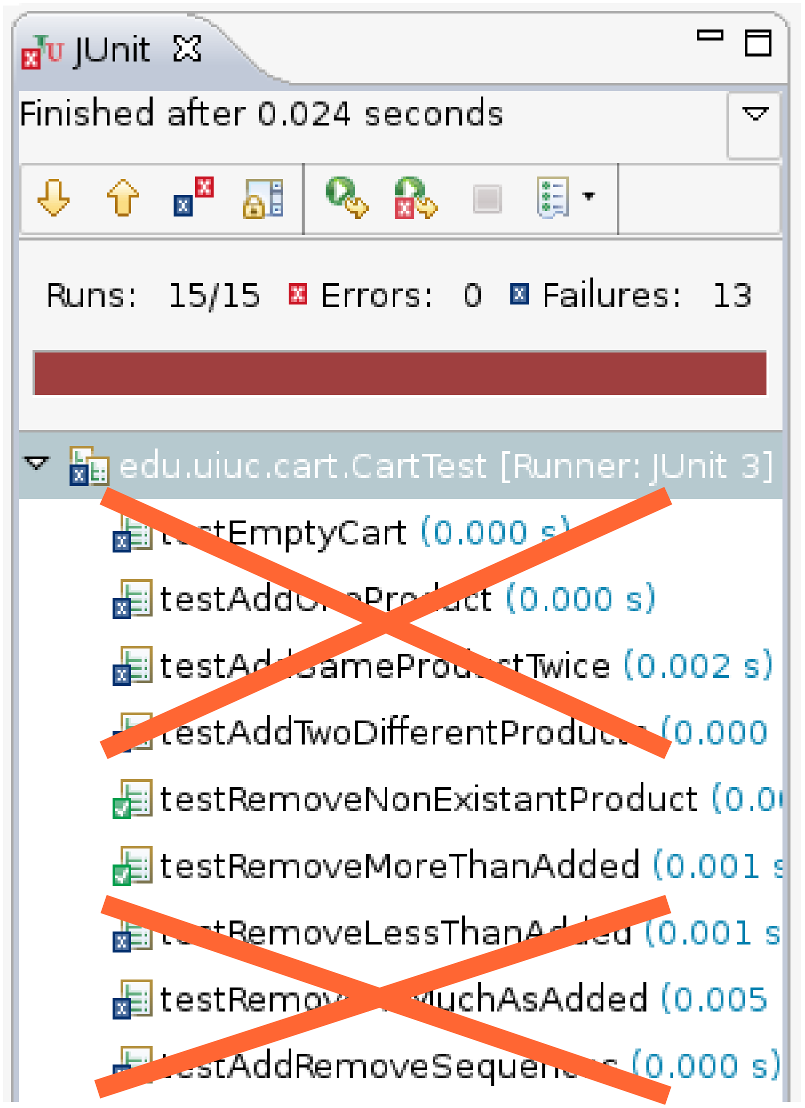
edu.uiuc.cart.CartTest [Runner: JUnit 3]

- testEmptyCart (0.000 s)
- testAddOneProduct (0.000 s)
- testAddSameProductTwice (0.002 s)
- testAddTwoDifferentProducts (0.000
- testRemoveNonExistantProduct (0.0
- testRemoveMoreThanAdded (0.001 s
- testRemoveLessThanAdded (0.001 s
- testRemoveAsMuchAsAdded (0.005
- testAddRemoveSequences (0.000 s)

```
public class Cart {
    ...
    public double getTotalPrice() {...}
    public String getPrintedBill() {...}
    ...
}
```

```
public void testAddTwoDifferentProducts() {
    Cart cart = ...
    assertEquals(3.0, cart.getTotalPrice());
    assertEquals(
        "Discount: -$1.00, Total: $3.00",
        cart.getPrintedBill());
}
```

She can delete broken tests



JUnit

Finished after 0.024 seconds

Runs: 15/15 Errors: 0 Failures: 13

edu.uiuc.cart.CartTest [Runner: JUnit 3]

~~testEmptyCart (0.000 s)~~

~~testAddOneProduct (0.000 s)~~

~~testAddSameProductTwice (0.002 s)~~

~~testAddTwoDifferentProducts (0.000~~

testRemoveNonExistantProduct (0.0

testRemoveMoreThanAdded (0.001 s

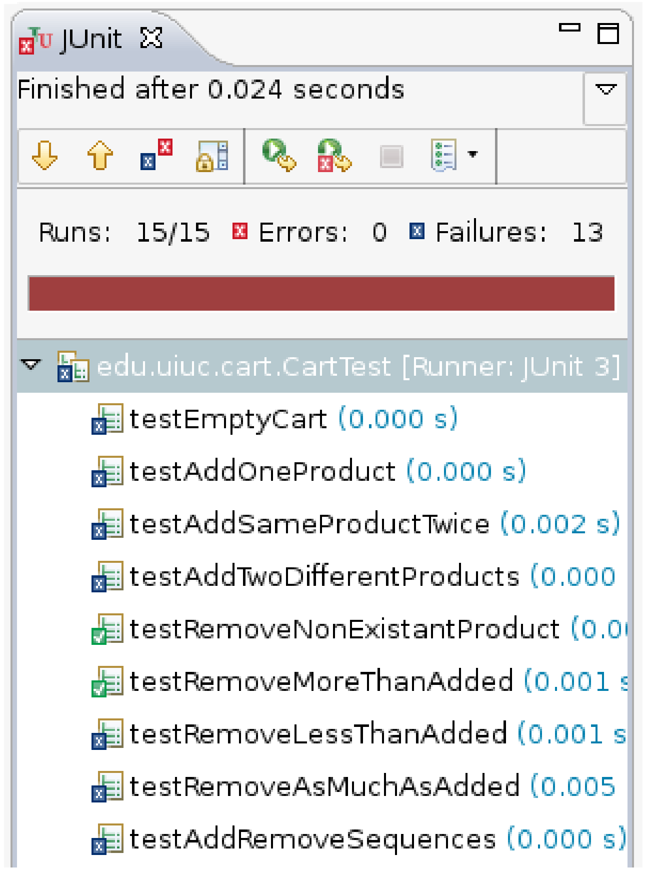
~~testRemoveLessThanAdded (0.001 s~~

~~testRemoveAsMuchAsAdded (0.005~~

~~testAddRemoveSequences (0.000 s)~~

But that reduces the quality of the test suite.

Repairing tests is preferable



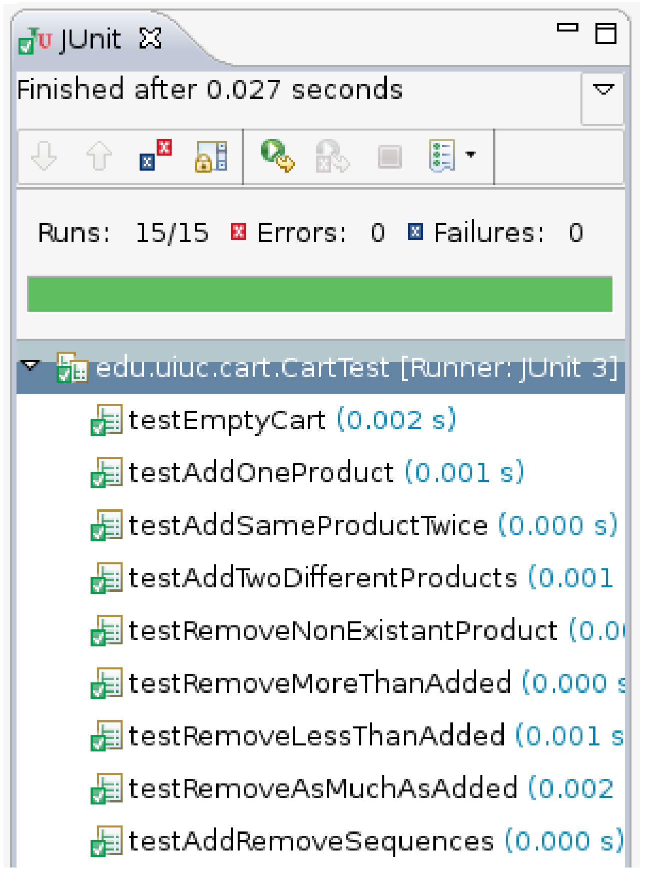
JUnit

Finished after 0.024 seconds

Runs: 15/15 Errors: 0 Failures: 13

edu.uiuc.cart.CartTest [Runner: JUnit 3]
- testEmptyCart (0.000 s)
- testAddOneProduct (0.000 s)
- testAddSameProductTwice (0.002 s)
- testAddTwoDifferentProducts (0.000
- testRemoveNonExistantProduct (0.0
- testRemoveMoreThanAdded (0.001 s
- testRemoveLessThanAdded (0.001 s
- testRemoveAsMuchAsAdded (0.005
- testAddRemoveSequences (0.000 s)

But that requires a lot of time and effort



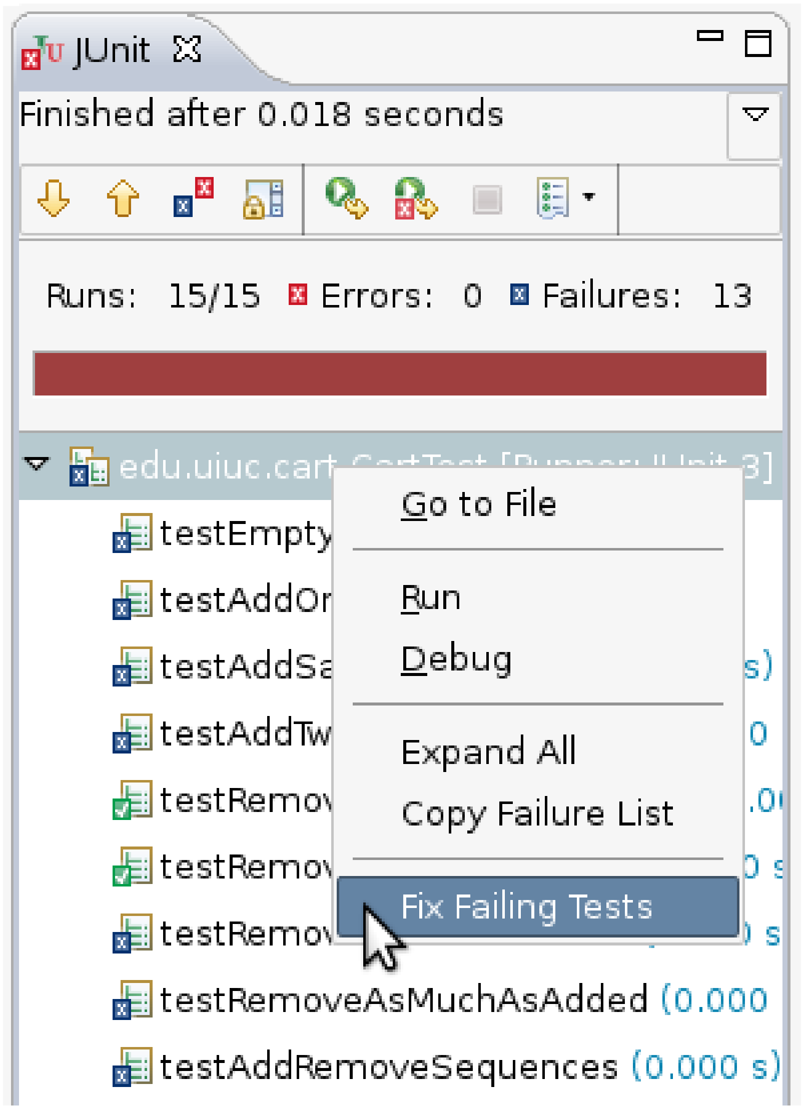
JUnit

Finished after 0.027 seconds

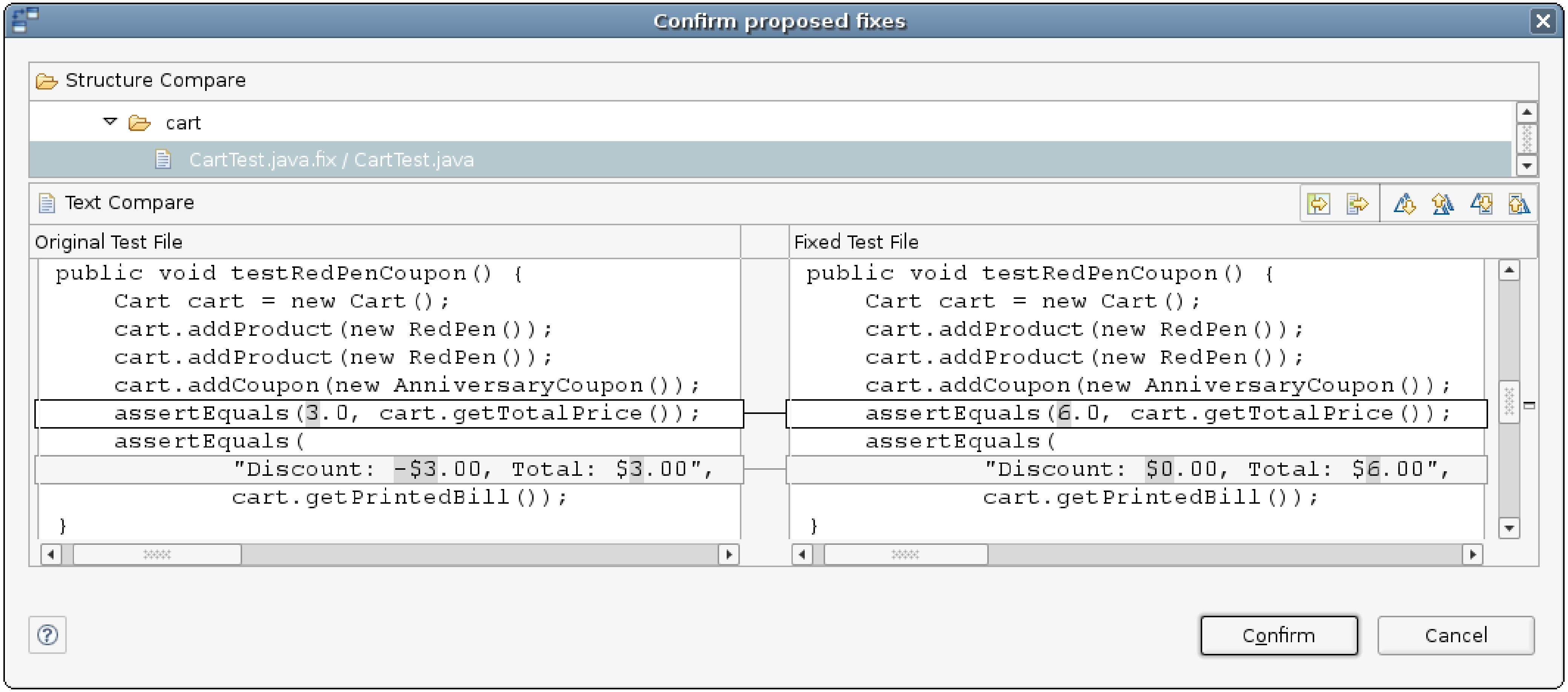
Runs: 15/15 Errors: 0 Failures: 0

edu.uiuc.cart.CartTest [Runner: JUnit 3]
- testEmptyCart (0.002 s)
- testAddOneProduct (0.001 s)
- testAddSameProductTwice (0.000 s)
- testAddTwoDifferentProducts (0.001
- testRemoveNonExistantProduct (0.0
- testRemoveMoreThanAdded (0.000 s
- testRemoveLessThanAdded (0.001 s
- testRemoveAsMuchAsAdded (0.002
- testAddRemoveSequences (0.000 s)

ReAssert suggests repairs



Alice decides whether to apply



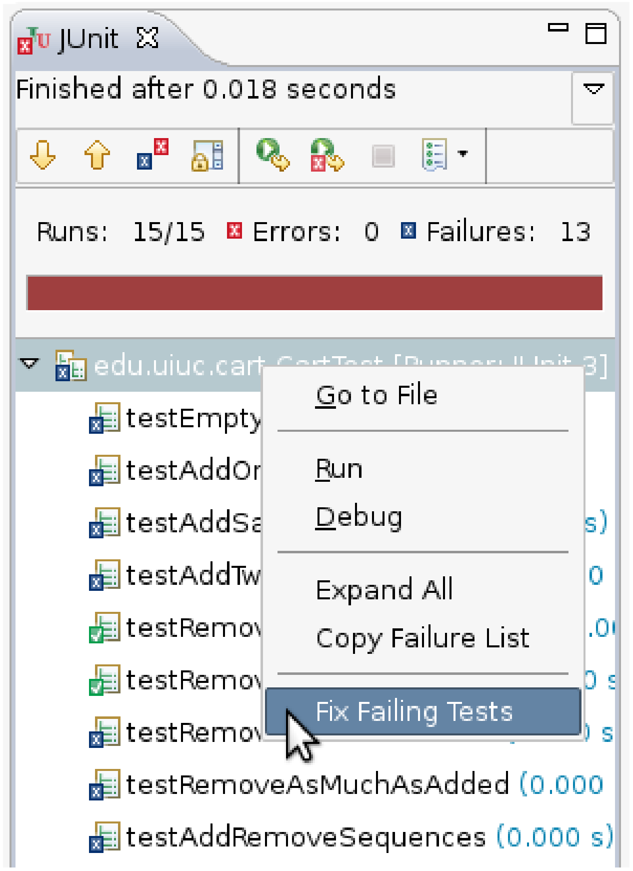
Original Test File

```
public void testRedPenCoupon() {
    Cart cart = new Cart();
    cart.addProduct(new RedPen());
    cart.addProduct(new RedPen());
    cart.addCoupon(new AnniversaryCoupon());
    assertEquals(3.0, cart.getTotalPrice());
    assertEquals(
            "Discount: -$3.00, Total: $3.00",
            cart.getPrintedBill());
}
```

Fixed Test File

```
public void testRedPenCoupon() {
    Cart cart = new Cart();
    cart.addProduct(new RedPen());
    cart.addProduct(new RedPen());
    cart.addCoupon(new AnniversaryCoupon());
    assertEquals(6.0, cart.getTotalPrice());
    assertEquals(
            "Discount: $0.00, Total: $6.00",
            cart.getPrintedBill());
}
```

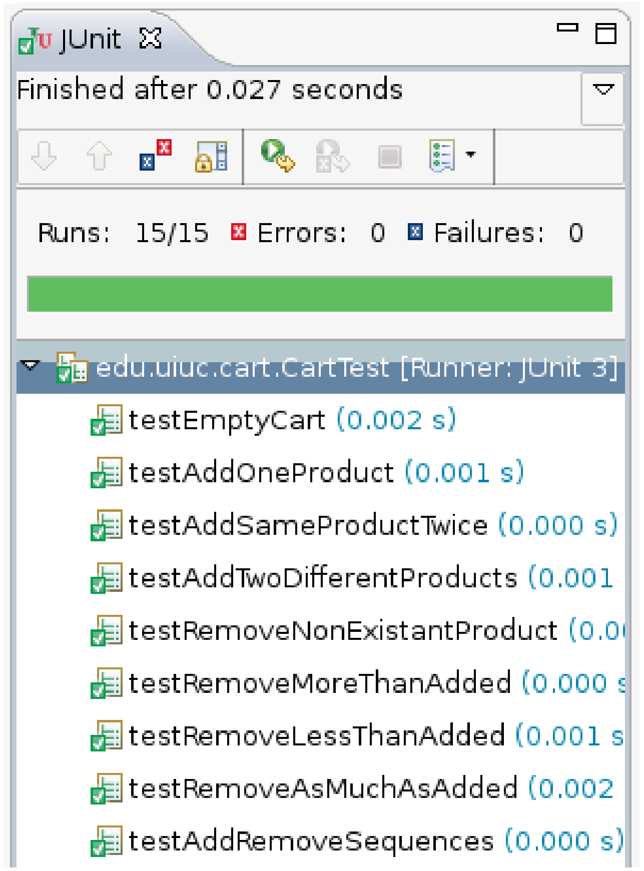
ReAssert reduces effort



Finished after 0.018 seconds

Runs: 15/15 Errors: 0 Failures: 13

edu.uiuc.cart.CartTest [Runner: JUnit 3]
testEmpty
testAddOr
testAddSa
testAddTw
testRemov
testRemov
testRemov
testRemoveAsMuchAsAdded (0.000
testAddRemoveSequences (0.000 s)



Finished after 0.027 seconds

Runs: 15/15 Errors: 0 Failures: 0

edu.uiuc.cart.CartTest [Runner: JUnit 3]
testEmptyCart (0.002 s)
testAddOneProduct (0.001 s)
testAddSameProductTwice (0.000 s)
testAddTwoDifferentProducts (0.001
testRemoveNonExistantProduct (0.0
testRemoveMoreThanAdded (0.000 s
testRemoveLessThanAdded (0.001 s
testRemoveAsMuchAsAdded (0.002
testAddRemoveSequences (0.000 s)

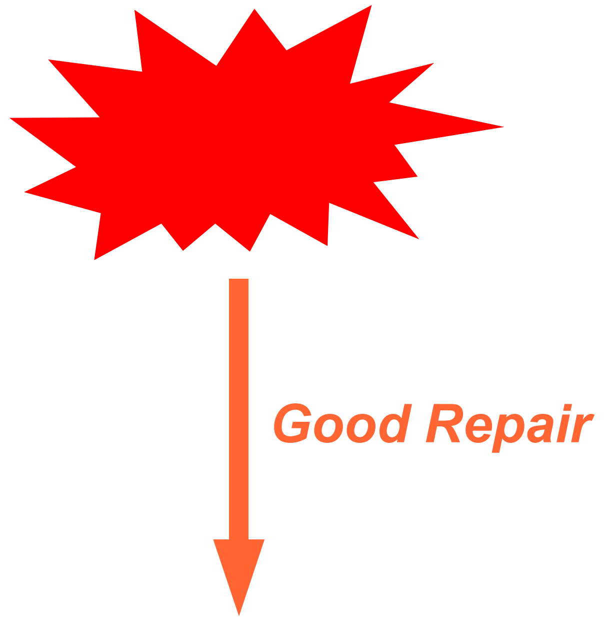
What is a Good Repair?

```
assertEquals(3.0, cart.getTotalPrice());
```

Bad Repair!

```
assertTrue(true);
```

Repair Criteria



Make tests **pass**

Make **minimal changes** to test code (not SUT)

Require developer **approval**

Produce **understandable** test code

Repair Strategies

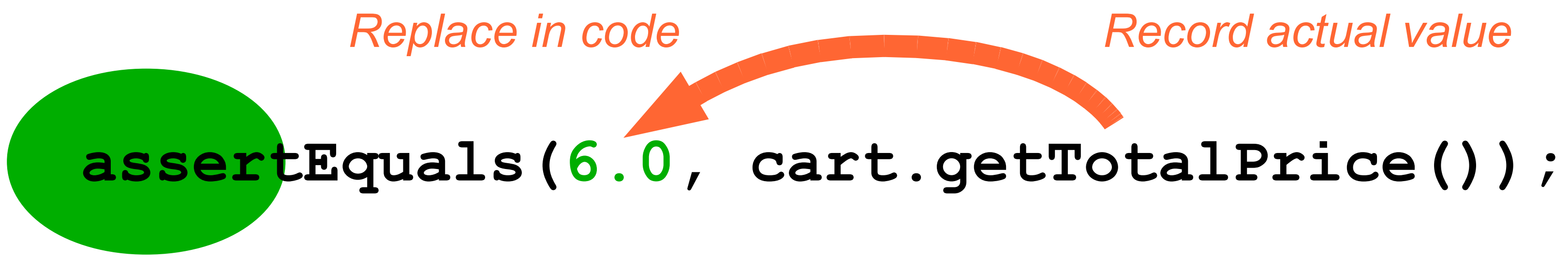
- Strategies specific to:
  - Static **structure** of the code
  - The **type** of failure
  - The **runtime values** that caused the failure

- Seven general strategies + custom strategies

Simple Assertion Failure

```java
assertEquals(3.0, cart.getTotalPrice());
```

Strategy: Replace Literal

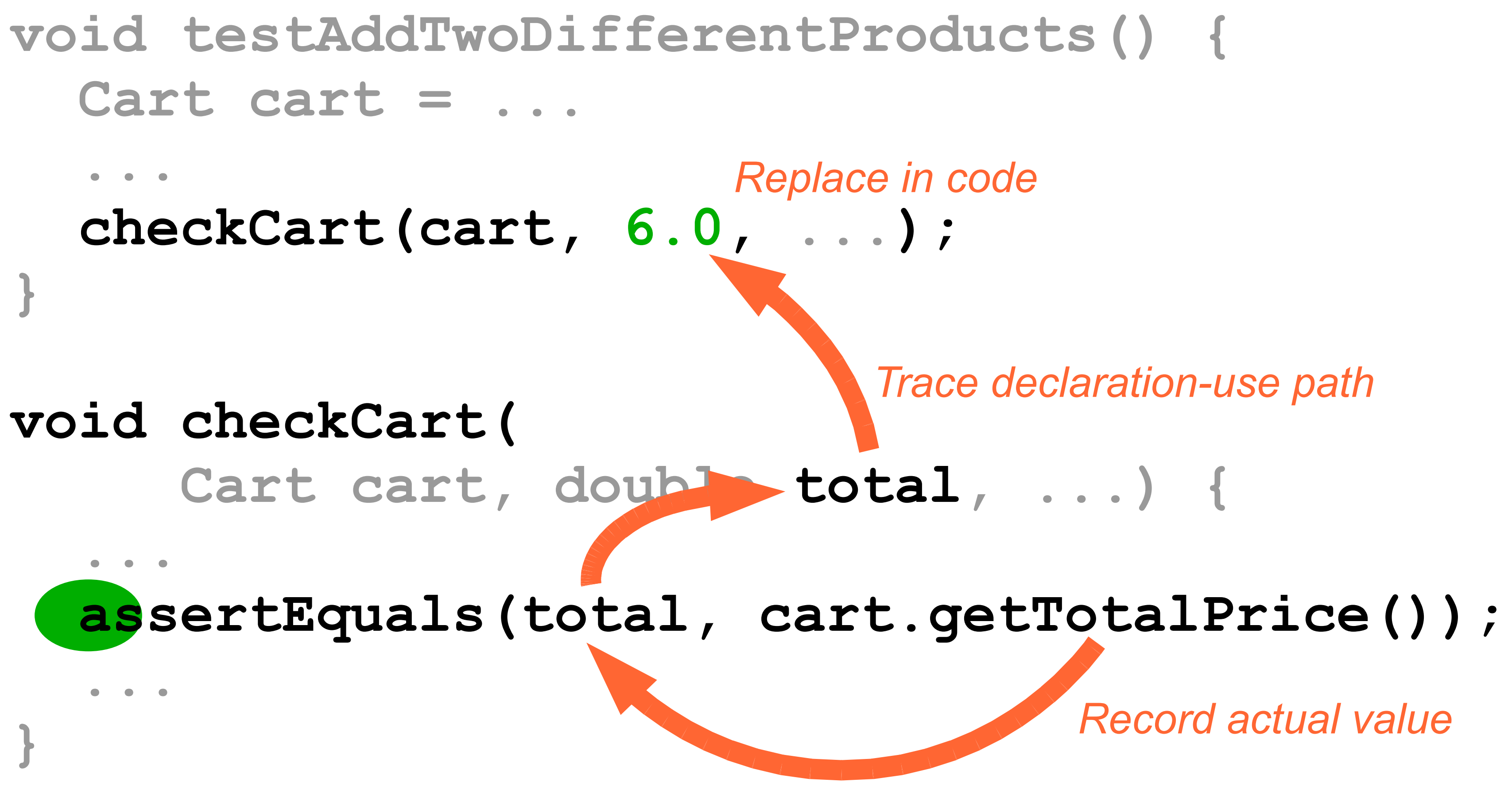


```
assertEquals(6.0, cart.getTotalPrice());
```

Failure in Helper Method

```
void testAddTwoDifferentProducts() {
  Cart cart = ...
  ...
  checkCart(cart, 3.0, ...);
}

void checkCart(
     Cart cart, double total, ...) {
  ...
  assertEquals(total, cart.getTotalPrice());
  ...
}
```

Strategy: Trace Declaration-Use Path



```
void testAddTwoDifferentProducts() {
   Cart cart = ...
   ...
   checkCart(cart, 6.0, ...);
}

void checkCart(
      Cart cart, double total, ...) {
   ...
   assertEquals(total, cart.getTotalPrice());
   ...
}
```

Object (In)Equality Failure

```
Product expected = ...
Product actual = ...
assertEquals(expected, actual);
```

Strategy: Expand Accessors

Expand accessors

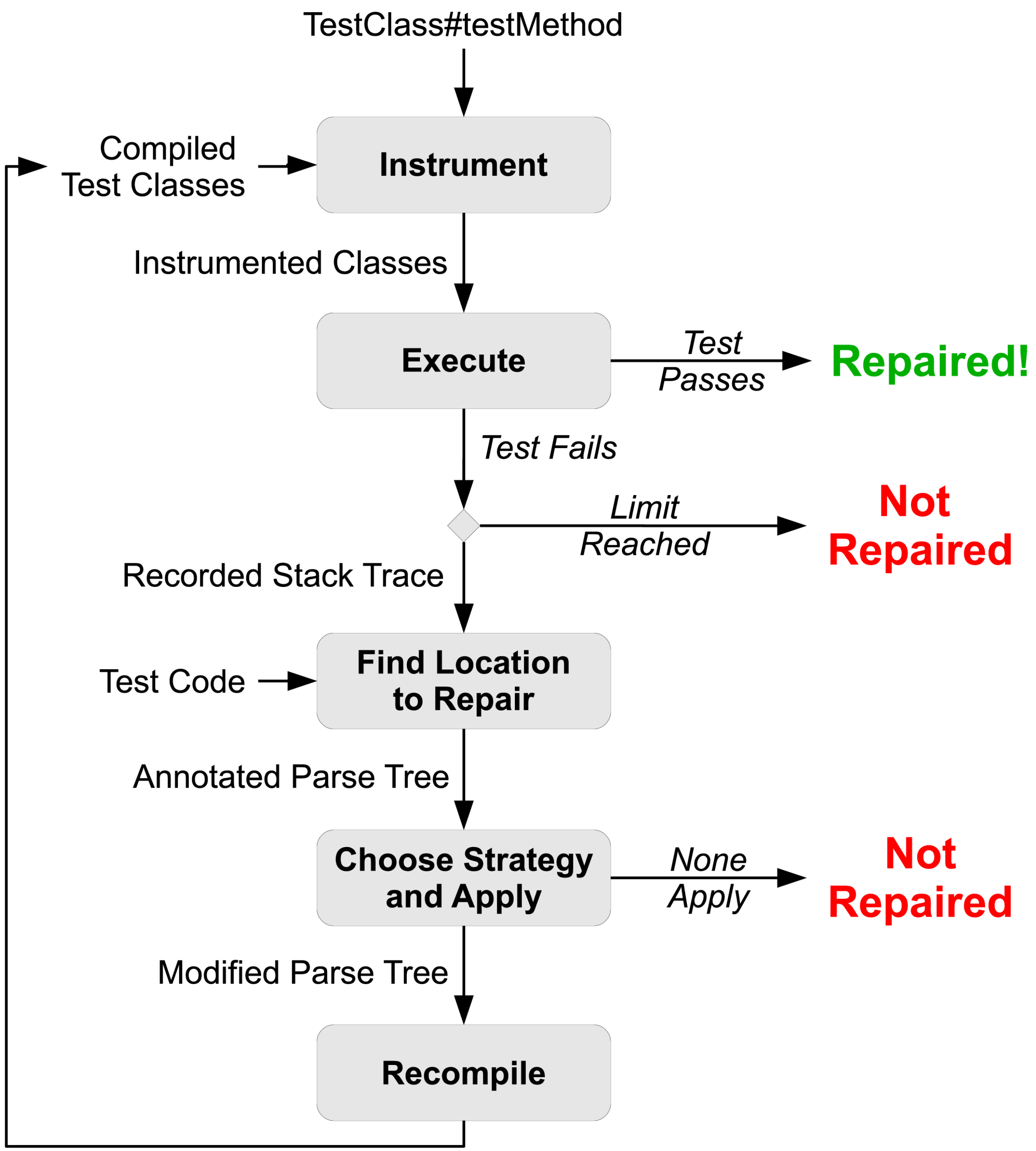
```
Product expected = ...
Product actual = ...
{
  assertEquals(         , actual.getPrice());
  assertEquals(         , actual.getDescription());
}
```

Strategy: Expand Accessors

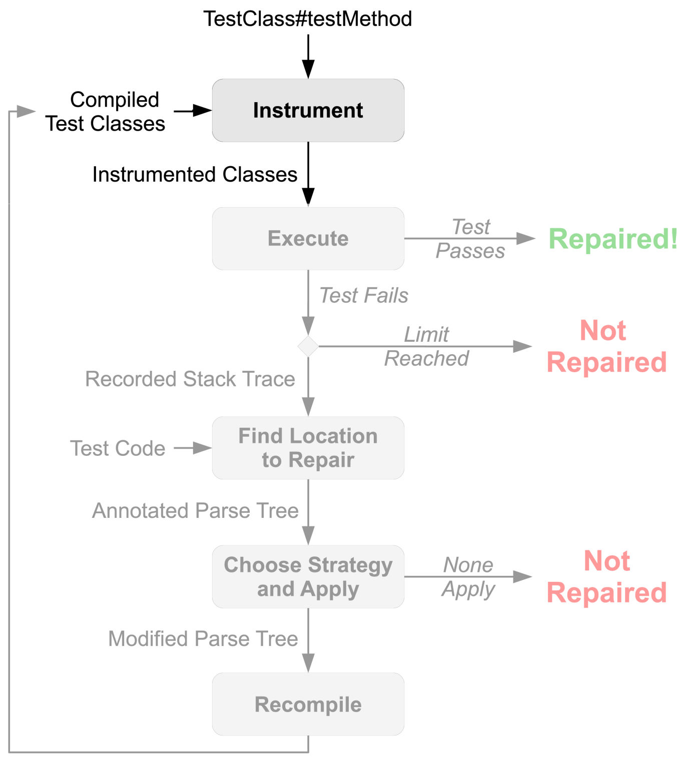
```
Product expected = ...
Product actual = ...
{
  assertEquals(expected.getPrice(), actual.getPrice());
  assertEquals("Red pen", actual.getDescription());
}
```

Expected and actual accessors equal

Actual accessor differs


TestClass#testMethod
Compiled Test Classes
Instrument
Instrumented Classes
Execute
Test Passes
Repaired!
Test Fails
Limit Reached
Not Repaired
Recorded Stack Trace
Test Code
Find Location to Repair
Annotated Parse Tree
Choose Strategy and Apply
None Apply
Not Repaired
Modified Parse Tree
Recompile

Instrument

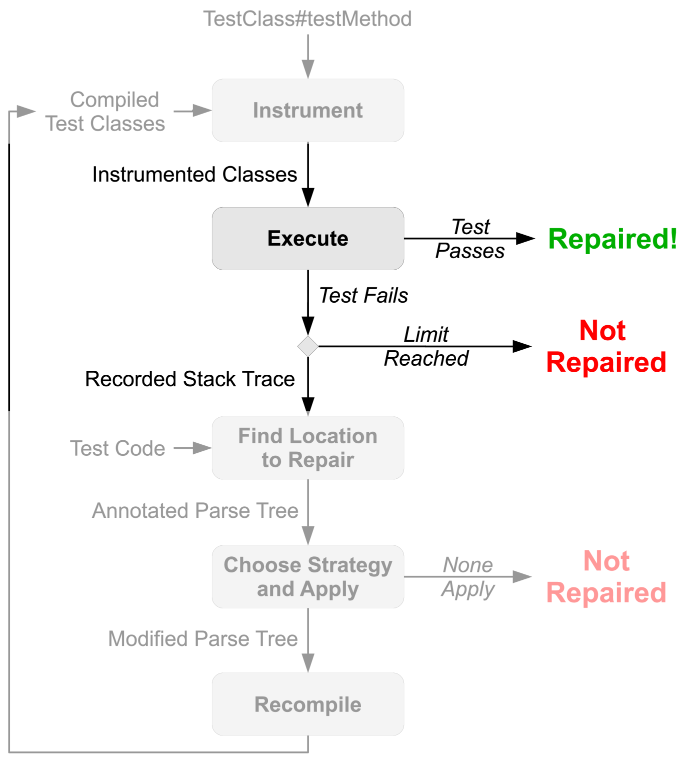


```
public static void assertEquals (
        Object expected,
        Object actual) {
    try {
        // ...assert expected.equals(actual)
    }
    catch (Error e) {
        throw new RecordedAssertFailure(
            e, expected, actual);
    }
}
```

If assertion fails...

...then record values that caused failure

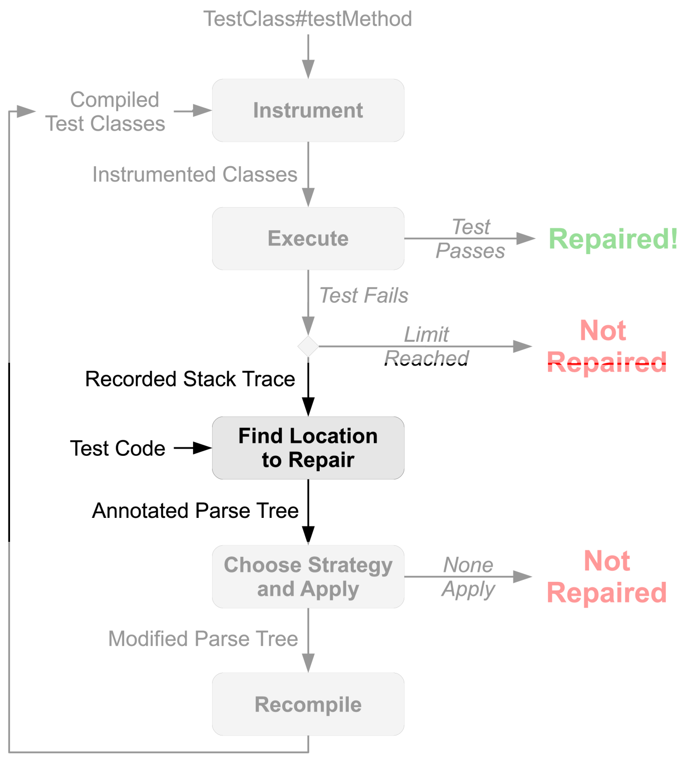
Execute



```
assertEquals(3.0, cart.getTotalPrice());
```

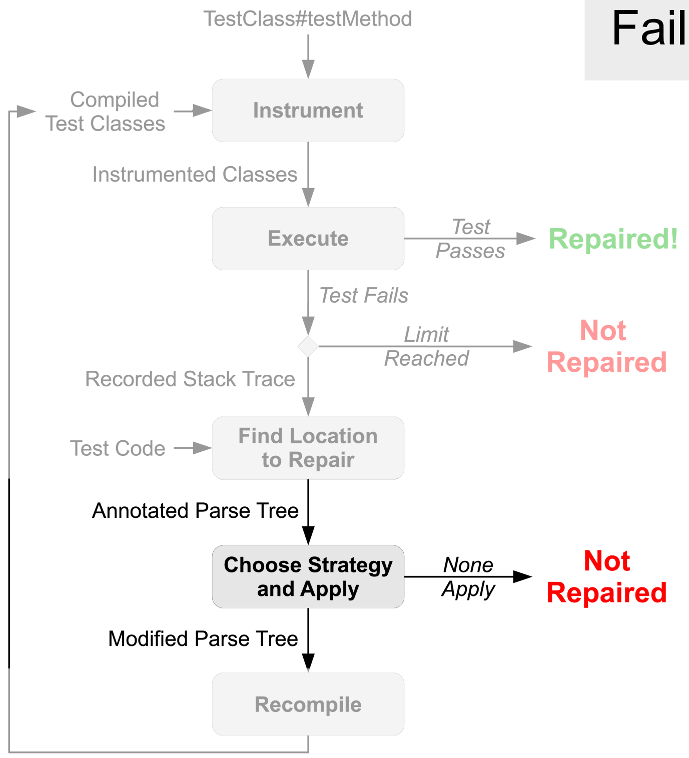
```
throw RecordedAssertFailure(e, 3.0, 6.0);

  edu.illinois.reassert.RecordedAssertFailure:
  org.junit.AssertionFailedError:
  expected:<3.0> but was:<6.0>
    at org.junit.Assert.assertEquals(Assert.java:116)
    at CartTest.testRedPenCoupon(CartTest.java:6)
    ...
```

Find Repair Location



```
edu.illinois.reassert.RecordedAssertFailure:
org.junit.AssertionFailedError:
expected:<3.0> but was:<6.0>
  at org.junit.Assert.assertEquals(Assert.java:116)
  at CartTest.testRedPenCoupon(CartTest.java:6)
  ...
```

Choose Strategy and Apply



Failure type: assertion failure

Recorded values: literals

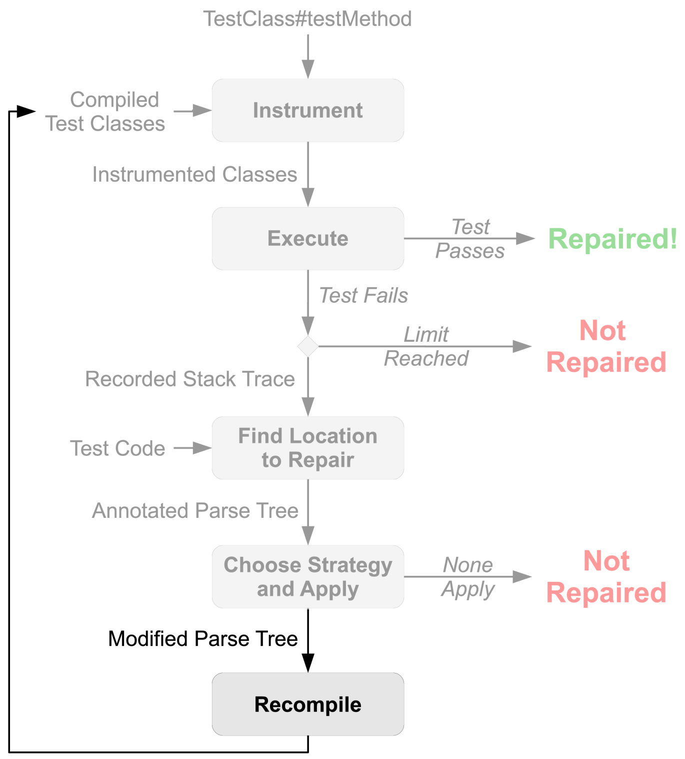
```
assertEquals(3.0, cart.getTotalPrice());
```

Structure: **`assertEquals`** with literal

∴ ***Replace Literal in Assertion*** **strategy**

```
assertEquals(6.0, cart.getTotalPrice());
```

Recompile and Repeat



```
assertEquals(6.0, cart.getTotalPrice());

assertEquals(
    "Discount: -$1.00, Total: $3.00",
    cart.getPrintedBill());
```

Evaluation

Q1: How many failures can ReAssert **repair**?

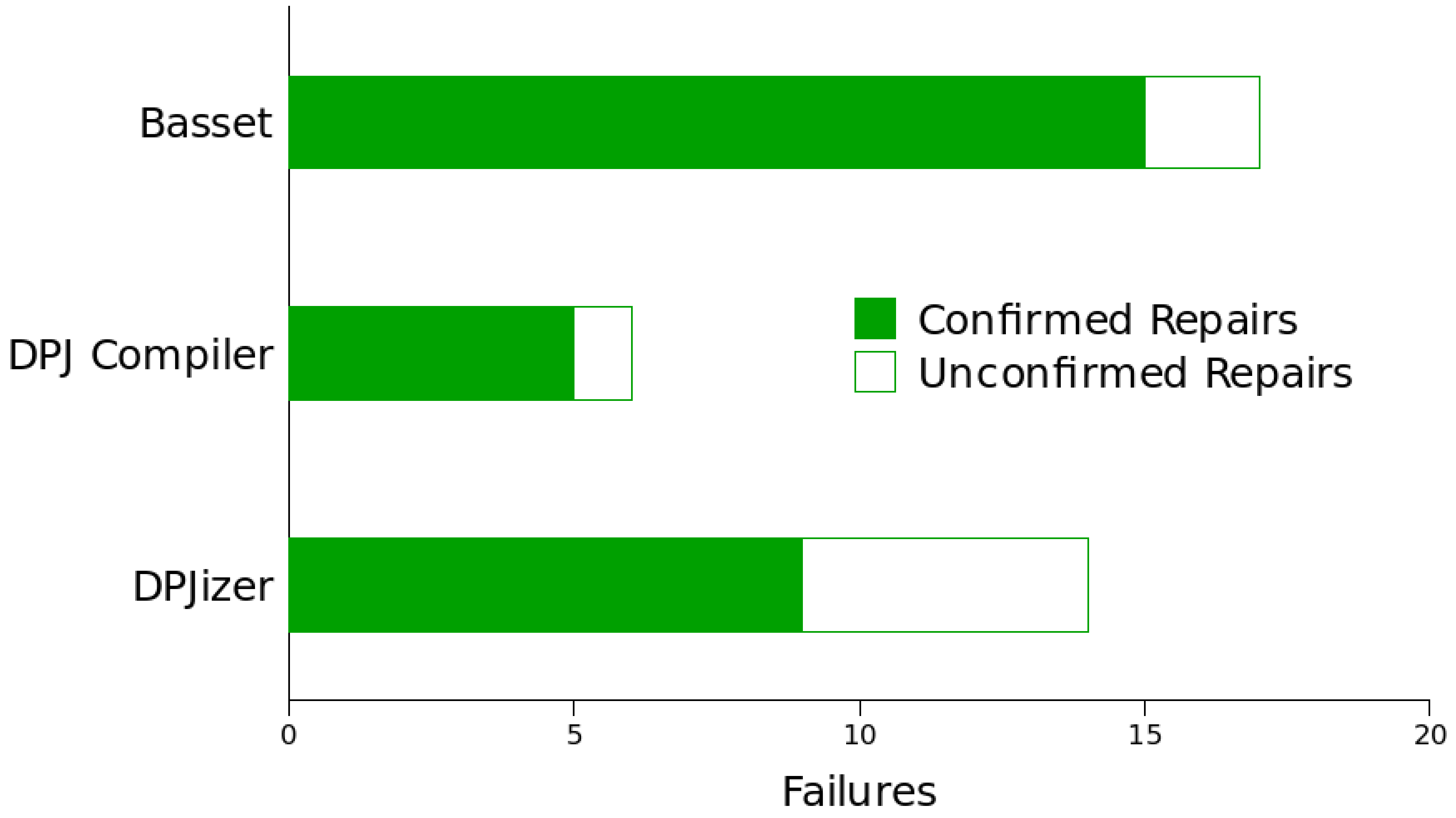
Q2: Are ReAssert's suggested repairs **useful**?

Q3: Does ReAssert **reveal** or **hide** regressions?

Evaluation

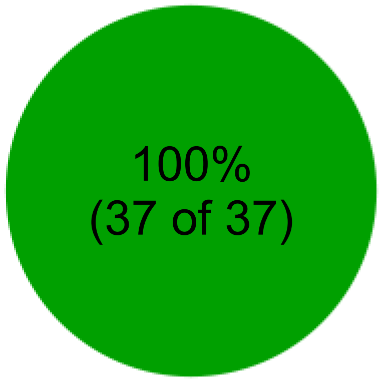
	Repairs?	Useful?	Regressions?
Case Studies	✓	✓	✓
Controlled User Study	✓	✓	✓
Failures in Open-Source Software	✓		

Case Studies

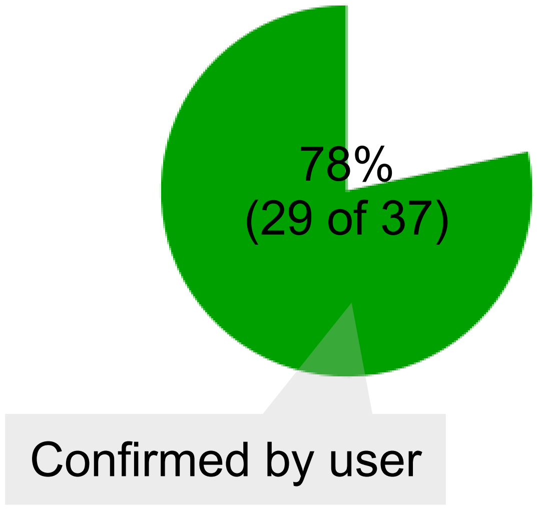


Case Studies

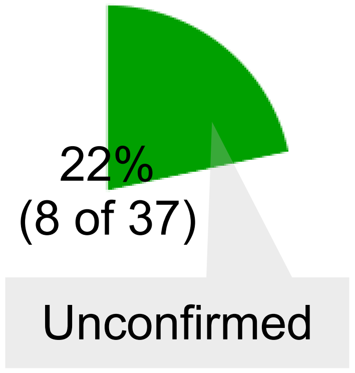
Repairs?



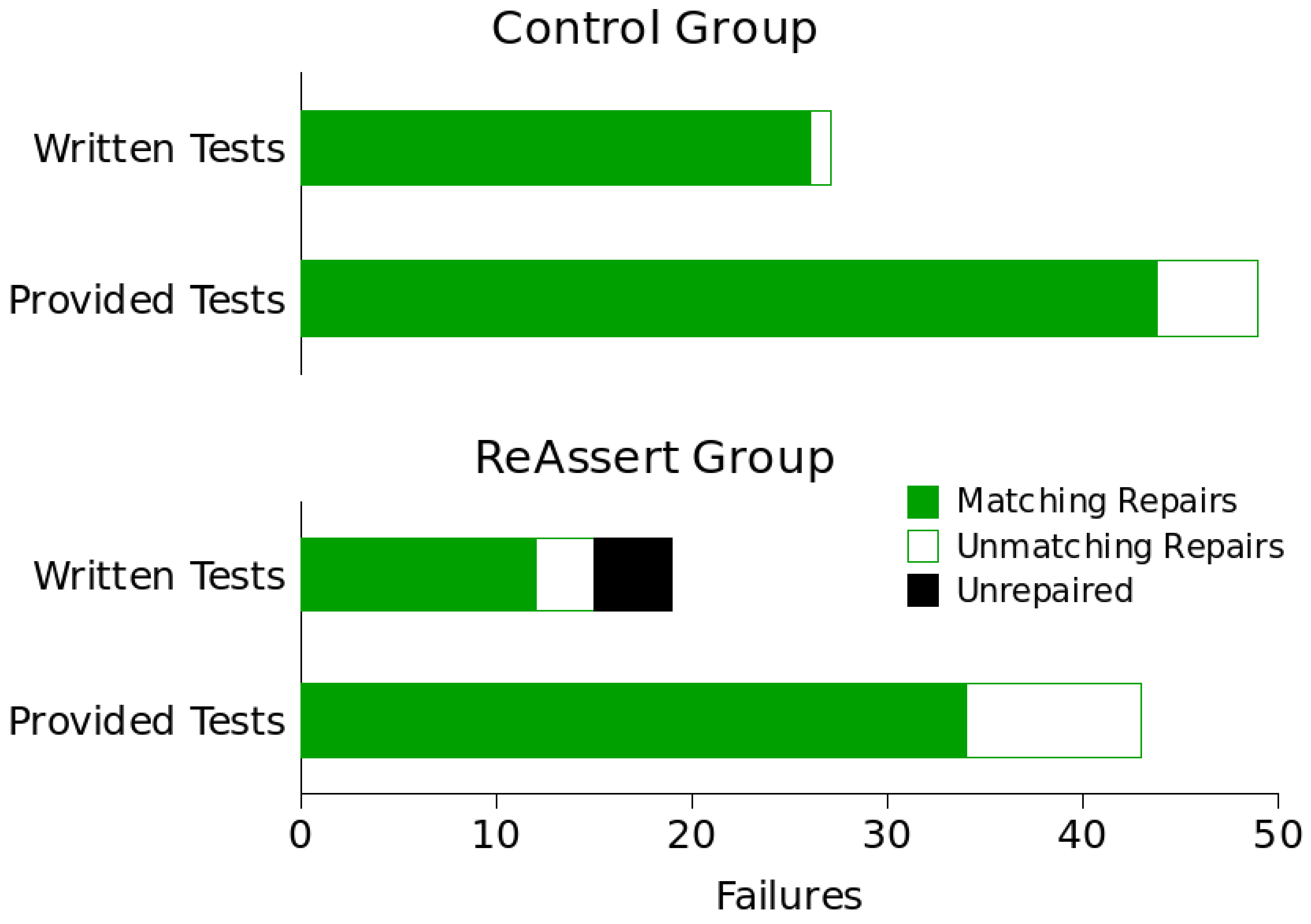
Useful?



Regressions?

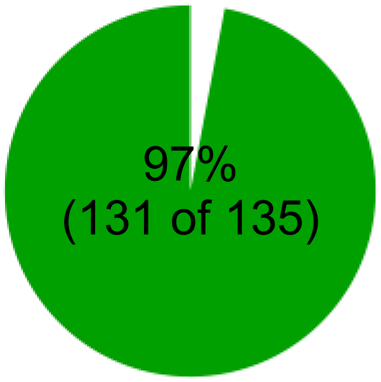


Controlled User Study

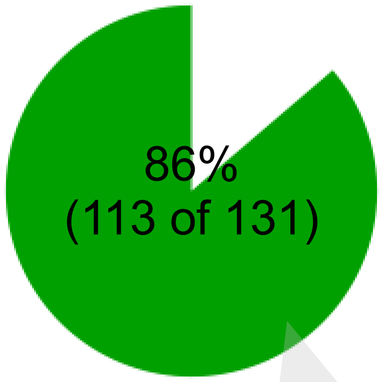


Controlled User Study

Repairs?

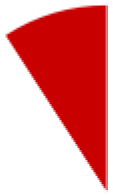


Useful?



Matching repairs

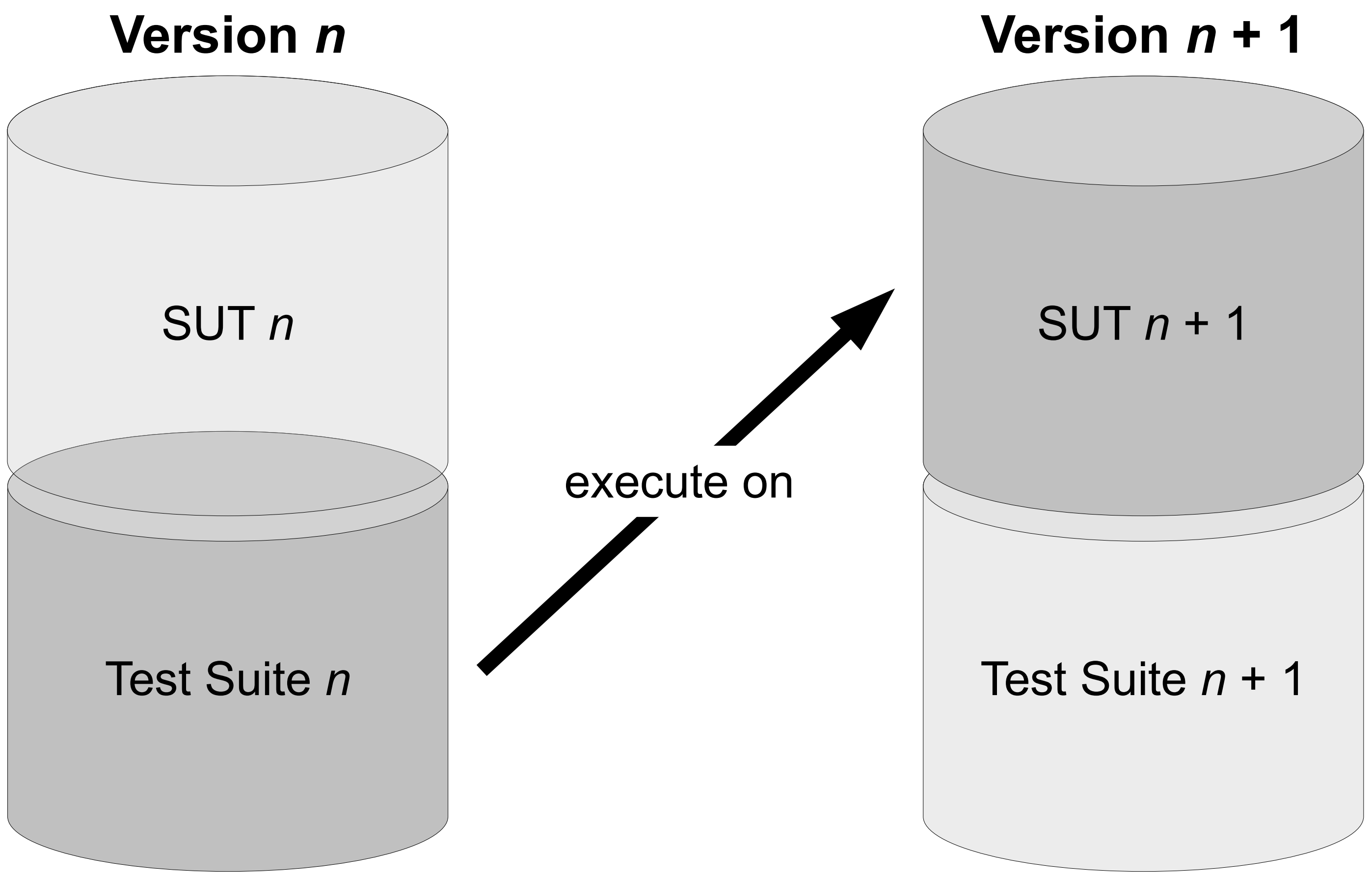
Regressions?



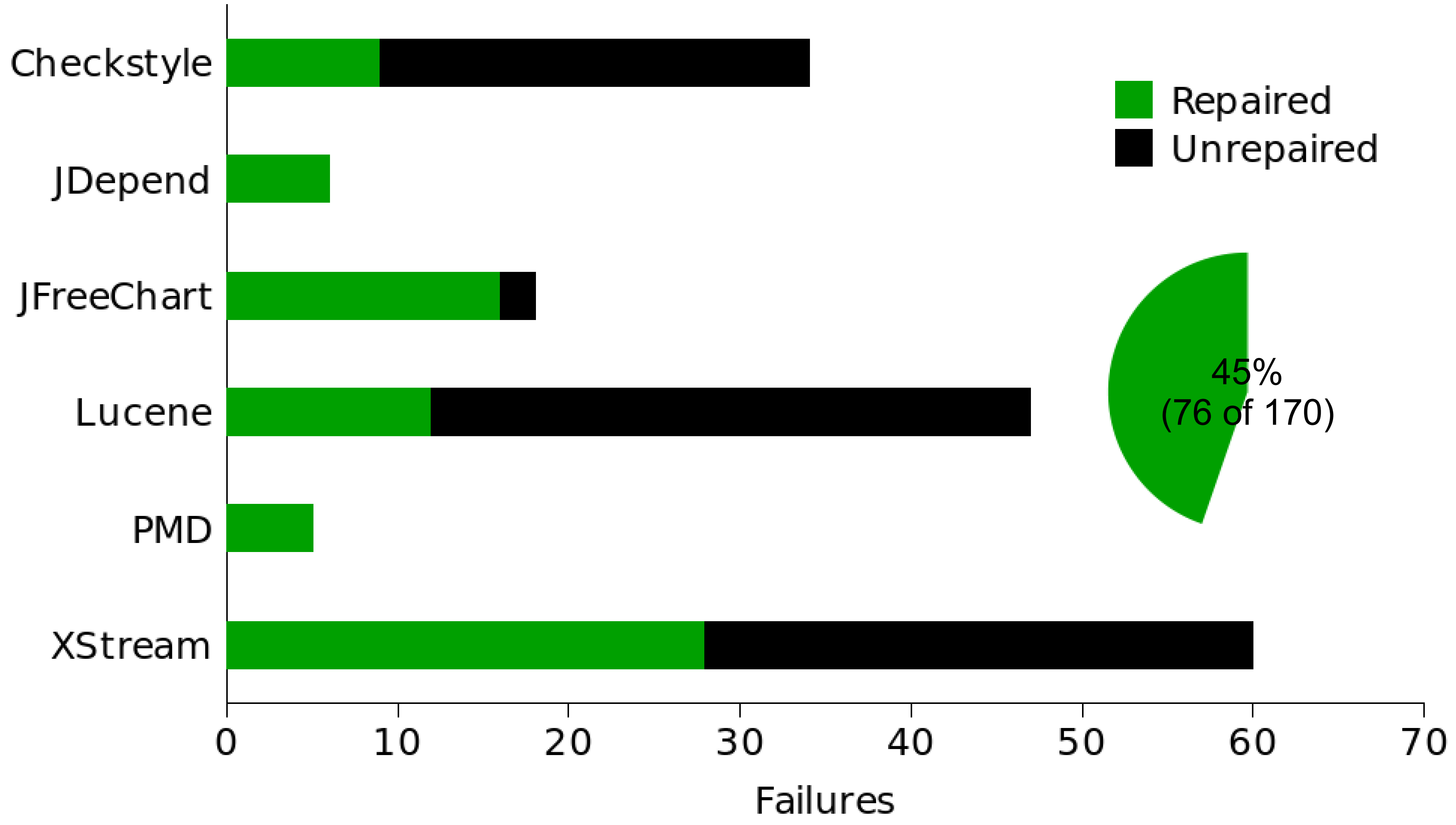
9%
(12 of 131)

vs. 8 introduced by the control group

Failures in Open-Source Software



Failures in Open-Source Software



Unrepairable Failures

- Nondeterminism

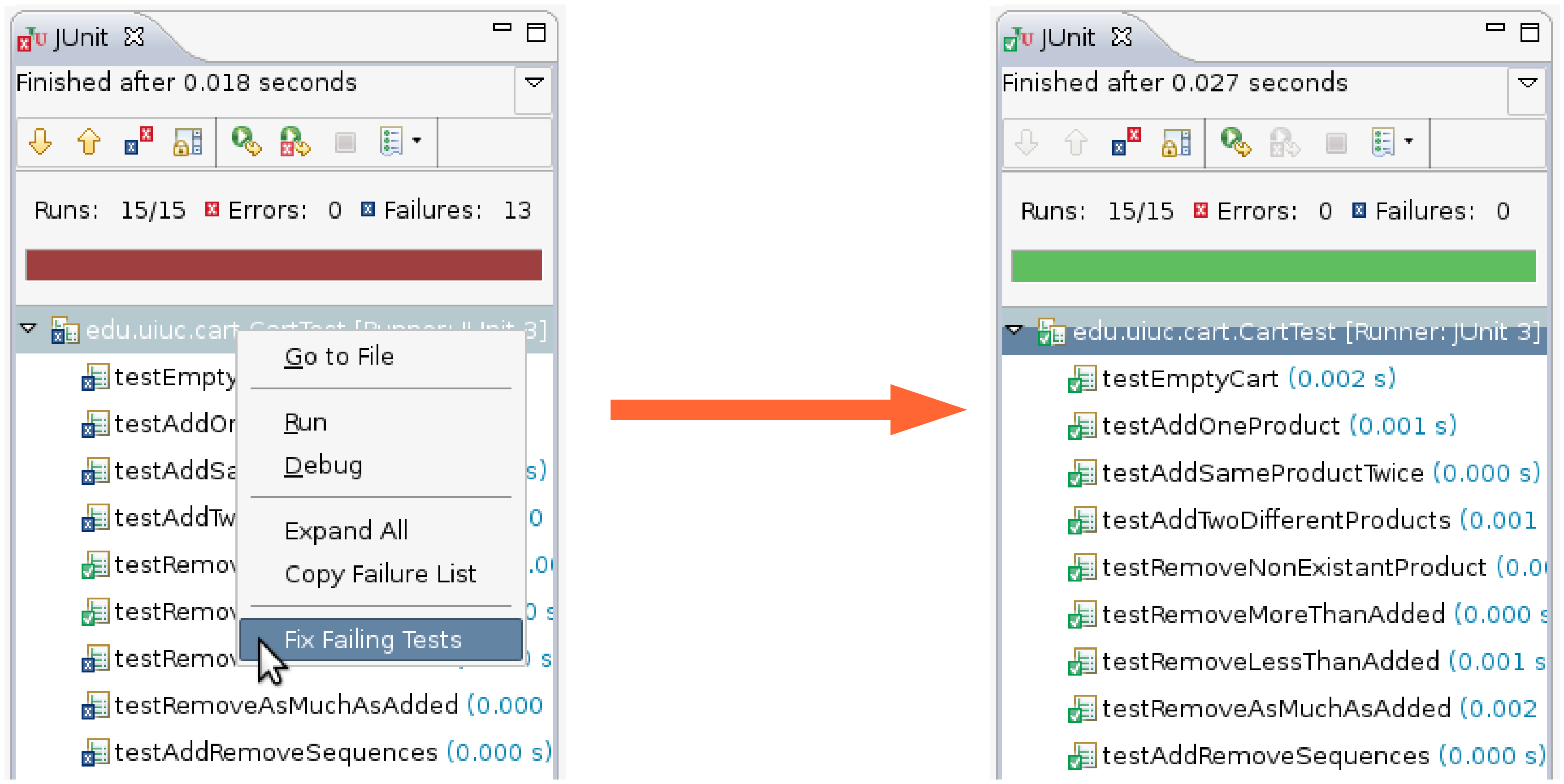
```
assertEquals(..., cart.getPurchaseDate());
```

- Multiple contexts

```
for (Product product : cart.getProducts()) {
   assertEquals(3.0, product.getPrice());
}
```

- No applicable strategies

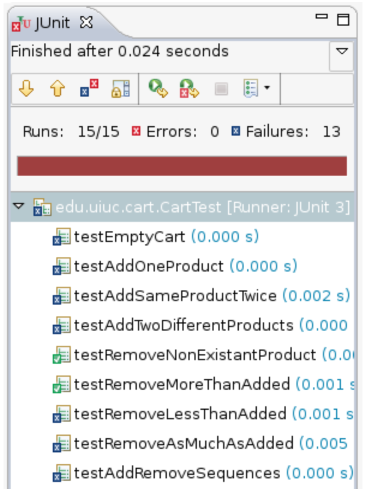
```
if (...) {
  expected = 3.0;
}
assertEquals(expected, cart.getTotalPrice());
```



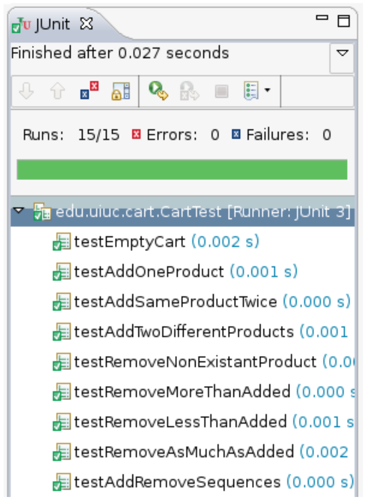
```
Finished after 0.018 seconds
Runs:  15/15   Errors:  0   Failures:  13
edu.uiuc.cart.CartTest [Runner: JUnit 3]
    testEmpty
    testAddOr
    testAddSa                                s)
    testAddTw                                0
    testRemov                                .0
    testRemov                                0 s
    testRemov                                ) s
    testRemoveAsMuchAsAdded (0.000
    testAddRemoveSequences (0.000 s)
```

```
Finished after 0.027 seconds
Runs:  15/15   Errors:  0   Failures:  0
edu.uiuc.cart.CartTest [Runner: JUnit 3]
    testEmptyCart (0.002 s)
    testAddOneProduct (0.001 s)
    testAddSameProductTwice (0.000 s)
    testAddTwoDifferentProducts (0.001
    testRemoveNonExistantProduct (0.0
    testRemoveMoreThanAdded (0.000 s
    testRemoveLessThanAdded (0.001 s
    testRemoveAsMuchAsAdded (0.002
    testAddRemoveSequences (0.000 s)
```

http://mir.cs.illinois.edu/reassert



```
JUnit
Finished after 0.024 seconds
Runs:  15/15   Errors:  0   Failures:  13
edu.uiuc.cart.CartTest [Runner: JUnit 3]
    testEmptyCart (0.000 s)
    testAddOneProduct (0.000 s)
    testAddSameProductTwice (0.002 s)
    testAddTwoDifferentProducts (0.000
    testRemoveNonExistantProduct (0.0
    testRemoveMoreThanAdded (0.001 s
    testRemoveLessThanAdded (0.001 s
    testRemoveAsMuchAsAdded (0.005
    testAddRemoveSequences (0.000 s)
```



```
JUnit
Finished after 0.027 seconds
Runs:  15/15   Errors:  0   Failures:  0
edu.uiuc.cart.CartTest [Runner: JUnit 3]
    testEmptyCart (0.002 s)
    testAddOneProduct (0.001 s)
    testAddSameProductTwice (0.000 s)
    testAddTwoDifferentProducts (0.001
    testRemoveNonExistantProduct (0.0
    testRemoveMoreThanAdded (0.000 s
    testRemoveLessThanAdded (0.001 s
    testRemoveAsMuchAsAdded (0.002
    testAddRemoveSequences (0.000 s)
```

Test-Driven Development	**Test Repair**
Make tests **fail**	
...by changing tests	...by changing SUT
Make tests **pass**	
...by changing SUT	...by changing tests