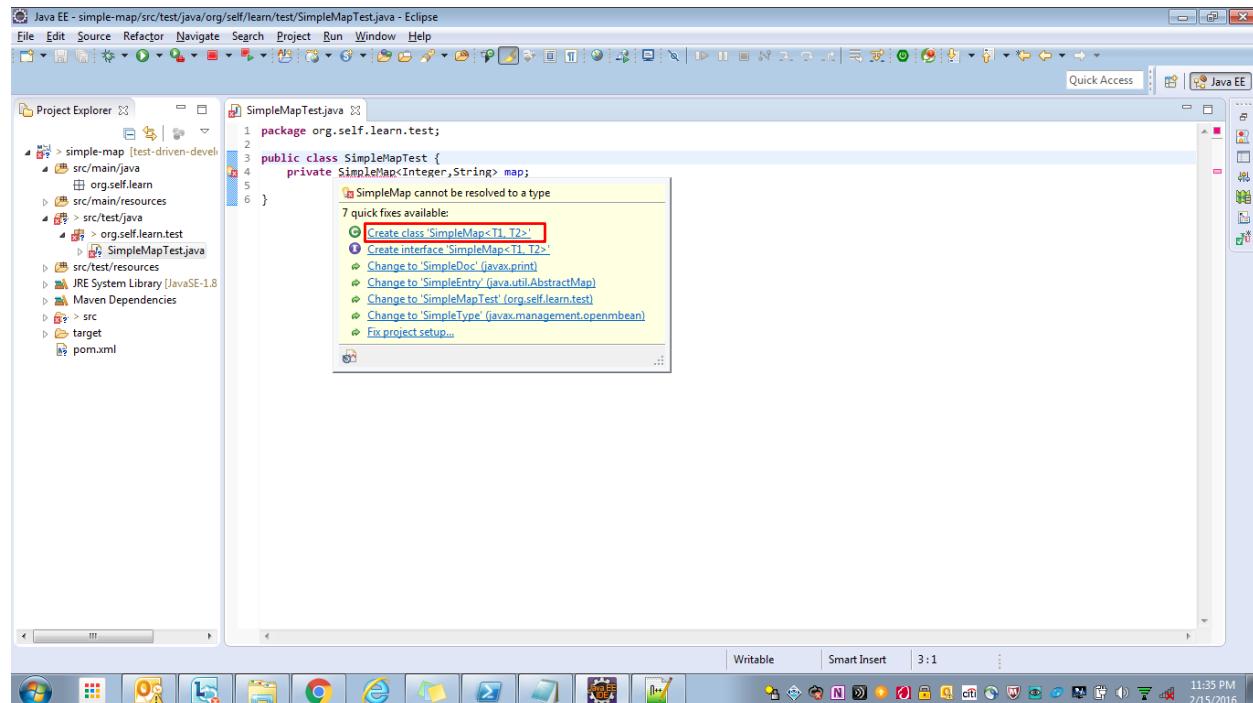


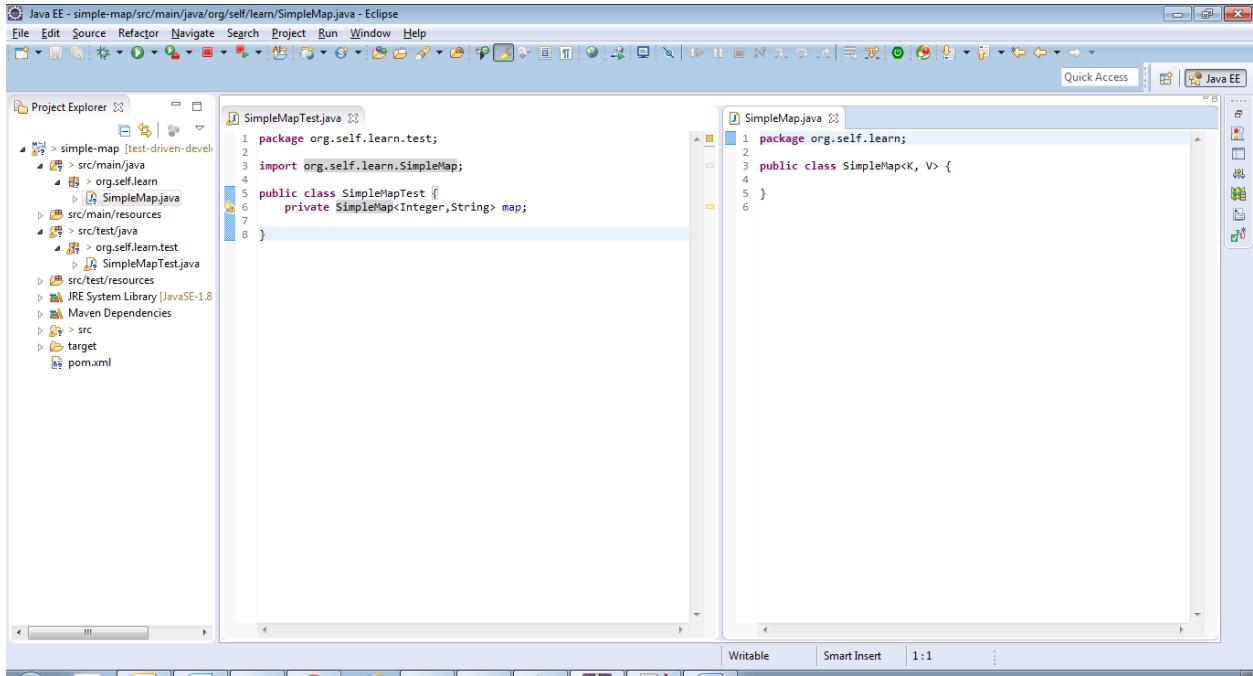
Simple Map without using Java Collections Framework (JCF)

1. Can have Null 'Key'
2. Can have null 'values'
3. Cannot contain duplicate 'keys'
4. May not retain order in which "keys/values" were saved →**No Test Required**
5. Performance is NOT a concern i.e. we will not implement Hash Table like features e.g. bucket, hash values etc. Also redundant iterations are allowed →**No Test Required**
6. Manages size dynamically
7. Throws exception if no such key found
8. Throws exception if duplicate key found
9. Should be able to perform the following operations:
 - a. Create Simple Parametrized Map (Generics)
 - b. Save 'key' and 'value' pair
 - c. Retrieve value given a 'key'
 - d. Remove a "key-value" pair given a 'key'
 - e. Retrieve size of the map i.e. number of "key-value" pairs
 - f. Check if a given 'key' is present
 - g. Check if a given 'value' is present

Let's start by creating the test class and trying to instantiate the test object. I prefer to use the quick fix to create the class/methods under test. This, although might sound a bit exaggerated, ensures that our test drives our implementation.



Make note of the eclipse window... this is our preferred style for TDD i.e. having test and implementation class side by side.



Key idea behind TDD is that you start by test driving that smallest, simplest and tiniest bit and build on top of it. Start with a blank mind without having any preconceived notion of the implementation. Think about what (**and not how!**) would you like your class to do.

Let's write our first test - "testEmptyMapCreation()"... Notice how your test tells what your implementation is.

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. In `SimpleMapTest.java`, there is a call to `map.size()`. A tooltip appears, indicating that the method `size()` is undefined for the type `SimpleMap<Integer, String>`. It provides two quick fixes: 'Create method size()' in type SimpleMap' and 'Add cast to 'map''. The `SimpleMap.java` file shows the implementation of the `size()` method returning -1.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4 import static org.junit.Assert.assertThat;
5
6 import org.junit.Test;
7 import org.self.learn.SimpleMap;
8
9 public class SimpleMapTest {
10     private SimpleMap<Integer, String> map;
11
12     @Test
13     public void testEmptyMapCreation() {
14         map = new SimpleMap<>();
15         assertThat(0, is(equalTo(map.size())));
16     }
17 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4
5     public int size() {
6         return -1;
7     }
8
9
10 }
11

```

I used quick fix on my test class to create my implementation again... With that, let's run the test which would obviously fail as we expect it to...

The screenshot shows the Eclipse IDE interface with the JUnit view open. The test `testEmptyMapCreation()` has run successfully, indicated by the status bar showing 'Runs: 1/1 Errors: 0 Failures: 1'. The failure trace shows a `java.lang.AssertionError` where the expected value is less than 1 but the actual value is 0.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.equalTo;
4
5 public class SimpleMapTest {
6     private SimpleMap<Integer, String> map;
7
8     @Test
9     public void testEmptyMapCreation() {
10         map = new SimpleMap<>();
11         assertThat(0, is(equalTo(map.size())));
12     }
13 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4
5     public int size() {
6         return -1;
7     }
8
9
10 }

Markers Properties Servers Data Source Explorer Snippets JUnit
Finished after 0.026 seconds
Runs: 1/1 Errors: 0 Failures: 1
Failure Trace
java.lang.AssertionError:
Expected: is <1>
  but: was <0>
at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
at org.self.learn.test.SimpleMapTest.testEmptyMapCreation(SimpleMapTest.java:16)

```

Returning zero takes care of that... (note that the test completes successfully). Idea is to only implement how much the tests force us to... It ensures that you have 100% code coverage.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

```

File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Java EE Quick Access
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4
5 public class SimpleMapTest {
6     private SimpleMap<Integer, String> map;
7
8     @Test
9     public void testEmptyMapCreation() {
10         map = new SimpleMap<>();
11         assertThat(0, is(equalTo(map.size())));
12     }
13 }

```

```

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     public int size() {
5         return 0;
6     }
7 }

```

Markers Properties Servers Data Source Explorer Snippets JUnit Finished after 0.022 seconds Runs: 1/1 Errors: 0 Failures: 0 Failure Trace

org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.000 s)

Writable Smart Insert 16:6 11:55 PM 2/15/2016

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

```

File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Java EE Quick Access
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4
5 @RunWith(BlockJUnit4ClassRunner.class)
6 public class SimpleMapTest {
7     private SimpleMap<Integer, String> map;
8
9     @Before
10    public void init() {
11        map = new SimpleMap<>();
12    }
13
14    @Test
15    public void testEmptyMapCreation() {
16        assertThat(0, is(equalTo(map.size()))); //Hamcrest Assertion
17        assertEquals(0, map.size()); //JUnit Assertion
18    }
19
20 }

```

```

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     public int size() {
5         return 0;
6     }
7 }

```

Markers Properties Servers Data Source Explorer Snippets JUnit Finished after 0.022 seconds Runs: 1/1 Errors: 0 Failures: 0 Failure Trace

org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.001 s)

Writable Smart Insert 26:5 11:55 PM 2/15/2016

Next I separate the test set up (which in this case is just the test object initialization) into the @Before so the test runs it every time before it runs my tests.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer SimpleMapTest.java SimpleMap.java
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.equalTo;
4
5 public class SimpleMapTest {
6     private SimpleMap<Integer, String> map;
7
8     @Before
9     public void init() {
10         map = new SimpleMap<>();
11     }
12
13     @Test
14     public void testEmptyMapCreation() {
15         assertThat(0, equalTo(map.size()));
16     }
17
18
19
20
21
22
23

```

```

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4
5     public int size() {
6         return 0;
7     }
8
9
10
11

```

Markers Properties Servers Data Source Explorer Snippets JUnit
Finished after 0.027 seconds
Runs: 1/1 Errors: 0 Failures: 0
Failure Trace

Let's get rolling with the subsequent tests based on a similar approach. The second test is "testNonEmptyMap()"...

Note again how the test demands implementation...

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer SimpleMapTest.java SimpleMap.java
SimpleMapTest.java
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4
5 @RunWith(BlockJUnit4ClassRunner.class)
6 public class SimpleMapTest {
7     private SimpleMap<Integer, String> map;
8
9     @Before
10    public void init() {
11        map = new SimpleMap<>();
12    }
13
14    @Test
15    public void testEmptyMapCreation() {
16        assertThat(0, equalTo(map.size()));
17        assertEquals(0, map.size());
18    }
19
20    @Test
21    public void testNonEmptyMap() {
22        map.save(1, "Priyan");
23    }
24
25
26
27
28
29
30
31
32

```

```

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4
5     public int size() {
6         return 0;
7     }
8
9
10
11

```

The method save(int, String) is undefined for the type SimpleMap<Integer, String>
2 quick fixes available:
Create method 'save(int, String)' in type 'SimpleMap'
Add cast to 'map'
Press F2 for focus

Test fails for obvious reason when ran...

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. The `SimpleMapTest.java` file contains JUnit tests for the `SimpleMap` class. The `SimpleMap.java` file defines the `SimpleMap` class with a field `size` and a method `size()` that returns 0. The JUnit results show 2 runs, 0 errors, and 1 failure. The failure trace indicates an `AssertionError` where the expected value is not less than the actual value.

```

SimpleMapTest.java
12 public class SimpleMapTest {
13     private SimpleMap<Integer, String> map;
14
15     @Before
16     public void init() {
17         map = new SimpleMap<>();
18     }
19
20     @Test
21     public void testEmptyMapCreation() {
22         assertEquals(0, map.size());
23         assertEquals(0, map.size());
24     }
25
26     @Test
27     public void testNonEmptyMap() {
28         map.save(1, "Priyan");
29         assertEquals(1, map.size());
30     }
31
32 }
33

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size;
5
6     public int size() {
7         return size;
8     }
9
10    public void save(int i, String string) {
11        size++;
12    }
13
14 }
15

```

I again did the smallest change that would make my tests run successfully and they do...

The screenshot shows the Eclipse IDE interface with the same files as before. The `SimpleMap.java` file has been refactored to initialize the `size` variable in the constructor. The JUnit results now show 2 runs, 0 errors, and 0 failures, indicating that the tests are passing successfully.

```

SimpleMapTest.java
12 public class SimpleMapTest {
13     private SimpleMap<Integer, String> map;
14
15     @Before
16     public void init() {
17         map = new SimpleMap<>();
18     }
19
20     @Test
21     public void testEmptyMapCreation() {
22         assertEquals(0, map.size());
23         assertEquals(0, map.size());
24     }
25
26     @Test
27     public void testNonEmptyMap() {
28         map.save(1, "Priyan");
29         assertEquals(1, map.size());
30     }
31
32 }
33

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size = 0;
5
6     public int size() {
7         return this.size;
8     }
9
10    public void save(int i, String string) {
11        this.size++;
12    }
13
14 }
15

```

Quick refactor i.e. initialize “int size” inside the constructor. Tests still run fine.

Note: Refactors are anything that improve code quality while not change the behavior of my implementation

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit
SimpleMapTest.java
1 package org.self.learn;
2 import org.junit.Test;
3 import org.junit.Before;
4 import org.junit.Assert;
5
6 public class SimpleMapTest {
7     @Before
8     public void init() {
9         map = new SimpleMap<Integer, String>();
10    }
11
12    @Test
13    public void testEmptyMapCreation() {
14        Assert.assertThat(0, is(equalTo(map.size())));
15        Assert.assertThat(0, isEquals(0, map.size()));
16    }
17
18    @Test
19    public void testNonEmptyMap() {
20        map.save(1, "Priyan");
21        Assert.assertThat(0, is(not(equalTo(map.size()))));
22    }
23
24    @Test
25    public void testSize() {
26        map.save(1, "Priyan");
27        Assert.assertThat(1, is(equalTo(map.size())));
28    }
29
30 }
31
32
33 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size;
5
6     public SimpleMap() {
7         this.size = 0;
8     }
9
10    public int size() {
11        return this.size;
12    }
13
14    public void save(int i, String string) {
15        this.size++;
16    }
17
18 }
19

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.024 seconds

Runs: 2/2 Errors: 0 Failures: 0

Failure Trace

Another “testSize()” test which the implementation has already taken care though...

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer JUnit
SimpleMapTest.java
1 package org.self.learn;
2 import org.junit.Test;
3 import org.junit.Before;
4 import org.junit.Assert;
5
6 public class SimpleMapTest {
7     @Test
8     public void testEmptyMapCreation() {
9         Assert.assertThat(0, is(equalTo(map.size())));
10        Assert.assertThat(0, isEquals(0, map.size()));
11    }
12
13    @Test
14    public void testNonEmptyMap() {
15        map.save(1, "Priyan");
16        Assert.assertThat(0, is(not(equalTo(map.size()))));
17    }
18
19    @Test
20    public void testSize() {
21        map.save(1, "Priyan");
22        Assert.assertThat(1, is(equalTo(map.size())));
23    }
24
25 }
26
27
28 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size;
5
6     public SimpleMap() {
7         this.size = 0;
8     }
9
10    public int size() {
11        return this.size;
12    }
13
14    public void save(int i, String string) {
15        this.size++;
16    }
17
18 }
19

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.024 seconds

Runs: 3/3 Errors: 0 Failures: 0

Failure Trace

Well, let's modify the test a little bit to make it more exhaustive... The test fails this time, however.

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. The `SimpleMapTest.java` file contains JUnit tests for the `SimpleMap` class. The `SimpleMap.java` file contains the implementation of the `SimpleMap` class. The test results show 3 runs, 0 errors, and 1 failure. The failure trace indicates an `AssertionError` where the expected size was less than or equal to the actual size.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
SimpleMapTest.java
21    public void testEmptyMapCreation() {
22        assertEquals(0, map.size());
23        assertEquals(0, map.size());
24    }
25}
26
27 @Test
28 public void testNonEmptyMap() {
29     map.save(1, "Priyan");
30     assertEquals(1, map.size());
31 }
32
33 @Test
34 public void testSize() {
35     map.save(1, "Priyan");
36     assertEquals(1, map.size());
37     map.save(2, "Priyadarshan");
38     assertEquals(2, map.size());
39 }
40 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size;
5
6     public SimpleMap() {
7         this.size = 0;
8     }
9
10    public int size() {
11        return this.size;
12    }
13
14    public void save(int i, String string) {
15        this.size++;
16    }
17
18 }
19

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.031 seconds

Runs: 3/3 Errors: 0 Failures: 1

Failure Trace

java.lang.AssertionError:
Expected: is <= 1
but: was < 2
at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
at org.self.learn.test.SimpleMapTest.testSize(SimpleMapTest.java:38)

Let's take the next leap to address the test for size of the simple map. And fix does work, as the test confirms.

The screenshot shows the Eclipse IDE interface with the same two files. The test results now show 3 runs, 0 errors, and 0 failures, indicating that the fix has been successful.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
SimpleMapTest.java
21    public void testEmptyMapCreation() {
22        assertEquals(0, map.size());
23        assertEquals(0, map.size());
24    }
25}
26
27 @Test
28 public void testNonEmptyMap() {
29     map.save(1, "Priyan");
30     assertEquals(1, map.size());
31 }
32
33 @Test
34 public void testSize() {
35     map.save(1, "Priyan");
36     assertEquals(1, map.size());
37     map.save(2, "Priyadarshan");
38     assertEquals(2, map.size());
39 }
40 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private int size;
5
6     public SimpleMap() {
7         this.size = 0;
8     }
9
10    public int size() {
11        return this.size;
12    }
13
14    public void save(int i, String string) {
15        this.size++;
16    }
17
18 }
19

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.021 seconds

Runs: 3/3 Errors: 0 Failures: 0

Failure Trace

Let's go an extra mile to make sure that our implementation really works! And it does because tests are green.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like `SimpleMap.java`, `SimpleMapTest.java`, and `pom.xml`.
- SimpleMapTest.java:** Contains JUnit tests for the `SimpleMap` class. The `testSize()` method asserts that saving 10 entries results in a size of 10. The `testNonEmptyMap()` method saves 1 entry and asserts that its size is not zero. The `testEmptyMapCreation()` method saves 10 entries and asserts that its size is 10.
- SimpleMap.java:** The implementation of the `SimpleMap` class, which contains a private `size` variable and a `save` method that increments `size`.
- JUnit View:** Shows the test results: 3/3 runs completed, 0 errors, and 0 failures.
- Failure Trace:** Displays the stack traces for the three test methods.

Let's test if our map is really saving keys/values...

The screenshot shows the Eclipse IDE interface with the title "Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, etc. The "Project Explorer" view on the left shows the project structure: simple-map [test-driven-devel] with src/main/java (org.self.learn, SimpleMap.java), src/main/resources, src/test/java (org.self.learn.test, SimpleMapTest.java), JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml. The "SimpleMapTest.java" editor window contains test cases for SimpleMap. The "SimpleMap.java" editor window contains the implementation of the SimpleMap class. A tooltip at the bottom left indicates that the method get(int) is undefined for the type SimpleMap<Integer, String>, with two quick fixes available: "Create method 'get(int)' in type 'SimpleMap'" and "Add cast to map".

```
SimpleMapTest.java
private SimpleMap<Integer, String> map;
@Before
public void init() {
    map = new SimpleMap<>();
}
@Test
public void testEmptyMapCreation() {
    assertEquals(0, map.size());
    assertEquals(0, map.size());
}
@Test
public void testNonEmptyMap() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
}
@Test
public void testGetSize() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
    map.save(2, "Priyadarshan");
    assertEquals(2, map.size());
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
@Test
public void testSaveKeyValuePair() {
    map.save(1, "Priyan");
}

SimpleMap.java
package org.self.learn;
public class SimpleMap<K, V> {
    private int size;
    public SimpleMap() {
        this.size = 0;
    }
    public int size() {
        return this.size;
    }
    public void save(int i, String string) {
        this.size++;
    }
}
```

Here is what the completed test along with the auto generated method looks like. The test fails for obvious reasons.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer

```

SimpleMapTest.java
28 public void testNonEmptyMap() {
29     map.save(1, "Priyan");
30     assertThat(0, is(not(equalTo(map.size()))));
31 }
32
33 @Test
34 public void testSize() {
35     map.save(1, "Priyan");
36     assertThat(1, is(equalTo(map.size())));
37     map.save(2, "Priyadarshan");
38     assertThat(2, is(equalTo(map.size())));
39     map.save(10, "Priyan Parida");
40     assertThat(3, is(equalTo(map.size())));
41 }
42
43 @Test
44 public void testSaveKeyValuePair() {
45     map.save(1, "Priyan");
46     assertThat("Priyan", is(equalTo(map.get(1))));
47 }
48
49 }

```

```

SimpleMap.java
3 package org.self.learn;
4
5 private int size;
6
7 public SimpleMap() {
8     this.size = 0;
9 }
10 public int size() {
11     return this.size;
12 }
13
14 public void save(int i, String string) {
15     this.size++;
16 }
17
18 public V get(K i) {
19     // TODO Auto-generated method stub
20     return null;
21 }
22
23 }
24

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.033 seconds

Runs: 4/4 Errors: 0 Failures: 1

Failure Trace

java.lang.AssertionError:
Expected: is null
but: was "Priyan"
at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
at org.self.learn.test.SimpleMapTest.testSaveKeyValuePair(SimpleMapTest.java:46)

I have gone ahead and provided implementation to address this part. The tests show that the implementation is correct indeed!

Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer

```

SimpleMapTest.java
26
27 @Test
28 public void testNonEmptyMap() {
29     map.save(1, "Priyan");
30     assertThat(0, is(not(equalTo(map.size()))));
31 }
32
33 @Test
34 public void testSize() {
35     map.save(1, "Priyan");
36     assertThat(1, is(equalTo(map.size())));
37     map.save(2, "Priyadarshan");
38     assertThat(2, is(equalTo(map.size())));
39     map.save(10, "Priyan Parida");
40     assertThat(3, is(equalTo(map.size())));
41 }
42
43 @Test
44 public void testSaveKeyValuePair() {
45     map.save(1, "Priyan");
46     assertThat("Priyan", is(equalTo(map.get(1))));
47 }
48
49 }

```

```

SimpleMap.java
2
3 public class SimpleMap<K, V> {
4     private int size;
5     private K key;
6     private V value;
7
8     public SimpleMap() {
9         this.size = 0;
10    }
11
12    public int size() {
13        return this.size;
14    }
15
16    public void save(K key, V value) {
17        this.key = key;
18        this.value = value;
19        this.size++;
20    }
21
22    public V get(K key) {
23        return value;
24    }
25
26 }
27

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.027 seconds

Runs: 4/4 Errors: 0 Failures: 0

Failure Trace

Let's make the test for values being saved a bit more elaborate. I have renamed my test to better tell what it is actually testing...

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

SimpleMapTest.java

```

32
33    @Test
34    public void testSize() {
35        map.save(1, "Priyan");
36        assertEquals(1, map.size());
37        map.save(2, "Priyadarshan");
38        assertEquals(2, map.size());
39        map.save(10, "Priyan Parida");
40        assertEquals(3, map.size());
41    }
42
43    @Test
44    public void testSingleSaveKeyValuePair() {
45        map.save(1, "Priyan");
46        assertEquals("Priyan", map.get(1));
47    }
48
49    @Test
50    public void testMultipleSaveKeyValuePair() {
51        map.save(1, "Priyan");
52        map.save(2, "Priyadarshan");
53        assertEquals("Priyan Parida", map.get(2));
54    }

```

SimpleMap.java

```

2
3    public class SimpleMap<K, V> {
4        private int size;
5        private K key;
6        private V value;
7
8        public SimpleMap() {
9            this.size = 0;
10       }
11
12        public int size() {
13            return this.size;
14        }
15
16        public void save(K key, V value) {
17            this.key = key;
18            this.value = value;
19            this.size++;
20        }
21
22        public V get(K key) {
23            return value;
24        }

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.027 seconds

Runs: 4/4 Errors: 0 Failures: 0

org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.000 s)

- testSize (0.000 s)
- testNonEmptyMap (0.000 s)
- testSaveKeyValuePair (0.000 s)
- testEmptyMapCreation (0.000 s)

Failure Trace

We want to test the map is able to save, not just a single key/value, but multiple keys/values. The below test shows, it does. But does it really? Check the next test.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

SimpleMapTest.java

```

36        assertEquals(1, map.size());
37        map.save(2, "Priyadarshan");
38        assertEquals(2, map.size());
39        map.save(10, "Priyan Parida");
40        assertEquals(3, map.size());
41    }
42
43    @Test
44    public void testSingleSaveKeyValuePair() {
45        map.save(1, "Priyan");
46        assertEquals("Priyan", map.get(1));
47    }
48
49    @Test
50    public void testMultipleSaveKeyValuePair() {
51        map.save(1, "Priyan");
52        map.save(2, "Priyadarshan");
53        assertEquals("Priyan Parida", map.get(2));
54    }

```

SimpleMap.java

```

2
3    public class SimpleMap<K, V> {
4        private int size;
5        private K key;
6        private V value;
7
8        public SimpleMap() {
9            this.size = 0;
10       }
11
12        public int size() {
13            return this.size;
14        }
15
16        public void save(K key, V value) {
17            this.key = key;
18            this.value = value;
19            this.size++;
20        }
21
22        public V get(K key) {
23            return value;
24        }

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.022 seconds

Runs: 5/5 Errors: 0 Failures: 0

org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.003 s)

- testMultipleSaveKeyValuePair (0.001 s)
- testSize (0.000 s)
- testNonEmptyMap (0.001 s)
- testSingleSaveKeyValuePair (0.000 s)
- testEmptyMapCreation (0.001 s)

Failure Trace

There we go... This test caught us. ☺

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer

SimpleMapTest.java

```

37     map.save(2, "Priyadarshan");
38     assertEquals(2, map.size());
39     map.save(10, "Priyan Parida");
40     assertEquals(3, map.size());
41 }
42
43 @Test
44 public void testSingleSaveKeyValuePair() {
45     map.save(1, "Priyan");
46     assertEquals("Priyan", map.get(1));
47 }
48
49 @Test
50 public void testMultipleSaveKeyValuePair() {
51     map.save(1, "Priyan");
52     map.save(2, "Priyadarshan");
53     assertEquals("Priyadarshan", map.get(2));
54     assertEquals("Priyan", map.get(1));
55 }
56
57 }

```

SimpleMap.java

```

private int size;
private K key;
private V value;
public SimpleMap() {
    this.size = 0;
}
public int size() {
    return this.size;
}
public void save(K key, V value) {
    this.key = key;
    this.value = value;
    this.size++;
}
public V get(K key) {
    return value;
}

```

Markers Properties Servers Data Source Explorer Snippets JUnit

Finished after 0.031 seconds

Runs: 5/5 Errors: 0 Failures: 1

Failure Trace

- java.lang.AssertionError: Expected: is "Priyadarshan" but was "Priyan"
- at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
- at org.self.learn.test.SimpleMapTest.testMultipleSaveKeyValuePair(SimpleMapTest.java:54)

Writable Smart Insert 54:54

Let's go a few miles to actually store keys/values into the map.

New Java Class

Create a new Java class.

Source folder: simple-map/src/main/java

Package: org.self.learn

Enclosing type:

Name: KeyValuePair<K, V>

Modifiers: public package private protected abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments

Finish Cancel

Here is what my Key Value Template looks like.

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `KeyValuePair.java`.

SimpleMapTest.java:

```
1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4
5 @RunWith(BlockingJUnit4ClassRunner.class)
6 public class SimpleMapTest {
7     private SimpleMap<Integer, String> map;
8
9     @Before
10    public void init() {
11         map = new SimpleMap<>();
12     }
13
14     @Test
15    public void testEmptyMapCreation() {
16         assertThat(0, is(equalTo(map.size())));
17         assertEquals(0, map.size());
18     }
19
20     @Test
21    public void testNonEmptyMap() {
22         map.save(1, "Priyan");
23         assertThat(1, is(not(equalTo(map.size()))));
24     }
25
26     @Test
27    public void testGetSize() {
28         map.save(1, "Priyan");
29         map.save(2, "Priyadarshan");
30         assertThat(2, is(equalTo(map.size())));
31         map.save(10, "Priyan Parida");
32         assertThat(3, is(equalTo(map.size())));
33     }
34
35     @Test
```

KeyValuePair.java:

```
1 package org.self.learn;
2
3 public class KeyValuePair<K, V> {
4     private K key;
5     private V value;
6
7     public KeyValuePair(K key, V value) {
8         this.key = key;
9         this.value = value;
10    }
11    public K getKey() {
12        return key;
13    }
14    public void setKey(K key) {
15        this.key = key;
16    }
17    public V getValue() {
18        return value;
19    }
20    public void setValue(V value) {
21        this.value = value;
22    }
23}
24}
```

Added array of key value template as a member variable, saved keys/values into the array. The “get()” method now returns the value corresponding to the key.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure with packages: simple-map [test-driven-devel], org.self.learn, and org.self.learn.test. It also lists src/main/java, src/main/resources, src/test/java, src/test/resources, JRE System Library [JavaSE-1.8], Maven Dependencies, src, target, and pom.xml.
- Left Editor:** Displays the content of SimpleMapTest.java, which contains several test methods using JUnit annotations (@Test) to verify the functionality of the SimpleMap class.
- Right Editor:** Displays the content of SimpleMap.java, which defines the SimpleMap class with methods like save, size, and get.

I run the tests and they complete successfully this time.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
public void testEmptyMapCreation() {
    assertEquals(0, map.size());
}
@Test
public void testNonEmptyMap() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
}
@Test
public void testSize() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
    map.save(2, "Priyadarshan");
    assertEquals(2, map.size());
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
JUnit
Markers Properties Servers Data Source Explorer Snippets Debug
Failure Trace
SimpleMap.java
package org.self.learn;
public class SimpleMap<K, V> {
    private KeyValuePair<K, V>[] keyValuePair;
    private int size;
    @SuppressWarnings("unchecked")
    public SimpleMap() {
        this.size = 0;
        this.keyValuePair = (KeyValuePair<K, V>[] ) new KeyValuePair[5];
    }
    public int size() {
        return this.size;
    }
    public void save(K key, V value) {
        KeyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
    }
}

SimpleMapTest.java
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
public void testSize() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
    map.save(2, "Priyadarshan");
    assertEquals(2, map.size());
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
public void testSingleSaveKeyValuePair() {
    map.save(1, "Priyan");
    assertEquals("Priyan", map.get(1));
}
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    assertEquals("Priyadarshan", map.get(2));
    assertEquals("Priyan", map.get(1));
}
public void testContainsKey() {
    map.save(1, "Priyan");
    assertEquals(true, map.containsKey(1));
}

```

Next, let's write test cases for "containsKey"

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
public void testNonEmptyMap() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
}
@Test
public void testSize() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
    map.save(2, "Priyadarshan");
    assertEquals(2, map.size());
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
@Test
public void testSingleSaveKeyValuePair() {
    map.save(1, "Priyan");
    assertEquals("Priyan", map.get(1));
}
@Test
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    assertEquals("Priyadarshan", map.get(2));
    assertEquals("Priyan", map.get(1));
}
public void testContainsKey() {
    map.save(1, "Priyan");
    assertEquals(true, map.containsKey(1));
}
KeyValuePair.java
package org.self.learn;
public class SimpleMap<K, V> {
    private KeyValuePair<K, V>[] keyValuePair;
    private int size;
    @SuppressWarnings("unchecked")
    public SimpleMap() {
        this.size = 0;
        this.keyValuePair = (KeyValuePair<K, V>[] ) new KeyValuePair[5];
    }
    public int size() {
        return this.size;
    }
    public void save(K key, V value) {
        KeyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
    }
    public V get(K key) {
        KeyValuePair<K, V> keyValuePair;
        for(int i=0;i<size;i++) {
            keyValuePair = this.keyValuePair[i];
            if(key.equals(keyValuePair.getKey())) {
                return keyValuePair.getValue();
            }
        }
        return null;
    }
}

SimpleMapTest.java
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
public void testSize() {
    map.save(1, "Priyan");
    assertEquals(1, map.size());
    map.save(2, "Priyadarshan");
    assertEquals(2, map.size());
    map.save(10, "Priyan Parida");
    assertEquals(3, map.size());
}
public void testSingleSaveKeyValuePair() {
    map.save(1, "Priyan");
    assertEquals("Priyan", map.get(1));
}
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    assertEquals("Priyadarshan", map.get(2));
    assertEquals("Priyan", map.get(1));
}
public void testContainsKey() {
    map.save(1, "Priyan");
    assertEquals(true, map.containsKey(1));
}

```

Successful completion of the test gives us an illusion that the implementation is complete. But, it is not. Let's add another test case to prove this.

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. The `SimpleMapTest.java` file contains JUnit tests for the `SimpleMap` class. The `containsKey` method in `SimpleMap.java` is currently implemented to return `true` for all keys. The test results show 6 runs, 0 errors, and 0 failures, indicating a failure.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
public void testSingleSaveKeyValuePair() {
    map.save(1, "Priyan");
    assertThat("Priyan", is(equalTo(map.get(1))));
}
@Test
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    assertThat("Priyadarshan", is(equalTo(map.get(2))));
    assertThat("Priyan", is(equalTo(map.get(1))));
}
@Test
public void testContainsKey() {
    map.save(1, "Priyan");
    assertThat(true, is(equalTo(map.containsKey(1))));
}
SimpleMap.java
public V get(K key) {
    KeyValuePair<K, V> keyValuePair;
    for(int i=0;i<size;i++) {
        keyValuePair = this.keyValuePair[i];
        if(key.equals(keyValuePair.getKey())) {
            return keyValuePair.getValue();
        }
    }
    return null;
}
public boolean containsKey(K key) {
    return true;
}

```

The test fails when ran now. We know returning “true” always isn’t the right implementation. So let’s go fix it.

The screenshot shows the Eclipse IDE interface with the same files and code as the previous one. However, the test results now show 6 runs, 0 errors, and 1 failure. The failure is in the `testContainsKey` test, which expected `false` but got `true`. The failure trace shows the assertion error.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
public void testSingleSaveKeyValuePair() {
    map.save(1, "Priyan");
    assertThat("Priyan", is(equalTo(map.get(1))));
}
@Test
public void testMultipleSaveKeyValuePair() {
    map.save(1, "Priyan");
    map.save(2, "Priyadarshan");
    assertThat("Priyadarshan", is(equalTo(map.get(2))));
    assertThat("Priyan", is(equalTo(map.get(1))));
}
@Test
public void testContainsKey() {
    map.save(1, "Priyan");
    assertThat(true, is(equalTo(map.containsKey(1))));
    assertThat(false, is(not(equalTo)(map.containsKey(35))));
}
SimpleMap.java
public V get(K key) {
    KeyValuePair<K, V> keyValuePair;
    for(int i=0;i<size;i++) {
        keyValuePair = this.keyValuePair[i];
        if(key.equals(keyValuePair.getKey())) {
            return keyValuePair.getValue();
        }
    }
    return null;
}
public boolean containsKey(K key) {
    return false;
}

```

The test result indicates that the implementation is complete now.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer

- simple-map [test-driven-developer]
 - src/main/java
 - org.self.learn
 - KeyValuePair.java
 - SimpleMap.java
 - src/main/resources
 - src/test/java
 - org.self.learn.test
 - SimpleMapTest.java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src/test/resources
 - target
 - pom.xml

SimpleMapTest.java

```

47     }
48     @Test
49     public void testMultipleSaveKeyValuePair() {
50         map.save(1, "Priyan");
51         map.save(2, "Priyadarshan");
52         assertEquals("Priyadarshan", is(equalTo(map.get(2)));
53         assertEquals("Priyan", is(equalTo(map.get(1)));
54     }
55     @Test
56     public void testContainsKey() {
57         map.save(1, "Priyan");
58         assertEquals(true, is(equalTo(map.containsKey(1)));
59         assertEquals(true, is(not(equalTo(map.containsKey(35))))));
60         assertFalse(map.containsKey(35));
61     }
62     @Test
63     public void testEmptyMapCreation() {
64     }
65 }

```

KeyValuePair.java

```

28     }
29     return null;
30 }
31
32
33
34     public boolean containsKey(K key) {
35         KeyValuePair<K, V> keyValuePair;
36         for(int i=0;i<size;i++) {
37             keyValuePair = this.keyValuePair[i];
38             if(key.equals(keyValuePair.getKey())) {
39                 return true;
40             }
41         }
42         return false;
43     }
44
45
46
47

```

SimpleMap.java

```

12     }
13     public int size() {
14         return this.size;
15     }
16
17     public void save(K key, V value) {
18         KeyValuePair<K, V> keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
19     }
20
21
22     public V get(K key) {
23         KeyValuePair<K, V> keyValuePair;
24         for(int i=0;i<size;i++) {
25             keyValuePair = this.keyValuePair[i];
26             if(key.equals(keyValuePair.getKey())) {
27                 return keyValuePair.getValue();
28             }
29         }
30         return null;
31     }
32
33
34     public boolean containsKey(K key) {
35         KeyValuePair<K, V> keyValuePair;
36         for(int i=0;i<size;i++) {
37             keyValuePair = this.keyValuePair[i];
38             if(key.equals(keyValuePair.getKey())) {
39                 return true;
40             }
41         }
42         return false;
43     }
44
45
46
47

```

InvocationTargetException.class

Markers Properties Servers Data Source Explorer Snippets JUnit Debug

Finished after 0.029 seconds

Runs: 6/6 Errors: 0 Failures: 0

Failure Trace

- org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.000 s)
 - testMultipleSaveKeyValuePair (0.000 s)
 - testSize (0.000 s)
 - testNonEmptyMap (0.000 s)
 - testSingleSaveKeyValuePair (0.000 s)
 - testContainsKey (0.000 s)
 - testEmptyMapCreation (0.000 s)

Next, let's test the edge cases w.r.t. getKey(). We want it to throw an exception when a key isn't found. Note, again, how the tests drive the implementation.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

File Edit Source Refactor Navigate Project Run Window Help

Project Explorer

- simple-map [test-driven-developer]
 - src/main/java
 - org.self.learn
 - KeyValuePair.java
 - SimpleMap.java
 - src/main/resources
 - src/test/java
 - org.self.learn.test
 - SimpleMapTest.java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src/test/resources
 - target
 - pom.xml

SimpleMapTest.java

```

51     assertEquals("Priyan", is(equalTo(map.get(1)));
52     }
53     @Test
54     public void testMultipleSaveKeyValuePair() {
55         map.save(1, "Priyan");
56         map.save(2, "Priyadarshan");
57         assertEquals("Priyadarshan", is(equalTo(map.get(2)));
58         assertEquals("Priyan", is(equalTo(map.get(1)));
59     }
60     @Test
61     public void testContainsKey() {
62         map.save(1, "Priyan");
63         assertEquals(true, is(equalTo(map.containsKey(1)));
64         assertEquals(true, is(not(equalTo(map.containsKey(35))))));
65         assertFalse(map.containsKey(35));
66     }
67     @Test
68     public void testNoSuchKeyException() {
69         map.save(1, "Priyan");
70         thrown.expect(NoSuchKeyException.class);
71         map.get(2);
72     }
73 }

```

NoSuchKeyException cannot be resolved to a type

17 quick fixes available:

- Create class 'NoSuchKeyException'
- Create interface 'NoSuchKeyException'
- Change to 'NamingException' (javax.naming)
- Change to 'NoSuchAlgorithmException' (java.security)
- Change to 'NoSuchAttributeException' (javax.naming.directory)
- Change to 'NoSuchElementException' (java.util)
- Change to 'NoSuchFieldException' (java.lang)
- Change to 'NoSuchElementException' (java.util)

KeyValuePair.java

```

12     }
13     public int size() {
14         return this.size;
15     }
16
17     public void save(K key, V value) {
18         KeyValuePair<K, V> keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
19     }
20
21
22     public V get(K key) {
23         KeyValuePair<K, V> keyValuePair;
24         for(int i=0;i<size;i++) {
25             keyValuePair = this.keyValuePair[i];
26             if(key.equals(keyValuePair.getKey())) {
27                 return keyValuePair.getValue();
28             }
29         }
30         return null;
31     }
32
33
34     public boolean containsKey(K key) {
35         KeyValuePair<K, V> keyValuePair;
36         for(int i=0;i<size;i++) {
37             keyValuePair = this.keyValuePair[i];
38             if(key.equals(keyValuePair.getKey())) {
39                 return true;
40             }
41         }
42         return false;
43     }
44
45
46
47

```

SimpleMap.java

```

12     }
13     public int size() {
14         return this.size;
15     }
16
17     public void save(K key, V value) {
18         KeyValuePair<K, V> keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
19     }
20
21
22     public V get(K key) {
23         KeyValuePair<K, V> keyValuePair;
24         for(int i=0;i<size;i++) {
25             keyValuePair = this.keyValuePair[i];
26             if(key.equals(keyValuePair.getKey())) {
27                 return keyValuePair.getValue();
28             }
29         }
30         return null;
31     }
32
33
34     public boolean containsKey(K key) {
35         KeyValuePair<K, V> keyValuePair;
36         for(int i=0;i<size;i++) {
37             keyValuePair = this.keyValuePair[i];
38             if(key.equals(keyValuePair.getKey())) {
39                 return true;
40             }
41         }
42         return false;
43     }
44
45
46
47

```

InvocationTargetException.class

Btw, “thrown” is a Junit rule of type ExpectedException and here is how it is defined.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

```

1 package org.self.learn.test;
2
3 import static org.hamcrest.CoreMatchers.*;
4
5 import org.junit.runner.RunWith;
6 import org.junit.runners.BlockJUnit4ClassRunner;
7
8 public class SimpleMapTest {
9     private SimpleMap<Integer, String> map;
10
11     @Rule
12     public ExpectedException thrown = ExpectedException.none();
13
14     @Before
15     public void init() {
16         map = new SimpleMap<>();
17     }
18
19     @Test
20     public void testEmptyMapCreation() {
21         assertEquals(0, map.size());
22         assertEquals(0, map.size());
23     }
24
25     @Test
26     public void testNonEmptyMap() {
27         map.save(1, "Priyan");
28         assertEquals(1, map.size());
29         map.save(2, "Priyadarshan");
30         assertEquals(2, map.size());
31         map.save(10, "Priyan Parida");
32         assertEquals(3, map.size());
33     }
34
35     @Test
36     public void testSize() {
37         map.save(1, "Priyan");
38         assertEquals(1, map.size());
39         map.save(2, "Priyadarshan");
40         assertEquals(2, map.size());
41         map.save(10, "Priyan Parida");
42         assertEquals(3, map.size());
43     }
44
45     @Test
46     public void testGet() {
47         map.save(2, "Priyadarshan");
48         assertEquals(2, map.size());
49         map.save(10, "Priyan Parida");
50         assertEquals(3, map.size());
51     }
52
53     @Test
54     public void testSingleSaveKeyValuePair() {
55         map.save(1, "Priyan");
56         assertEquals("Priyan", map.get(1));
57     }
58
59     @Test
60     public void testMultipleSaveKeyValuePair() {
61         map.save(1, "Priyan");
62         map.save(2, "Priyadarshan");
63         assertEquals("Priyadarshan", map.get(2));
64         assertEquals("Priyan", map.get(1));
65     }
66
67     @Test
68     public void testContainsKey() {
69         map.save(1, "Priyan");
70         assertEquals(true, map.containsKey(1));
71         assertEquals(true, map.containsKey(35));
72         assertEquals(false, map.containsKey(35));
73     }
74
75     @Test
76     public void testNoSuchKeyException() {
77         map.save(1, "Priyan");
78         thrown.expect(NoSuchKeyException.class);
79         map.get(2);
80     }
81 }

```

Java EE - simple-map/src/main/java/org/self/learn/test/SimpleMap.java - Eclipse

```

1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private KeyValuePairs<K, V> keyValuePair;
5
6     public int size() {
7         return this.size;
8     }
9
10    public void save(K key, V value) {
11        keyValuePair[this.size] = new KeyValuePair<K, V>(key, value);
12    }
13
14    public V get(K key) {
15        KeyValuePairs<K, V> keyValuePair;
16        for(int i=0;i<size;i++) {
17            keyValuePair = this.keyValuePair[i];
18            if(key.equals(keyValuePair.getKey())) {
19                return keyValuePair.getValue();
20            }
21        }
22        return null;
23    }
24
25    public boolean containsKey(K key) {
26        KeyValuePairs<K, V> keyValuePair;
27        for(int i=0;i<size;i++) {
28            keyValuePair = this.keyValuePair[i];
29            if(key.equals(keyValuePair.getKey())) {
30                return true;
31            }
32        }
33        return false;
34    }
35
36    public void clear() {
37        keyValuePair = null;
38    }
39
40    public void remove(K key) {
41        KeyValuePairs<K, V> keyValuePair;
42        for(int i=0;i<size;i++) {
43            keyValuePair = this.keyValuePair[i];
44            if(key.equals(keyValuePair.getKey())) {
45                keyValuePair = null;
46            }
47        }
48    }
49
50    public void merge(SimpleMap<K, V> other) {
51        for(int i=0;i<other.size();i++) {
52            other.get(i);
53        }
54    }
55
56    public void swap(SimpleMap<K, V> other) {
57        for(int i=0;i<other.size();i++) {
58            other.get(i);
59        }
60    }
61
62    public void copy(SimpleMap<K, V> other) {
63        for(int i=0;i<other.size();i++) {
64            other.get(i);
65        }
66    }
67
68    public void swap(SimpleMap<K, V> other) {
69        for(int i=0;i<other.size();i++) {
70            other.get(i);
71        }
72    }
73
74    public void copy(SimpleMap<K, V> other) {
75        for(int i=0;i<other.size();i++) {
76            other.get(i);
77        }
78    }
79
80    public void swap(SimpleMap<K, V> other) {
81        for(int i=0;i<other.size();i++) {
82            other.get(i);
83        }
84    }
85
86    public void copy(SimpleMap<K, V> other) {
87        for(int i=0;i<other.size();i++) {
88            other.get(i);
89        }
90    }
91
92    public void swap(SimpleMap<K, V> other) {
93        for(int i=0;i<other.size();i++) {
94            other.get(i);
95        }
96    }
97
98    public void copy(SimpleMap<K, V> other) {
99        for(int i=0;i<other.size();i++) {
100            other.get(i);
101        }
102    }
103
104    public void swap(SimpleMap<K, V> other) {
105        for(int i=0;i<other.size();i++) {
106            other.get(i);
107        }
108    }
109
110    public void copy(SimpleMap<K, V> other) {
111        for(int i=0;i<other.size();i++) {
112            other.get(i);
113        }
114    }
115
116    public void swap(SimpleMap<K, V> other) {
117        for(int i=0;i<other.size();i++) {
118            other.get(i);
119        }
120    }
121
122    public void copy(SimpleMap<K, V> other) {
123        for(int i=0;i<other.size();i++) {
124            other.get(i);
125        }
126    }
127
128    public void swap(SimpleMap<K, V> other) {
129        for(int i=0;i<other.size();i++) {
130            other.get(i);
131        }
132    }
133
134    public void copy(SimpleMap<K, V> other) {
135        for(int i=0;i<other.size();i++) {
136            other.get(i);
137        }
138    }
139
140    public void swap(SimpleMap<K, V> other) {
141        for(int i=0;i<other.size();i++) {
142            other.get(i);
143        }
144    }
145
146    public void copy(SimpleMap<K, V> other) {
147        for(int i=0;i<other.size();i++) {
148            other.get(i);
149        }
150    }
151
152    public void swap(SimpleMap<K, V> other) {
153        for(int i=0;i<other.size();i++) {
154            other.get(i);
155        }
156    }
157
158    public void copy(SimpleMap<K, V> other) {
159        for(int i=0;i<other.size();i++) {
160            other.get(i);
161        }
162    }
163
164    public void swap(SimpleMap<K, V> other) {
165        for(int i=0;i<other.size();i++) {
166            other.get(i);
167        }
168    }
169
170    public void copy(SimpleMap<K, V> other) {
171        for(int i=0;i<other.size();i++) {
172            other.get(i);
173        }
174    }
175
176    public void swap(SimpleMap<K, V> other) {
177        for(int i=0;i<other.size();i++) {
178            other.get(i);
179        }
180    }
181
182    public void copy(SimpleMap<K, V> other) {
183        for(int i=0;i<other.size();i++) {
184            other.get(i);
185        }
186    }
187
188    public void swap(SimpleMap<K, V> other) {
189        for(int i=0;i<other.size();i++) {
190            other.get(i);
191        }
192    }
193
194    public void copy(SimpleMap<K, V> other) {
195        for(int i=0;i<other.size();i++) {
196            other.get(i);
197        }
198    }
199
200    public void swap(SimpleMap<K, V> other) {
201        for(int i=0;i<other.size();i++) {
202            other.get(i);
203        }
204    }
205
206    public void copy(SimpleMap<K, V> other) {
207        for(int i=0;i<other.size();i++) {
208            other.get(i);
209        }
210    }
211
212    public void swap(SimpleMap<K, V> other) {
213        for(int i=0;i<other.size();i++) {
214            other.get(i);
215        }
216    }
217
218    public void copy(SimpleMap<K, V> other) {
219        for(int i=0;i<other.size();i++) {
220            other.get(i);
221        }
222    }
223
224    public void swap(SimpleMap<K, V> other) {
225        for(int i=0;i<other.size();i++) {
226            other.get(i);
227        }
228    }
229
230    public void copy(SimpleMap<K, V> other) {
231        for(int i=0;i<other.size();i++) {
232            other.get(i);
233        }
234    }
235
236    public void swap(SimpleMap<K, V> other) {
237        for(int i=0;i<other.size();i++) {
238            other.get(i);
239        }
240    }
241
242    public void copy(SimpleMap<K, V> other) {
243        for(int i=0;i<other.size();i++) {
244            other.get(i);
245        }
246    }
247
248    public void swap(SimpleMap<K, V> other) {
249        for(int i=0;i<other.size();i++) {
250            other.get(i);
251        }
252    }
253
254    public void copy(SimpleMap<K, V> other) {
255        for(int i=0;i<other.size();i++) {
256            other.get(i);
257        }
258    }
259
260    public void swap(SimpleMap<K, V> other) {
261        for(int i=0;i<other.size();i++) {
262            other.get(i);
263        }
264    }
265
266    public void copy(SimpleMap<K, V> other) {
267        for(int i=0;i<other.size();i++) {
268            other.get(i);
269        }
270    }
271
272    public void swap(SimpleMap<K, V> other) {
273        for(int i=0;i<other.size();i++) {
274            other.get(i);
275        }
276    }
277
278    public void copy(SimpleMap<K, V> other) {
279        for(int i=0;i<other.size();i++) {
280            other.get(i);
281        }
282    }
283
284    public void swap(SimpleMap<K, V> other) {
285        for(int i=0;i<other.size();i++) {
286            other.get(i);
287        }
288    }
289
290    public void copy(SimpleMap<K, V> other) {
291        for(int i=0;i<other.size();i++) {
292            other.get(i);
293        }
294    }
295
296    public void swap(SimpleMap<K, V> other) {
297        for(int i=0;i<other.size();i++) {
298            other.get(i);
299        }
299    }
300
301    public void copy(SimpleMap<K, V> other) {
302        for(int i=0;i<other.size();i++) {
303            other.get(i);
304        }
305    }
306
307    public void swap(SimpleMap<K, V> other) {
308        for(int i=0;i<other.size();i++) {
309            other.get(i);
310        }
311    }
312
313    public void copy(SimpleMap<K, V> other) {
314        for(int i=0;i<other.size();i++) {
315            other.get(i);
316        }
317    }
318
319    public void swap(SimpleMap<K, V> other) {
320        for(int i=0;i<other.size();i++) {
321            other.get(i);
322        }
323    }
324
325    public void copy(SimpleMap<K, V> other) {
326        for(int i=0;i<other.size();i++) {
327            other.get(i);
328        }
329    }
330
331    public void swap(SimpleMap<K, V> other) {
332        for(int i=0;i<other.size();i++) {
333            other.get(i);
334        }
335    }
336
337    public void copy(SimpleMap<K, V> other) {
338        for(int i=0;i<other.size();i++) {
339            other.get(i);
340        }
341    }
342
343    public void swap(SimpleMap<K, V> other) {
344        for(int i=0;i<other.size();i++) {
345            other.get(i);
346        }
347    }
348
349    public void copy(SimpleMap<K, V> other) {
350        for(int i=0;i<other.size();i++) {
351            other.get(i);
352        }
353    }
354
355    public void swap(SimpleMap<K, V> other) {
356        for(int i=0;i<other.size();i++) {
357            other.get(i);
358        }
359    }
360
361    public void copy(SimpleMap<K, V> other) {
362        for(int i=0;i<other.size();i++) {
363            other.get(i);
364        }
365    }
366
367    public void swap(SimpleMap<K, V> other) {
368        for(int i=0;i<other.size();i++) {
369            other.get(i);
370        }
371    }
372
373    public void copy(SimpleMap<K, V> other) {
374        for(int i=0;i<other.size();i++) {
375            other.get(i);
376        }
377    }
378
379    public void swap(SimpleMap<K, V> other) {
380        for(int i=0;i<other.size();i++) {
381            other.get(i);
382        }
383    }
384
385    public void copy(SimpleMap<K, V> other) {
386        for(int i=0;i<other.size();i++) {
387            other.get(i);
388        }
389    }
390
391    public void swap(SimpleMap<K, V> other) {
392        for(int i=0;i<other.size();i++) {
393            other.get(i);
394        }
395    }
396
397    public void copy(SimpleMap<K, V> other) {
398        for(int i=0;i<other.size();i++) {
399            other.get(i);
400        }
401    }
402
403    public void swap(SimpleMap<K, V> other) {
404        for(int i=0;i<other.size();i++) {
405            other.get(i);
406        }
407    }
408
409    public void copy(SimpleMap<K, V> other) {
410        for(int i=0;i<other.size();i++) {
411            other.get(i);
412        }
413    }
414
415    public void swap(SimpleMap<K, V> other) {
416        for(int i=0;i<other.size();i++) {
417            other.get(i);
418        }
419    }
420
421    public void copy(SimpleMap<K, V> other) {
422        for(int i=0;i<other.size();i++) {
423            other.get(i);
424        }
425    }
426
427    public void swap(SimpleMap<K, V> other) {
428        for(int i=0;i<other.size();i++) {
429            other.get(i);
430        }
431    }
432
433    public void copy(SimpleMap<K, V> other) {
434        for(int i=0;i<other.size();i++) {
435            other.get(i);
436        }
437    }
438
439    public void swap(SimpleMap<K, V> other) {
440        for(int i=0;i<other.size();i++) {
441            other.get(i);
442        }
443    }
444
445    public void copy(SimpleMap<K, V> other) {
446        for(int i=0;i<other.size();i++) {
447            other.get(i);
448        }
449    }
450
451    public void swap(SimpleMap<K, V> other) {
452        for(int i=0;i<other.size();i++) {
453            other.get(i);
454        }
455    }
456
457    public void copy(SimpleMap<K, V> other) {
458        for(int i=0;i<other.size();i++) {
459            other.get(i);
460        }
461    }
462
463    public void swap(SimpleMap<K, V> other) {
464        for(int i=0;i<other.size();i++) {
465            other.get(i);
466        }
467    }
468
469    public void copy(SimpleMap<K, V> other) {
470        for(int i=0;i<other.size();i++) {
471            other.get(i);
472        }
473    }
474
475    public void swap(SimpleMap<K, V> other) {
476        for(int i=0;i<other.size();i++) {
477            other.get(i);
478        }
479    }
480
481    public void copy(SimpleMap<K, V> other) {
482        for(int i=0;i<other.size();i++) {
483            other.get(i);
484        }
485    }
486
487    public void swap(SimpleMap<K, V> other) {
488        for(int i=0;i<other.size();i++) {
489            other.get(i);
490        }
491    }
492
493    public void copy(SimpleMap<K, V> other) {
494        for(int i=0;i<other.size();i++) {
495            other.get(i);
496        }
497    }
498
499    public void swap(SimpleMap<K, V> other) {
500        for(int i=0;i<other.size();i++) {
501            other.get(i);
502        }
503    }
504
505    public void copy(SimpleMap<K, V> other) {
506        for(int i=0;i<other.size();i++) {
507            other.get(i);
508        }
509    }
510
511    public void swap(SimpleMap<K, V> other) {
512        for(int i=0;i<other.size();i++) {
513            other.get(i);
514        }
515    }
516
517    public void copy(SimpleMap<K, V> other) {
518        for(int i=0;i<other.size();i++) {
519            other.get(i);
520        }
521    }
522
523    public void swap(SimpleMap<K, V> other) {
524        for(int i=0;i<other.size();i++) {
525            other.get(i);
526        }
527    }
528
529    public void copy(SimpleMap<K, V> other) {
530        for(int i=0;i<other.size();i++) {
531            other.get(i);
532        }
533    }
534
535    public void swap(SimpleMap<K, V> other) {
536        for(int i=0;i<other.size();i++) {
537            other.get(i);
538        }
539    }
540
541    public void copy(SimpleMap<K, V> other) {
542        for(int i=0;i<other.size();i++) {
543            other.get(i);
544        }
545    }
546
547    public void swap(SimpleMap<K, V> other) {
548        for(int i=0;i<other.size();i++) {
549            other.get(i);
550        }
551    }
552
553    public void copy(SimpleMap<K, V> other) {
554        for(int i=0;i<other.size();i++) {
555            other.get(i);
556        }
557    }
558
559    public void swap(SimpleMap<K, V> other) {
560        for(int i=0;i<other.size();i++) {
561            other.get(i);
562        }
563    }
564
565    public void copy(SimpleMap<K, V> other) {
566        for(int i=0;i<other.size();i++) {
567            other.get(i);
568        }
569    }
570
571    public void swap(SimpleMap<K, V> other) {
572        for(int i=0;i<other.size();i++) {
573            other.get(i);
574        }
575    }
576
577    public void copy(SimpleMap<K, V> other) {
578        for(int i=0;i<other.size();i++) {
579            other.get(i);
580        }
581    }
582
583    public void swap(SimpleMap<K, V> other) {
584        for(int i=0;i<other.size();i++) {
585            other.get(i);
586        }
587    }
588
589    public void copy(SimpleMap<K, V> other) {
590        for(int i=0;i<other.size();i++) {
591            other.get(i);
592        }
593    }
594
595    public void swap(SimpleMap<K, V> other) {
596        for(int i=0;i<other.size();i++) {
597            other.get(i);
598        }
599    }
599
600    public void copy(SimpleMap<K, V> other) {
601        for(int i=0;i<other.size();i++) {
602            other.get(i);
603        }
604    }
605
606    public void swap(SimpleMap<K, V> other) {
607        for(int i=0;i<other.size();i++) {
608            other.get(i);
609        }
610    }
611
612    public void copy(SimpleMap<K, V> other) {
613        for(int i=0;i<other.size();i++) {
614            other.get(i);
615        }
616    }
617
618    public void swap(SimpleMap<K, V> other) {
619        for(int i=0;i<other.size();i++) {
620            other.get(i);
621        }
622    }
623
624    public void copy(SimpleMap<K, V> other) {
625        for(int i=0;i<other.size();i++) {
626            other.get(i);
627        }
628    }
629
630    public void swap(SimpleMap<K, V> other) {
631        for(int i=0;i<other.size();i++) {
632            other.get(i);
633        }
634    }
635
636    public void copy(SimpleMap<K, V> other) {
637        for(int i=0;i<other.size();i++) {
638            other.get(i);
639        }
640    }
641
642    public void swap(SimpleMap<K, V> other) {
643        for(int i=0;i<other.size();i++) {
644            other.get(i);
645        }
646    }
647
648    public void copy(SimpleMap<K, V> other) {
649        for(int i=0;i<other.size();i++) {
650            other.get(i);
651        }
652    }
653
654    public void swap(SimpleMap<K, V> other) {
655        for(int i=0;i<other.size();i++) {
656            other.get(i);
657        }
658    }
659
660    public void copy(SimpleMap<K, V> other) {
661        for(int i=0;i<other.size();i++) {
662            other.get(i);
663        }
664    }
665
666    public void swap(SimpleMap<K, V> other) {
667        for(int i=0;i<other.size();i++) {
668            other.get(i);
669        }
670    }
671
672    public void copy(SimpleMap<K, V> other) {
673        for(int i=0;i<other.size();i++) {
674            other.get(i);
675        }
676    }
677
678    public void swap(SimpleMap<K, V> other) {
679        for(int i=0;i<other.size();i++) {
680            other.get(i);
681        }
682    }
683
684    public void copy(SimpleMap<K, V> other) {
685        for(int i=0;i<other.size();i++) {
686            other.get(i);
687        }
688    }
689
690    public void swap(SimpleMap<K, V> other) {
691        for(int i=0;i<other.size();i++) {
692            other.get(i);
693        }
694    }
695
696    public void copy(SimpleMap<K, V> other) {
697        for(int i=0;i<other.size();i++) {
698            other.get(i);
699        }
700    }
600
601    public void swap(SimpleMap<K, V> other) {
701        for(int i=0;i<other.size();i++) {
702            other.get(i);
703        }
704    }
705
706    public void copy(SimpleMap<K, V> other) {
707        for(int i=0;i<other.size();i++) {
708            other.get(i);
709        }
710    }
711
712    public void swap(SimpleMap<K, V> other) {
713        for(int i=0;i<other.size();i++) {
714            other.get(i);
715        }
716    }
717
718    public void copy(SimpleMap<K, V> other) {
719        for(int i=0;i<other.size();i++) {
720            other.get(i);
721        }
722    }
723
724    public void swap(SimpleMap<K, V> other) {
725        for(int i=0;i<other.size();i++) {
726            other.get(i);
727        }
728    }
729
730    public void copy(SimpleMap<K, V> other) {
731        for(int i=0;i<other.size();i++) {
732            other.get(i);
733        }
734    }
735
736    public void swap(SimpleMap<K, V> other) {
737        for(int i=0;i<other.size();i++) {
738            other.get(i);
739        }
740    }
741
742    public void copy(SimpleMap<K, V> other) {
743        for(int i=0;i<other.size();i++) {
744            other.get(i);
745        }
746    }
747
748    public void swap(SimpleMap<K, V> other) {
749        for(int i=0;i<other.size();i++) {
750            other.get(i);
751        }
752    }
753
754    public void copy(SimpleMap<K, V> other) {
755        for(int i=0;i<other.size();i++) {
756            other.get(i);
757        }
758    }
759
760    public void swap(SimpleMap<K, V> other) {
761        for(int i=0;i<other.size();i++) {
762            other.get(i);
763        }
764    }
765
766    public void copy(SimpleMap<K, V> other) {
767        for(int i=0;i<other.size();i++) {
768            other.get(i);
769        }
770    }
771
772    public void swap(SimpleMap<K, V> other) {
773        for(int i=0;i<other.size();i++) {
774            other.get(i);
775        }
776    }
777
778    public void copy(SimpleMap<K, V> other) {
779        for(int i=0;i<other.size();i++) {
780            other.get(i);
781        }
782    }
783
784    public void swap(SimpleMap<K, V> other) {
785        for(int i=0;i<other.size();i++) {
786            other.get(i);
787        }
788    }
789
790    public void copy(SimpleMap<K, V> other) {
791        for(int i=0;i<other.size();i++) {
792            other.get(i);
793        }
794    }
795
796    public void swap(SimpleMap<K, V> other) {
797        for(int i=0;i<other.size();i++) {
798            other.get(i);
799        }
800    }
801
802    public void copy(SimpleMap<K, V> other) {
803        for(int i=0;i<other.size();i++) {
804            other.get(i);
805        }
806    }
807
808    public void swap(SimpleMap<K, V> other) {
809        for(int i=0;i<other.size();i++) {
810            other.get(i);
811        }
812    }
813
814    public void copy(SimpleMap<K, V> other) {
815        for(int i=0;i<other.size();i++) {
816            other.get(i);
817        }
818    }
819
820    public void swap(SimpleMap<K, V> other) {
821        for(int i=0;i<other.size();i++) {
822            other.get(i);
823        }
824    }
825
826    public void copy(SimpleMap<K, V> other) {
827        for(int i=0;i<other.size();i++) {
828            other.get(i);
829        }
830    }
831
832    public void swap(SimpleMap<K, V> other) {
833        for(int i=0;i<other.size();i++) {
834            other.get(i);
835        }
836    }
837
838    public void copy(SimpleMap<K, V> other) {
839        for(int i=0;i<other.size();i++) {
840            other.get(i);
841        }
842    }
843
844    public void swap(SimpleMap<K, V> other) {
845        for(int i=0;i<other.size();i++) {
846            other.get(i);
847        }
848    }
849
850    public void copy(SimpleMap<K, V> other) {
851        for(int i=0;i<other.size();i++) {
852            other.get(i);
853        }
854    }
855
856    public void swap(SimpleMap<K, V> other) {
857        for(int i=0;i<other.size();i++) {
858            other.get(i);
859        }
860    }
861
862    public void copy(SimpleMap<K, V> other) {
863        for(int i=0;i<other.size();i++) {
864            other.get(i);
865        }
866    }
867
868    public void swap(SimpleMap<K, V> other) {
869        for(int i=0;i<other.size();i++) {
870            other.get(i);
871        }
872    }
873
874    public void copy(SimpleMap<K, V> other) {
875        for(int i=0;i<other.size();i++) {
876            other.get(i);
877        }
878    }
879
880    public void swap(SimpleMap<K, V> other) {
881        for(int i=0;i<other.size();i++) {
882            other.get(i);
883        }
884    }
885
886    public void copy(SimpleMap<K, V> other) {
887        for(int i=0;i<other.size();i++) {
888            other.get(i);
889        }
890    }
891
892    public void swap(SimpleMap<K, V> other) {
893        for(int i=0;i<other.size();i++) {
894            other.get(i);
895        }
896    }
897
898    public void copy(SimpleMap<K, V> other) {
899        for(int i=0;i<other.size();i++) {
900            other.get(i);
901        }
902    }
903
904    public void swap(SimpleMap<K, V> other) {
905        for(int i=0;i<other.size();i++) {
906            other.get(i);
907        }
908    }
909
910    public void copy(SimpleMap<K, V> other) {
911        for(int i=0;i<other.size();i++) {
912            other.get(i);
913        }
914    }
915
916    public void swap(SimpleMap<K, V> other) {
917        for(int i=0;i<other.size();i++) {
918            other.get(i);
919        }
920    }
921
922    public void copy(SimpleMap<K, V> other) {
923        for(int i=0;i<other.size();i++) {
924            other.get(i);
925        }
926    }
927
928    public void swap(SimpleMap<K, V> other) {
929        for(int i=0;i<other.size();i++) {
930            other.get(i);
931        }
932    }
933
934    public void copy(SimpleMap<K, V> other) {
935        for(int i=0;i<other.size();i++) {
936            other.get(i);
937        }
938    }
939
940    public void swap(SimpleMap<K, V> other) {
941        for(int i=0;i<other.size();i++) {
942            other.get(i);
943        }
944    }
945
946    public void copy(SimpleMap<K, V> other) {
947        for(int i=0;i<other.size();i++) {
948            other.get(i);
949        }
950    }
951
952    public void swap(SimpleMap<K, V> other) {
953        for(int i=0;i<other.size();i++) {
954            other.get(i);
955        }
956    }
957
958    public void copy(SimpleMap<K, V> other) {
959        for(int i=0;i<other.size();i++) {
960            other.get(i);
961        }
962    }
963
964    public void swap(SimpleMap<K, V> other) {
965        for(int i=0;i<other.size();i++) {
966            other.get(i);
967        }
968    }
969
970    public void copy(SimpleMap<K, V> other) {
971        for(int i=0;i<other.size();i++) {
972            other.get(i);
973        }
974    }
975
976    public void swap(SimpleMap<K, V> other) {
977        for(int i=0;i<other.size();i++) {
978            other.get(i);
979        }
980    }
981
982    public void copy(SimpleMap<K, V> other) {
983        for(int i=0;i<other.size();i++) {
984            other.get(i);
985        }
986    }
987
988    public void swap(SimpleMap<K, V> other) {
989        for(int i=0;i<other.size();i++) {
990            other.get(i);
991        }
992    }
993
994    public void copy(SimpleMap<K, V> other) {
995        for(int i=0;i<other.size();i++) {
996            other.get(i);
997        }
998    }
999
1000    public void swap(SimpleMap<K, V> other) {
1001        for(int i=0;i<other.size();i++) {
1002            other.get(i);
1003        }
1004    }
1005
1006    public void copy(SimpleMap<K, V> other) {
1007        for(int i=0;i<other.size();i++) {
1008            other.get(i);
1009        }
1010    }
1011
1012    public void swap(SimpleMap<K, V> other) {
1013        for(int i=0;i<other.size();i++) {
1014            other.get(i);
1015        }
1016    }
1017
1018    public void copy(SimpleMap<K, V> other) {
1019        for(int i=0;i<other.size();i++) {
1020            other.get(i);
1021        }
1022    }
1023
1024    public void swap(SimpleMap<K, V> other) {
1025        for(int i=0;i<other.size();i++) {
1026            other.get(i);
1027        }
1028    }
1029
1030    public void copy(SimpleMap<K, V> other) {
1031        for(int i=0;i<other.size();i++) {
1032            other.get(i);
1033        }
1034    }
1035
1036    public void swap(SimpleMap<K, V> other) {
1037        for(int i=0;i<other.size();i++) {
1038            other.get(i);
1039        }
1040    }
1041
1042    public void copy(SimpleMap<K, V> other) {
1043        for(int i=0;i<other.size();i++) {
1044            other.get(i);
1045        }
1046    }
1047
1048    public void swap(SimpleMap<K, V> other) {
1049        for(int i=0;i<other.size();i++) {
1050            other.get(i);
1051        }
1052    }
1053
1054    public void copy(SimpleMap<K, V> other) {
1055        for(int i=0;i<other.size();i++) {
1056            other.get(i);
1057        }
1058    }
1059
1060    public void swap(SimpleMap<K, V> other) {
1061        for(int i=0;i<other.size();i++) {
1062            other.get(i);
1063        }
1064    }
1065

```

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Data Source Explorer Servers Properties Markers Snippets Quick Access Java EE
SimpleMapTest.java
59     assertEquals("Priyadarshan",is(equalTo(map.get(2))));
60     assertEquals("Priyan",is(equalTo(map.get(1))));
61 }
62
63 @Test
64 public void testContainsKey() {
65     map.save(1, "Priyan");
66     assertEquals(true, is(equalTo(map.containsKey(1))));
67     assertEquals(true, is(not(equalTo(map.containsKey(35)))));
68     assertFalse(map.containsKey(35));
69 }
70
71 @Test
72 public void testNoSuchKeyException() {
73     map.save(1, "Priyan");
74     thrown.expect(NoSuchKeyException.class);
75     map.get(2);
76 }
77
78 }

SimpleMap.java
14 public int size() {
15     return this.size;
16 }
17
18 public void save(K key, V value) {
19     KeyValuePair<K, V> keyValuePair = new KeyValuePair<K, V>(key, value);
20 }
21
22 public V get(K key) {
23     KeyValuePair<K, V> keyValuePair;
24     for(int i=0;i<size;i++) {
25         keyValuePair = this.keyValuePair[i];
26         if(key.equals(keyValuePair.getKey())) {
27             return keyValuePair.getValue();
28         }
29     }
30
31     return null;
32 }

Markers Properties Servers Data Source Explorer Snippets JUnit
JUnit
Finished after 0.05 seconds
Runs: 7/7 Errors: 0 Failures: 1
org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.000 s)
testMultipleSaveKeyValuePair (0.000 s)
testSize (0.000 s)
testNonEmptyMap (0.000 s)
testNoSuchKeyException (0.000 s)
testSingleSaveKeyValuePair (0.000 s)
testContainsKey (0.000 s)
testEmptyMapCreation (0.000 s)

Failure Trace
java.lang.AssertionError: Expected test to throw an instance of org.self.learn.NoSuchKeyException

```

Let's quickly get this working... The test completes successfully.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Data Source Explorer Servers Properties Markers Snippets Quick Access Java EE
SimpleMapTest.java
59     assertEquals("Priyadarshan",is(equalTo(map.get(2))));
60     assertEquals("Priyan",is(equalTo(map.get(1))));
61 }
62
63 @Test
64 public void testContainsKey() {
65     map.save(1, "Priyan");
66     assertEquals(true, is(equalTo(map.containsKey(1))));
67     assertEquals(true, is(not(equalTo(map.containsKey(35)))));
68     assertFalse(map.containsKey(35));
69 }
70
71 @Test
72 public void testNoSuchKeyException() throws NoSuchKeyException {
73     map.save(1, "Priyan");
74     thrown.expect(NoSuchKeyException.class);
75     map.get(2);
76 }
77
78 }

SimpleMap.java
19     KeyValuePair<K, V> keyValuePair[this.size++] = new KeyValuePair<K, V>(key, val);
20 }
21
22 public V get(K key) throws NoSuchKeyException {
23     KeyValuePair<K, V> keyValuePair;
24
25     if(!containsKey(key)) {
26         throw new NoSuchKeyException();
27     }
28
29     for(int i=0;i<size;i++) {
30         keyValuePair = this.keyValuePair[i];
31         if(key.equals(keyValuePair.getKey())) {
32             return keyValuePair.getValue();
33         }
34     }
35
36     return null;
37 }

Markers Properties Servers Data Source Explorer Snippets JUnit
JUnit
Finished after 0.03 seconds
Runs: 7/7 Errors: 0 Failures: 0
org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.007 s)
testMultipleSaveKeyValuePair (0.001 s)
testSize (0.000 s)
testNonEmptyMap (0.001 s)
testNoSuchKeyException (0.004 s)
testSingleSaveKeyValuePair (0.000 s)
testContainsKey (0.000 s)
testEmptyMapCreation (0.001 s)

Failure Trace

```

Next up is implementation of “containsValue()”

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. In the `SimpleMapTest.java` file, there is a test method `testContainsValue()` that calls `map.containsValue("Priyan")`. A tooltip appears over this line, stating: "The method containsValue(String) is undefined for the type SimpleMap<Integer, String>". It provides three quick fixes:

- Change to 'containsKey(...)' (highlighted)
- Create method 'containsValue(String)' in type 'SimpleMap'
- Add cast to map

I have skipped a couple of steps here i.e. auto generate method, implement the smallest bit, run test and start over with the next bit of implementation... Here is what the final test and implementation looks like. That took care of it, the test results indicate that.

The screenshot shows the Eclipse IDE interface with the same two files. The `SimpleMapTest.java` file now includes the fix from the previous screenshot, changing `containsValue` to `containsKey`. The `SimpleMap.java` file has been updated to implement the `containsValue` method. The status bar at the bottom indicates "Runs: 8/8 Errors: 0 Failures: 0", confirming that all tests have passed.

We do not want to allow our map to contain duplicate key and throw an exception when that is the case. Let's write a test for that.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

Project Explorer

```

70     @Test
71     public void testNoSuchKeyException() throws NoSuchKeyException {
72         map.save(1, "Priyan");
73         thrown.expect(NoSuchKeyException.class);
74         map.get(2);
75     }
76
77
78     @Test
79     public void testContainsValue() {
80         map.save(1, "Priyan");
81         map.save(2, "Priyan");
82         map.save(3, "Priyadarshan");
83         assertTrue(map.containsValue("Priyan"));
84         assertTrue(map.containsValue("Priyadarshan"));
85         assertFalse(map.containsValue("Priyan Parida"));
86     }
87
88     public void testDuplicateKeyFoundException() {
89         map.save(1, "Priyan");
90         thrown.expect(DuplicateKeyFoundException.class);
91         map.save(1, "Priyan");
92     }
93
94
95
96
97
98
99
100
101
102
103
104
105

```

SimpleMap.java

```

20     }
21
22     public V get(K key) throws NoSuchKeyException {
23         KeyValuePair<K, V> keyValuePair;
24
25         if(!containsKey(key)) {
26             throw new NoSuchKeyException();
27         }
28
29         for(int i=0;i<size();i++) {
30             keyValuePair = this.keyValuePair[i];
31             if(key.equals(keyValuePair.getKey())) {
32                 return keyValuePair.getValue();
33             }
34         }
35
36         return null;
37     }
38
39     public boolean containsKey(K key) {
40         KeyValuePair<K, V> keyValuePair;
41         for(int i=0;i<size();i++) {
42             keyValuePair = this.keyValuePair[i];
43             if(key.equals(keyValuePair.getKey())) {
44                 return true;
45             }
46         }
47
48         return false;
49     }
50
51     public boolean containsValue(V value) {
52         KeyValuePair<K, V> keyValuePair;
53         for(int i=0;i<size();i++) {
54             keyValuePair = this.keyValuePair[i];
55             if(value.equals(keyValuePair.getValue())) {
56                 return true;
57             }
58         }
59
60         return false;
61     }
62
63
64     @Test
65     public void testContainsKey() {
66         map.save(1, "Priyan");
67         assertEquals(true, is(equalTo(map.containsKey(1))));
68         assertEquals(false, is(notEqualTo(map.containsKey(35))));
69         assertFalse(map.containsKey(35));
70     }
71
72     @Test
73     public void testNoSuchKeyException() throws NoSuchKeyException {
74         map.save(1, "Priyan");
75         thrown.expect(NoSuchKeyException.class);
76         map.get(2);
77     }
78
79     @Test
80     public void testContainsValue() {
81         map.save(1, "Priyan");
82         map.save(2, "Priyan");
83         map.save(3, "Priyadarshan");
84         assertTrue(map.containsValue("Priyan"));
85         assertTrue(map.containsValue("Priyadarshan"));
86         assertFalse(map.containsValue("Priyan Parida"));
87     }
88
89     public void testDuplicateKeyFoundException() {
90         map.save(1, "Priyan");
91         thrown.expect(DuplicateKeyFoundException.class);
92         map.save(1, "Priyadarshan");
93     }
94
95

```

Writable Smart Insert | 101:5

4:25 AM 2/16/2016

Create the custom exception class to resolve compilation errors.

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

Project Explorer

```

60         assertEquals("Priyadarshan", isEqualTo(map.get(2)));
61         assertEquals("Priyan", isEqualTo(map.get(1)));
62     }
63
64     @Test
65     public void testContainsKey() {
66         map.save(1, "Priyan");
67         assertEquals(true, is(equalTo(map.containsKey(1))));
68         assertEquals(false, is(notEqualTo(map.containsKey(35))));
69         assertFalse(map.containsKey(35));
70     }
71
72     @Test
73     public void testNoSuchKeyException() throws NoSuchKeyException {
74         map.save(1, "Priyan");
75         thrown.expect(NoSuchKeyException.class);
76         map.get(2);
77     }
78
79     @Test
80     public void testContainsValue() {
81         map.save(1, "Priyan");
82         map.save(2, "Priyan");
83         map.save(3, "Priyadarshan");
84         assertTrue(map.containsValue("Priyan"));
85         assertTrue(map.containsValue("Priyadarshan"));
86         assertFalse(map.containsValue("Priyan Parida"));
87     }
88
89     public void testDuplicateKeyFoundException() {
90         map.save(1, "Priyan");
91         thrown.expect(DuplicateKeyFoundException.class);
92         map.save(1, "Priyadarshan");
93     }
94
95

```

SimpleMap.java

```

1 package org.self.learn;
2
3 public class DuplicateKeyFoundException extends Exception {
4     private static final long serialVersionUID = 3949478449695281;
5
6 }
7

```

Writable Smart Insert | 92:37

I refactored my test a bit to make it more comprehensive i.e. not just throw an exception; ensure that the value inserted with duplicate key is discarded.

Make note of the assertion style we used in this case. Junit rule could not have been used in this case because anything after exception is thrown, won't be executed i.e. the following would not work:

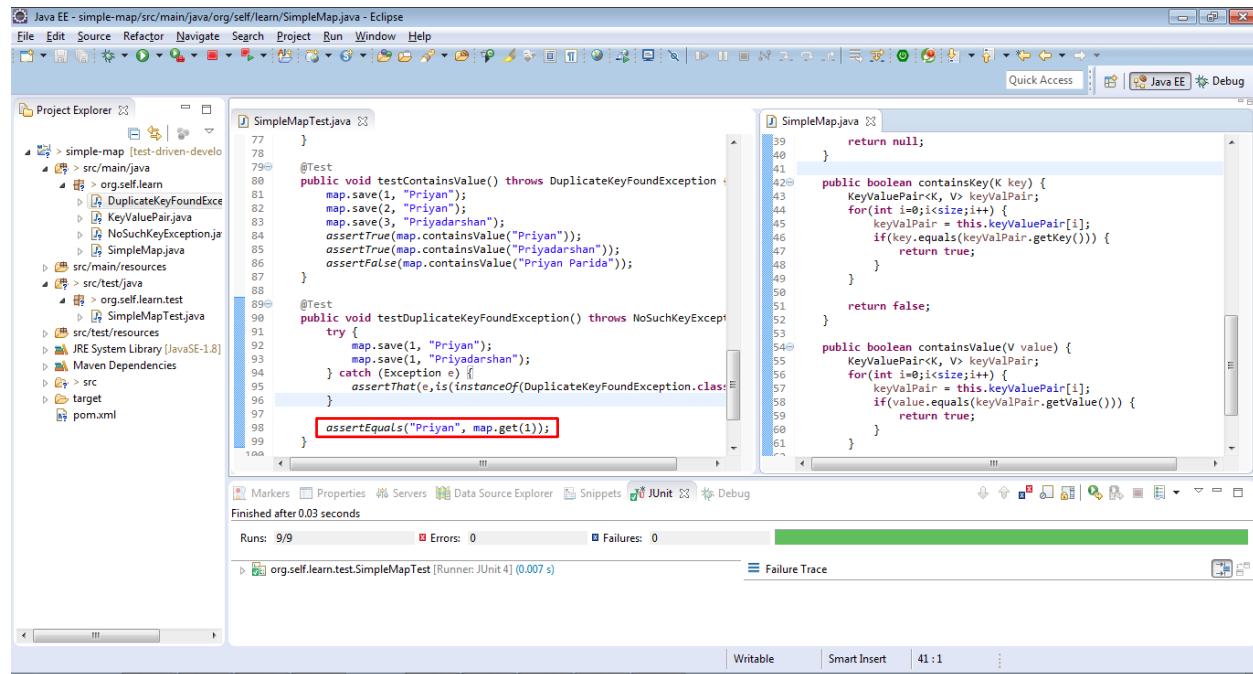
```
map.save(1, "Priyan");
```

```

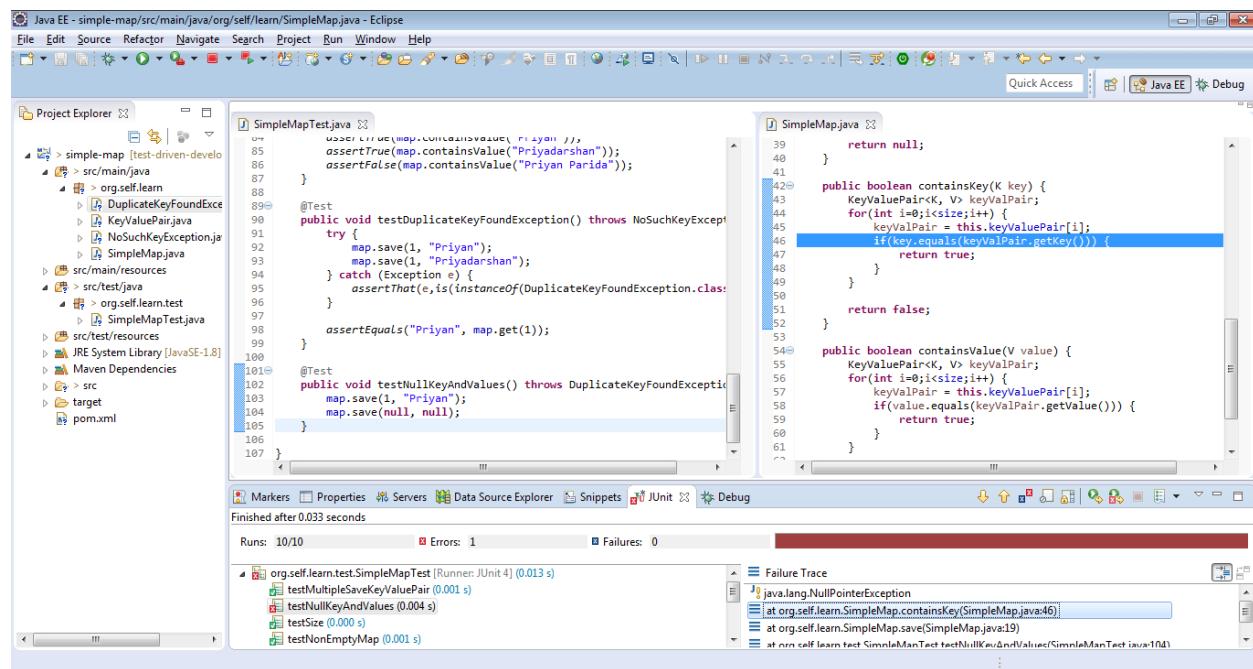
thrown.expect(DuplicateKeyFoundException.class);
map.save(1, "Priyadarshan");
assertEquals("Priyan", map.get(1)); →This assertion will never be executed.

```

The test did complete successfully.



Let's test if the map is able to handle null key (and not keys) and values. I ran a premature test to find that the map does not handle that yet! Make note of the NullPointerException which leads to test failure.



Refer to the fix highlighted in RED. Test results confirms that the fix took care of the issue.

The screenshot shows the Eclipse IDE interface with two open files: `SimpleMapTest.java` and `SimpleMap.java`. The `SimpleMapTest.java` file contains test cases for a `SimpleMap` class, specifically testing for duplicate keys and null keys. The `SimpleMap.java` file contains the implementation of the `SimpleMap` class, which uses an array of `KeyValuePair` objects to store key-value pairs. A red box highlights a condition in the `containsKey` method that checks for both `key==null` and `key.equals(keyValuePair.getKey())`, indicating a potential bug or redundancy.

```
Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer
SimpleMapTest.java
import java.util.*;
assertTrue(map.containsKey("Priyan"));
assertTrue(map.containsValue("Priyadarshan"));
assertFalse(map.containsValue("Priyan Parida"));
}
@Test
public void testDuplicateKeyFoundException() throws IOException {
try {
map.save(1, "Priyan");
map.save(1, "Priyadarshan");
} catch (Exception e) {
assertThat(e, isInstanceOf(DuplicateKeyFoundException.class));
}
assertEquals("Priyan", map.get(1));
}
@Test
public void testNullKeyAndValues() throws DuplicateKeyFoundException {
map.save(1, "Priyan");
map.save(null, null);
}

SimpleMap.java
return keyValuePair.getValue();
}
boolean containsKey(K key) {
ValuePair[K, V] keyValuePair;
'int i=0;i<size;i++) {
keyValuePair = this.keyValuePair[i];
if((key==null && keyValuePair.getKey()==null) || (key!=null && key.equals(keyValuePair.getKey())))
return true;
}
return false;
}
boolean containsValue(V value) {
ValuePair[K, V] keyValuePair;
'int i=0;i<size;i++) {
keyValuePair = this.keyValuePair[i];
if((value==null && keyValuePair.getValue()==null) || (value!=null && value.equals(keyValuePair.getValue())))
return true;
}
return false;
}

Markers Properties Servers Data Source Explorer Snippets JUnit Debug
Finished after 0.026 seconds
Runs: 10/10 Errors: 0 Failures: 0
org.self.learn.test.SimpleMapTest [Runner: JUnit 4] [0.008 s]
Failure Trace
Writable Smart Insert 46 : 55
```

Here is an attempt to make the test more comprehensive and test result indicates that we haven't cared for all scenarios yet.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse
- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar with various icons for file operations.
- Project Explorer:** Shows the project structure with packages like simple-map, src/main/java, and src/test/java.
- SimpleMapTest.java:** Contains JUnit test cases for SimpleMap. A red box highlights the assertion `assertNull(map.get(null));` at line 111.
- SimpleMap.java:** The implementation of the SimpleMap class. A blue box highlights the return statement `return null;` at line 46.
- Bottom Status Bar:** Writable, Smart Insert, 35:1

Refer to the highlighted code fix, tests look good with that.

```

Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
92     map.save(1, "Priyan");
93     map.save(1, "Priyadarshan");
94 } catch (Exception e) {
95     assertEquals("Priyan", map.get(1));
96 }
97 assertEquals("Priyan", map.get(1));
98 }
99 }
100 @Test
101 public void testNullKeyAndValues() throws DuplicateKeyException {
102     map.save(null, null);
103     map.save(1, null);
104     assertTrue(map.containsKey(null));
105     try {
106         map.save(null, "Priyan");
107     } catch (Exception e) {
108         assertEquals("Priyan", map.get(1));
109     }
110     assertNull(map.get(null));
111 }
112 }
113 }
114 }

SimpleMap.java
22     KeyValuePair.this.size++ = new KeyValuePair<K, V>(key, value);
23 }
24
25 public V get(K key) throws NoSuchKeyException {
26     KeyValuePair<K, V> keyValuePair;
27
28     if(!containsKey(key)) {
29         throw new NoSuchKeyException();
30     }
31
32     for(int i=0;i<size;i++) {
33         keyValuePair = this.keyValuePair[i];
34         if((key==null && keyValuePair.getKey()==null) || 
35             (key!=null && key.equals(keyValuePair.getKey()))) {
36             return keyValuePair.getValue();
37         }
38     }
39
40     return null;
41 }
42
43 public boolean containsKey(K key) {
44     KeyValuePair<K, V> keyValuePair;

```

Markers Properties Servers Data Source Explorer Snippets JUnit Debug
Finished after 0.029 seconds
Runs: 10/10 Errors: 0 Failures: 0

Failure Trace

Following test-build iteration is to ensure that the same handled for containsValue() method as well.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Debug
SimpleMapTest.java
92     map.save(1, "Priyan");
93     map.save(1, "Priyadarshan");
94 } catch (Exception e) {
95     assertEquals("Priyan", map.get(1));
96 }
97 assertEquals("Priyan", map.get(1));
98 }
99 }
100 @Test
101 public void testNullKeyAndValues() throws DuplicateKeyException {
102     map.save(null, null);
103     map.save(1, null);
104     assertTrue(map.containsKey(null));
105     try {
106         map.save(null, "Priyan");
107     } catch (Exception e) {
108         assertEquals("Priyan", map.get(1));
109     }
110     assertNull(map.get(null));
111     assertTrue(map.containsValue(null));
112 }
113 }

SimpleMap.java
47     KeyValuePair.this.size++ = new KeyValuePair<K, V>(key, value);
48     if((key==null && keyValuePair.getKey()==null) || 
49         (key!=null && key.equals(keyValuePair.getKey()))) {
50         return true;
51     }
52
53     return false;
54 }
55
56 public boolean containsValue(V value) {
57     KeyValuePair<K, V> keyValuePair;
58     for(int i=0;i<size;i++) {
59         keyValuePair = this.keyValuePair[i];
60         if(value.equals(keyValuePair.getValue())) {
61             return true;
62         }
63     }
64
65     return false;
66 }
67
68 }

Markers Properties Servers Data Source Explorer Snippets JUnit Debug  
Finished after 0.039 seconds  
Runs: 10/10 Errors: 1 Failures: 0
```

Failure Trace

That will take care of it!

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "simple-map [test-driven-devel]".
- SimpleMapTest.java:** Contains test cases for the SimpleMap class. One test method, `testNullKeyAndValues()`, is currently selected.
- SimpleMap.java:** The source code for the SimpleMap class. A red box highlights a specific line of code in the `containsValue` method:

```
if((value==null && keyValPair.getValue()==null)||  
    (value!=null && value.equals(keyValPair.getValue())))
```
- Run Status:** Shows "Runs: 10/10", "Errors: 0", and "Failures: 0".
- Failure Trace:** Displays a tree view of failed test cases, all of which are green (indicating they passed).
 - org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.008 s)
 - testMultipleSaveKeyValuePair (0.002 s)
 - testNullKeyAndValues (0.001 s)
 - testSize (0.001 s)
 - testNonEmptyMap (0.000 s)

Next up is remove... Following few screenshots depict how we put its implementation together using TDD approach.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows the project structure with packages like simple-map [test-driven-develop], org.self.learn, and org.self.learn.test.
- Sources View:** Displays two files:
 - SimpleMapTest.java:** Contains test methods for SimpleMap. It includes tests for null key and values, and remove operations.
 - SimpleMap.java:** Contains the implementation of the SimpleMap interface. It includes methods like containsKey, containsValue, and remove.
- Quick Access:** Shows Java EE and Debug options.
- Bottom Status Bar:** Shows "Finished after 0.027 second".

Java EE - simple-map[src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

```

87 }
88 }
89 @Test
90 public void testDuplicateKeyFoundException() throws NoSuchElementException {
91     try {
92         map.save(1, "Priyan");
93         map.save(1, "Priyadarshan");
94     } catch (Exception e) {
95         assertEqualsInstanceOf(DuplicateKeyFoundException.class, e);
96     }
97     assertEquals("Priyan", map.get(1));
98 }
99
100 @Test
101 public void testNullKeyAndValues() throws DuplicateKeyFoundException {
102     map.save(null, null);
103     map.save(1, null);
104     assertTrue(map.containsKey(null));
105     try {
106         map.save(null, "Priyan");
107     } catch (Exception e) {
108         assertEqualsInstanceOf(DuplicateKeyFoundException.class, e);
109     }
110     assertNull(map.get(null));
111     assertTrue(map.containsValue(null));
112 }
113
114 @Test
115 public void testRemove() throws DuplicateKeyFoundException {
116     map.save(1, "Priyan");
117     map.save(2, "Priyadarshan");
118     map.remove(2);
119     assertFalse(isEqualTo(map.containsKey(2)));
120     assertThat(isEqualTo(map.containsValue("Priyadarshan")));
121     assertThat(1, isEqualTo(map.size()));
122     thrown.expect(NoSuchKeyException.class);
123     map.get(2);
124 }
125 }
126 }
127 }

```

Java EE - simple-map[src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse

```

100 i
101     map.save(null, "Priyan");
102 } catch (Exception e) {
103     assertEqualsInstanceOf(DuplicateKeyFoundException.class, e);
104 }
105     assertNull(map.get(null));
106     assertTrue(map.containsValue(null));
107 }
108
109 @Test
110 public void testRemove() throws DuplicateKeyFoundException {
111     map.save(1, "Priyan");
112     map.save(2, "Priyadarshan");
113     map.remove(2);
114     assertFalse(isEqualTo(map.containsKey(2)));
115     assertThat(isEqualTo(map.containsValue("Priyadarshan")));
116     assertThat(1, isEqualTo(map.size()));
117     thrown.expect(NoSuchKeyException.class);
118     map.get(2);
119 }
120
121 }
122
123 }
124
125 }
126
127 }

```

Markers Properties Servers Data Source Explorer Snippets JUnit Debug

Runs: 11/11 Errors: 0 Failures: 1

Failure Trace

- java.lang.AssertionError:
Expected: is<true>
but: was <false>
- at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
- at org.self.learn.test.SimpleMapTest.testRemove(SimpleMapTest.java:120)

And that should be it for remove()

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Quick Access Java EE Debug
SimpleMapTest.java
105     assertTrue(map.containsKey(null));
106     try {
107         map.save(null, "Priyan");
108     } catch (Exception e) {
109         assertEquals(e, instanceof(DuplicateKeyFoundException));
110     }
111     assertNull(map.get(null));
112     assertTrue(map.containsValue(null));
113 }
114 @Test
115 public void testRemove() throws DuplicateKeyFoundException {
116     map.save(1, "Priyan");
117     map.save(2, "Priyadarshan");
118     map.remove(2);
119     assertFalse(false, is(equalTo(map.containsKey(2)));
120     assertFalse(false, is(equalTo(map.containsValue("Priyadarshan")));
121     assertEquals(1, is(equalTo(map.size())));
122     exception.expect(NoSuchKeyException.class);
123     map.get(2);
124 }
125 }
126
127 }

SimpleMap.java
public void remove(K key) throws NoSuchKeyException {
    KeyValuePair<K, V> keyValuePair;
    if(!containsKey(key)) {
        throw new NoSuchKeyException();
    }
    for(int i=0;i<size;i++) {
        keyValuePair = this.keyValuePair[i];
        if((key==null && keyValuePair.getKey()==null) || 
           (key!=null && key.equals(keyValuePair.getKey())))
        if(this.size==1){
            this.keyValuePair[1] = null;
        } else {
            this.keyValuePair[i] = this.keyValuePair[this.size-1];
            this.keyValuePair[this.size-1] = null;
        }
        this.size--;
    }
}

Markers Properties Servers Data Source Explorer Snippets JUnit Debug
Finished after 0.031 seconds
Runs: 11/11 Errors: 0 Failures: 0
org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.009 s)
testMultipleSaveKeyValuePair (0.002 s)
testNullKeyAndValues (0.001 s)
testSize (0.000 s)
testNonEmptyMap (0.001 s) ...
Failure Trace

```

The final one is support for dynamic size. I have added six key/value pairs to the map to end up with `ArrayIndexOutOfBoundsException` which lead to test failure.

```

Java EE - simple-map/src/test/java/org/self/learn/test/SimpleMapTest.java - Eclipse
File Edit Source Refactor Navigate Project Run Window Help
Project Explorer JUnit Quick Access Java EE Debug
SimpleMapTest.java
114
115 @Test
116 public void testRemove() throws DuplicateKeyFoundException {
117     map.save(1, "Priyan");
118     map.save(2, "Priyadarshan");
119     map.remove(2);
120     assertFalse(false, is(equalTo(map.containsKey(2)));
121     assertFalse(false, is(equalTo(map.containsValue("Priyadarshan")));
122     assertEquals(1, is(equalTo(map.size())));
123     exception.expect(NoSuchKeyException.class);
124     map.get(2);
125 }
126
127 @Test
128 public void testDynamicMapSize() throws DuplicateKeyFoundException {
129     map.save(1, "Priyan");
130     map.save(2, "Priyan");
131     map.save(3, "Priyan");
132     map.save(4, "Priyan");
133     map.save(5, "Priyan");
134     map.save(6, "Priyan");
135 }
136 }

SimpleMap.java
1 package org.self.learn;
2
3 public class SimpleMap<K, V> {
4     private KeyValuePair<K, V>[] keyValuePair;
5     private int size;
6
7     @SuppressWarnings("unchecked")
8     public SimpleMap() {
9         this.size = 0;
10        this.keyValuePair = (KeyValuePair<K, V>) new KeyValuePair[5];
11    }
12
13    public int size() {
14        return this.size;
15    }
16
17    public void save(K key, V value) throws DuplicateKeyFoundException {
18        if(containsKey(key)) {
19            throw new DuplicateKeyFoundException();
20        }
21        keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
22    }
23 }

Markers Properties Servers Data Source Explorer Snippets JUnit Debug
Finished after 0.038 seconds
Runs: 12/12 Errors: 1 Failures: 0
org.self.learn.test.SimpleMapTest [Runner: JUnit 4] (0.001 s)
testMultipleSaveKeyValuePair (0.000 s)
testNullKeyAndValues (0.000 s)
testSize (0.000 s)
testNonEmptyMap (0.000 s) ...
Failure Trace
java.lang.ArrayIndexOutOfBoundsException: 5
at org.self.learn.SimpleMap.save(SimpleMap.java:22)
at org.self.learn.test.SimpleMapTest.testDynamicMapSize(SimpleMapTest.java:134)

```

Following is the completed test and corresponding implementation. Note that all the tests complete successfully at this point. And we have built all desired functionality into our map as well. That concludes this exercise.

Java EE - simple-map/src/main/java/org/self/learn/SimpleMap.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

SimpleMapTest.java

```
121     assertThat(false, isEqualTo(map.containsValue("Priyadarshi")));
122     assertThat(1, isEqualTo(map.size()));
123     exception.expect(NoSuchKeyException.class);
124     map.get(2);
125 }
126
127 @Test
128 public void testDynamicMapSize() throws DuplicateKeyFoundException {
129     map.save(1, "Priyan-1");
130     map.save(2, "Priyan-2");
131     map.save(3, "Priyan-3");
132     map.save(4, "Priyan-4");
133     map.save(5, "Priyan-5");
134     map.save(6, "Priyan-6");
135     assertThat("Priyan-6", isEqualTo(map.get(6)));
136     assertThat("Priyan-5", isEqualTo(map.get(5)));
137     assertThat("Priyan-4", isEqualTo(map.get(4)));
138     assertThat("Priyan-3", isEqualTo(map.get(3)));
139     assertThat("Priyan-2", isEqualTo(map.get(2)));
140     assertThat("Priyan-1", isEqualTo(map.get(1)));
141     assertEquals(6, map.size());
142 }
```

SimpleMap.java

```
11
12 }
13
14 public int size() {
15     return this.size;
16 }
17
18 public void save(K key, V value) throws DuplicateKeyFoundException {
19     if(!containsKey(key)) {
20         throw new DuplicateKeyFoundException();
21     }
22
23     if(this.size<this.keyValuePair.length) {
24         this.keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
25     } else {
26         @SuppressWarnings("unchecked")
27         KeyValuePair<K, V>[] newKeyValuePair = (KeyValuePair<K, V>[] ) new
28         System.arraycopy(this.keyValuePair, 0, newKeyValuePair, 0, this.size);
29         this.keyValuePair = newKeyValuePair;
30         this.keyValuePair[this.size++] = new KeyValuePair<K, V>(key, value);
31     }
32 }
```

Markers Properties Servers Data Source Explorer Snippets JUnit Debug

Finished after 0.033 seconds

Runs: 12/12 Errors: 0 Failures: 0

Failure Trace

Writable Smart Insert 28:25