private static <K, V> void moveToFront(Queue<Pair<K, V>> q, K key) {

Pair<K,V> temp = null;

for (int I =0; I < q.length(); i++) {

temp = q.dequeue();

if (temp.key().equals(key)) {

I = q.length();

}

q.enqueue(temp);

}

}

*@Test*

public void addTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2");

Map<String, String> ref = this.createFromArgsRef("1", "1", "2", "2",

"3", "3");

test.add("3", "3");

*assertEquals*(ref, test);

}

*@Test*

public void removeTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",

"3", "3");

Map<String, String> ref = this.createFromArgsRef("1", "1", "2", "2");

test.remove("3");

*assertEquals*(ref, test);

}

*@Test*

public void removeAnyTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",

"3", "3");

Map<String, String> ref = this.createFromArgsRef("2", "2", "3", "3");

test.removeAny();

*assertEquals*(ref, test);

}

*@Test*

public void valueTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",

"3", "3");

*assertEquals*("2", test.value("2"));

}

*@Test*

public void hasKeyTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",

"3", "3");

*assertEquals*(true, test.hasKey("1"));

*assertEquals*(true, test.hasKey("2"));

*assertEquals*(true, test.hasKey("3"));

*assertEquals*(false, test.hasKey("4"));

*assertEquals*(false, test.hasKey("0"));

}

*@Test*

public void sizeTest() {

Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",

"3", "3");

*assertEquals*(3, test.size());

}

A screenshot of a computer

Description automatically generated