```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code Map<String, String>}'s constructor and kernel
 * methods.
 *
 * @author Put your name here
 *
 */
public abstract class MapTest {

    /**
     * Invokes the appropriate {@code Map} constructor for the implementation
     * under test and returns the result.
     *
     * @return the new map
     * @ensures constructorTest = {}
     */
    protected abstract Map<String, String> constructorTest();

    /**
     * Invokes the appropriate {@code Map} constructor for the reference
     * implementation and returns the result.
     *
     * @return the new map
     * @ensures constructorRef = {}
     */
    protected abstract Map<String, String> constructorRef();

    /**
     *
     * Creates and returns a {@code Map<String, String>} of the implementation
     * under test type with the given entries.
     *
     * @param args
     *            the (key, value) pairs for the map
     * @return the constructed map
     * @requires <pre>
     * [args.length is even]  and
     * [the 'key' entries in args are unique]
     * </pre>
     * @ensures createFromArgsTest = [pairs in args]
     */
    private Map<String, String> createFromArgsTest(String... args) {
        assert args.length % 2 == 0 : "Violation of: args.length is even";
        Map<String, String> map = this.constructorTest();
        for (int i = 0; i < args.length; i += 2) {
            assert !map.hasKey(args[i]) : ""
                    + "Violation of: the 'key' entries in args are unique";
            map.add(args[i], args[i + 1]);
        }
        return map;
    }

    /**
     *
     * Creates and returns a {@code Map<String, String>} of the reference
     * implementation type with the given entries.
     *
```

```
64      * @param args
65      *            the (key, value) pairs for the map
66      * @return the constructed map
67      * @requires <pre>
68      * [args.length is even]  and
69      * [the 'key' entries in args are unique]
70      * </pre>
71      * @ensures createFromArgsRef = [pairs in args]
72      */
73     private Map<String, String> createFromArgsRef(String... args) {
74         assert args.length % 2 == 0 : "Violation of: args.length is even";
75         Map<String, String> map = this.constructorRef();
76         for (int i = 0; i < args.length; i += 2) {
77             assert !map.hasKey(args[i]) : ""
78                     + "Violation of: the 'key' entries in args are unique";
79             map.add(args[i], args[i + 1]);
80         }
81         return map;
82     }
83
84     @Test
85     public void addTest() {
86
87         Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2");
88         Map<String, String> ref = this.createFromArgsRef("1", "1", "3", "3",
89                 "2", "2");
90
91         test.add("3", "3");
92
93         assertEquals(ref, test);
94     }
95
96     @Test
97     public void removeTest() {
98
99         Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",
100                "3", "3");
101        Map<String, String> ref = this.createFromArgsRef("1", "1", "2", "2");
102
103        test.remove("3");
104
105        assertEquals(ref, test);
106    }
107
108    @Test
109    public void removeAnyTest() {
110
111        Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",
112                "3", "3");
113        Map<String, String> ref = this.createFromArgsRef("2", "2", "3", "3");
114
115        test.removeAny();
116
117        assertEquals(ref, test);
118    }
119
120    @Test
121    public void valueTest() {
122
```

```
123         Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",
124                 "3", "3");
125 
126         assertEquals("2", test.value("2"));
127     }
128 
129     @Test
130     public void hasKeyTest() {
131 
132         Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",
133                 "3", "3");
134 
135         assertEquals(true, test.hasKey("1"));
136         assertEquals(true, test.hasKey("2"));
137         assertEquals(true, test.hasKey("3"));
138         assertEquals(false, test.hasKey("4"));
139         assertEquals(false, test.hasKey("0"));
140     }
141 
142     @Test
143     public void sizeTest() {
144 
145         Map<String, String> test = this.createFromArgsTest("1", "1", "2", "2",
146                 "3", "3");
147 
148         assertEquals(3, test.size());
149     }
150 
151 }
152 
```