```java

/**
 *
 * @author ehn19
 */
public class ExercisesTest {

    Exercises ex = new Exercises();
    private ArrayList<Cat> cats;
    private ArrayList<Integer> data;
    /**
     * Test of readFile method, of class Exercises.
     */
    @org.junit.Test
    public void testReadFile() {
        System.out.println("Read from file");
        cats = ex.readFile("Cats.txt");

        int exp = 8;
        int result = cats.size();

        assertEquals(exp, result);

    }

    @Test(expected = IndexOutOfBoundsException.class)
    public void testReadWrongFile() {
        cats = ex.readFile("Cat.txt");
        cats.get(0);
    }

```

```
59      /**
60       * Test of getOldest method, of class Exercises.
61       */
62      @org.junit.Test
63      public void testGetOldest() {
64          System.out.println("getOldest");
65          cats = ex.readFile("Cats.txt");
66
67          String expRes = "Whiskers";
68          String res = ex.getOldest(cats);
69          assertEquals(expRes, res);
70      }
71
72      /**
73       * Test of removeSickCats method, of class Exercises.
74       */
75      @org.junit.Test
76      public void testRemoveSickCats() {
77          System.out.println("removeSickCats");
78          cats = ex.readFile("Cats.txt");
79          int expBefore = 8;
80          assertEquals(expBefore, cats.size());
81          ex.removeSickCats(cats);
82          int expAfter = 5;
83          assertEquals(expAfter, cats.size());
84      }
85
```

```
86      /**
87       * Test of updateName method, of class Exercises.
88       */
89      @org.junit.Test
90      public void testUpdateName() {
91          System.out.println("updateName");
92          cats = ex.readFile("Cats.txt");
93
94          String expResBefore = "Felix";
95          String resBefore = cats.get(1).getName();
96
97          assertEquals(expResBefore, resBefore);
98
99          ex.updateName(cats, "Felix", "Ole");
100
101         String expResAfter = "Ole";
102         String resAfter = cats.get(1).getName();
103
104         assertEquals(expResAfter, resAfter);
105     }
106
107     @org.junit.Test
108     public void testUpdateNameH() {
109         System.out.println("updateName with Hamcrest");
110         cats = ex.readFile("Cats.txt");
111         String expResAfter = "Ole";
112         String resBefore = cats.get(1).getName();
113
114         assertThat("Ole", not(equalTo(resBefore)));
115         ex.updateName(cats, "Felix", "Ole");
116         String resAfter = cats.get(1).getName();
117         assertThat(expResAfter, is(equalTo(resAfter)));
118     }
119
```

```
120     /**
121      * Test of updateCat method, of class Exercises.
122      */
123     @org.junit.Test
124     public void testUpdateCat() {
125         System.out.println("updateCat");
126         cats = ex.readFile("Cats.txt");
127 
128         Cat newCat = new Cat("Bo", 13, "gray", false);
129         ex.updateCat(cats, "Whiskers", newCat);
130         assertEquals(newCat, cats.get(2));
131 
132     }
133 
134     @org.junit.Test
135     public void testUpdateCatH() {
136         System.out.println("updateCat with Hamcrest");
137         cats = ex.readFile("Cats.txt");
138 
139         Cat newCat = new Cat("Bo", 13, "gray", false);
140         ex.updateCat(cats, "Whiskers", newCat);
141         assertThat(newCat, is(equalTo(cats.get(2))));
142     }
143 
```

```
154     /**
155      * Test of getSomeCats method, of class Exercises.
156      */
157     @org.junit.Test
158     public void testGetSomeCats() {
159         System.out.println("getSomeCats");
160         cats = ex.readFile("Cats.txt");
161 
162         int expBefore = 8;
163         assertEquals(expBefore, cats.size());
164         ArrayList<String> subList = ex.getSomeCats(cats, "B");
165         int expAfter = 2;
166         assertEquals(expAfter, subList.size());
167     }
168 
169     /**
170      * Test of sortedCats method, of class Exercises.
171      */
172     @org.junit.Test
173     public void sortedCats() {
174         System.out.println("sortedCats");
175         cats = ex.readFile("Cats.txt");
176 
177         String expBefore = "Bailey";
178         assertEquals(expBefore, cats.get(0).getName());
179         ArrayList<String> subList = ex.sortedCats(cats);
180         String expAfter = "Alex";
181         assertEquals(expAfter, subList.get(0));
182     }
183 
184     /**
185      * Test of sortedCats method, of class Exercises.
186      */
187     @org.junit.Test
188     public void getKittens() {
189         System.out.println("sortedCats");
190         cats = ex.readFile("Cats.txt");
191 
192         int expBefore = 8;
193         assertEquals(expBefore, cats.size());
194         ArrayList<Cat> subList = ex.getKittens(cats);
195         int expAfter = 1;
196         assertEquals(expAfter, subList.size());
197     }
```

```
198
199     @Test
200     public void testReadData() {
201         System.out.println("Read data from file");
202         data = ex.readData("Data.txt");
203
204         int exp = 90;
205         int result = data.size();
206
207         assertEquals(exp, result);
208
209     }
210
211     @Test
212     public void dataDrivenTestDataFileEndsWith() {
213         int lastValue = 90;
214         data = ex.readData("Data.txt");
215
216         assertThat(lastValue, is(equalTo(data.size())));
217     }
218 }
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running sem.firstsemexam.ExercisesTest
getSomeCats
updateCat
updateName with Hamcrest
updateCat with Hamcrest
updateName
Could not find file!
java.io.FileNotFoundException: Cat.txt (Den angivne fil blev ikke fundet)
sortedCats
Read data from file
Could not read from file!
java.io.IOException: Stream closed
Read from file
Could not read from file!
java.io.IOException: Stream closed
sortedCats
getOldest
removeSickCats
Tests run: 13, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.272 sec

Results :

Tests run: 13, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 2.493s
Finished at: Sun Mar 11 10:42:16 CET 2018
Final Memory: 7M/245M
------------------------------------------------------------------------
```