```
 1
 2 package staffTestsrc;
 3
 4 import static org.junit.jupiter.api.Assertions.*;
14
15 class StaffTest {
16     // create a new CharityStaff object
17     static CharityStaff cs = new CharityStaff();
18
19     @BeforeAll
20     static void setUpBeforeClass() throws Exception {
21         // populate the staff list with volunteers and employees
22         cs.populate();
23     }
24
25     @AfterAll
26     static void tearDownAfterClass() throws Exception {
27         // remove population at end of test
28         cs = null;
29     }
30
31     @Test
32     public void testGetTotalHours() {
33         // get the total hours worked by all staff
34         int totalHours = cs.getTotalHours();
35         // assert that the total hours is equal to the expected value
36         assertEquals(75, totalHours);
37     }
38
39     @Test
40     public void testFindPerson() {
41         // search for a person by name
42         Person p = cs.findPerson("Mary Munro");
43
44         // assert that the person object is not null and has the correct name
45         assertNotNull(p);
46         assertEquals("Mary Munro", p.getName().getFullName());
47     }
48
49     @Test
50     public void testListAllVolunteers() {
51         // get the list of all volunteers
52         String volunteerList = cs.listAllVolunteers();
53
54         // assert that the list of volunteers contains the expected names
55         assertTrue(volunteerList.contains("Helen Scott"));
56         assertTrue(volunteerList.contains("James Jackson"));
57     }
58
59     @Test
60     public void testPopulate() {
61         // get the list of all staff
62         String staffList = cs.listAllNames();
63
64         // assert that the list of staff contains the expected names
65         assertTrue(staffList.contains("Helen Scott"));
66         assertTrue(staffList.contains("James Jackson"));
67         assertTrue(staffList.contains("Tim Moore"));
68         assertTrue(staffList.contains("Mary Munro"));
69         assertTrue(staffList.contains("Keith Clark"));
```

```
70      }
71
72      @Test
73      public void testVolunteerEquals_True() {
74          // create two new Volunteer objects with the same values
75          Volunteer v1 = new Volunteer(21, new Name("Helen Scott"),
    "helen@hotmail.com", "0131 123 1234", 5);
76          Volunteer v2 = new Volunteer(21, new Name("Helen Scott"),
    "helen@hotmail.com", "0131 123 1234", 5);
77
78          // assert that the two Volunteer objects are not equal
79          assertTrue(v1.equals(v2));
80      }
81
82      @Test
83      public void testVolunteerEquals_False() {
84          // create two new Volunteer objects with different values
85          Volunteer v1 = new Volunteer(21, new Name("Helen Scott"),
    "helen@hotmail.com", "0131 123 1234", 5);
86          Volunteer v2 = new Volunteer(23, new Name("Paul Smith "),
    "Paul@hotmail.com", "0121 456 4567", 8);
87
88          // assert that the two Volunteer objects are not equal
89          assertFalse(v1.equals(v2));
90      }
91
92  }
93
```