New test cases report

We planned to write three test cases as required in the assignment description, but we found that some existing functionalities lack the same kind of testing, so we added them together. Also, we wrote a test case to check if we handled the issue we chose correctly.

TaskInfoTest

The ElasticSearch project does not have a test class for TaskInfo.java. So we think we can write some simple test cases to test its basic functions.

testToString

The toString method is an override method used to convert the TaskInfo object to String. We use some sample information to build an object for testing.

```
    public void testToString() {

        TaskInfo taskInfo1 = new TaskInfo(new TaskId("001"));
        taskInfo1.setType("type");
        taskInfo1.setAction("action");
        taskInfo1.setDescription("description");

        String expected = "TaskInfo{" +
            "taskId=001, type='type', action='action,
description='description', startTime=, runningTimeNanos=, cancellable=,
parentTaskId=, status=, headers=}";
        assertEquals(expected,taskInfo1.toString());

    }
```

testEquals

The Equals method is used to determine whether two TaskInfo objects are totally the same. We build four instances of the class. The first two have the exact same parameters, while the taskinfo3 has different taskId and the last instance has a different description. We need to check whether the equals method works as expected. The object should be equal to itself. The different objects with the same parameters should be recognized as the same. The objects with different parameters(no matter which) should be judged as not equal.

```java
public void testEquals() {
   TaskInfo taskInfo1 = new TaskInfo(new TaskId("001"));
   taskInfo1.setType("type");
   taskInfo1.setAction("action");
   taskInfo1.setDescription("description");

   TaskInfo taskInfo2 = new TaskInfo(new TaskId("001"));
   taskInfo2.setType("type");
   taskInfo2.setAction("action");
   taskInfo2.setDescription("description");

   TaskInfo taskInfo3 = new TaskInfo(new TaskId("003"));
   taskInfo3.setType("type");
   taskInfo3.setAction("action");
   taskInfo3.setDescription("description");

   TaskInfo taskInfo4 = new TaskInfo(new TaskId("001"));
   taskInfo4.setType("type");
   taskInfo4.setAction("action");
   taskInfo4.setDescription("description4");

   assertEquals(taskInfo1,taskInfo1);
   assertEquals(taskInfo1,taskInfo2);
   assertNotEquals(taskInfo1,taskInfo3);
   assertNotEquals(taskInfo1,taskInfo4);
}
```

testHashCode

HashCode should perform almost the same with equal, which means equal objects should have the same hashcode while the unequal ones should have different hashcode.

```java
public void testHashCode() {
   assertEquals(taskInfo1.hashCode(),taskInfo1.hashCode());
   assertEquals(taskInfo1.hashCode(),taskInfo2.hashCode());
   assertNotEquals(taskInfo1.hashCode(),taskInfo3.hashCode());
   assertNotEquals(taskInfo1.hashCode(),taskInfo4.hashCode());
}
```

IndexTest

There is already an IndexTest class. But some of the methods in the Index.java are not covered in the test cases. So we add some test cases to improve the method coverage.

testHashCode

The test for the equals method is provided. But the test for the hashcode method is missing. So we should test it from four aspects, in the exact same way as in the TaskInfo Test.

```
public void testHashCode() {
   Index index1 = new Index("a", "a");
   Index index2 = new Index("a", "a");
   Index index3 = new Index("a", "b");
   Index index4 = new Index("b", "a");
   //test that the same object has the same hashcode
   assertEquals(index1.hashCode(), index1.hashCode());
   //test that different objects with same name and uuid have the same hashcode
   assertEquals(index1.hashCode(), index2.hashCode());
   //test that objects with different name have different hashcode
   assertNotEquals(index1.hashCode(), index3.hashCode());
   //test that objects with different uuid have different hashcode
   assertNotEquals(index1.hashCode(), index4.hashCode());
}
```

testToXContent

XContent is the self-defined class for handling json objects and parsing in ElasticSearch. The index class has a method toXContent to transfer Index objects to XContentBuilder. The test case is designed to check whether the transferring function is behaving correctly.

```
public void testToXContent() throws Exception  {
   XContentBuilder builder1 = XContentFactory.jsonBuilder();
   builder1.startObject();
   builder1.field("index_name", "a");
   builder1.field("index_uuid", "b");
   builder1.endObject();
   Index index = new Index("a", "b");
   XContentBuilder builder2 = XContentFactory.jsonBuilder();
   index.toXContent(builder2,EMPTY_PARAMS);
   assertEquals(Strings.toString(builder1), Strings.toString(builder2));
}
```

Test for our issue

In our issue, we deprecate four gateway settings. A test case is written to check whether the deprecation is working.

```
public void testDeprecatedSettings() {
   GatewayService service = createService(Settings.builder());

   service = createService(Settings.builder().put("gateway.expected_nodes", 1));
   assertSettingDeprecationsAndWarnings(new Setting<?>[]
{GatewayService.EXPECTED_NODES_SETTING });

   service = createService(Settings.builder().put("gateway.expected_master_nodes",
1));
   assertSettingDeprecationsAndWarnings(new Setting<?>[]
{GatewayService.EXPECTED_MASTER_NODES_SETTING });

   service = createService(Settings.builder().put("gateway.recover_after_nodes",
1));
   assertSettingDeprecationsAndWarnings(new Setting<?>[]
{GatewayService.RECOVER_AFTER_NODES_SETTING });

   service =
createService(Settings.builder().put("gateway.recover_after_master_nodes", 1));
   assertSettingDeprecationsAndWarnings(new Setting<?>[]
{GatewayService.RECOVER_AFTER_MASTER_NODES_SETTING });
}
```