**import** org.junit.Test;  
  
**public class** Main<CreateLabResult, TestRequest> {  
  
  
 **private** Object **HttpStatus**;  
 **private** Object **User**;  
  
 @Test  
 @WithUserDetails(value = **"tester"**)  
 **public** <TestRequest> **void** calling\_assignForLabTest\_with\_valid\_test\_request\_id\_should\_update\_the\_request\_status(){  
  
 TestRequest testRequest = getTestRequestByStatus(RequestStatus.***INITIATED***);  
 *//Implement this method  
  
 //Create another object of the TestRequest method and explicitly assign this object for Lab Test using assignForLabTest() method  
 // from labRequestController class. Pass the request id of testRequest object.  
  
 //Use assertThat() methods to perform the following two comparisons  
 // 1. the request ids of both the objects created should be same  
 // 2. the status of the second object should be equal to 'INITIATED'  
 // make use of assertNotNull() method to make sure that the lab result of second object is not null  
 // use getLabResult() method to get the lab result* Object testRequestQueryService =**null**;  
 **if** (testRequestQueryService != **null**) {  
 **return** testRequestQueryService.findByTester(RequestStatus.***INITIATED***);  
 }  
  
  
 }  
  
 **public** <TestRequest> TestRequest getTestRequestByStatus(RequestStatus status) {  
 Object testRequestQueryService = **new** Object();  
 **return** testRequestQueryService.findBy(status).stream().findFirst().get();  
 }  
  
 @Test  
 @WithUserDetails(value = **"tester"**)  
 **public void** calling\_assignForLabTest\_with\_valid\_test\_request\_id\_should\_throw\_exception(){  
  
 Long InvalidRequestId= -34L;  
  
 *//Implement this method  
  
  
 // Create an object of ResponseStatusException . Use assertThrows() method and pass assignForLabTest() method  
 // of labRequestController with InvalidRequestId as Id  
  
  
 //Use assertThat() method to perform the following comparison  
 // the exception message should be contain the string "Invalid ID"*   
 }  
  
 @Test  
 @WithUserDetails(value = **"tester"**)  
 **public void** calling\_updateLabTest\_with\_valid\_test\_request\_id\_should\_update\_the\_request\_status\_and\_update\_test\_request\_details(){  
  
 TestRequest testRequest = Object userLoggedInService = **new** Object();  
 User tester = userLoggedInService.getLoggedInUser();  
  
 Object testRequestUpdateService = **null**;  
 Object id = **new** Object();  
 **assert** testRequestUpdateService != **null**;  
 **return** testRequestUpdateService.assignForLabTest(id,tester);  
getTestRequestByStatus(RequestStatus.***LAB\_TEST\_IN\_PROGRESS***);  
  
 *//Implement this method  
 //Create an object of CreateLabResult and call getCreateLabResult() to create the object. Pass the above created object as the parameter  
  
 //Create another object of the TestRequest method and explicitly update the status of this object  
 // to be 'LAB\_TEST\_IN\_PROGRESS'. Make use of updateLabTest() method from labRequestController class (Pass the previously created two objects as parameters)  
  
 //Use assertThat() methods to perform the following three comparisons  
 // 1. the request ids of both the objects created should be same  
 // 2. the status of the second object should be equal to 'LAB\_TEST\_COMPLETED'  
 // 3. the results of both the objects created should be same. Make use of getLabResult() method to get the results.* Object userLoggedInService = **null**;  
 User tester = userLoggedInService.getLoggedInUser();  
  
 Object testRequestUpdateService = **null**;  
 Object createLabResult = **null**;  
 Object id = **null**;  
 **return** testRequestUpdateService.updateLabTest(id,createLabResult,tester);  
  
  
  
 }  
  
  
 @Test  
 @WithUserDetails(value = **"tester"**)  
 **public void** calling\_updateLabTest\_with\_invalid\_test\_request\_id\_should\_throw\_exception(){  
  
 TestRequest testRequest = getTestRequestByStatus(RequestStatus.***LAB\_TEST\_IN\_PROGRESS***);  
  
  
 *//Implement this method  
  
 //Create an object of CreateLabResult and call getCreateLabResult() to create the object. Pass the above created object as the parameter  
  
 // Create an object of ResponseStatusException . Use assertThrows() method and pass updateLabTest() method  
 // of labRequestController with a negative long value as Id and the above created object as second parameter  
 //Refer to the TestRequestControllerTest to check how to use assertThrows() method  
  
  
 //Use assertThat() method to perform the following comparison  
 // the exception message should be contain the string "Invalid ID"* Object userLoggedInService = **null**;  
 User tester = userLoggedInService.getLoggedInUser();  
  
 Object testRequestUpdateService = **null**;  
 Object createLabResult = **null**;  
 Object id = **null**;  
 **return** testRequestUpdateService.updateLabTest(id,createLabResult,tester);  
  
 **throw new** ResponseStatusException(**HttpStatus**.NOT\_IMPLEMENTED,**"Not implemented"**);  
  
 }  
  
 @Test  
 @WithUserDetails(value = **"tester"**)  
 **public** <TestRequest> **void** calling\_updateLabTest\_with\_invalid\_empty\_status\_should\_throw\_exception(){  
  
 TestRequest testRequest = getTestRequestByStatus(RequestStatus.***LAB\_TEST\_IN\_PROGRESS***);  
  
 *//Implement this method  
  
 //Create an object of CreateLabResult and call getCreateLabResult() to create the object. Pass the above created object as the parameter  
 // Set the result of the above created object to null.  
  
 // Create an object of ResponseStatusException . Use assertThrows() method and pass updateLabTest() method  
 // of labRequestController with request Id of the testRequest object and the above created object as second parameter  
 //Refer to the TestRequestControllerTest to check how to use assertThrows() method  
  
  
 //Use assertThat() method to perform the following comparison  
 // the exception message should be contain the string "ConstraintViolationException"* Object userLoggedInService = **null**;  
 User tester = userLoggedInService.getLoggedInUser();  
  
 Object testRequestUpdateService = **null**;  
 Object createLabResult = **null**;  
 Object id = **null**;  
 **return** testRequestUpdateService.updateLabTest(id,createLabResult,tester);  
  
 **throw new** ResponseStatusException(**HttpStatus**.NOT\_IMPLEMENTED,**"ConstraintViolationException"**);  
  
 }  
  
 **public** <TestRequest> CreateLabResult getCreateLabResult(TestRequest testRequest) {  
  
 *//Create an object of CreateLabResult and set all the values  
 // Return the object* Object CreateLabResult = (Long Object id;  
 id, @Valid CreateLabResult Object createLabResult;  
 createLabResult, **User** Object tester;  
 tester);  
 Object newObject;  
 CreateLabResult createLabResult1 = (CreateLabResult) (CreateLabResult = newObject);  
 **return** createLabResult1;  
   
  
  
 **return null**; *// Replace this line with your code* }  
  
 **private class** ResponseStatusException {  
 **public** ResponseStatusException(Object p0, String constraintViolationException) {  
 }  
 }  
}