Spring AU ’21 – Junit, Mockito, Logger – Afternoon Session

Name: Sheik Abudhahir K

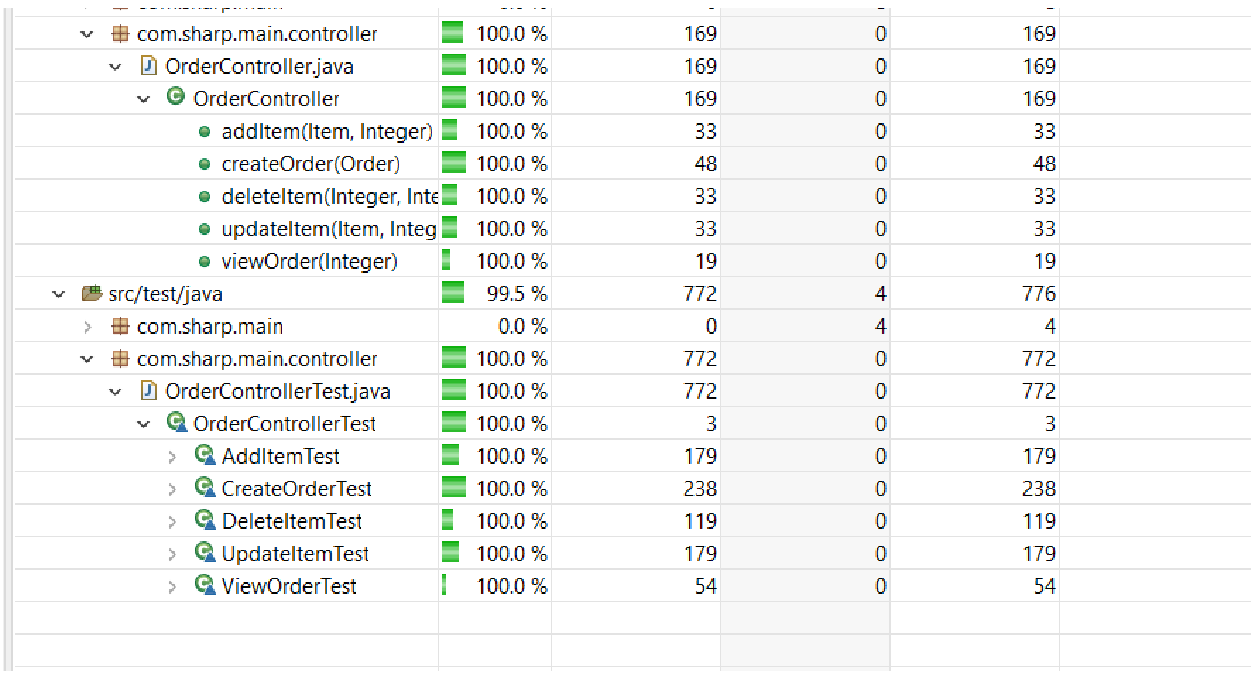
Date: 25/01/2021

Source code uploaded in Git

Question:

<u>Write Junit test cases for each and every use case with 100% coverage</u>

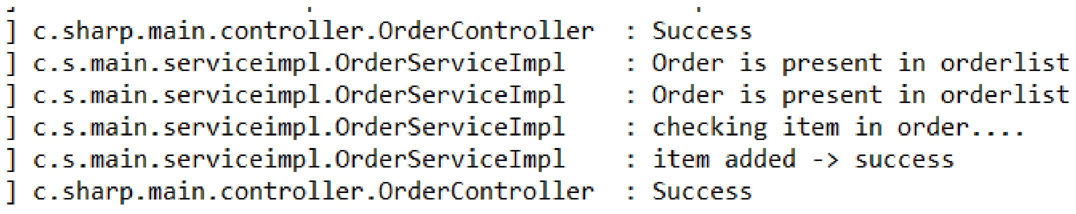
All apis are tested with 100% coverage



com.sharp.main.controller	100.0 %	169	0	169
OrderController.java	100.0 %	169	0	169
OrderController	100.0 %	169	0	169
addItem(Item, Integer)	100.0 %	33	0	33
createOrder(Order)	100.0 %	48	0	48
deleteItem(Integer, Inte	100.0 %	33	0	33
updateItem(Item, Integ	100.0 %	33	0	33
viewOrder(Integer)	100.0 %	19	0	19
src/test/java	99.5 %	772	4	776
com.sharp.main	0.0 %	0	4	4
com.sharp.main.controller	100.0 %	772	0	772
OrderControllerTest.java	100.0 %	772	0	772
OrderControllerTest	100.0 %	3	0	3
AddItemTest	100.0 %	179	0	179
CreateOrderTest	100.0 %	238	0	238
DeleteItemTest	100.0 %	119	0	119
UpdateItemTest	100.0 %	179	0	179
ViewOrderTest	100.0 %	54	0	54

Totally 5 apis and 16 testcases are there, both are had 100% coverage

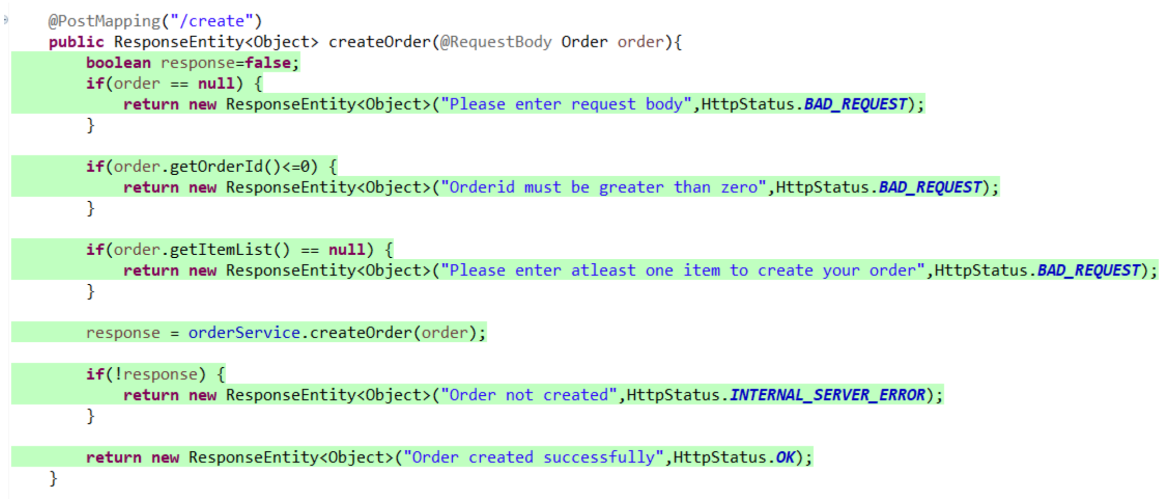
Logger output: create, view, add item



```
] c.sharp.main.controller.OrderController  : Success
] c.s.main.serviceimpl.OrderServiceImpl    : Order is present in orderlist
] c.s.main.serviceimpl.OrderServiceImpl    : Order is present in orderlist
] c.s.main.serviceimpl.OrderServiceImpl    : checking item in order....
] c.s.main.serviceimpl.OrderServiceImpl    : item added -> success
] c.sharp.main.controller.OrderController  : Success
```

Each test case should contain at least one assert statement and No real time calls should happen, all calls should be completely mocked

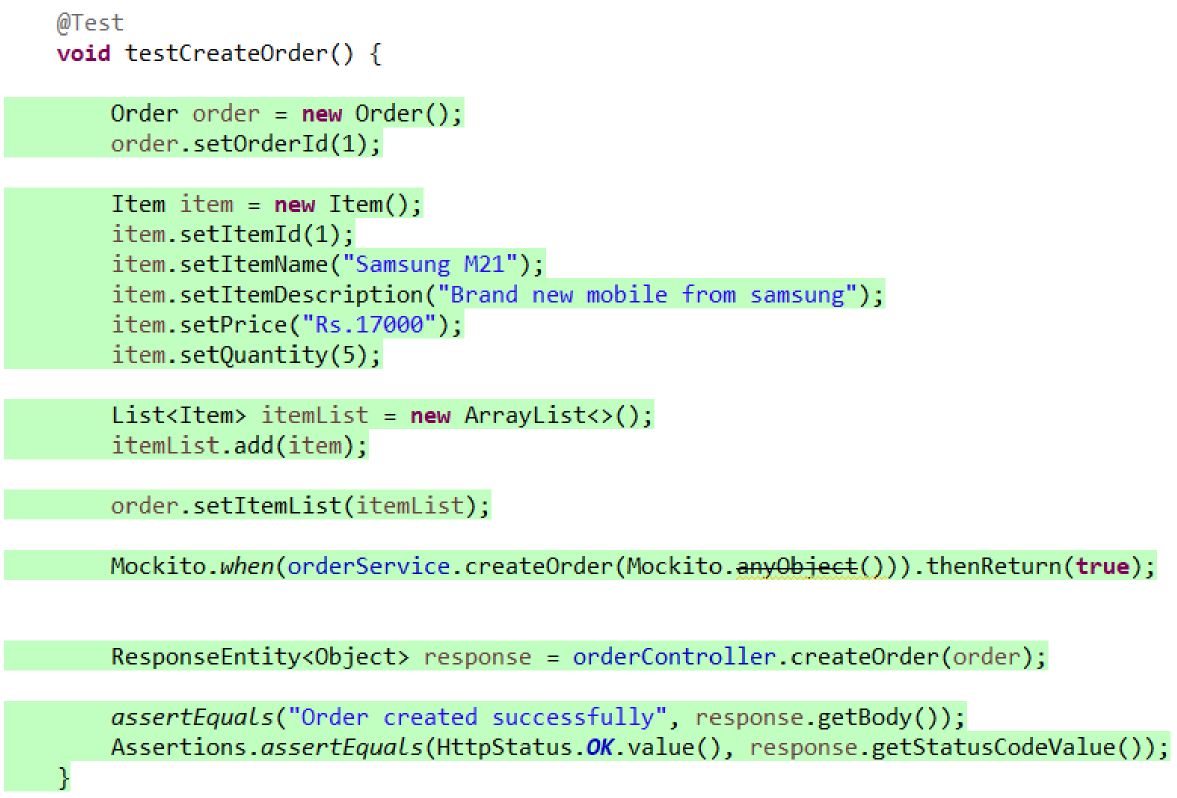
Normal java code: only for createOrder api



```java
    @PostMapping("/create")
    public ResponseEntity<Object> createOrder(@RequestBody Order order){
        boolean response=false;
        if(order == null) {
            return new ResponseEntity<Object>("Please enter request body",HttpStatus.BAD_REQUEST);
        }

        if(order.getOrderId()<=0) {
            return new ResponseEntity<Object>("Orderid must be greater than zero",HttpStatus.BAD_REQUEST);
        }

        if(order.getItemList() == null) {
            return new ResponseEntity<Object>("Please enter atleast one item to create your order",HttpStatus.BAD_REQUEST);
        }

        response = orderService.createOrder(order);

        if(!response) {
            return new ResponseEntity<Object>("Order not created",HttpStatus.INTERNAL_SERVER_ERROR);
        }

        return new ResponseEntity<Object>("Order created successfully",HttpStatus.OK);
    }
```

Testcase code(5 cases in createOrder api): with asset statements and Mockito

Order created successfully



```java
    @Test
    void testCreateOrder() {

        Order order = new Order();
        order.setOrderId(1);

        Item item = new Item();
        item.setItemId(1);
        item.setItemName("Samsung M21");
        item.setItemDescription("Brand new mobile from samsung");
        item.setPrice("Rs.17000");
        item.setQuantity(5);

        List<Item> itemList = new ArrayList<>();
        itemList.add(item);

        order.setItemList(itemList);

        Mockito.when(orderService.createOrder(Mockito.~~anyObject~~())).thenReturn(true);


        ResponseEntity<Object> response = orderController.createOrder(order);

        assertEquals("Order created successfully", response.getBody());
        Assertions.assertEquals(HttpStatus.OK.value(), response.getStatusCodeValue());
    }
```

Order not created case

```
@Test
void testCreateNotOrder() {

    Order order = new Order();
    order.setOrderId(1);

    Item item = new Item();
    item.setItemId(1);
    item.setItemName("Samsung M21");
    item.setItemDescription("Brand new mobile from samsung");
    item.setPrice("Rs.17000");
    item.setQuantity(5);

    List<Item> itemList = new ArrayList<>();
    itemList.add(item);

    order.setItemList(itemList);

    Mockito.when(orderService.createOrder(Mockito.~~anyObject~~())).thenReturn(false);


    ResponseEntity<Object> response = orderController.createOrder(order);

    assertEquals("Order not created", response.getBody());
    Assertions.assertEquals(HttpStatus.INTERNAL_SERVER_ERROR.value(), response.getStatusCodeValue());
}
```

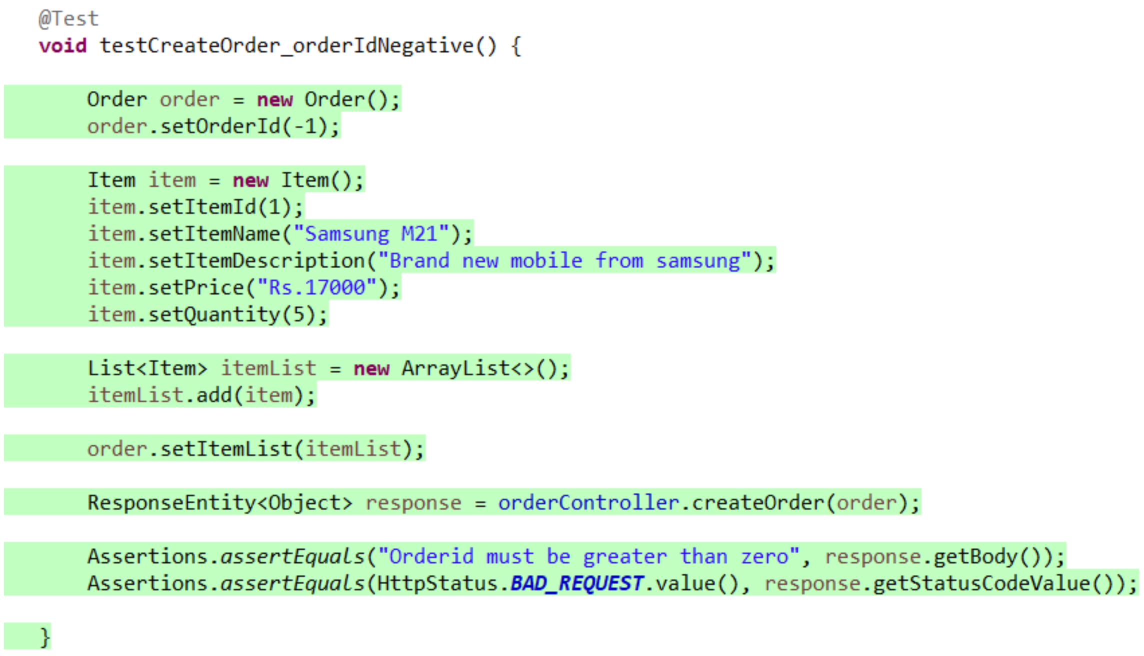
Check api data is null

```
@Test
void testCreateOrder_nullCheck() {

    ResponseEntity<Object> response = orderController.createOrder(null);

    Assertions.assertEquals("Please enter request body", response.getBody());
    Assertions.assertEquals(HttpStatus.BAD_REQUEST.value(), response.getStatusCodeValue());

}
```

Check item in order is null

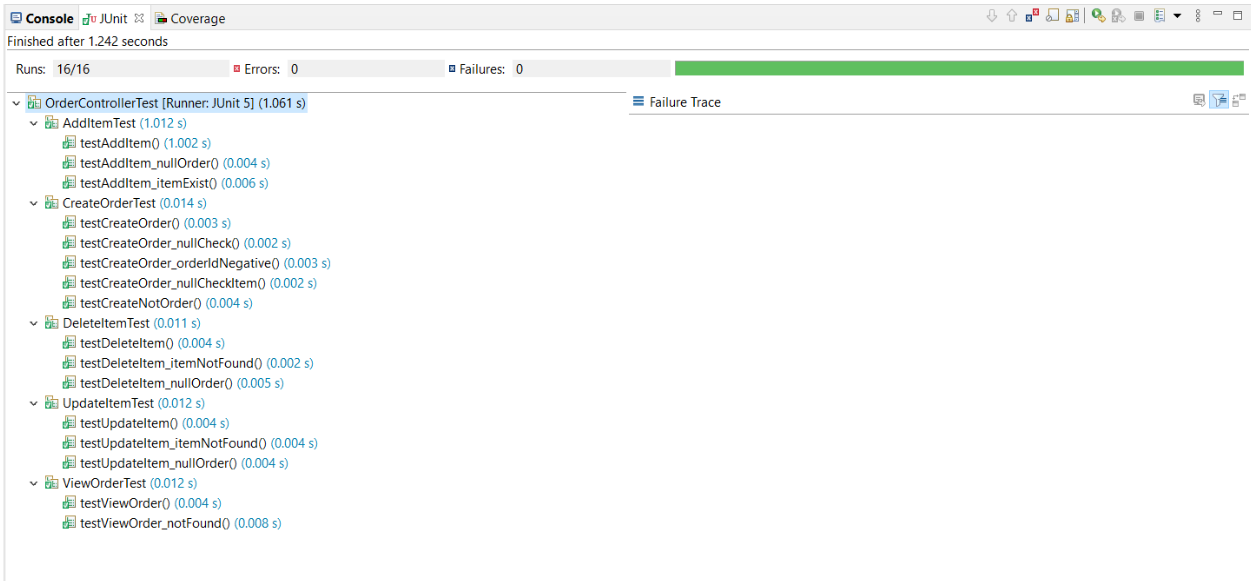
```
@Test
void testCreateOrder_nullCheckItem() {

    Order order = new Order();
    order.setOrderId(1);

    ResponseEntity<Object> response = orderController.createOrder(order);

    Assertions.assertEquals("Please enter atleast one item to create your order", response.getBody());
    Assertions.assertEquals(HttpStatus.BAD_REQUEST.value(), response.getStatusCodeValue());

}
```

Check for orderId validation



```
@Test
void testCreateOrder_orderIdNegative() {

    Order order = new Order();
    order.setOrderId(-1);

    Item item = new Item();
    item.setItemId(1);
    item.setItemName("Samsung M21");
    item.setItemDescription("Brand new mobile from samsung");
    item.setPrice("Rs.17000");
    item.setQuantity(5);

    List<Item> itemList = new ArrayList<>();
    itemList.add(item);

    order.setItemList(itemList);

    ResponseEntity<Object> response = orderController.createOrder(order);

    Assertions.assertEquals("Orderid must be greater than zero", response.getBody());
    Assertions.assertEquals(HttpStatus.BAD_REQUEST.value(), response.getStatusCodeValue());

}
```

Junit testcase output:



```
Finished after 1.242 seconds
Runs: 16/16    Errors: 0    Failures: 0

OrderControllerTest [Runner: JUnit 5] (1.061 s)
    AddItemTest (1.012 s)
        testAddItem() (1.002 s)
        testAddItem_nullOrder() (0.004 s)
        testAddItem_itemExist() (0.006 s)
    CreateOrderTest (0.014 s)
        testCreateOrder() (0.003 s)
        testCreateOrder_nullCheck() (0.002 s)
        testCreateOrder_orderIdNegative() (0.003 s)
        testCreateOrder_nullCheckItem() (0.002 s)
        testCreateNotOrder() (0.004 s)
    DeleteItemTest (0.011 s)
        testDeleteItem() (0.004 s)
        testDeleteItem_itemNotFound() (0.002 s)
        testDeleteItem_nullOrder() (0.005 s)
    UpdateItemTest (0.012 s)
        testUpdateItem() (0.004 s)
        testUpdateItem_itemNotFound() (0.004 s)
        testUpdateItem_nullOrder() (0.004 s)
    ViewOrderTest (0.012 s)
        testViewOrder() (0.004 s)
        testViewOrder_notFound() (0.008 s)
```

Loggers are added in the code