UNIT TEST AND SPOCK

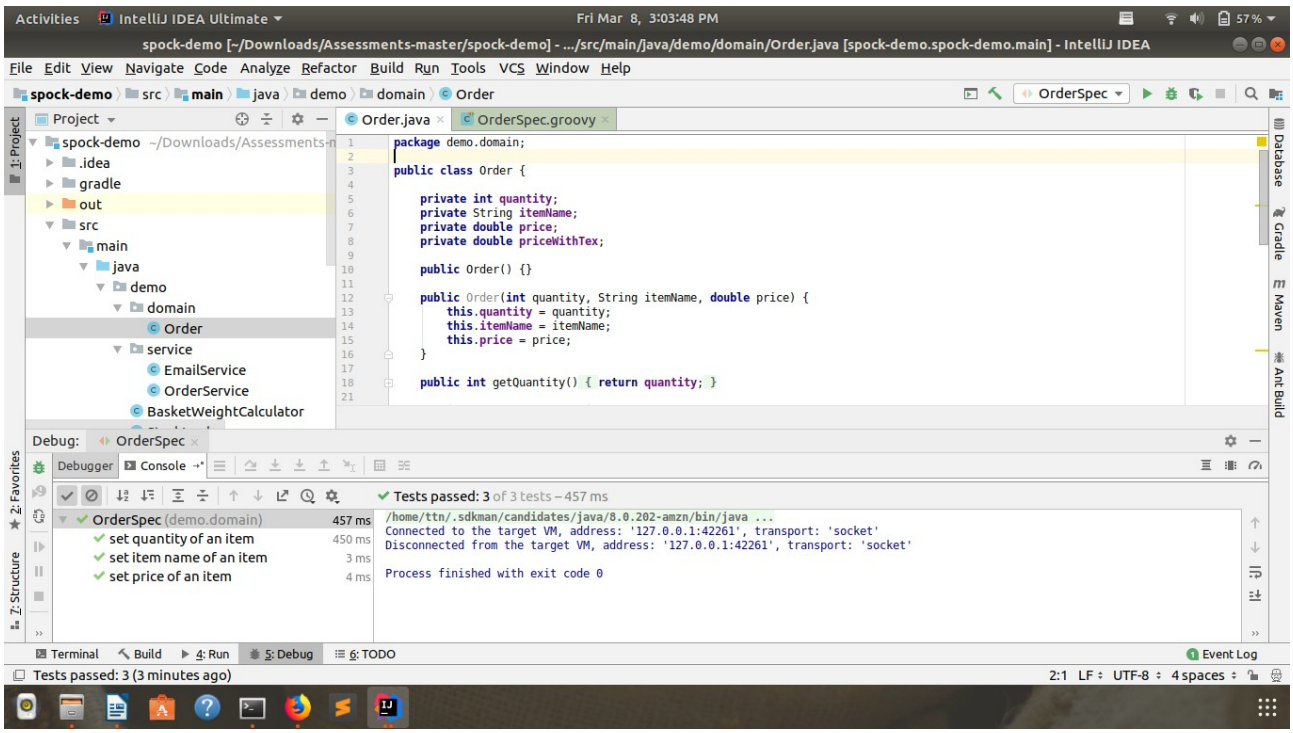
Finish the incomplete test cases for following:

- Order.java
- EmailService.java

ANSWERS

ANS1.

Order.java

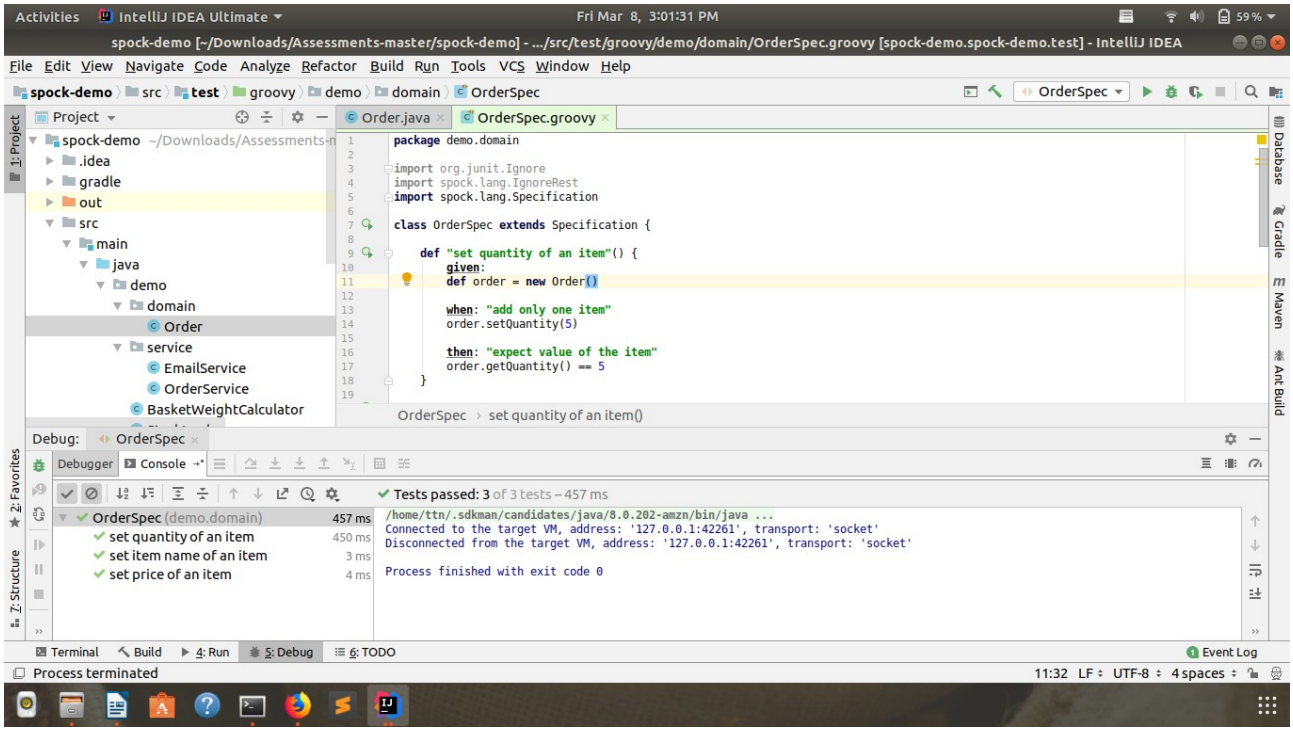



```
package demo.domain;

public class Order {
     private int quantity;
     private String itemName;
     private double price;
     private double priceWithTex;
     public Order() {}
     public Order(int quantity, String itemName, double price) {
          this.quantity = quantity;
          this.itemName = itemName;
          this.price = price;
     }
     public int getQuantity() {
          return quantity;
     }
     public Order setQuantity(int quantity) {
          this.quantity = quantity;
          return this;
     }
     public String getItemName() {
          return itemName;
     }
     public Order setItemName(String itemName) {
          this.itemName = itemName;
```

```
        return this;
    }
    public double getPrice() {
        return price;
    }
    public Order setPrice(double price) {
        this.price = price;
        return this;
    }
    public double getPriceWithTex() {
        return priceWithTex;
    }
    public Order setPriceWithTex(double priceWithTex) {
        this.priceWithTex = priceWithTex;
        return this;
    }
}
```

OderSpec.groovy



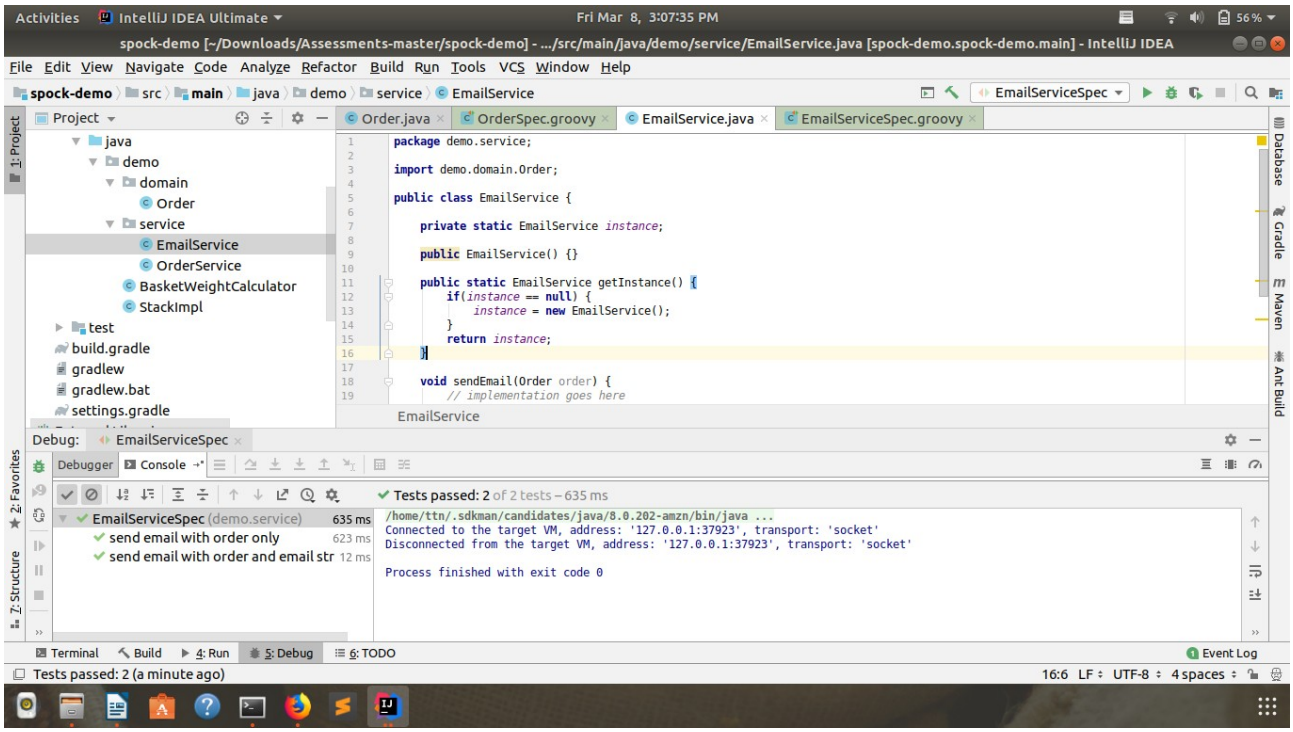

```
package demo.domain
import org.junit.Ignore
import spock.lang.IgnoreRest
import spock.lang.Specification
class OrderSpec extends Specification {
    def "set quantity of an item"() {
        given:
        def order = new Order()
        when: "add only one item"
        order.setQuantity(5)
        then: "expect value of the item"
        order.getQuantity() == 5
    }
    def "set item name of an item"() {
        given:
        def order = new Order()
        when: "add only one item"
        order.setItemName("apple")
```

```
        then: "expect value of the item"
        order.getItemName() == "apple"
    }

    def "set price of an item"() {
        given:
        def order = new Order()
        when: "add only one item"
        order.setPrice(55)
        then: "expect value of the item"
        order.getPrice() == 55
    }
}
```

ANS2.

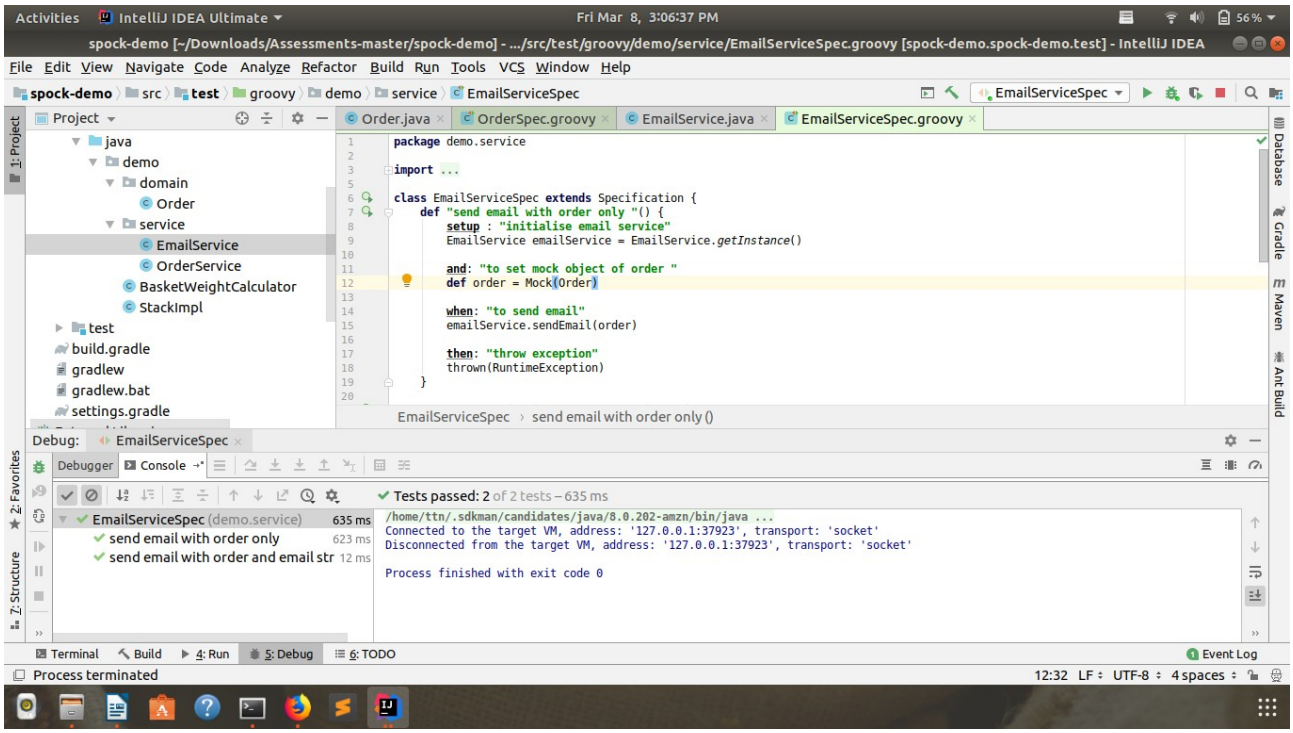
EmailService.java




```
package demo.service;
import demo.domain.Order;
public class EmailService {
    private static EmailService instance;
    public EmailService() {}
    public static EmailService getInstance() {
        if(instance == null) {
            instance = new EmailService();
        }
        return instance;
    }
    void sendEmail(Order order) {
        // implementation goes here
        throw new RuntimeException();
    }
    boolean sendEmail(Order order, String cc) {
        // implementation goes here
        return true;
```

```
    }
}
```

EmailServiceSpec.groovy




```groovy
package demo.service
import demo.domain.Order
import spock.lang.Specification
class EmailServiceSpec extends Specification {
    def "send email with order only "() {
        setup : "initialise email service"
        EmailService emailService = EmailService.getInstance()
        and: "to set mock object of order "
        def order = Mock(Order)
        when: "to send email"
        emailService.sendEmail(order)
        then: "throw exception"
        thrown(RuntimeException)
    }
    def "send email with order and email string "() {
        setup: "initialise email service "
        EmailService emailService = EmailService.getInstance()
        and: "to set mock object of order "
        def order = Mock(Order)
        and:"initialise string"
        String str="emailName"
        when: "to send email"
        Boolean result =emailService.sendEmail(order,str)
        then: "throw exception"
        result
    }
}
```