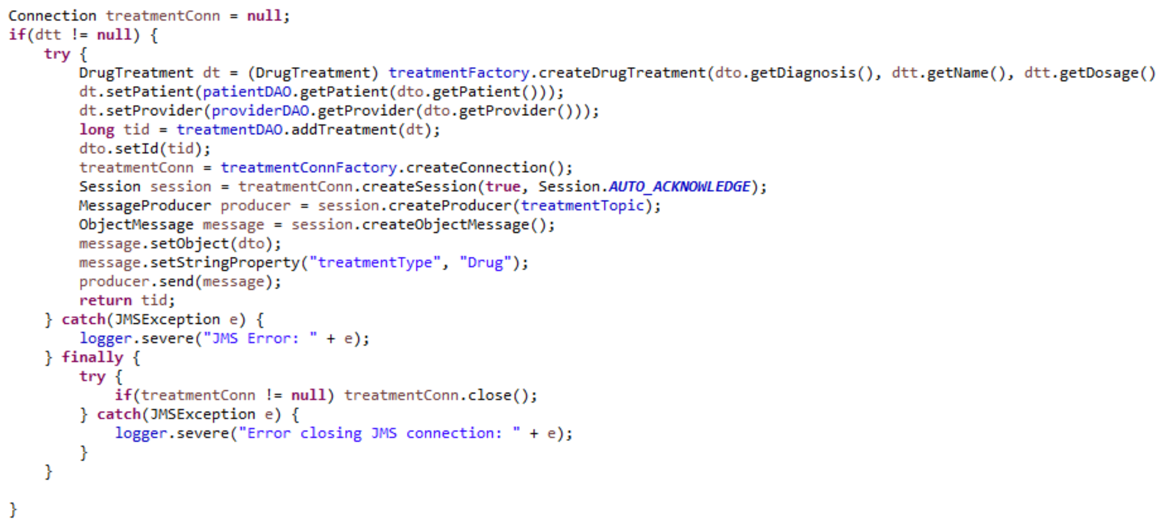
Assignment 9 Report

Rong Wan

This assignment is about implementing JMS on the basis of the code from previous assignments. I downloaded external JAR file for JMS to work. The most important parts of this assignment are annotations and calls into DTO and DAO with factory methods, and dependency injection is also used in this assignment. Below are some fractions of the code which reflects the main points of this assignment.

All settings stay the same as previous assignments, except that the context name specified in the POM file may not work so the name in the URL could be clinic-rest or ClinicRestWebService-1.0.0, but all functions and resources are fine.

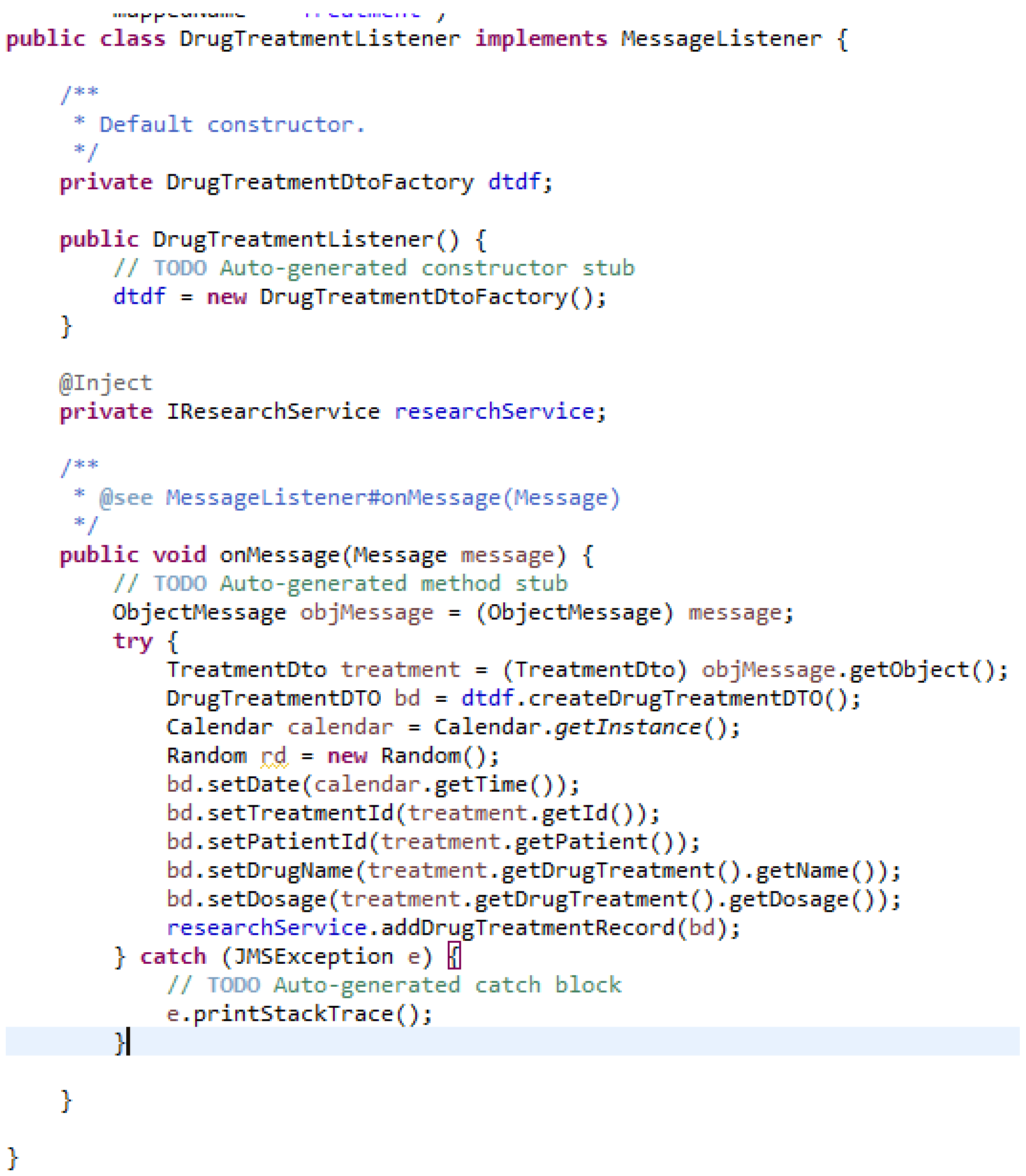


```
Connection treatmentConn = null;
if(dtt != null) {
    try {
        DrugTreatment dt = (DrugTreatment) treatmentFactory.createDrugTreatment(dto.getDiagnosis(), dtt.getName(), dtt.getDosage()
        dt.setPatient(patientDAO.getPatient(dto.getPatient()));
        dt.setProvider(providerDAO.getProvider(dto.getProvider()));
        long tid = treatmentDAO.addTreatment(dt);
        dto.setId(tid);
        treatmentConn = treatmentConnFactory.createConnection();
        Session session = treatmentConn.createSession(true, Session.AUTO_ACKNOWLEDGE);
        MessageProducer producer = session.createProducer(treatmentTopic);
        ObjectMessage message = session.createObjectMessage();
        message.setObject(dto);
        message.setStringProperty("treatmentType", "Drug");
        producer.send(message);
        return tid;
    } catch(JMSException e) {
        logger.severe("JMS Error: " + e);
    } finally {
        try {
            if(treatmentConn != null) treatmentConn.close();
        } catch(JMSException e) {
            logger.severe("Error closing JMS connection: " + e);
        }
    }

}
```

The ProviderService part is mostly the same for all three kinds of treatments

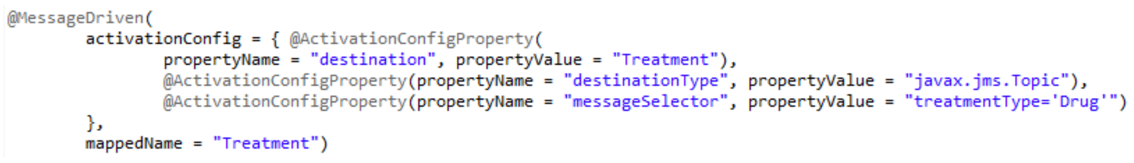
```java
public class TreatmentListener implements MessageListener {

    /**
     * Default constructor.
     */
    private BillingDtoFactory billingDtoFactory;

    public TreatmentListener() {
        // TODO Auto-generated constructor stub
        billingDtoFactory = new BillingDtoFactory();
    }

    @Inject
    private IBillingService billingService;

    /**
     * @see MessageListener#onMessage(Message)
     */
    public void onMessage(Message message) {
        // TODO Auto-generated method stub
        ObjectMessage objMessage = (ObjectMessage) message;
        try {
            TreatmentDto treatment = (TreatmentDto) objMessage.getObject();
            BillingDTO bd = billingDtoFactory.createBillingDTO();
            Calendar calendar = Calendar.getInstance();
            Random rd = new Random();
            bd.setDate(calendar.getTime());
            bd.setTreatmentId(treatment.getId());
            bd.setDescription(treatment.getDiagnosis());
            bd.setAmount(rd.nextFloat()*500);
            billingService.addBillingRecord(bd);
        } catch (JMSException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

Treatment listener is created using wizard according to demo videos



```java
        mappedName = "Treatment")
public class DrugTreatmentListener implements MessageListener {

    /**
     * Default constructor.
     */
    private DrugTreatmentDtoFactory dtdf;

    public DrugTreatmentListener() {
        // TODO Auto-generated constructor stub
        dtdf = new DrugTreatmentDtoFactory();
    }

    @Inject
    private IResearchService researchService;

    /**
     * @see MessageListener#onMessage(Message)
     */
    public void onMessage(Message message) {
        // TODO Auto-generated method stub
        ObjectMessage objMessage = (ObjectMessage) message;
        try {
            TreatmentDto treatment = (TreatmentDto) objMessage.getObject();
            DrugTreatmentDTO bd = dtdf.createDrugTreatmentDTO();
            Calendar calendar = Calendar.getInstance();
            Random rd = new Random();
            bd.setDate(calendar.getTime());
            bd.setTreatmentId(treatment.getId());
            bd.setPatientId(treatment.getPatient());
            bd.setDrugName(treatment.getDrugTreatment().getName());
            bd.setDosage(treatment.getDrugTreatment().getDosage());
            researchService.addDrugTreatmentRecord(bd);
        } catch (JMSException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

So does the Research Listener, but the requirement on the spec for Subject Entity has already been completed in the addDrugTreatmentRecord function



```java
@MessageDriven(
        activationConfig = { @ActivationConfigProperty(
                propertyName = "destination", propertyValue = "Treatment"),
                @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Topic"),
                @ActivationConfigProperty(propertyName = "messageSelector", propertyValue = "treatmentType='Drug'")
        },
        mappedName = "Treatment")
```

Annotations for filtering treatments

The demo video tries to add two treatments, one drug treatments and one non-drug treatment, and according to the spec, both will appear on the billing page but only the drug treatment will appear on the research page.