| Tshwane University of Technology<br>*We empower people*<br><br>____________________<br><br>Signature<br><br>**I hereby subject myself to the examination rules and regulations of Tshwane University of Technology** | **Faculty of Information And Communication Technology (ICT)**<br>**Department of Computer Science**<br>**Subject: Distributed Programming**<br>DSD117V<br><br>**Tutorial 3 MEMO** | |
|---|---|---|
| | **Date 8 October 2025**<br>**Number of Pages: 14** | 1st Examiner:<br>Dr. ML. Gadebe |
| | **Total Marks 100** | |
| **Group Number:** | **Student Number:** | |

Bopha traders requested you to develop an enterprise application that will allow their customers to login and to view and buy available products by product type online. The application uses point to point mode to validate customers to logons. Also, the application will inform customers about recent bought products according to product type. Client's application must subscribe to recent bought product topic.

Note:

Study the system architecture in the next page and answer all questions.

**Question 1** **(28)**

1. Create a persistence.xml that will interact with the bophaDbase which is mapped with entity managers. //1/

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence                version="1.0"                xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="SemesterTestProject-ejbPU" transaction-type="JTA"> //1 mark
    <jta-data-source>jdbc/sample</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties>
  </persistence-unit>
</persistence>
```

2. Create two entities (Person and Customer) map their existing relationship and link them to tblPerson and tblCustomer respectively. //6/

```
@Entity
@Table(name="tblPerson")//1 mark
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS) //2 mark
public class Person implements Serializable
{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO) //1 mark
   private int id;
   private String name;
   private String surname;
   private String address;
>>>>>>
}
```

```
Entity
@Table(name="tblCustomer")//1 mark
public class Customer extends Person//1 mark
{
   private int cusomerNo;
   private String userName;
   private String password;
   private String email;
   private double creditBalance;
……………….
}
```

3. Create a local session bean called CustomerBean that will store customer object and to validate customer login using email address and password and return an object of type Customer if the logons are valid otherwise return null. //5/

```
package za.ac.tut.person;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;


public class CustomerBean implements CustomerBeanService
{
    @PersistenceContext(unitName="SemesterTestProject-ejbPU")//1 mark
    private EntityManager entityManager;

    @Override
    public void storeCustomer(Customer customer) {
      entityManager.persist(customer); //1 mark
    }

    @Override
    public Customer validateLogon(String email, String password)
    {
      String sql = "select customer from Customer customer where customer.email like '"+email+"' and customer.password '"+password+"'";//1
mark
      Customer cust = null;
      Query query = entityManager.createQuery(sql); //1 mark
      try
      {
        cust = (Customer) query.getSingleResult();//1 mark
      }
      catch(Exception er)
      {
         cust = null;
      }
      return cust;
    }
}
```

4. Create another entity named Product //2/

```
@Entity//1 mark
public class Product implements Serializable//1 mark
{
```

```
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private int productID;
   private String name;
   private String productType;
   private int qty;
   private double price
5.
```

6. Create a local stateless session bean called ProductBean with storeProduct(), updateProduct(),getProduct() and getAllProduct() methods as shown in system architecture. The getAllProducts() retrieves all products by product type. //14/

```
@Stateless
public class ProductBean implements ProductBeanLocal//1 mark
{
   @PersistenceContext(unitName="SemesterTestProject-ejbPU")

   private EntityManager entityManager;
   @Override
   public List<Product> getAllProducts()
   {
     String sql = "select product from Product product";//1 mark

     List<Product> products;
     Query query = entityManager.createQuery(sql); //1 mark

     try
     {
       products = (List<Product>) query.getResultList();//1 mark

     }
     catch(Exception er)
```

```
@Local
public interface ProductBeanLocal //1 mark
{
   public List<Product> getAllProducts();//1 mark
   public void storeProduct(Product product); //1 mark
   public Product getProduct(int productId); //1 mark
   public void updateProduct(Product product); //1 mark

}
```

```
      {
        products = null;
      }
      return products;
    }

    @Override
    public void storeProduct(Product product) //1 mark
    {
      entityManager.persist(product); //1 mark

    }

    @Override
    public Product getProduct(int productId)
    {
      return entityManager.find(Product.class, productId); //1 mark

    }

    @Override
    public void updateProduct(Product product)
    {
      Product productGet = getProduct(product.getProductID());//1 mark

      if (productGet != null)
      {
        entityManager.merge(product); //1 mark
      }
    }

  }
```

## Question 2 **(32)**

1. Create a message driven bean called LogonMDB reads text messaged from queue named “jms/logonQueue”. The text contains email address and password separated by a # key. Then, the message driven bean will pass the logons to the CustomerBean to validate the logons. If the Customer is found the LogonMDB will publish an object of type Customer otherwise it will publish “customer not found” message to a queue named “jms/logonQueue” as a confirmation.
//12/

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "jms/logonQueues"),
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")
})
public class LogonMDB implements MessageListener {//1 mark
    @Resource(mappedName="jms/logonQueueFactory")
    ConnectionFactory connectFactory;
    @EJB
    CustomerBeanService customerBean; //1 mark
    public LogonMDB() {
    }

    @Override
    public void onMessage(Message message)
    {
        if (message instanceof TextMessage)
        {
            Connection connect = null;
            try {
                TextMessage txtMsg = (TextMessage) message; //1 mark
                String[] logons = txtMsg.getText().split("#");//1 mark
                //Validate logons
                Customer cust = customerBean.validateLogon(logons[0], logons[1]); //1 mark
                //Send the feedback
                 connect = connectFactory.createConnection();//1 mark
                 //Create session
                 Session session = connect.createSession(false, Session.AUTO_ACKNOWLEDGE); //1 mark
                 //Publisher
                 Queue queue = (Queue) message.getJMSDestination();//1 mark
                 MessageProducer publish = session.createProducer(queue);
                 //determine

                 if (cust != null)
                 {
                   ObjectMessage objMsg = session.createObjectMessage();
                   objMsg.setObject(cust); //1 mark
                   publish.send(objMsg); //1 mark
                 }
                 else
                 {
                     txtMsg.setText("customer not found");//1 mark
                     publish.send(txtMsg); //1 mark
                 }

            } catch (JMSException ex) {
                Logger.getLogger(LogonMDB.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        }

    }
```

2. Create another message driven bean called RecentBoughtProductTopic that subscribes to a topic named "jms/productTopic". The message driven bean will read any changes made on the product topic and display a list of recent bought products on a console screen. //10/

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "clientId", propertyValue = "jms/productTopic"),
    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "jms/productTopic"),
    @ActivationConfigProperty(propertyName = "subscriptionDurability", propertyValue = "Durable"),
    @ActivationConfigProperty(propertyName = "subscriptionName", propertyValue = "jms/productTopic"),
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Topic")
})
public class RecentBoughtProductTopic implements MessageListener {//1 mark

    public RecentBoughtProductTopic() {
    }

    @Override
    public void onMessage(Message message) // 2 marks
    {
        if(message instanceof ObjectMessage) //1 mark
        {
            try {
                ObjectMessage objMsg = (ObjectMessage) message; //1 mark
                List<Product> list = (List<Product>) objMsg.getObject();//1 mark
                System.out.println("Product recently bought");
                for(Product product : list) //2 mark
                {
                    System.out.println(product.toString());//2 mark
                }
            } catch (JMSException ex) {
                Logger.getLogger(RecentBoughtProductTopic.class.getName()).log(Level.SEVERE, null, ex);
            }

        }
    }

}
```

3. Create a stateful session bean named ShoppingCartBean that will initialise the shopping cart data member, allow a user to add and remove products to/from a shopping cart. Also the stateful session bean will return all products added to a shopping cart. //10/

```
@Stateful
public class ShoppingCartBean implements ShoppingCartBeanService {//1 mark

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")
    private List<Product> products;

    @Override
    @PostConstruct
    public void initialliseShopCart() {
        products = new ArrayList<Product>();//1 mark
    }
```

```
@Remote//1 mark
public interface ShoppingCartBeanService//1 mark
{

public void initialliseShopCart();//1 mark
public void addItem(Product product);
public void removeItem(int productId);
public List<Product> getAllItem();
```

```
        @Override
        public void addItem(Product product) {
          products.add(product); //1 mark
        }

        @Override
        public void removeItem(int productId)
        {
           for(Product product: products) //1 mark
           {
              if (product.getProductID() == productId) //1 mark
              {
                 products.remove(product); //1 mark
                 break;
              }
           }
        }

        @Override
        public List<Product> getAllItem()
        {
           return products; //1 mark
        }
    }
```

```
}
```

## Question 3 (20)

1. Create a controller class called LogonServlet that override the doPost method to do the following:
   - The doPost method will accept the email address and password to validate a customer user. The method will publish the combined logons “email#password” to a queue named “jms/logonQueue”. Then it will wait for confirmation. The

servlet will act as publisher and consumer of the message. If the confirmation is of type object display Customer's name, surname, address, email and credit balance otherwise if the confirmation is of type text display the message. Use the same LogonServlet to display the messages. //20/

```
@WebServlet(urlPatterns = {"/LoginServlet"})
public class LoginServlet extends HttpServlet {
@Resource(mappedName="jms/logonQueueFactory")
ConnectionFactory connectFactory;
@Resource(mappedName="jms/logonQueue")
Queue queue;


    @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
  {
    String email, password, logons;
    email = request.getParameter("email");//1 mark
    password = request.getParameter("password");//1 mark
    logons = email+"#"+password; //1 mark

    //Connection
    Connection connection = null;
    Connection connection2 = null;
  try {
     connection = connectFactory.createConnection();//1 mark
     Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); //1 mark
```

```
TextMessage txtMsg = session.createTextMessage();//1 mark
txtMsg.setText(logons); //1 mark
txtMsg.setJMSReplyTo(queue); //1 mark
MessageProducer publish = session.createProducer(queue); //1 mark
publish.send(txtMsg); //1 mark
//Consume the returned message
connection2 = connectFactory.createConnection();//1 mark
Session session2 = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); //1 mark
MessageConsumer consumer = session2.createConsumer(queue); //1 mark
connection2.start();//1 mark
Message message = consumer.receive(0); //1 mark
 try (PrintWriter out = response.getWriter()) {
   /* TODO output your page here. You may use following sample code. */
   out.println("<!DOCTYPE html>");
   out.println("<html>");
   out.println("<head>");
   out.println("<title>Servlet NewServlet</title>");
   out.println("</head>");
   out.println("<body>");


       if (message instanceof TextMessage) //1 mark
       {
           TextMessage txtMsgOut = (TextMessage) message;
           out.println("<h1> " + txtMsgOut.getText() + "</h1>");//1 mark

       }
```

```
                else if (message instanceof ObjectMessage) //1 mark
                {
                    ObjectMessage objMsg = (ObjectMessage) message;
                    Customer cust = (Customer) objMsg.getObject();
                    out.println("<h1> Name" + cust.getName() + "</h1>");//1 mark
                    out.println("<h1> Surname" + cust.getSurname() + "</h1>");//1 mark
                    out.println("<h1> Credit balance R " + cust.getCreditBalance() + "</h1>");
                }

            out.println("</body>");
            out.println("</html>");

            }

        } catch (JMSException ex) {
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
        }

        }



    }
```