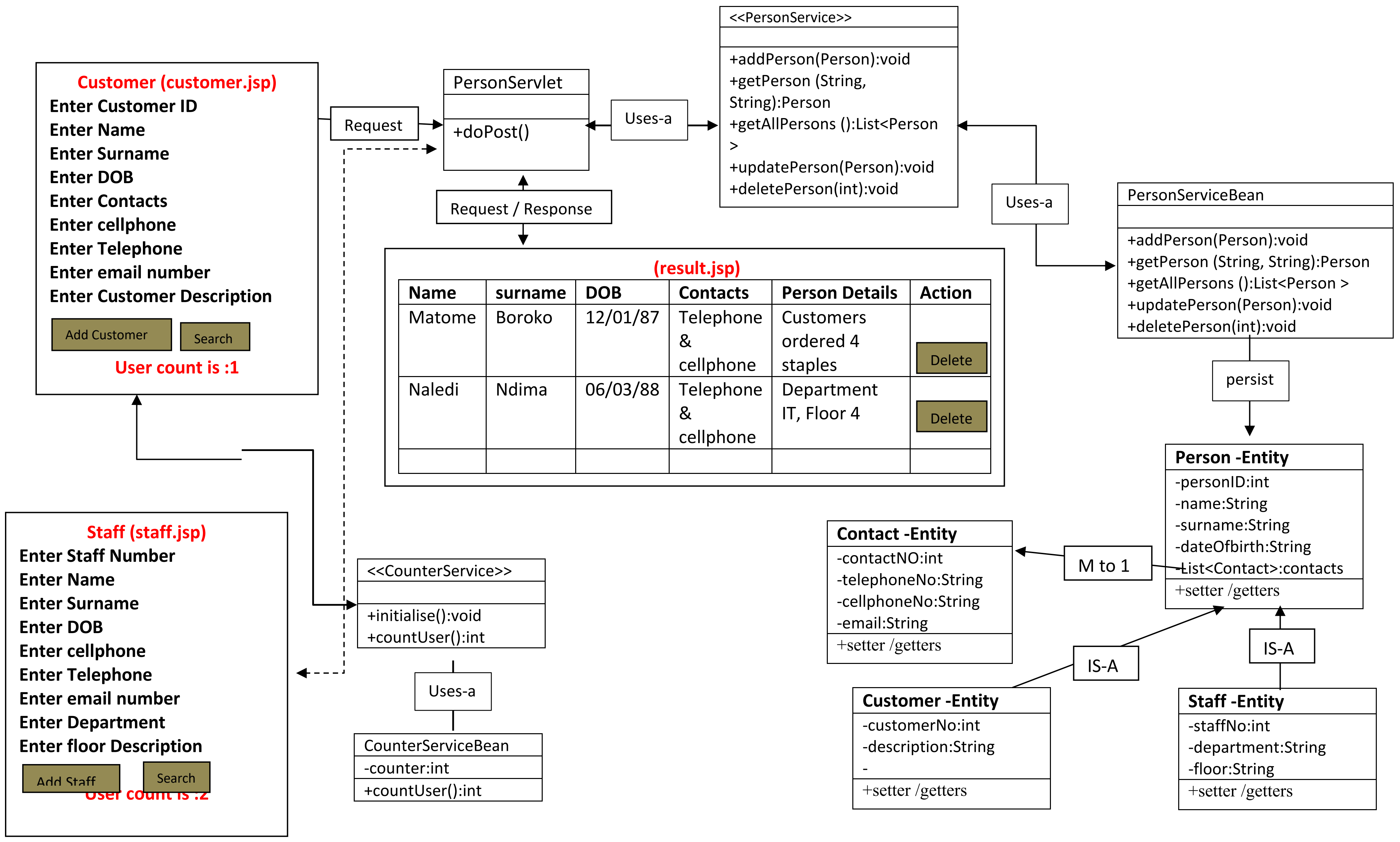
Tshwane University of Technology We empower people **I hereby subject myself to the examination rules and regulations of Tshwane University of Technology**	**Faculty of Information And Communication Technology (ICT)** **Department of Computer Science** **Subject: Distributed Programming** **DSD117V**	
	Tutorial 2 memo **August 2025** **Number of Pages: 2**	1st Examiner: Dr. M. L. Gadebe
	Duration: 3 hours **Total Mark: 70**	
Group Number:	**Student Number:**	

Create an enterprise that allows the Noko wholesalers to add, update, delete, search, retrieve customer and staff using name and surname. See the diagram in the next page.



Name	surname	DOB	Contacts	Person Details	Action
Matome	Boroko	12/01/87	Telephone & cellphone	Customers ordered 4 staples	Delete
Naledi	Ndima	06/03/88	Telephone & cellphone	Department IT, Floor 4	Delete

Question 1 //39/

1. Create a database called NokoDbase using Derby database management service and create the JDBC Pool and resource using glassfish.
2. Crete a persistence.xml to persist the entities and to configure the database connection. [2]

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="abcDB" transaction-type="JTA">/1 Mark
    <jta-data-source>jdbc/abcConnection</jta-data-source>/1 Mark
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties>
  </persistence-unit>
</persistence>
```

3. Create an entity name Person consisting of Contact also create its sub classes Customer and Staff, use the joined relationship strategy. Also map entities to tblPerson, tblContact, tblCustomer and tblStaff respectively. [10]

```
@Entity/1 Mark
@Inheritance(strategy=InheritanceType.JOINED) /1 Mark
 @DiscriminatorColumn(name="subType")
@Table(name="tblPerson")/1 Mark
public class Person
{
 @Id
 @GeneratedValue(strategy=GenerationType.AUTO) /1 Mark
 private int ID;
 @Column(name="first_name")
 private String name;
```

```
@Table(name="tblCustomer")/1 Mark
@DiscriminatorValue("customer")
@Entity
public class Customer extends Person/1 Mark
{
    private int customerNo;
    private string description;
}
```

```
             private String surname;
            private String DOB;
            @OneToMany( cascade=CascadeType.ALL) /1 Mark
            private List<Contact> contacts; /1 Mark
}
```

```
            @Table(name="tblStaff")/1 Mark
            @DiscriminatorValue("staff")
            @Entity
            public class Staff extends Person
            {
              private int staffNo;
              private String department;
              private String floor;
            }
```

```
            @Entity
            @Table(name="tblContact")/1 Mark
            public class Contact
            {
               @Id
               @GeneratedValue
                   private int contactNO;
                   private String telephoneNo;
                   private String cellphoneNo;
                   private String email;
```

4. Create a local stateless session bean called PersonServiceBean and mapped it to the persistence.xml to do the following: [10]

```
            @Stateless
            public class PersonServiceBean implements PersonService/1 Mark
            {
               @PersistenceContext(unitName="abcDB")/1 Mark
               private EntityManager entity; /1 Mark

               @Override
               public void store(Person person) {/1 Mark
                   entity.persist(person); /1 Mark
               }

               @Override
               public Person getPerson(int id) {
```

```
            @Local/1 Mark
            public interface PersonService/1 Mark
            {
               public void store(Person person); /1 Mark
               public void delete(int id); /1 Mark
               public Person getByName(String name, String
            surname); /1 Mark
               public List<Person> getAllPersons();/1 Mark
              public void updatePersons(Person person); /1 Mark
 }
```

```
        return entity.find(Person.class, id);
      }

      @Override
      public void delete(int id) {/1 Mark
        entity.remove(getPerson(id)); /1 Mark
      }

      @Override
      public Person getByName(String ename, String esurname) {
        String sql = "select person from Person person where person.name Like
   :name AND person.surname Like :surname";/1 Mark
        Query query = entity.createQuery(sql); /1 Mark
        query.setParameter("name", ename); /1 Mark
        query.setParameter("surname", esurname); /1 Mark
        Person person = (Person) query.getSingleResult();
        return person;
      }

      @Override
      public void updatePerson(Person objPerson)
      {
         Person objPerson = getPerson(objPerson.getId());/1 Mark
         if (objPerson != null)
         {
             entity.merge(objPerson); /1 Mark
         }
      }

      public List<Person> getAllPersons() {
        String sql = "Select person from Person person";/1 Mark
        Query query = entity.createQuery(sql); /1 Mark
```

```
          List<Person> persons = (List<Person>)query. getResultList();/1 Mark
           return persons; /1 Mark
         }

}
```

a. addPerson the method receives a composite value object Book and persist it to the database [2]
b. getPerson the method receives the name and surname of a person, then retrieve and return Person. [4]
c. getAllPersons the method retrieves all Person and return a List of Person [4]
d. updatePerson the method receives the Person and update the instance of the Person to the database. [2]
e. deletePerson the method receives the person id to the delete a Person instance from the database [2]

5. Create a remote singleton session bean called CounterServiceBean that will keep the count of all clients calls [5]

```
@Remote/1 Mark
public interface SingletonInterface
{
   public int couter();/1 Mark
}
```

```
@Singleton/1 Mark
@Stateful
public class CounterServiceBean implements SingletonInterface{/1 Mark

   private int count = 0;
   @Override
   public int couter()
   {
      return count++;/1 Mark
   }

}
```

Question 2 //31/

Create a controller class called PersonServlet that overrides the doPost method to do the following:

```
protected void doPost Request(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException
```

```
{
      response.setContentType("text/html;charset=UTF-8");
      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 PersonServlet</title>");
         out.println("</head>");   out.println("<body>");

         if (request.getParameter("select").equals("add customer"))/1 Mark
         {
             Customer objCust = new Customer();
            objCust.setName(request.getParameter("name"));
            objCust.setSurname(request.getParameter("surname"));
            objCust.setDoB(request.getParameter("dob"));
            objCust.setCustomerNo(Integer.parseInt(request.getParameter("customerNo")));/1 Mark
           objCust.setDescription(request.getParameter("description"));
           Contact contact = new Contact();
           contact.setTelephoneNo(request.getParameter("telephoneNo"));
           contact.setCellphoneNo(request.getParameter("cellphoneNo"));
            contact.setCellphoneNo(request.getParameter("email"));
            List<Contact> contacts = new ArrayList();
            contacts.add(contact) /1 Mark
            objCust.setContact(contacts);
            serviceBean.add(objCust); /1 Mark
            out.println("<p>Customer Record  Added </p>");
         }
         else  if (request.getParameter("select").equals("add staff"))/1 Mark
         {
            Staff objStaff = new Staff();
            objStaff.setName(request.getParameter("name"));
            objStaff.setSurname(request.getParameter("surname"));
            objStaff.setDoB(request.getParameter("dob"));
            objStaff.setStaffNo(Integer.parseInt(request.getParameter("staffNo")));/1 Mark
            objStaff.setDepartment(request.getParameter("department"));
            objStaff.setFloor(request.getParameter("floor"));
            Contact contact = new Contact();
            contact.setTelephoneNo(request.getParameter("telephoneNo"));
            contact.setCellphoneNo(request.getParameter("cellphoneNo"));
            contact.setCellphoneNo(request.getParameter("email"));
            List<Contact> contacts = new ArrayList();
            contacts.add(contact)
            objStaff.setContact(contacts); /1 Mark
            serviceBean.add(objStaff);
            out.println("<p>Staff Record  Added </p>");/1 Mark
        }
        else if (request.getParameter("select").equals("search"))/1 Mark
        {
             Person person = serviceBean.getPerson(Integer.parseInt(request.getParameter("personID")));/1 Mark
             Request.setAttribute("person",person); /1 Mark
             request.getRequestDispatcher("result.jsp()").forward(request, response); /1 Mark
        }
        else if (request.getParameter("select").equals("delete"))/1 Mark
        {
            serviceBean.deletePerson(Integer.parseInt(request.getParameter("personID")));/1 Mark
            out.println("<p>Person Record deleted </p>");    /1 Mark
        }


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

Results.jsp

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
                         <table>
                           <tr>
                                  <td>Name</td><td>Surname</td><td>DOB</td><td>Contacts</td><td>Person details</td><td>Action</td>
                           </tr>
                         <form action="PersonServlet" method="POST">
                           <%
                                Person person = (Person) request.getAttribute("person"); /1 Mark
                                if (person != null)
                                 {
                                   <input type="hidden" name="personID" value="<%=person.getPersonID()%>" /> /1 Mark
                                    <tr>
                                            <td><%=person.getName()%></td>/1 Mark
                                            <td><%=person.getSurname()%></td>
                                            <td><%=person.getDob()%></td>/1 Mark
                                            <td><%=person.getContact().getCellphoneNo()%>
                                               <%=person.getContact().getTelephoneNo()%>/1 Mark
                                            </td>
                                             <%
                                               if (person instanceof Customer ) /1 Mark
                                                 {
                                                        Customer customer = (Customer) person; /1 Mark
                                             %>
                                               <td>
                                                        <%=customer.getDescription()%> /1 Mark
                                                     </td>
```

```
                                                <td>
                                                    <input type="submit" name="select" value="delete" />
                                                </td>
                                            <%
                                                }
                                                else
                                                {
                                                  Staff staff = (Staff) person; /1 Mark
                                                %>
                                                 <td>
                                                    <%=staff.getDepartment()%><%=staff.getFloor()%>/1 Mark
                                                 </td>
                                                 <td>
                                                     <input type="submit" name="select" value="delete" />
                                                 </td>
                                                <%
                                                }
                                            %>
                                    </tr>
                              <%} %>

        </form>
                    </table>
    </body>
</html>
```

Customer.jsp

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
```

```
   <body>

        <form action="PersonServlet" method="POST">

        </form>
                        <%
          InitialContext context = new InitialContext();/2 Mark
          SingletonInterface singleService = (SingletonInterface) context.lookup("za.ac.tut.session.SingletonInterface");/2 Mark


                        %>
                        <p> User count is <%=singleService.couter() + "</p>");/2 Mark


   </body>
</html>
```

1. The doPost method will accept a request to add a customer or staff to the database and display the message "Person Record is inserted" [8]
2. The doPost method will accept a request to search for a person using name and surname and display the person records using result.jsp [13]
3. The doPost method will accept a request to delete the instance of Person from the database table using person id and display the message "Person is Deleted" [4]
4. Create an customer,jsp and staff.jsp pages as shown in system architecture and keep track of all active clients on the system [6]

Note : Submit the following on EC: Persistence.xml, Person.java, Contact.java, Customer.java, Staff.java, PersonService.java, PersonServiceBean.java,CounterService.java, CounterServiceBean.java, customer.jsp, staff.jsp, PersonServlet.java, result.jsp