ASSIGNMENT-5

HIBERNATE

Question 1:

Objective: Develop a Java application using Hibernate for performing CRUD (Create, Read,

Update, Delete) operations on a Customer entity.

Steps:

- **Set up your development environment by installing JDK, Hibernate, and a database**

management system.

- **Create a new Java project in your IDE.**
- **Configure Hibernate properties such as database connection details in a hibernate.cfg.xml**

file.

- **Define a Customer entity class with attributes like id, name, email, and phone number.**
- **Implement CRUD operations for the Customer entity using Hibernate APIs.**
- **Write a test class to demonstrate the CRUD operations on the Customer entity**

SOLUTION

Customer.java

```
package com.customer;

import javax.persistence.*;

@Entity
@Table(name = "customers")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "email")
    private String email;

    @Column(name = "phone")
    private String phone;


    public Customer() {}

    public Customer(String name, String email, String phone) {
        this.name = name;
        this.email = email;
        this.phone = phone;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", email="
+ email + ", phone=" + phone + "]";
    }
}
```

CustomerDAO.java

```
package com.customer;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class CustomerDAO {
    private SessionFactory factory;

    public CustomerDAO() {
        Configuration cfg = new
Configuration().configure("hibernate.cfg.xml");
        factory = cfg.buildSessionFactory();
    }

    public void saveOrUpdate(Customer customer) {
        Session session = factory.openSession();
        Transaction tx = null;
        try {
```

```java
            tx = session.beginTransaction();
            session.saveOrUpdate(customer);
            tx.commit();
        } catch (Exception e) {
            if (tx != null) {
                tx.rollback();
            }
            e.printStackTrace();
        } finally {
            session.close();
        }
    }

    public Customer getById(int id) {
        Session session = factory.openSession();
        Customer customer = null;
        try {
            customer = session.get(Customer.class, id);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return customer;
    }

    public void delete(int id) {
        Session session = factory.openSession();
        Transaction tx = null;
        try {
            tx = session.beginTransaction();
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                session.delete(customer);
            }
            tx.commit();
        } catch (Exception e) {
            if (tx != null) {
                tx.rollback();
            }
            e.printStackTrace();
        } finally {
            session.close();
        }
    }
}
```

Main.java

```java
package com.customer;

public class Main {
    public static void main(String[] args) {
        CustomerDAO dao = new CustomerDAO();

        // Create
        Customer customer = new Customer("mp", "mp@gmail.com",
"5534567800");

        dao.saveOrUpdate(customer);
        System.out.println("Customer created: " + customer);

        // Read
        Customer retrievedCustomer = dao.getById(customer.getId());
        System.out.println("Retrieved customer: " +
retrievedCustomer);

        // Update
        retrievedCustomer.setName("surya");
        dao.saveOrUpdate(retrievedCustomer);
        System.out.println("Updated customer: " +
retrievedCustomer);

        // Delete
        dao.delete(retrievedCustomer.getId());
        System.out.println("Customer deleted");

        // Read again (should return null)
        Customer deletedCustomer =
dao.getById(retrievedCustomer.getId());
        System.out.println("Deleted customer: " + deletedCustomer);
    }
}
```

hibernate.cfg.xml

```xml
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
     <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property
name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
```

```
        <!-- <property
name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
-->
        <property
name="connection.url">jdbc:mysql://localhost:3306/testdb</property>
           <property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="show_sql">true</property>
        <mapping class="com.customer.Customer"/>
    </session-factory>

</hibernate-configuration>
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.emp</groupId>
  <artifactId>01-EmployeeHibernateXML</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>

  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java
-->
           <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>8.0.13</version>
           </dependency>

           <!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
           <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-core</artifactId>
                <version>5.3.7.Final</version>
           </dependency>

           <!-- API, java.xml.bind module -->
           <dependency>
                <groupId>jakarta.xml.bind</groupId>
                <artifactId>jakarta.xml.bind-api</artifactId>
                <version>2.3.2</version>
           </dependency>
		<!-- Runtime, com.sun.xml.bind module -->
		<dependency>
			<groupId>org.glassfish.jaxb</groupId>
			<artifactId>jaxb-runtime</artifactId>
			<version>2.3.2</version>
		</dependency>
		<!-- Runtime, com.sun.xml.bind module -->
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.2</version>
</dependency>
	</dependencies>

		<build>
		<sourceDirectory>src/main/java</sourceDirectory>
		<plugins>
			<plugin>
				<artifactId>maven-compiler-plugin</artifactId>
				<version>3.5.1</version>
				<configuration>
					<source>1.8</source>
					<target>1.8</target>
				</configuration>
			</plugin>
		</plugins>
	</build>

</project>
```

Question 2: Develop a Java application using Hibernate for performing CRUD (Create, Read, Update,

Delete) operations on Order and OrderItem entities, demonstrating a basic e-commerce scenario.

Steps:

• Set up your development environment by installing JDK, Hibernate, and a database

management system.

• Create a new Java project in your IDE.

• Configure Hibernate properties such as database connection details in a hibernate.cfg.xml

file.

• Define Order with

```
@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Temporal(TemporalType.TIMESTAMP) private Date orderDate;

@OneToMany(mappedBy = "order", cascade = CascadeType.ALL)

private List<OrderItem> orderItems = new ArrayList<>();

//Getter/ Setter
```

• Define OrderItem entity classes with appropriate attributes and relationships.

```
@Entity

@Table(name = "order_items")

public class OrderItem { @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@ManyToOne @JoinColumn(name = "order_id")

private Order order;
```

private String product;

private int quantity;

//Getter/ Setter

- **Implement CRUD operations for the Order and OrderItem entities using Hibernate APIs.**
- **Write a test class to demonstrate the CRUD operations on Order and OrderItem entities**

Solution

Main.java

```
package com.order;

import java.util.Date;
import java.util.List;

import com.orderDao.OrderDAO;
import com.orderDao.OrderItemDAO;

public class Main {
    public static void main(String[] args) {
        OrderDAO orderDAO = new OrderDAO();
        OrderItemDAO orderItemDAO = new OrderItemDAO();

        // Create Order
        Order order = new Order();
        order.setOrderDate(new Date());
        orderDAO.saveOrUpdate(order);
        System.out.println("Order created: " + order);

        // Create OrderItems
        OrderItem orderItem1 = new OrderItem(0, order, "Product-1", 2);
        orderItemDAO.saveOrUpdate(orderItem1);
        System.out.println("Order item created: " + orderItem1);

        OrderItem orderItem2 = new OrderItem(0, order, "Product-2", 5);
        orderItemDAO.saveOrUpdate(orderItem2);
        System.out.println("Order item created: " + orderItem2);

        // Read Order
        Order retrievedOrder = orderDAO.getById(order.getId());
        System.out.println("Retrieved order: " + retrievedOrder);

        // Read OrderItems
        List<OrderItem> orderItems = orderItemDAO.getByOrderId(order.getId());
        System.out.println("Order items for order " + order.getId() + ": " +
orderItems);
```

```
        // Update OrderItem
        orderItem1.setQuantity(3);
        orderItemDAO.saveOrUpdate(orderItem1);
        System.out.println("Updated order item: " + orderItem1);

        // Delete OrderItem
        orderItemDAO.delete(orderItem2.getId());
        System.out.println("Order item deleted: " + orderItem2);

        // Read OrderItems again
        orderItems = orderItemDAO.getByOrderId(order.getId());
        System.out.println("Order items for order " + order.getId() + " after
deletion: " + orderItems);

        // Delete Order
        orderDAO.delete(order.getId());
        System.out.println("Order deleted: " + order);
    }
}
```

Order.java

```
package com.order;

import javax.persistence.*;
import java.util.*;

@Entity
@Table(name = "orders")
public class Order {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	@Temporal(TemporalType.TIMESTAMP)
	private Date orderDate;
	@OneToMany(mappedBy = "order", cascade = CascadeType.ALL)
	private List<OrderItem> orderItems = new ArrayList<>();


	public Order(int id, Date orderDate, List<OrderItem> orderItems) {
		super();
		this.id = id;
		this.orderDate = orderDate;
		this.orderItems = orderItems;
	}


	public Order() {
		// TODO Auto-generated constructor stub
	}


	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}


	public Date getOrderDate() {
		return orderDate;
	}


	public void setOrderDate(Date orderDate) {
		this.orderDate = orderDate;
	}


	public List<OrderItem> getOrderItems() {
		return orderItems;
	}


	public void setOrderItems(List<OrderItem> orderItems) {
		this.orderItems = orderItems;
	}


	@Override
	public String toString() {
		return "Order [id=" + id + ", orderDate=" + orderDate + ", 
orderItems=" + orderItems + "]";
	}


}
```

OrderItem.java

```
package com.order;

import javax.persistence.*;
	@Entity
	@Table(name = "order_items")
	public class OrderItem {
	 @Id
	 @GeneratedValue(strategy = GenerationType.IDENTITY)
	 private int id;
	 @ManyToOne
	 @JoinColumn(name = "order_id")
	 private Order order;
	 private String product;
	 private int quantity;

	 public OrderItem(int id, Order order, String product, int quantity) {
		super();
		this.id = id;
		this.order = order;
		this.product = product;
		this.quantity = quantity;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public Order getOrder() {
		return order;
	}
	public void setOrder(Order order) {
		this.order = order;
	}
	public String getProduct() {
		return product;
	}
	public void setProduct(String product) {
		this.product = product;
	}
	public int getQuantity() {
		return quantity;
	}
	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}
	@Override
	public String toString() {
		return "OrderItem [id=" + id + ", order=" + order + ", product=" +
product + ", quantity=" + quantity + "]";
	}



}
```

OrderDAO.java

```
package com.orderDao;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import com.order.Order;
public class OrderDAO {
	 private SessionFactory factory;
	 public OrderDAO() {
		 Configuration cfg = new
Configuration().configure("hibernate.cfg.xml");
		 factory = cfg.buildSessionFactory();
	 }

	 public void saveOrUpdate(Order order) {
		 Session session = factory.openSession();
		 Transaction tx = null;
		 try {
```

```java
                        tx = session.beginTransaction();
                        session.saveOrUpdate(order);
                        tx.commit();
                } catch (Exception e) {
                        if (tx != null) {
                                tx.rollback();
                        }
                        e.printStackTrace();
                } finally {
                        session.close();
                }
        }

        public Order getById(int id) {
                Session session = factory.openSession();
                Order order = null;
                try {
                        order = session.get(Order.class, id);
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        session.close();
                }
                return order;
        }

        public void delete(int id) {
                Session session = factory.openSession();
                Transaction tx = null;
                try {
                        tx = session.beginTransaction();
                        Order order = session.get(Order.class, id);
                        if (order != null) {
                                session.delete(order);
                        }
                        tx.commit();
                } catch (Exception e) {
                        if (tx != null) {
                                tx.rollback();
                        }
                        e.printStackTrace();
                } finally {
                session.close();
                }
                }
       }
```

OrderItemDAO.java

```java
package com.orderDao;

import java.util.List;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

import com.order.OrderItem;
```

```java
public class OrderItemDAO {
	private SessionFactory factory;

	public OrderItemDAO() {
		Configuration cfg = new
Configuration().configure("hibernate.cfg.xml");
		factory = cfg.buildSessionFactory();
	}

	public void saveOrUpdate(OrderItem orderItem) {
		Session session = factory.openSession();
		Transaction tx = null;
		try {
			tx = session.beginTransaction();
			session.saveOrUpdate(orderItem);
			tx.commit();
		} catch (Exception e) {
			if (tx != null) {
//Call rollback if transaction is null
				tx.rollback();
			}
			e.printStackTrace();
		} finally {
			session.close();
		}
	}

	public OrderItem getById(int id) {
		Session session = factory.openSession();
		OrderItem orderItem = null;
		try {
			orderItem = session.get(OrderItem.class, id);

		} catch (Exception e) {
			e.printStackTrace();
		} finally {
			session.close();
		}
		return orderItem;
	}

	public void delete(int id) {
		Session session = factory.openSession();
		Transaction tx = null;
		try {
			tx = session.beginTransaction();
			OrderItem orderItem = session.get(OrderItem.class, id);
			if (orderItem != null) {
				session.delete(orderItem);
			}
			tx.commit();
		} catch (Exception e) {
			if (tx != null) {
				tx.rollback();
			}
			e.printStackTrace();
		} finally {
			session.close();
		}
	}

	public List<OrderItem> getByOrderId(int id) {
		// TODO Auto-generated method stub
		return null;
	}
}
```

hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
     <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
        <!-- <property
name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property> -->
        <property
name="connection.url">jdbc:mysql://localhost:3306/testdb</property>
		<property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="show_sql">true</property>
        <mapping class="com.order.Order"/>
		<mapping class="com.order.OrderItem"/>
    </session-factory>

</hibernate-configuration>
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.emp</groupId>
  <artifactId>01-EmployeeHibernateXML</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>

  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>8.0.13</version>
		</dependency>

		<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core
-->
```

```xml
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
			<version>5.3.7.Final</version>
		</dependency>

		<!-- API, java.xml.bind module -->
		<dependency>
			<groupId>jakarta.xml.bind</groupId>
			<artifactId>jakarta.xml.bind-api</artifactId>
			<version>2.3.2</version>
		</dependency>

		<!-- Runtime, com.sun.xml.bind module -->
		<dependency>
			<groupId>org.glassfish.jaxb</groupId>
			<artifactId>jaxb-runtime</artifactId>
			<version>2.3.2</version>
		</dependency>
		<!-- Runtime, com.sun.xml.bind module -->
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.2</version>
</dependency>
	</dependencies>

		<build>
		<sourceDirectory>src/main/java</sourceDirectory>
		<plugins>
			<plugin>
				<artifactId>maven-compiler-plugin</artifactId>
				<version>3.5.1</version>
				<configuration>
					<source>1.8</source>
					<target>1.8</target>
				</configuration>
			</plugin>
		</plugins>
	</build>

</project>
```