Experiment - 10

Student Name: Manjot Singh
UID: 23BCS12549
Branch: BE-CSE
Section/Group: KRG-2B
Semester: 5th
Date of Performance:21/10/25
Subject Name: PBLJ
Subject Code: 23CSH-304

1. Aim:

Develop a Spring-based application integrated with Hibernate to manage transactions. Create a banking system where users can transfer money between accounts, ensuring transaction consistency.

2. Objective:

Understand transaction management in Spring-Hibernate apps and implement atomic operations.

3. Apparatus / Input Used:

Java, Hibernate, MySQL, Eclipse / IntelliJ, hibernate.cfg.xml

4. Procedure:

- Configure MySQL database and add Hibernate dependencies.
- Create hibernate.cfg.xml with DB credentials.
- Create Student.java with @Entity, @Id, @GeneratedValue annotations.
- Create HibernateUtil class for SessionFactory.
- Implement CRUD using session.save(), session.get(), session.update(),session.delete().
- Test using a main class

5. Code

Account.java:

```
package com.bank.entity;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Account {
   @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String holderName;
    private double balance;

    public Account() {}
    public Account(String holderName, double balance) {
        this.holderName = holderName;
        this.balance = balance;
    }

    public int getId() { return id; }
    public String getHolderName() { return holderName; }
    public void setHolderName(String holderName) { this.holderName = holderName; }
    public double getBalance() { return balance; }
    public void setBalance(double balance) { this.balance = balance; }
}
```

AccountService.java

```
package com.bank.service;

import com.bank.entity.Account;
import jakarta.transaction.Transactional;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class AccountService {

    @Autowired
    private SessionFactory sessionFactory;

    @Transactional
    public void transferMoney(int fromId, int toId, double amount) {
        Session session = sessionFactory.getCurrentSession();

        Account from = session.get(Account.class, fromId);
        Account to = session.get(Account.class, toId);

        if (from == null || to == null) throw new RuntimeException("Invalid account");
```

```
        if (from.getBalance() < amount) throw new RuntimeException("Insufficient balance");

        from.setBalance(from.getBalance() - amount);
        to.setBalance(to.getBalance() + amount);

        session.update(from);
        session.update(to);
    }
}
```

Main.java

```
package com.bank;

import com.bank.entity.Account;
import com.bank.service.AccountService;
import org.hibernate.SessionFactory;
import org.springframework.context.annotation.*;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

import javax.sql.DataSource;
import java.util.Properties;

@Configuration
@ComponentScan(basePackages = "com.bank")
@EnableTransactionManagement
public class Main {

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
        ds.setUrl("jdbc:mysql://localhost:3306/bankdb");
        ds.setUsername("root");
        ds.setPassword("password");
        return ds;
    }
```

```
    @Bean
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean factory = new LocalSessionFactoryBean();
        factory.setDataSource(dataSource());
        factory.setPackagesToScan("com.bank.entity");
        Properties props = new Properties();
        props.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
        props.put("hibernate.hbm2ddl.auto", "update");
        props.put("hibernate.show_sql", "true");
        factory.setHibernateProperties(props);
        return factory;
    }

    @Bean
    public org.springframework.orm.hibernate5.HibernateTransactionManager transactionManager(SessionFactory
sf) {
        return new org.springframework.orm.hibernate5.HibernateTransactionManager(sf);
    }

    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(MainApp.class);
        SessionFactory sf = context.getBean(SessionFactory.class);
        var service = context.getBean(AccountService.class);

        var session = sf.openSession();
        session.beginTransaction();
        session.save(new Account("Alice", 5000));
        session.save(new Account("Bob", 3000));
        session.getTransaction().commit();
        session.close();

        try {
            service.transferMoney(1, 2, 1500);
            System.out.println("Transfer successful");
        } catch (Exception e) {
            System.out.println("Transfer failed: " + e.getMessage());
        }
```

```
        context.close();
    }
}
```

Sample Output:

```
Transaction Successful: ₹500 transferred from Acc101 to Acc102
Transaction Failed: Insufficient balance — transaction rolled back
```