Experiment-9

StudentName: Vivek **UID:22BCS13384**
Branch:BE-CSE **Section/Group:IOT_639-A**
Semester:6th **DateofPerformance:04/04/2025**
Subject Name: Project-Based Learning in Java withLab **Subject Code: 22CSH-359**

9.1.1.Aim: TodemonstratedependencyinjectionusingSpringFramework withJava-based configuration.

Objective:

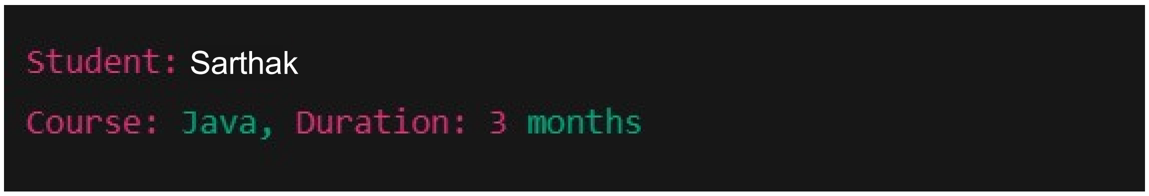
DefineCourseandStudentclasses.
UseConfigurationandBeanannotationstoinjectdependencies.
Load Spring context and print student details.

Code:

```
//Course.java
publicclassCourse{
   privateStringcourseName;
   private String duration;

   publicCourse(StringcourseName,Stringduration){
      this.courseName = courseName;
      this.duration =duration;
   }

   publicStringgetCourseName(){returncourseName;}
   public String getDuration() { return duration; }

   @Override
   publicStringtoString(){
      return"Course:"+courseName+",Duration:"+duration;
   }
}

//Student.java
public class Student {
   privateStringname;
```

```
    privateCoursecourse;
    publicStudent(Stringname,Coursecourse){
        this.name = name;
        this.course=course;
    }

    public void showDetails() {
        System.out.println("Student:"+name);
        System.out.println(course);
    }
}//AppConfig.java
importorg.springframework.context.annotation.*;

@Configuration
publicclassAppConfig{
    @Bean
    publicCoursecourse(){
        returnnewCourse("Java","3months");
    }

    @Bean
    publicStudentstudent(){
        returnnewStudent("Aman",course());
    }
}//MainApp.java
importorg.springframework.context.ApplicationContext;
importorg.springframework.context.annotation.AnnotationConfigApplicationContext;

publicclassMainApp{
    publicstaticvoidmain(String[]args){
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        Student student = context.getBean(Student.class);
        student.showDetails();
    }
}
```

Output:



```
Student: Sarthak
Course: Java, Duration: 3 months
```

Aim: ToperformCRUDoperationsonaStudententityusingHibernate ORM with MySQL.

Objective:DefineCourseandStudentclasses.

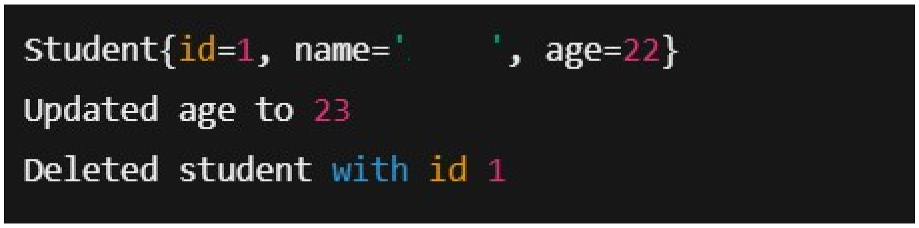
UseConfigurationandBeanannotationstoinjectdependencies.

Load Spring context and print student details.

Code:

```
    <hibernate-configuration>
        <session-factory>
            <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
            <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/testdb</property>
            <propertyname="hibernate.connection.username">root</property>
            <propertyname="hibernate.connection.password">password</property>
            <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
            <propertyname="hibernate.hbm2ddl.auto">update</property>
            <mappingclass="Student"/>
        </session-factory>
    </hibernate-configuration>

importjavax.persistence.*;

Entity
publicclassStudent{ Id
    GeneratedValue(strategy=GenerationType.IDENTITY) private
    int id;
    privateStringname;
    private int age;

    publicStudent(){}
    publicStudent(Stringname,intage){
        this.name = name;
        this.age=age;
    }
    //Getters,setters,toString
}
import org.hibernate.SessionFactory;
importorg.hibernate.cfg.Configuration;

publicclassHibernateUtil{
    privatestaticfinalSessionFactory sessionFactory;

    static{
        sessionFactory=newConfiguration().configure().buildSessionFactory();
    }

    publicstatic SessionFactorygetSessionFactory() { return
        sessionFactory;
    }
}

import org.hibernate.*;

publicclassMainCRUD{
    publicstaticvoidmain(String[] args){
        Sessionsession=HibernateUtil.getSessionFactory().openSession();

        //Create
        Transactiontx =session.beginTransaction(); Student
        s1 = new Student("Aman", 22); session.save(s1);
        tx.commit();

        //Read
        Studentstudent =session.get(Student.class,1); System.out.println(student);

        //Update
        tx=session.beginTransaction();
        student.setAge(23);
        session.update(student);
        tx.commit();

        //Delete
        tx=session.beginTransaction();
        session.delete(student);
        tx.commit();

        session.close();
    }
}
```

Output:



```
Student{id=1, name='     ', age=22}
Updated age to 23
Deleted student with id 1
```

9.3.1Aim: ToimplementabankingsystemusingSpringandHibernatethatensures transaction consistency during fund transfers.

Objective:

IntegrateSpring+Hibernate.
Handletransactionsatomically(rollbackonfailure).
Demonstrate success and failure cases.

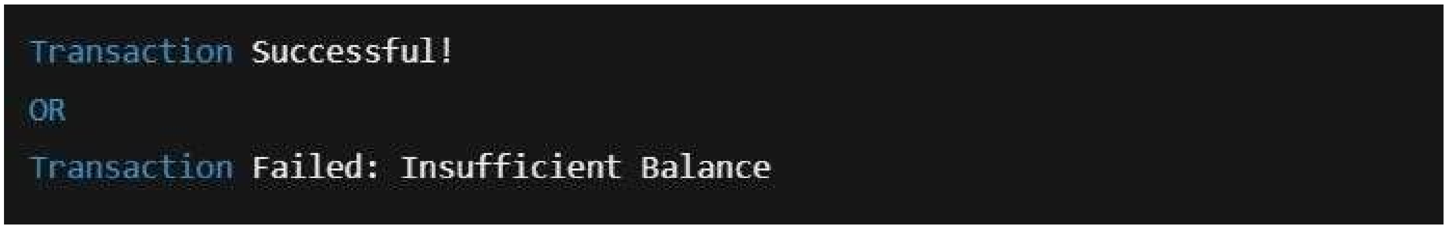
Code:

```
importjavax.persistence.*;

Entity
publicclassAccount{
   @Id
   private int accountId;
   privateStringholderName;
   private double balance;

   //Constructors,getters,setters
}

importjavax.persistence.*;
import java.util.Date;

@Entity
publicclassBankTransaction{ @Id
   @GeneratedValue(strategy=GenerationType.IDENTITY)
   private int txnId;
   private int fromAcc;
   private int toAcc;
   privatedoubleamount;
   privateDatetxnDate=newDate();

   //Constructors,getters,setters
}
importorg.hibernate.*;
importorg.springframework.transaction.annotation.Transactional;
```

```
publicclassBankService{
   privateSessionFactorysessionFactory;

   publicBankService(SessionFactorysessionFactory){
      this.sessionFactory = sessionFactory;
   }

   @Transactional
   publicvoidtransferMoney(intfromId,inttoId,doubleamount){ Session
      session = sessionFactory.getCurrentSession();

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

      if(from.getBalance()<amount){
         thrownewRuntimeException("InsufficientBalance");
      }

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

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

      BankTransactiontxn=newBankTransaction(fromId,toId,amount);
      session.save(txn);
   }
}
@Configuration
@EnableTransactionManagement
public class AppConfig {
   @Bean
   publicDataSourcedataSource(){
      DriverManagerDataSource ds = new DriverManagerDataSource();
      ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
      ds.setUrl("jdbc:mysql://localhost:3306/testdb");
      ds.setUsername("root");
      ds.setPassword("password");
```

```
        returnds;
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean lsf = new LocalSessionFactoryBean();
        lsf.setDataSource(dataSource());
        lsf.setPackagesToScan("your.package");
        Propertiesprops=newProperties();
        props.put("hibernate.dialect","org.hibernate.dialect.MySQL8Dialect");
        props.put("hibernate.hbm2ddl.auto", "update");
        lsf.setHibernateProperties(props);
        returnlsf;
    }

    @Bean
    publicHibernateTransactionManagertransactionManager(SessionFactorysf){
        return new HibernateTransactionManager(sf);
    }

    @Bean
    publicBankServicebankService(SessionFactorysf){
        return new BankService(sf);
    }
}

publicclassMainApp{
    public static void main(String[] args) {
        AnnotationConfigApplicationContextctx =new
AnnotationConfigApplicationContext(AppConfig.class);
        BankServiceservice=ctx.getBean(BankService.class);

        try{
            service.transferMoney(101, 102, 500);
            System.out.println("TransactionSuccessful!");
        }catch(Exceptione){
            System.out.println("TransactionFailed:"+e.getMessage());
        }
        ctx.close();
    }
}
```

OUTPUT



```
Transaction Successful!
OR
Transaction Failed: Insufficient Balance
```