NAME	RAMENDRA SHUKLA
UID	23BCS12146
CLASS	622-B

Experiment -2.3

Part a: Dependency Injection in Spring Using Java-Based Configuration

Objective:
To create a simple Spring application that demonstrates Dependency Injection (DI) using Java-based configuration instead of XML. CODE

```
import org.springframework.context.annotation.*;


interface MessageService {
    String getMessage();
}


class EmailService implements MessageService {
```

```
    public String getMessage() {
return "Hello from Email Service";
    }
}

class MessagePrinter {     private MessageService
service;     public MessagePrinter(MessageService
service) {         this.service = service;
    }
    public void printMessage() {
        System.out.println(service.getMessage());
    }
}

@Configuration class AppConfig {
@Bean     public MessageService
messageService() {         return new
EmailService();
    }
```

```
    @Bean     public MessagePrinter
messagePrinter() {         return new
MessagePrinter(messageService());
    }
}


public class MainApp {     public static
void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        MessagePrinter printer = context.getBean(MessagePrinter.class);
printer.printMessage();         context.close();
    }
}
```

OUTPUT

Hello from Email Service

Part b: Hibernate Application for Student CRUD Operations Objective: To develop a Hibernate application that performs CRUD operations on a Student entity stored in a MySQL database using Hibernate ORM.

CODE

```
import org.hibernate.*; import
org.hibernate.cfg.*; import
javax.persistence.*; import
java.util.*;

@Entity
@Table(name = "student") class
Student {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;
private String name;
private String course;
private double marks;

   public Student() {}
   public Student(String name, String course, double marks) {
this.name = name;        this.course = course;        this.marks
= marks;
```

```
    }     public int getId() { return id; }     public String
getName() { return name; }     public String getCourse() {
return course; }     public double getMarks() { return marks; }
public void setName(String name) { this.name = name; }
public void setCourse(String course) { this.course = course; }
public void setMarks(double marks) { this.marks = marks; } }


public class StudentCRUD {     private
static SessionFactory factory;     public
static void main(String[] args) {
        factory = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Student.class).buildSes
sionFactory();
        StudentCRUD app = new StudentCRUD();
int id = app.addStudent("John", "Java", 85.5);
app.listStudents();         app.updateStudent(id,
90.0);         app.getStudent(id);
app.deleteStudent(id);         factory.close();
    }


    public int addStudent(String name, String course, double marks) {
        Session s = factory.openSession();
```

```
        Transaction tx = s.beginTransaction();
        Student st = new Student(name, course, marks);
int id = (int) s.save(st);          tx.commit();
        s.close();
return id;
    }

    public void listStudents() {
        Session s = factory.openSession();
        List<Student> list = s.createQuery("from Student", Student.class).list();
for (Student st : list)

            System.out.println(st.getId() + " " + st.getName() + " " + st.getCourse() + " " +
st.getMarks());
        s.close();
    }

    public void updateStudent(int id, double marks) {
        Session s = factory.openSession();
        Transaction tx = s.beginTransaction();
Student st = s.get(Student.class, id);          if
(st != null) {              st.setMarks(marks);
```

```
            s.update(st);
}
        tx.commit();
        s.close();
    }

    public void getStudent(int id) {
        Session s = factory.openSession();
Student st = s.get(Student.class, id);          if
(st != null)
System.out.println(st.getId() + " " +
st.getName() + " " + st.getCourse() + " " +
st.getMarks());

        s.close();
    }

    public void deleteStudent(int id) {
        Session s = factory.openSession();
```

```
        Transaction tx = s.beginTransaction();
Student st = s.get(Student.class, id);          if
(st != null)
            s.delete(st);
tx.commit();
        s.close();
    }
}
```

Part c: Transaction Management Using Spring and Hibernate

Objective:
To create a banking system using Spring integrated with Hibernate that allows users to transfer money between accounts while ensuring transaction consistency.

CODE

```
 import org.springframework.context.annotation.*;
import org.springframework.stereotype.*; import
org.springframework.beans.factory.annotation.*; import
org.springframework.transaction.annotation.*; import
org.springframework.orm.hibernate5.*; import
org.hibernate.*; import javax.persistence.*; import
java.util.*;
```

```
@Entity
@Table(name = "account") class
Account {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;     private String name;     private
double balance;     public Account() {}     public
Account(String name, double balance) {
this.name = name;         this.balance = balance;

   }     public int getId() { return id; }     public String getName() {
return name; }     public double getBalance() { return balance; }
public void setBalance(double balance) { this.balance = balance; }
}

@Repository class AccountDAO {     @Autowired     private
SessionFactory sessionFactory;     public Account getAccount(int id)
{         return sessionFactory.getCurrentSession().get(Account.class,
id);
   }
```

```
   public void updateAccount(Account acc) {
sessionFactory.getCurrentSession().update(acc);
   }
}

@Service
@Transactional
class BankService {    @Autowired    private
AccountDAO dao;    public void transfer(int fromId, int
toId, double amount) {
       Account from = dao.getAccount(fromId);
Account to = dao.getAccount(toId);
from.setBalance(from.getBalance() - amount);
to.setBalance(to.getBalance() + amount);
dao.updateAccount(from);
dao.updateAccount(to);
   }
}

@Configuration
```

```
@ComponentScan(basePackages = "bank")
@EnableTransactionManagement class AppConfig {
@Bean     public LocalSessionFactoryBean
sessionFactory() {
        LocalSessionFactoryBean factory = new LocalSessionFactoryBean();
factory.setPackagesToScan("bank");

        Properties props = new Properties();
props.put("hibernate.connection.driver_class", "com.mysql.cj.jdbc.Driver");
props.put("hibernate.connection.url", "jdbc:mysql://localhost:3306/bankdb");
props.put("hibernate.connection.username", "root");
props.put("hibernate.connection.password", "1234");
props.put("hibernate.dialect", "org.hibernate.dialect.MySQL8Dialect");
props.put("hibernate.hbm2ddl.auto", "update");
factory.setHibernateProperties(props);         return factory;
    }

    @Bean
    public HibernateTransactionManager transactionManager(SessionFactory
sessionFactory) {         return new
HibernateTransactionManager(sessionFactory);
    }
```

```
}


public class BankingApp {     public

static void main(String[] args) {

        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

        BankService service = context.getBean(BankService.class);

service.transfer(1, 2, 500.0);

        System.out.println("Transaction Successful");

context.close();

    }

}
```

OUTPUT

Transaction Successful