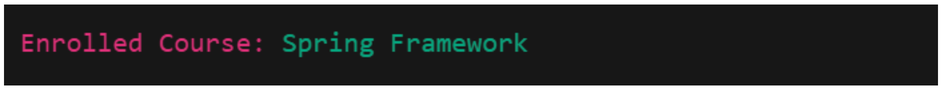
```
public class Course {
  public String getCourseName() {
    return "Spring Framework";
  }
}
public class Student {
  private Course course;

  public Student(Course course) {
    this.course = course;
  }

  public void showCourse() {
    System.out.println("Enrolled Course: " + course.getCourseName());
  }
}
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

  @Bean
  public Course course() {
    return new Course();
```

```
                                                                   }

    @Bean
    public Student student() {
        return new Student(course());
    }
}
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);
        Student student = context.getBean(Student.class);
        student.showCourse();
    }
}
```



```
Enrolled Course: Spring Framework
```

B)

Student.java

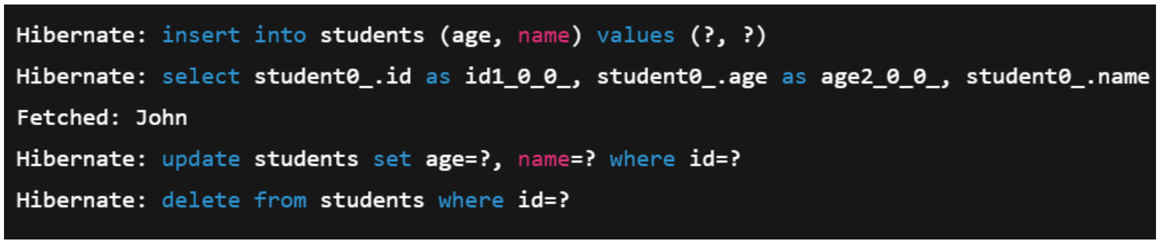
java

CopyEdit

```
import javax.persistence.*;

@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private int age;

    // Constructors, getters and setters
}

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

public class StudentDao {
    private static SessionFactory factory = new
Configuration().configure().buildSessionFactory();

    public void addStudent(Student s) {
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        session.save(s);
        tx.commit();
```

```
                                                session.close();
    }

    public Student getStudent(int id) {
        Session session = factory.openSession();
        Student s = session.get(Student.class, id);
        session.close();
        return s;
    }

    public void updateStudent(Student s) {
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        session.update(s);
        tx.commit();
        session.close();
    }

    public void deleteStudent(int id) {
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        Student s = session.get(Student.class, id);
        session.delete(s);
        tx.commit();
        session.close();
    }
```

```
                                                    }


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


    Student s = new Student();
    s.setName("John");
    s.setAge(21);
    dao.addStudent(s);


    Student fetched = dao.getStudent(1);
    System.out.println("Fetched: " + fetched.getName());


    fetched.setName("John Updated");
    dao.updateStudent(fetched);


    dao.deleteStudent(1);
  }
}
```



```
Hibernate: insert into students (age, name) values (?, ?)
Hibernate: select student0_.id as id1_0_0_, student0_.age as age2_0_0_, student0_.name
Fetched: John
Hibernate: update students set age=?, name=? where id=?
Hibernate: delete from students where id=?
```

C)

```
import javax.persistence.*;

@Entity
public class Account {
   @Id
   private int id;
   private String name;
   private double balance;

   // Getters, setters, constructors
}
```

BankService.java

java

CopyEdit

```
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.*;

@Service
public class BankService {

   @PersistenceContext
   private EntityManager em;
```

```
    @Transactional
    public void transfer(int fromId, int toId, double amount) {
        Account from = em.find(Account.class, fromId);
        Account to = em.find(Account.class, toId);

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

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

        em.persist(from);
        em.persist(to);
    }
}
```

AppConfig.java

java

CopyEdit

```
import org.springframework.context.annotation.*;
import org.springframework.orm.jpa.*;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.annotation.EnableTransactionManagement;

import javax.persistence.EntityManagerFactory;
import java.util.Properties;
```

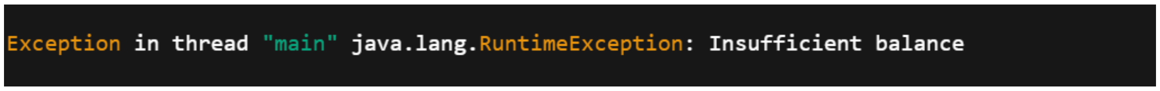
```
@Configuration
@ComponentScan("your.package.name")
@EnableTransactionManagement
public class AppConfig {

    @Bean
    public LocalContainerEntityManagerFactoryBean emf() {
        LocalContainerEntityManagerFactoryBean emf = new
LocalContainerEntityManagerFactoryBean();
        emf.setPersistenceUnitName("myPU");
        emf.setPackagesToScan("your.package.name");

        Properties props = new Properties();
        props.setProperty("hibernate.hbm2ddl.auto", "update");
        props.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");
        emf.setJpaProperties(props);

        return emf;
    }

    @Bean
    public PlatformTransactionManager transactionManager(EntityManagerFactory emf) {
        return new JpaTransactionManager(emf);
    }
}
```

MainBanking.java

java

CopyEdit

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


public class MainBanking {
  public static void main(String[] args) {
    var context = new AnnotationConfigApplicationContext(AppConfig.class);
    BankService bankService = context.getBean(BankService.class);


    bankService.transfer(1, 2, 500.0);
    System.out.println("Transfer successful.");
  }
}
```



```
Exception in thread "main" java.lang.RuntimeException: Insufficient balance
```