EXPERIMENT NO. 10

Title: Application using Hibernate Mapping Collections.

Outcome: Students will be able to develop application using Hibernate Mapping Collections.

Theory:

Simple Web Application using Hibernate Mapping with Collections in **Eclipse (Dynamic Web Project)**.
This example shows how Hibernate handles **collection mapping** — specifically a **one-to-many relationship** using List.
We'll create:
✅ Student (one) → Subjects (many)
A student can have multiple subjects stored as a collection.

🧩 **Step 1: Create a New Dynamic Web Project**

1. **Eclipse → File → New → Dynamic Web Project**
2. Project name: HibernateCollectionDemo
3. Target runtime: **Apache Tomcat 8.5 or 9.0**
4. Dynamic Web Module: **3.1**
5. Click **Finish**

🗃️ **Step 2: Database Setup (MySQL)**
Create database and tables.

```
CREATE DATABASE hibernatedb;
USE hibernatedb;

CREATE TABLE student (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50)
);

CREATE TABLE subject (
  id INT PRIMARY KEY AUTO_INCREMENT,
  subject_name VARCHAR(50),
  student_id INT,
  FOREIGN KEY (student_id) REFERENCES student(id)
);
```

📦 Step 3: Add Hibernate & MySQL JARs

Copy these JARs to your project's WebContent/WEB-INF/lib or src/main/webapp/WEB-INF/lib:

- hibernate-core-5.x.x.jar
- hibernate-commons-annotations.jar
- hibernate-jpa-2.1-api.jar
- dom4j.jar
- javassist.jar
- antlr.jar
- mysql-connector-java-8.x.jar
- commons-logging.jar
- (any slf4j/log4j JARs required)

⚙️ Step 4: Create Hibernate Configuration File

📄 src/hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<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/hibernatedb?useSSL=false
&amp;amp;serverTimezone=UTC</property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.connection.password">your_password</property>
      <property
name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>
      <property name="show_sql">true</property>
      <property name="hibernate.hbm2ddl.auto">update</property>

      <mapping resource="student.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

📜 Step 5: Mapping File with Collection

📄 src/student.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="com.example.bean.Student" table="student">
      <id name="id" column="id">
         <generator class="increment"/>
      </id>
      <property name="name" column="name"/>

      <!-- Collection Mapping -->
      <set name="subjects" cascade="all" inverse="true" table="subject">
         <key column="student_id"/>
         <one-to-many class="com.example.bean.Subject"/>
      </set>
   </class>

   <class name="com.example.bean.Subject" table="subject">
      <id name="id" column="id">
         <generator class="increment"/>
      </id>
      <property name="subjectName" column="subject_name"/>
   </class>
</hibernate-mapping>
```

👷 Step 6: Entity Classes

📄 src/com/example/bean/Student.java

```
package com.example.bean;
import java.util.Set;

public class Student {
   private int id;
   private String name;
   private Set<Subject> subjects;

   public Student() {}

   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 Set<Subject> getSubjects() { return subjects; }
    public void setSubjects(Set<Subject> subjects) { this.subjects = subjects; }
}
```

src/com/example/bean/Subject.java

```
package com.example.bean;

public class Subject {
    private int id;
    private String subjectName;

    public Subject() {}

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

    public String getSubjectName() { return subjectName; }
    public void setSubjectName(String subjectName) { this.subjectName = subjectName;
}
}
```

🧩 Step 7: Hibernate Utility Class

src/com/example/util/HibernateUtil.java

```
package com.example.util;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }
```

```
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

💾 Step 8: Servlet to Save Student and Subjects

📄 src/com/example/servlet/AddStudentServlet.java

```
package com.example.servlet;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import javax.servlet.*;
import javax.servlet.http.*;

import org.hibernate.Session;
import org.hibernate.Transaction;

import com.example.bean.Student;
import com.example.bean.Subject;
import com.example.util.HibernateUtil;

public class AddStudentServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String name = request.getParameter("name");
        String sub1 = request.getParameter("sub1");
        String sub2 = request.getParameter("sub2");

        Subject s1 = new Subject();
        s1.setSubjectName(sub1);
        Subject s2 = new Subject();
        s2.setSubjectName(sub2);

        Set<Subject> subjects = new HashSet<>();
        subjects.add(s1);
        subjects.add(s2);

        Student student = new Student();
```

```
        student.setName(name);
        student.setSubjects(subjects);

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.save(student);
        tx.commit();
        session.close();

        response.sendRedirect("success.jsp");
    }
}
```

Step 9: web.xml

WebContent/WEB-INF/web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" version="3.1">
  <servlet>
    <servlet-name>AddStudentServlet</servlet-name>
    <servlet-class>com.example.servlet.AddStudentServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>AddStudentServlet</servlet-name>
    <url-pattern>/addStudent</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Step 10: JSP Pages

index.jsp

```
<html>
<head><title>Hibernate Collection Example</title></head>
<body>
<h2>Student Registration</h2>
<form action="addStudent" method="post">
  Student Name: <input type="text" name="name"/><br/>
  Subject 1: <input type="text" name="sub1"/><br/>
```

```
    Subject 2: <input type="text" name="sub2"/><br/>
    <input type="submit" value="Save Student"/>
</form>
</body>
</html>
```

📄 **success.jsp**

```
<html><body>
<h3>Student and Subjects saved successfully!</h3>
<a href="index.jsp">Add Another Student</a>
</body></html>
```

▶️ **Step 11: Run and Test**

1. Right-click project → **Run As** → **Run on Server**.
2. Open browser:
   👉 http://localhost:8080/HibernateCollectionDemo/
3. Enter:
   - Name: Ravi
   - Subject1: Maths
   - Subject2: Physics
4. Click **Save Student**.
5. Check MySQL:

```
SELECT * FROM student;
SELECT * FROM subject;
```

✅ You'll see:

student.id student.name

1 Ravi

subject.id subject_name student_id

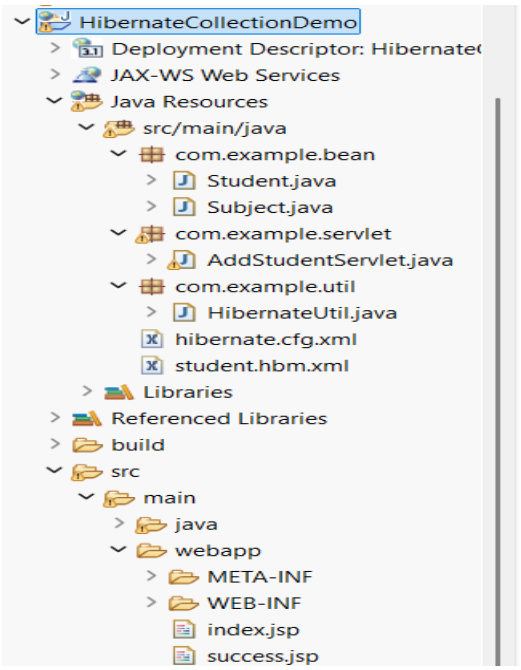
1 Maths 1

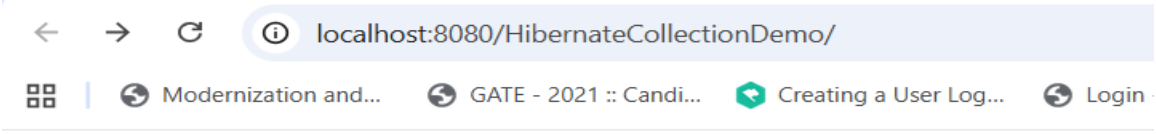
2 Physics 1

🧠 **Understanding the Concept**

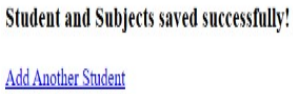
- Student → parent entity
- Subject → child entity
- <set> mapping defines a one-to-many relationship.
- cascade="all" means saving a student automatically saves its subjects.
- inverse="true" prevents circular updates.

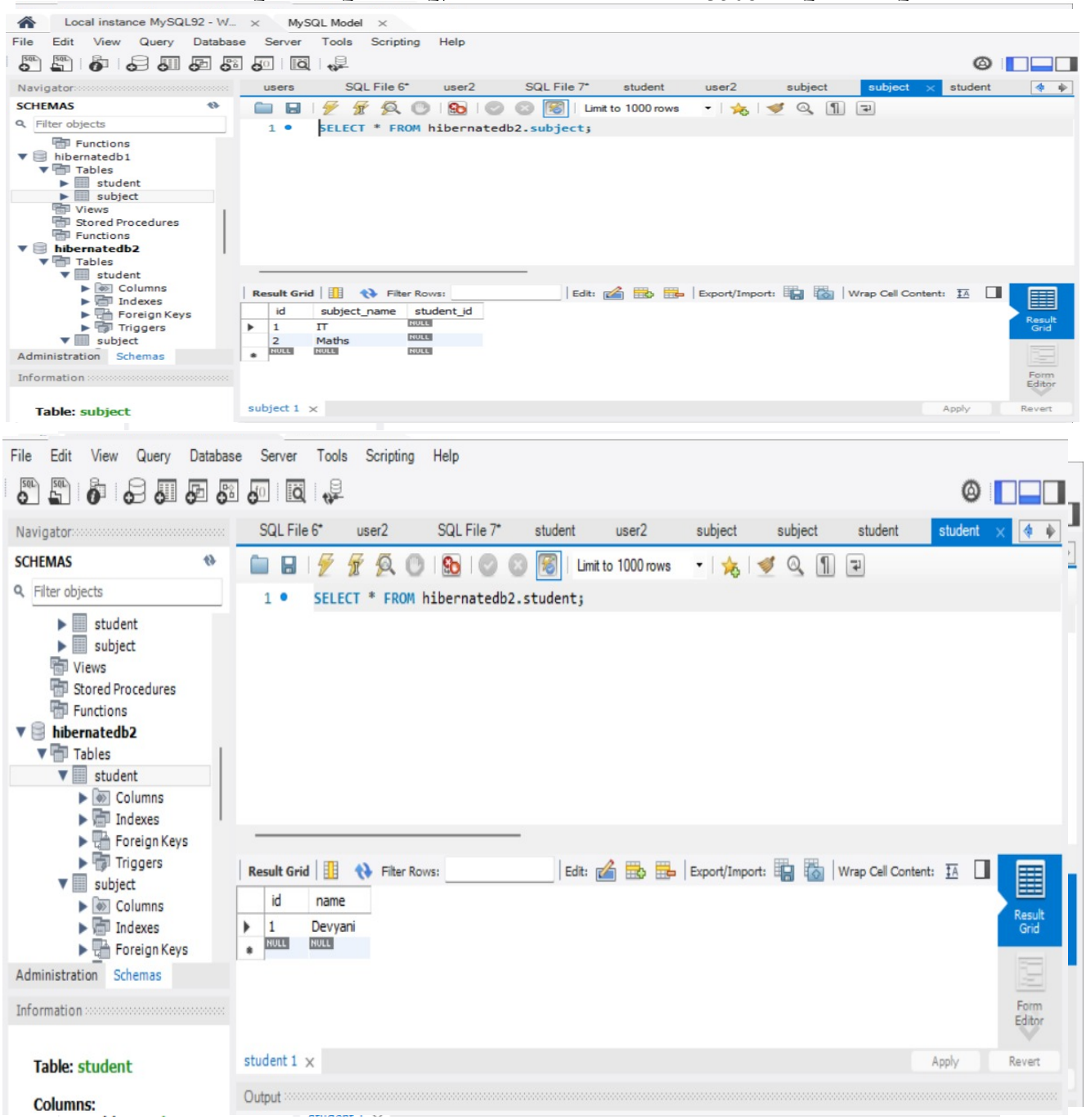
Output:-


HibernateCollectionDemo
Deployment Descriptor: Hibernate
JAX-WS Web Services
Java Resources
src/main/java
com.example.bean
Student.java
Subject.java
com.example.servlet
AddStudentServlet.java
com.example.util
HibernateUtil.java
hibernate.cfg.xml
student.hbm.xml
Libraries
Referenced Libraries
build
src
main
java
webapp
META-INF
WEB-INF
index.jsp
success.jsp


localhost:8080/HibernateCollectionDemo/
Modernization and...
GATE - 2021 :: Candi...
Creating a User Log...
Login


localhost:8080/exp_10/
Student Registration
Student Name: Devyani
Subject 1: IT
Subject 2: Maths
Save Student


Student and Subjects saved successfully!
Add Another Student



Conclusion : Thus, I have developed application using Hibernate Mapping Collections.