Table of Contents

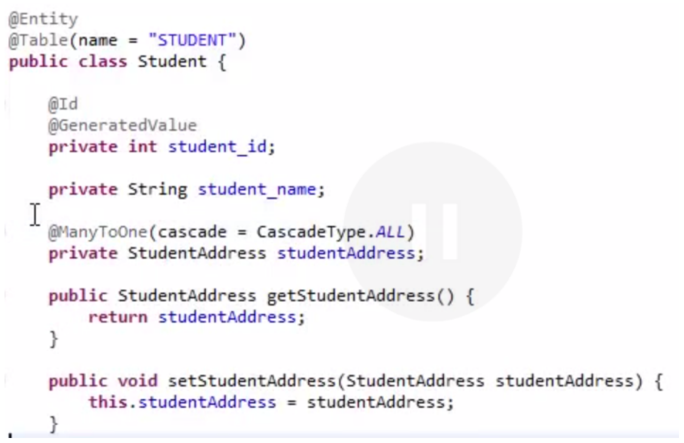
http://www.java4s.com/hibernate/difference-between-merge-and-update-methods-in-hibernate/

http://www.concretepage.com/hibernate/hibernate-object-states

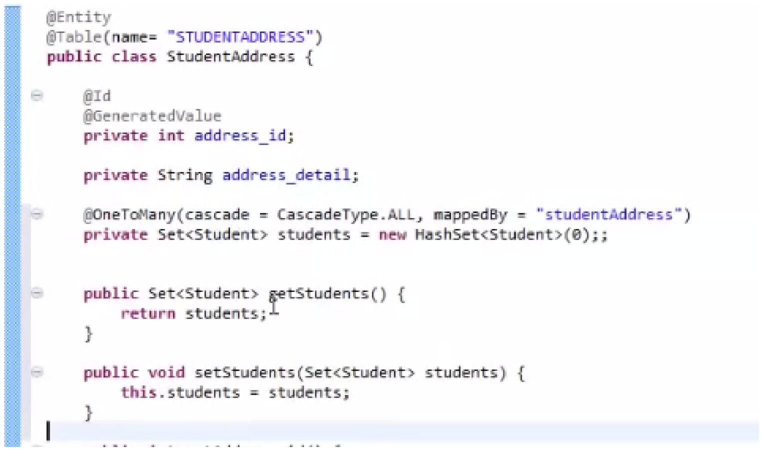
Hibernate defines and supports the following object states:

- Transient - an object is transient if it has just been instantiated using the new operator, and it is not associated with a Hibernate Session . ...
- Persistent - a persistent instance has a representation in the database and an identifier value.
- Mapping in this image is the right way.

ManytoOne OneToMany mapping - Bidirectional

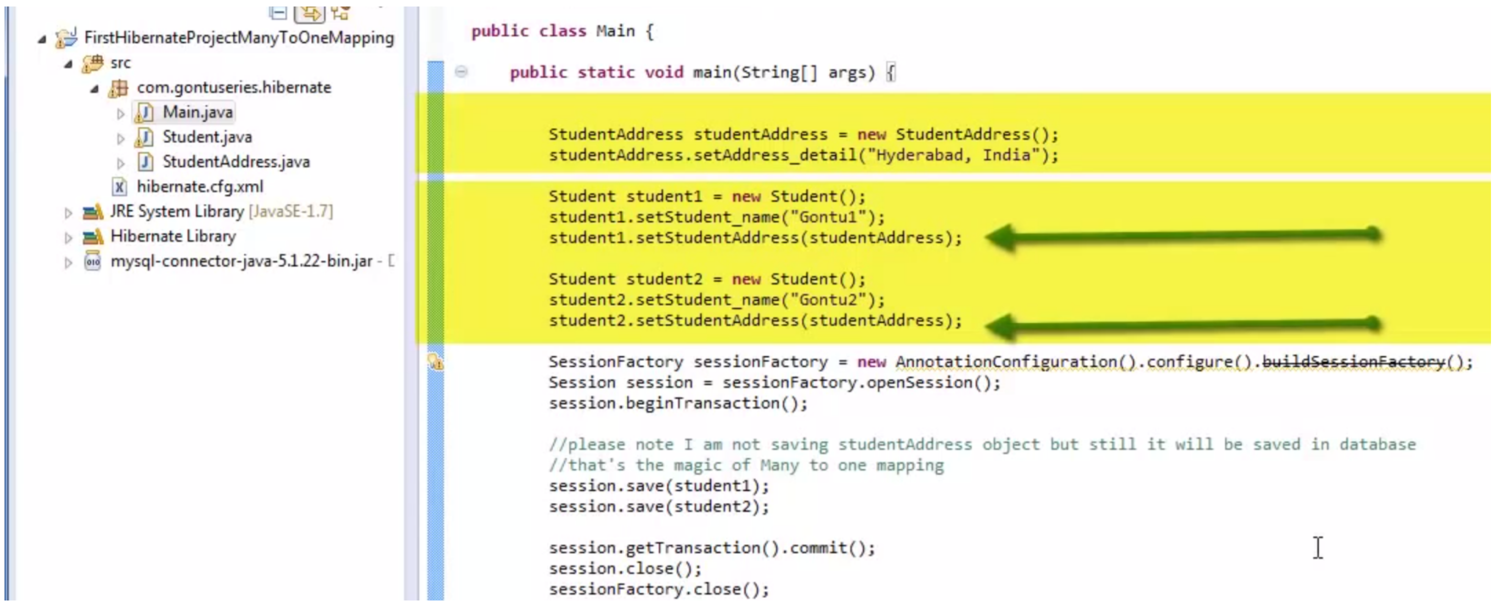


```
@Entity
@Table(name = "STUDENT")
public class Student {

    @Id
    @GeneratedValue
    private int student_id;

    private String student_name;

    @ManyToOne(cascade = CascadeType.ALL)
    private StudentAddress studentAddress;

    public StudentAddress getStudentAddress() {
        return studentAddress;
    }

    public void setStudentAddress(StudentAddress studentAddress) {
        this.studentAddress = studentAddress;
    }
```



```
@Entity
@Table(name= "STUDENTADDRESS")
public class StudentAddress {

    @Id
    @GeneratedValue
    private int address_id;

    private String address_detail;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "studentAddress")
    private Set<Student> students = new HashSet<Student>(0);;


    public Set<Student> getStudents() {
        return students;
    }

    public void setStudents(Set<Student> students) {
        this.students = students;
    }
```

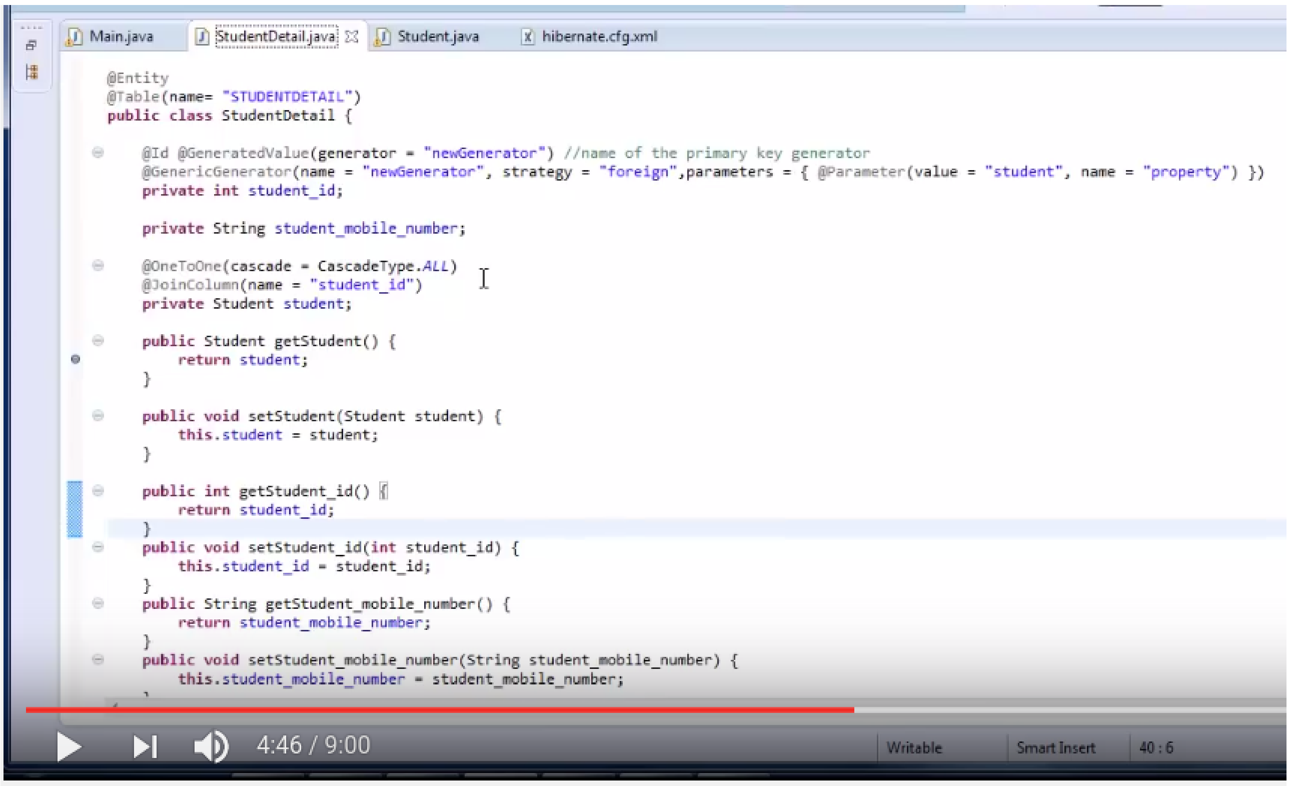
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name="egal welche name")



FirstHibernateProjectManyToOneMapping
src
com.gontuseries.hibernate
Main.java
Student.java
StudentAddress.java
hibernate.cfg.xml
JRE System Library [JavaSE-1.7]
Hibernate Library
mysql-connector-java-5.1.22-bin.jar

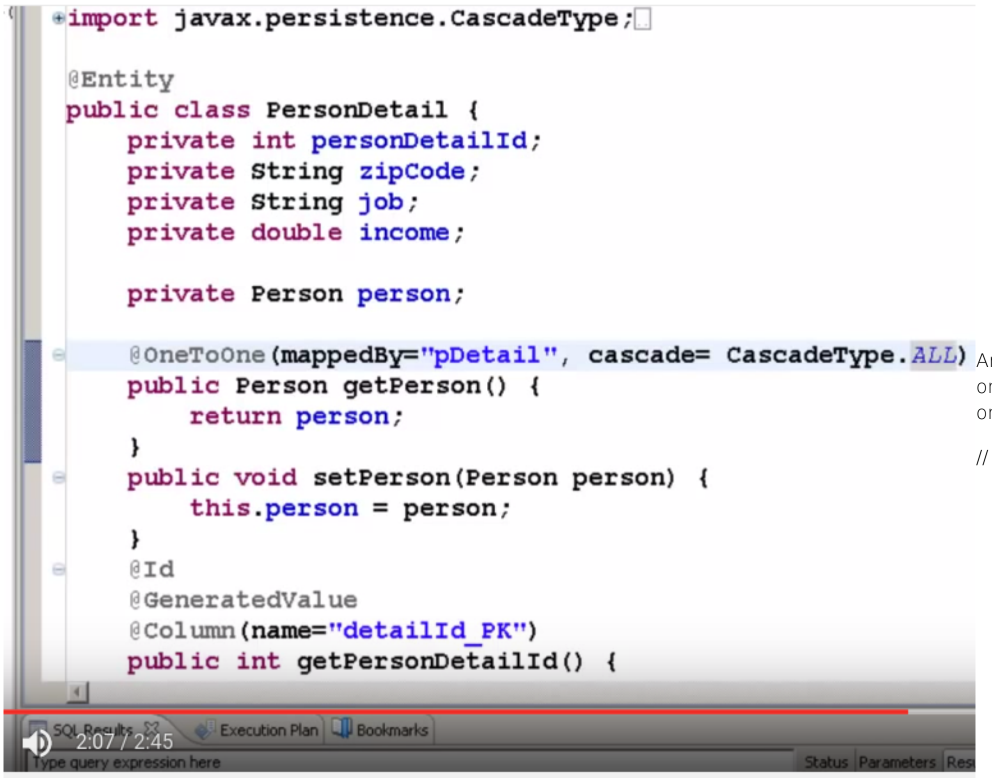
```
public class Main {

    public static void main(String[] args) {

        StudentAddress studentAddress = new StudentAddress();
        studentAddress.setAddress_detail("Hyderabad, India");

        Student student1 = new Student();
        student1.setStudent_name("Gontu1");
        student1.setStudentAddress(studentAddress);

        Student student2 = new Student();
        student2.setStudent_name("Gontu2");
        student2.setStudentAddress(studentAddress);

        SessionFactory sessionFactory = new AnnotationConfiguration().configure().~~buildSessionFactory~~();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        //please note I am not saving studentAddress object but still it will be saved in database
        //that's the magic of Many to one mapping
        session.save(student1);
        session.save(student2);

        session.getTransaction().commit();
        session.close();
        sessionFactory.close();
```

OneToOne Bidirectional



```java
@Entity
@Table(name= "STUDENTDETAIL")
public class StudentDetail {

    @Id @GeneratedValue(generator = "newGenerator") //name of the primary key generator
    @GenericGenerator(name = "newGenerator", strategy = "foreign",parameters = { @Parameter(value = "student", name = "property") })
    private int student_id;

    private String student_mobile_number;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "student_id")
    private Student student;

    public Student getStudent() {
        return student;
    }

    public void setStudent(Student student) {
        this.student = student;
    }

    public int getStudent_id() {
        return student_id;
    }
    public void setStudent_id(int student_id) {
        this.student_id = student_id;
    }
    public String getStudent_mobile_number() {
        return student_mobile_number;
    }
    public void setStudent_mobile_number(String student_mobile_number) {
        this.student_mobile_number = student_mobile_number;
```

Hibernate Tutorial part 9 - Unidirectional VS Bidirectional One to One Mapping, CascadeType



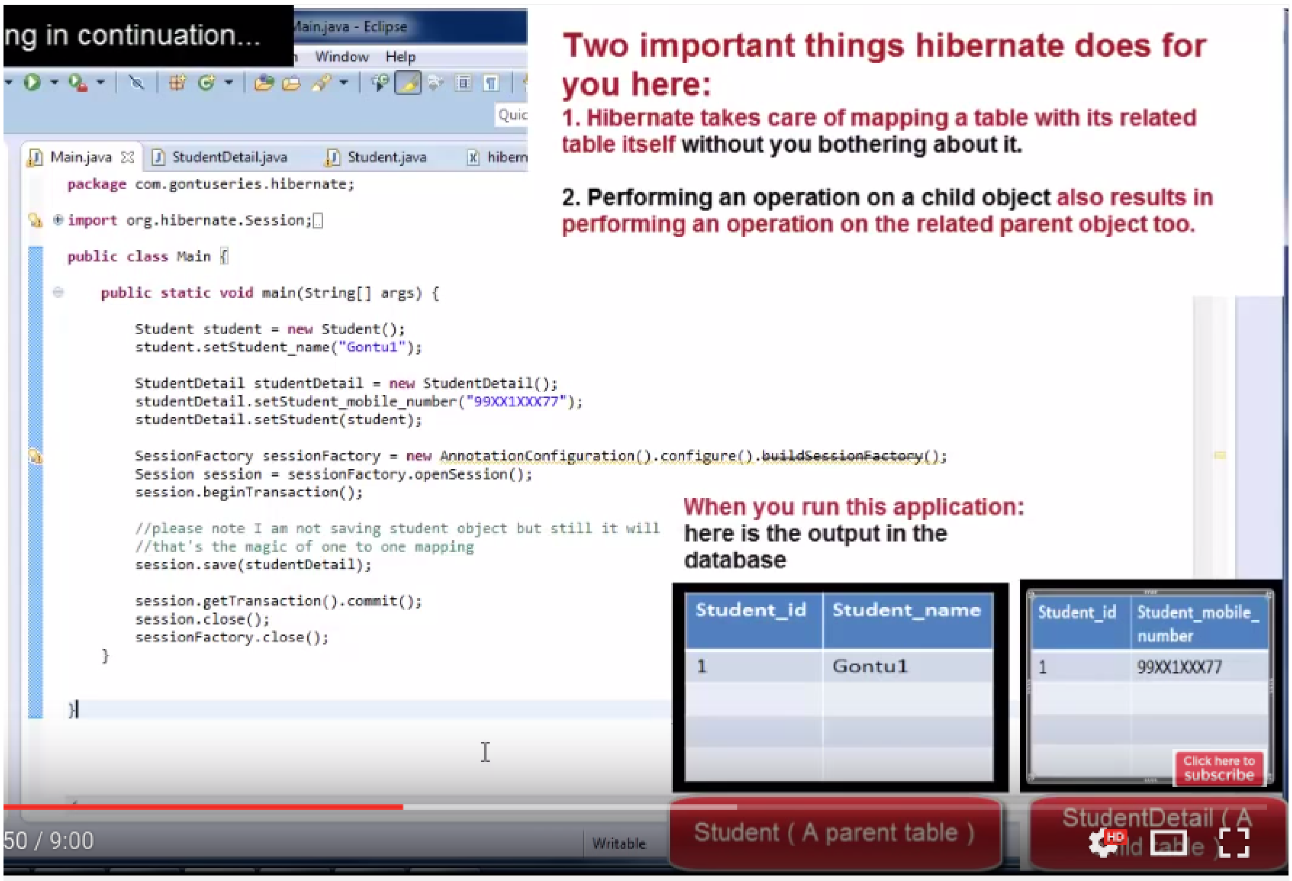
```java
import javax.persistence.CascadeType;

@Entity
public class PersonDetail {
    private int personDetailId;
    private String zipCode;
    private String job;
    private double income;

    private Person person;

    @OneToOne(mappedBy="pDetail", cascade= CascadeType.ALL)
    public Person getPerson() {
        return person;
    }
    public void setPerson(Person person) {
        this.person = person;
    }
    @Id
    @GeneratedValue
    @Column(name="detailId_PK")
    public int getPersonDetailId() {
```

Annotations should be placed on Object declaration and not on getters and setters.

// Private Person person

One to One Bi-directional

Up ne



```java
package com.gontuseries.hibernate;

import org.hibernate.Session;

public class Main {

    public static void main(String[] args) {

        Student student = new Student();
        student.setStudent_name("Gontu1");

        StudentDetail studentDetail = new StudentDetail();
        studentDetail.setStudent_mobile_number("99XX1XXX77");
        studentDetail.setStudent(student);

        SessionFactory sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        //please note I am not saving student object but still it will
        //that's the magic of one to one mapping
        session.save(studentDetail);

        session.getTransaction().commit();
        session.close();
        sessionFactory.close();
    }

}
```

Two important things hibernate does for you here:

1. Hibernate takes care of mapping a table with its related table itself without you bothering about it.

2. Performing an operation on a child object also results in performing an operation on the related parent object too.

When you run this application: here is the output in the database

Student_id	Student_name
1	Gontu1

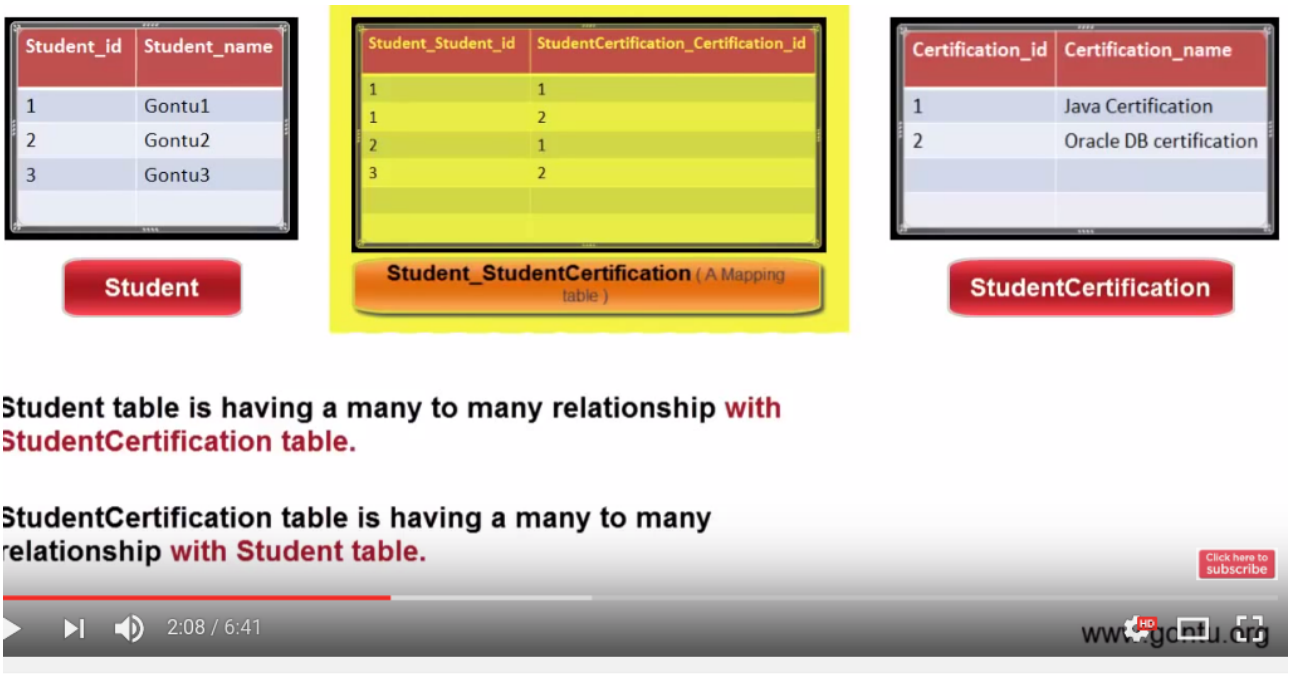
Student (A parent table)

Student_id	Student_mobile_number
1	99XX1XXX77

StudentDetail (A child table)

rial part 9 - Unidirectional VS Bidirectional One to One
adeType

ManyToMany bidirectional Relationships.



Student_id	Student_name
1	Gontu1
2	Gontu2
3	Gontu3

Student

Student_Student_id	StudentCertification_Certification_id
1	1
1	2
2	1
3	2

Student_StudentCertification (A Mapping table)

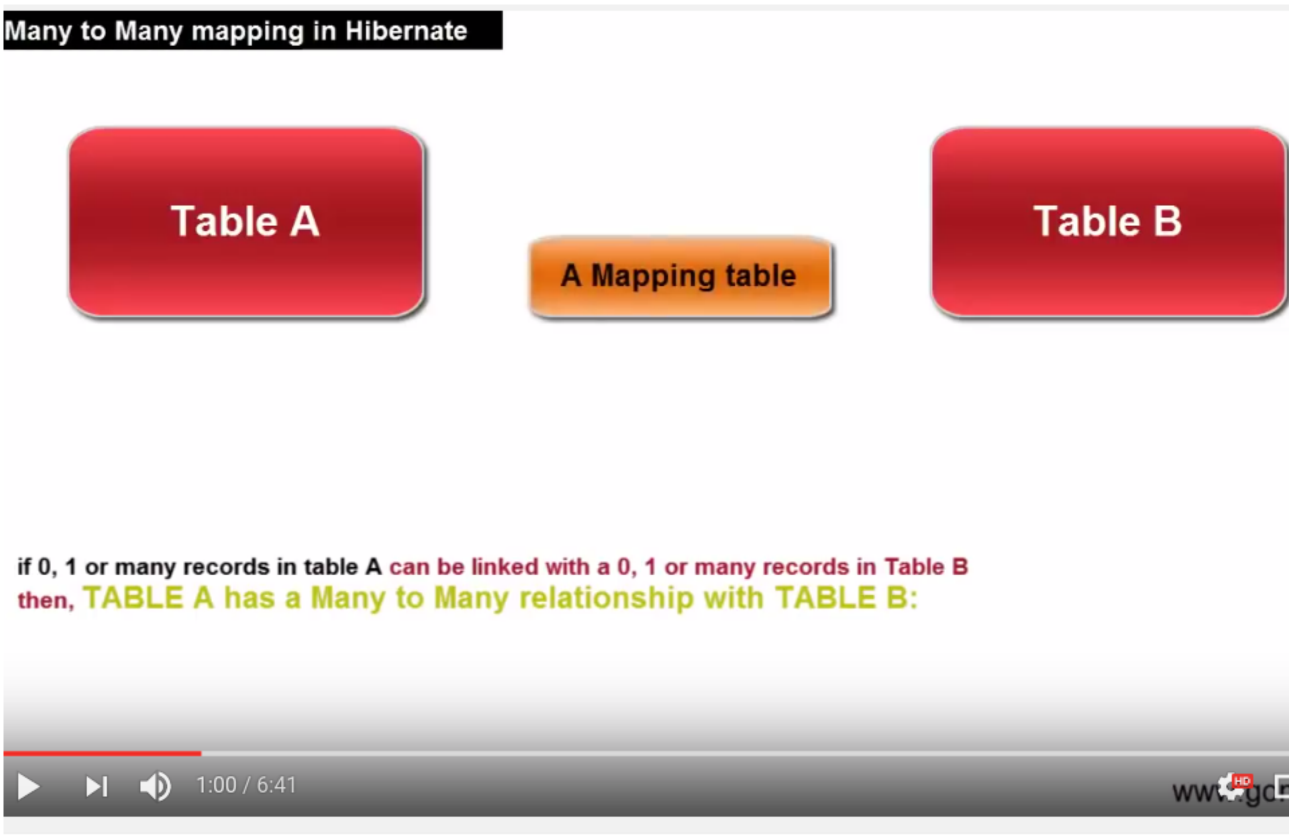
Certification_id	Certification_name
1	Java Certification
2	Oracle DB certification

StudentCertification

Student table is having a many to many relationship with StudentCertification table.

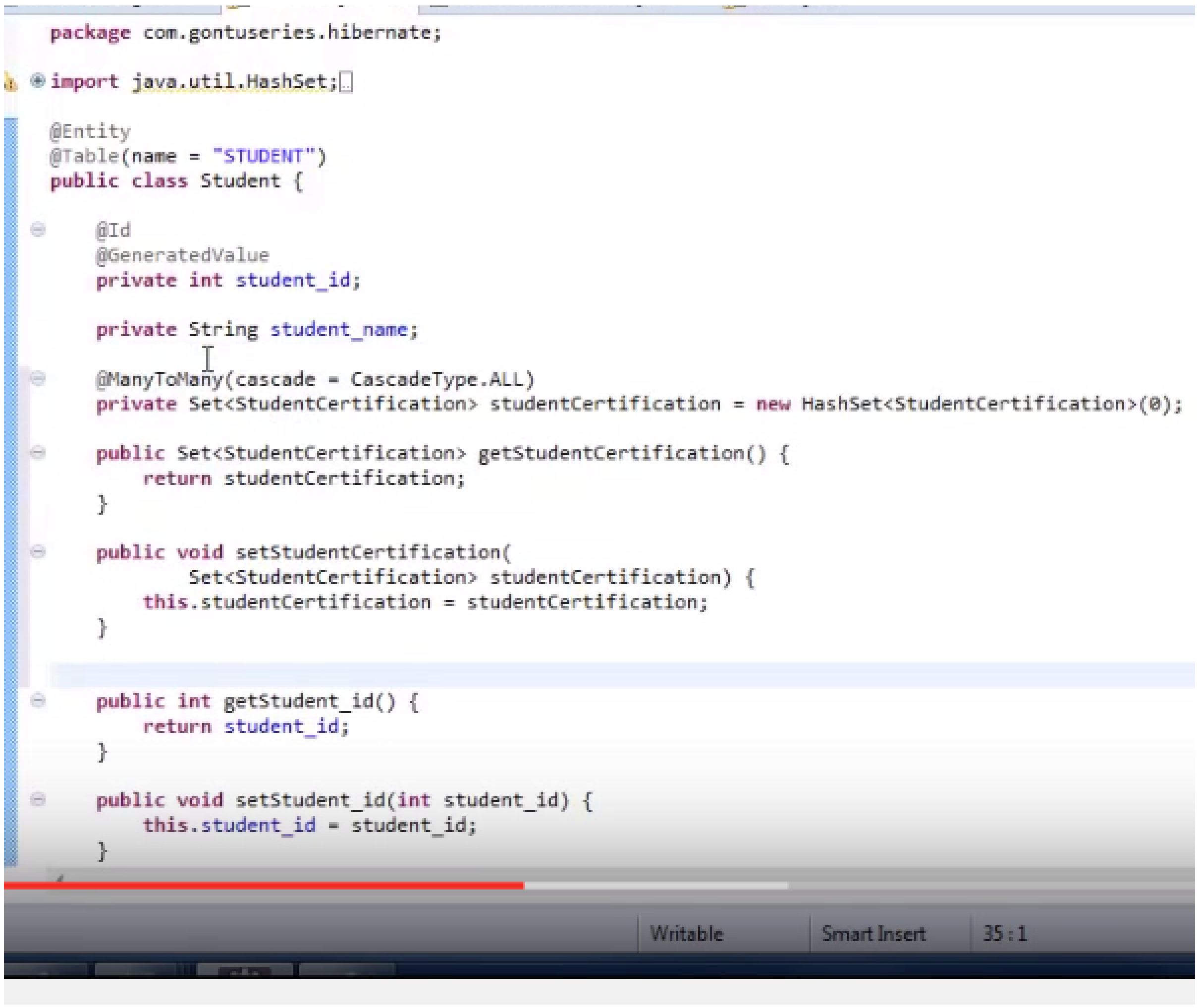
StudentCertification table is having a many to many relationship with Student table.

ibernate Tutorial part 13 - Many to Many mapping in detail


Many to Many mapping in Hibernate

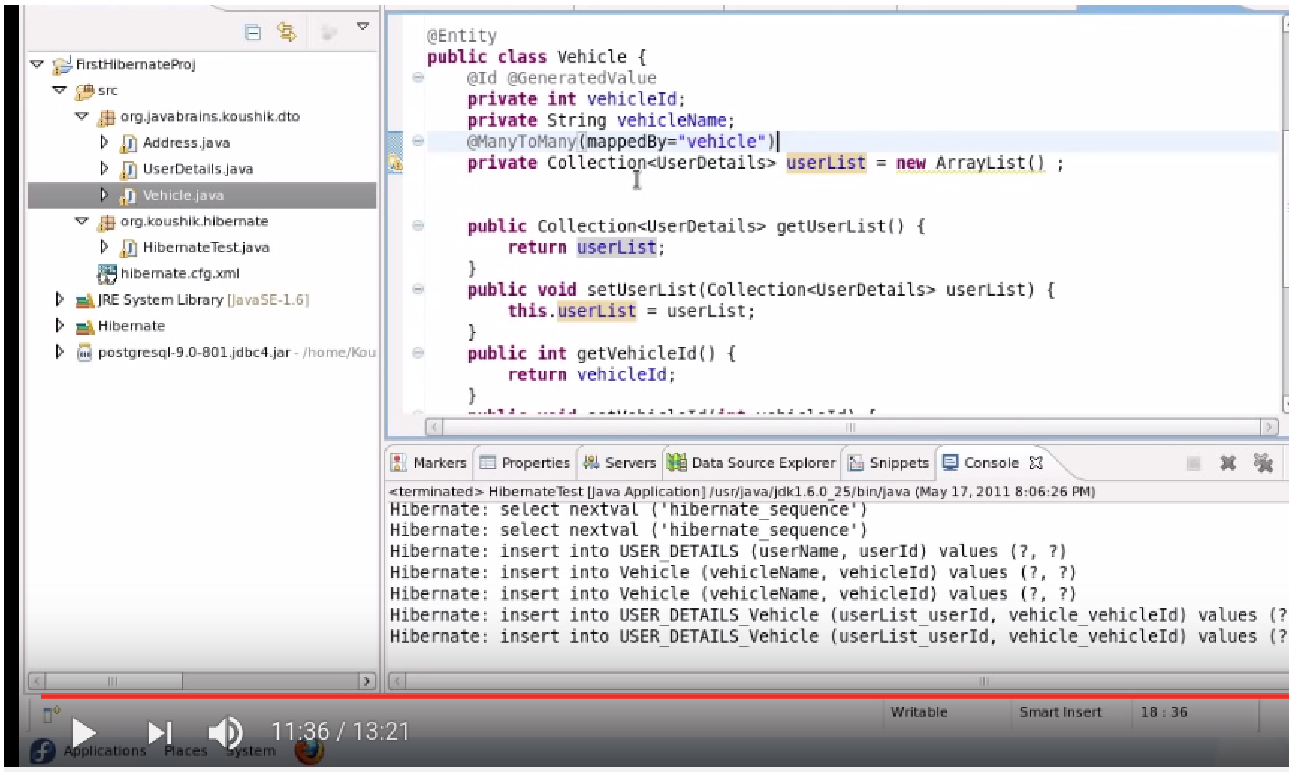
if 0, 1 or many records in table A can be linked with a 0, 1 or many records in Table B
then, TABLE A has a Many to Many relationship with TABLE B:

Hibernate Tutorial part 13 - Many to Many mapping in detail


```
package com.gontuseries.hibernate;

import java.util.HashSet;

@Entity
@Table(name = "STUDENT")
public class Student {

    @Id
    @GeneratedValue
    private int student_id;

    private String student_name;

    @ManyToMany(cascade = CascadeType.ALL)
    private Set<StudentCertification> studentCertification = new HashSet<StudentCertification>(0);

    public Set<StudentCertification> getStudentCertification() {
        return studentCertification;
    }

    public void setStudentCertification(
            Set<StudentCertification> studentCertification) {
        this.studentCertification = studentCertification;
    }


    public int getStudent_id() {
        return student_id;
    }

    public void setStudent_id(int student_id) {
        this.student_id = student_id;
    }
```

13 - Many to Many mapping in detail

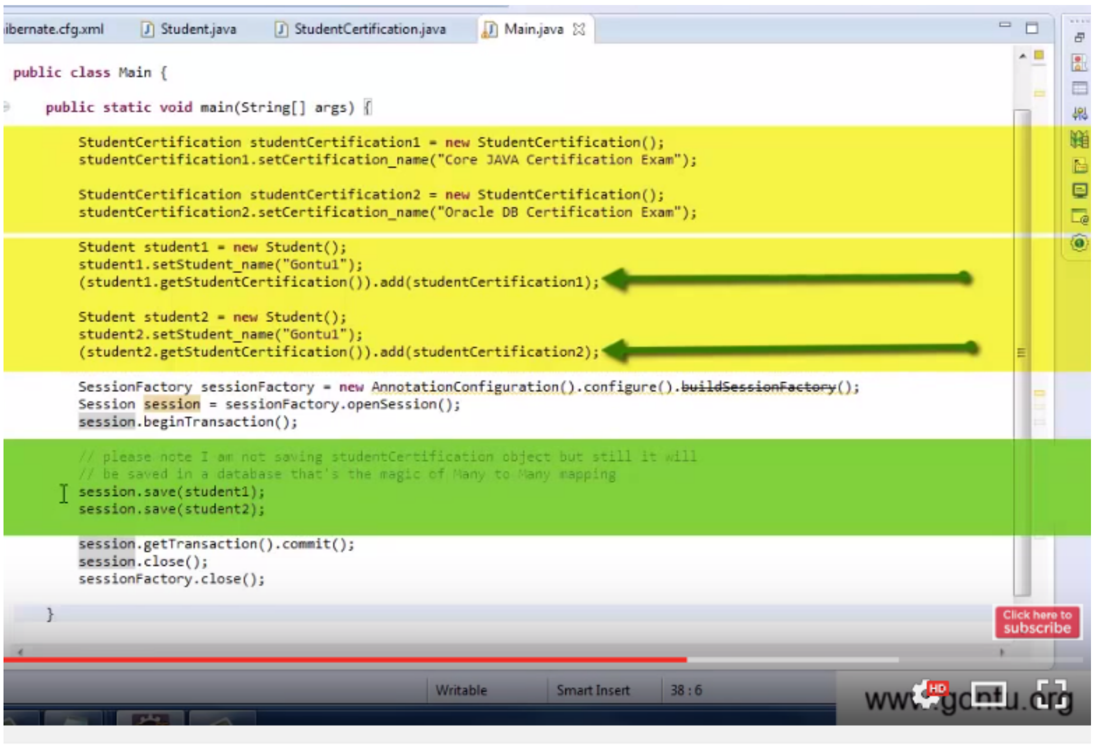
Mapping done here. Note you can also include : cascade = CascadeType.ALL annotation.



```
@Entity
public class Vehicle {
    @Id @GeneratedValue
    private int vehicleId;
    private String vehicleName;
    @ManyToMany(mappedBy="vehicle")
    private Collection<UserDetails> userList = new ArrayList() ;


    public Collection<UserDetails> getUserList() {
        return userList;
    }
    public void setUserList(Collection<UserDetails> userList) {
        this.userList = userList;
    }
    public int getVehicleId() {
        return vehicleId;
    }
```

```
<terminated> HibernateTest [Java Application] /usr/java/jdk1.6.0_25/bin/java (May 17, 2011 8:06:26 PM)
Hibernate: select nextval ('hibernate_sequence')
Hibernate: select nextval ('hibernate_sequence')
Hibernate: insert into USER_DETAILS (userName, userId) values (?, ?)
Hibernate: insert into Vehicle (vehicleName, vehicleId) values (?, ?)
Hibernate: insert into Vehicle (vehicleName, vehicleId) values (?, ?)
Hibernate: insert into USER_DETAILS_Vehicle (userList_userId, vehicle_vehicleId) values (?
Hibernate: insert into USER_DETAILS_Vehicle (userList_userId, vehicle_vehicleId) values (?
```

- mappedBy and Many To Many Mapping

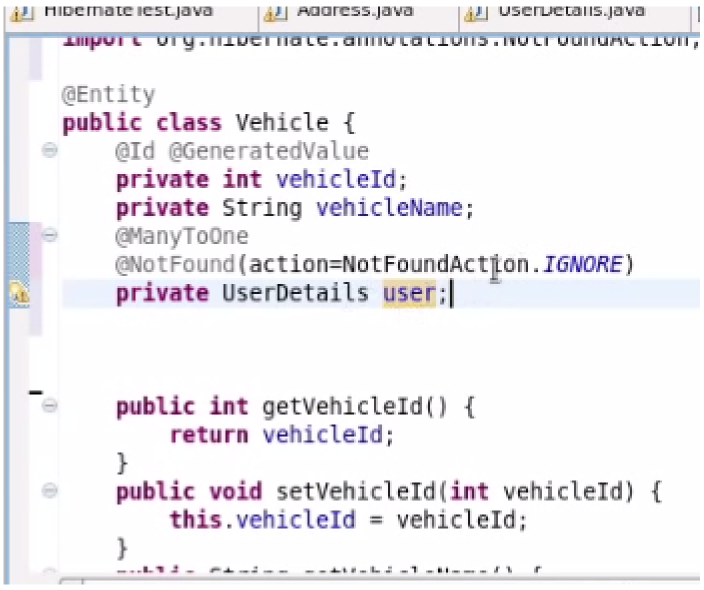


```
public class Main {

    public static void main(String[] args) {

        StudentCertification studentCertification1 = new StudentCertification();
        studentCertification1.setCertification_name("Core JAVA Certification Exam");

        StudentCertification studentCertification2 = new StudentCertification();
        studentCertification2.setCertification_name("Oracle DB Certification Exam");

        Student student1 = new Student();
        student1.setStudent_name("Gontu1");
        (student1.getStudentCertification()).add(studentCertification1);

        Student student2 = new Student();
        student2.setStudent_name("Gontu1");
        (student2.getStudentCertification()).add(studentCertification2);

        SessionFactory sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        // please note I am not saving studentCertification object but still it will
        // be saved in a database that's the magic of Many to Many mapping
        session.save(student1);
        session.save(student2);

        session.getTransaction().commit();
        session.close();
        sessionFactory.close();

    }
```

13 - Many to Many mapping in detail

https://www.youtube.com/watch?v=jAi8bY-H_ek

How to avoid NotFound Exception in hibernate



```java
import org.hibernate.annotations.NotFoundAction;

@Entity
public class Vehicle {
    @Id @GeneratedValue
    private int vehicleId;
    private String vehicleName;
    @ManyToOne
    @NotFound(action=NotFoundAction.IGNORE)
    private UserDetails user;


    public int getVehicleId() {
        return vehicleId;
    }
    public void setVehicleId(int vehicleId) {
        this.vehicleId = vehicleId;
    }
    public String getVehicleName() {
```

With the @NotFound annotation hibernate will ignore not throw an exception when it does not find a table on the database.