### Unidirectional one-to-one mapping:

```
public class Student {
    @Id
    private long studentId;

    3 usages
    @Column(name = "student_name")
    private String studentName;

2 usages
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "Student_fK")
    private StudentDetails studentDetails;

    no usages
    public long getStudentId() { return studentId; }

2 usages
    public void setStudentId(long studentId) { this.studentId = studentId; }
```

```
StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);

StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);

Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);

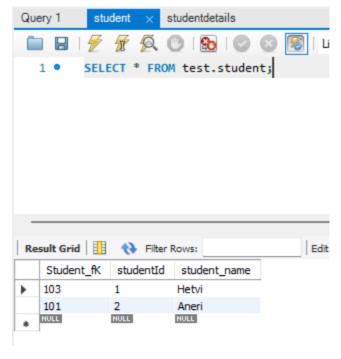
Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);
```

# Bidirectional one-to-one mapping:

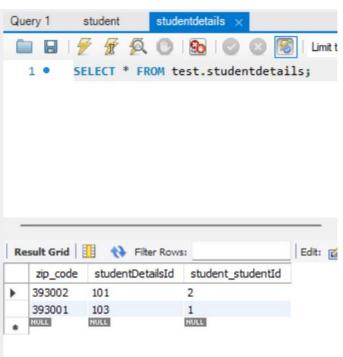
```
@Entity
public class StudentDetails {
    @Id
    private Long studentDetailsId;
    2 usages
    @Column(name = "zip_code")
    private int zipCode;
    2 usages
    @OneToOne(cascade = CascadeType.ALL)
    private Student student;

    no usages
    public Student getStudent() {
        return student;
    }

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



```
StudentDetails sd1 = new StudentDetails();
sd1.setStudentDetailsId(101L);
sd1.setZipCode(393002);
StudentDetails sd2 = new StudentDetails();
sd2.setStudentDetailsId(103L);
sd2.setZipCode(393001);
Student s1 = new Student();
s1.setStudentId(1);
s1.setStudentName("Hetvi");
s1.setStudentDetails(sd2);
Student s2 = new Student();
s2.setStudentId(2);
s2.setStudentName("Aneri");
s2.setStudentDetails(sd1);
sd1.setStudent(s2);
sd2.setStudent(s1);
Query 1
         student
                  studentdetails ×
```



# Many-to-one: (Many students are in one college)

### One-to-many: (one college have many students)

```
@Entity
public class Student_College {
    @Id
    @Column(name = "college_id")
    private int collegeId;

2 usages
    @Column(name = "college_name")
    private String collegeName;

2 usages
    @OneToMany(mappedBy = "studentCollege", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Student> students;

    no usages
    public List<Student> getStudents() { return students; }

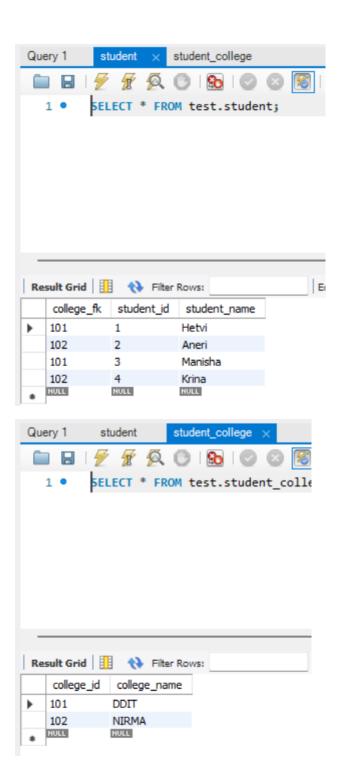
2 usages
    public void setStudents(List<Student> students) { this.students = students; }
```

```
student1.setStudentCollege(college1);
student2.setStudentCollege(college2);
student3.setStudentCollege(college1);
student4.setStudentCollege(college2);

List<Student> ddit_students = new ArrayList<>();
ddit_students.add(student1);
ddit_students.add(student3);

List<Student> nirma_students = new ArrayList<>();
nirma_students.add(student2);
nirma_students.add(student4);

college1.setStudents(ddit_students);
college2.setStudents(nirma_students);
```



# Many-to-many: (many students have many certificates) Create certificate entity class with id and name

```
@Entity
public class Student {
    @Id
    @Column(name = "student_id")
    private int studentId;

2 usages
    @Column(name = "student_name")
    private String studentName;

2 usages
    @ManyToMany(cascade = CascadeType.ALL)
    private Set<Certification> studentCertificates;

no usages

public Set<Certification> getStudentCertificates() {
        return studentCertificates;
    }

4 usages
    public void setStudentCertificates(Set<Certification> studentCertificates) {
        this.studentCertificates = studentCertificates;
    }
```

