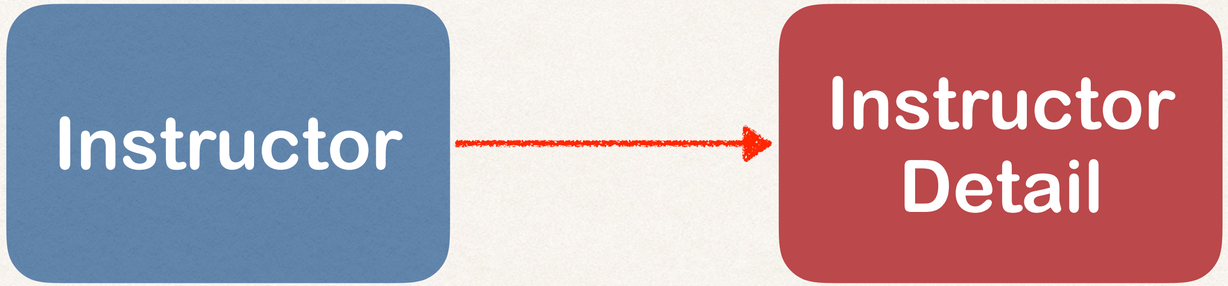
luv2code

Hibernate One-to-One Bi-Directional

One-to-One Mapping

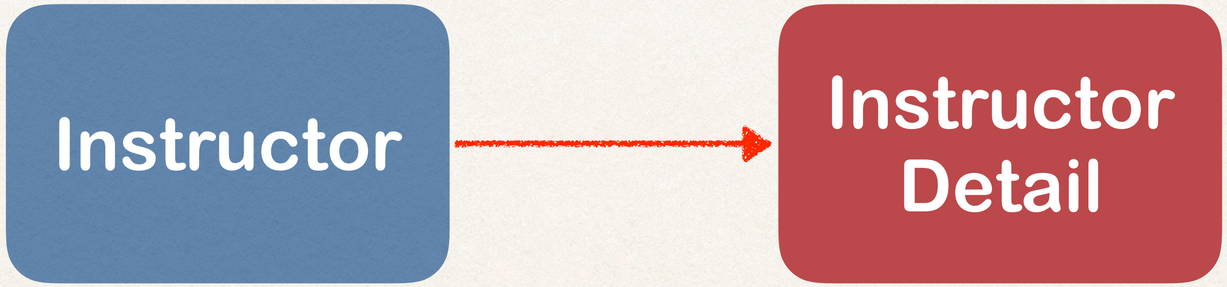
- We currently have a uni-directional mapping



New Use Case

- If we load an InstructorDetail
  - Then we'd like to get the associated Instructor

- Can't do this with current uni-directional relationship :-(



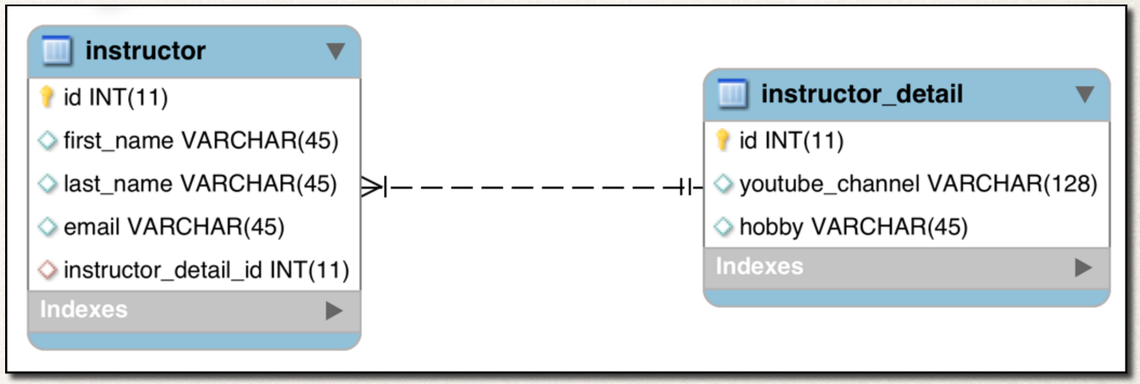
Bi-Directional

- Bi-Directional relationship is the solution

- We can start with InstructorDetail and make it back to the Instructor

Bi-Directional - The Good News

- To use Bi-Directional, we can keep the existing database schema
  - No changes required to database
- Simply update the Java code



Development Process: One-to-One (Bi-Directional)

Step-By-Step

1. Make updates to InstructorDetail class:
    1. Add new field to reference Instructor
    2. Add getter/setter methods for Instructor
    3. Add @OneToOne annotation

2. Create Main App

Step 1.1: Add new field to reference Instructor

```
@Entity
@Table(name="instructor_detail")
public class InstructorDetail {
    ...

    private Instructor instructor;

    ...

}
```

Step 1.2: Add getter/setter methods Instructor

```
@Entity
@Table(name="instructor_detail")
public class InstructorDetail {
    ...


    private Instructor instructor;

    public Instructor getInstructor() {
        return instructor;
    }

    public void setInstructor(Instructor instructor) {
        this.instructor = instructor;
    }
    ...

}
```

Step 1.3: Add @OneToOne annotation

```
@Entity
@Table(name="instructor_detail")
public class InstructorDetail {
    ...

    @OneToOne(mappedBy="instructorDetail")
    private Instructor instructor;

    public Instructor getInstructor() {
        return instructor;
    }

    public void setInstructor(Instructor instructor) {
        this.instructor = instructor;
    }
    ...

}
```

Refers to “instructorDetail” property in “Instructor” class

More on mappedBy

```
public class InstructorDetail {
    ...

    @OneToOne(mappedBy="instructorDetail")
    private Instructor instructor;
```

- **mappedBy** tells Hibernate
  - Look at the instructorDetail property in the Instructor class
  - Use information from the Instructor class @JoinColumn
  - To help find associated instructor

```
public class Instructor {
    ...

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="instructor_detail_id")
    private InstructorDetail instructorDetail;
```

Add support for Cascading

Cascade all operations
to the associated Instructor

```
@Entity
@Table(name="instructor_detail")
public class InstructorDetail {
    ...

    @OneToOne(mappedBy="instructorDetail", cascade=CascadeType.ALL)
    private Instructor instructor;

    public Instructor getInstructor() {
        return instructor;
    }

    public void setInstructor(Instructor instructor) {
        this.instructor = instructor;
    }
    ...

}
```

Step 2: Create Main App


```
public static void main(String[] args) {
        ...

        // get the instructor detail object
        int theId = 1;
        InstructorDetail tempInstructorDetail = session.get(InstructorDetail.class, theId);

        // print detail
        System.out.println("tempInstructorDetail: " + tempInstructorDetail);

        // print the associated instructor
        System.out.println("the associated instructor: "
                                    + tempInstructorDetail.getInstructor());
        ...
}
```