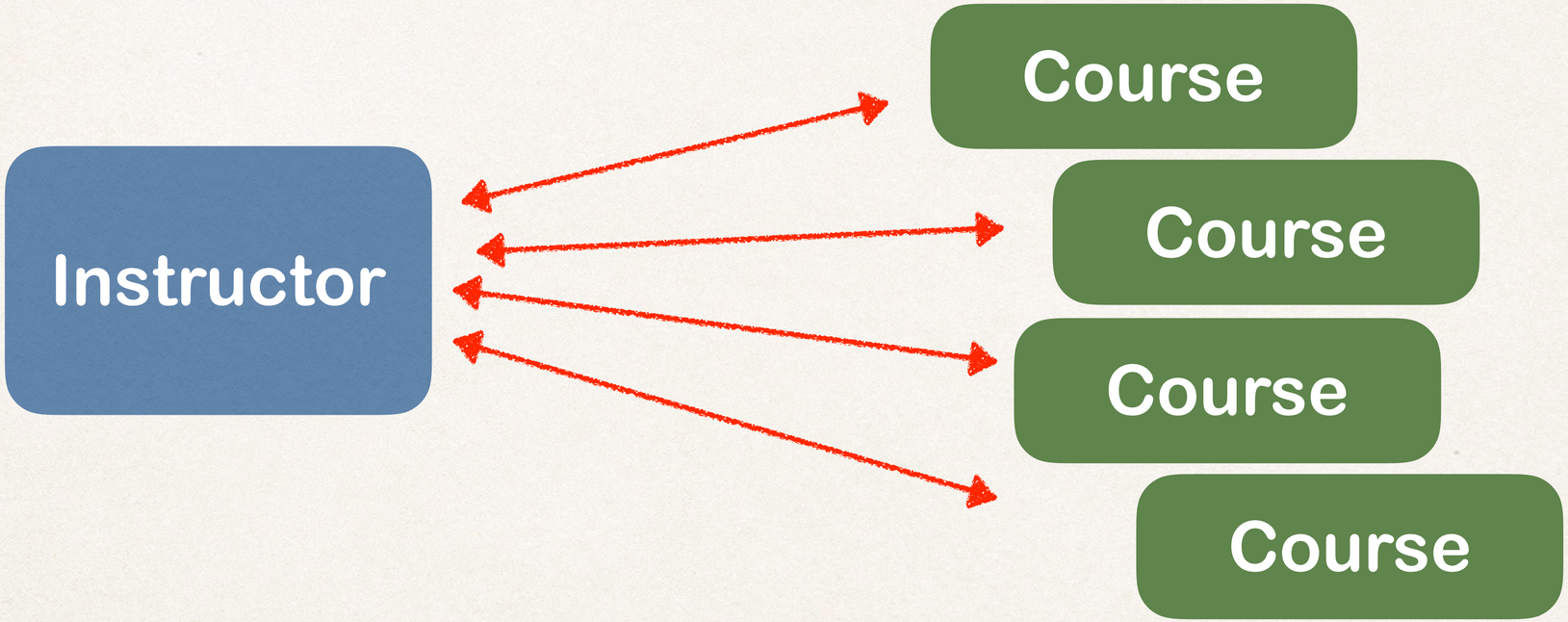
luv2code

Hibernate One-to-Many Bi-Directional

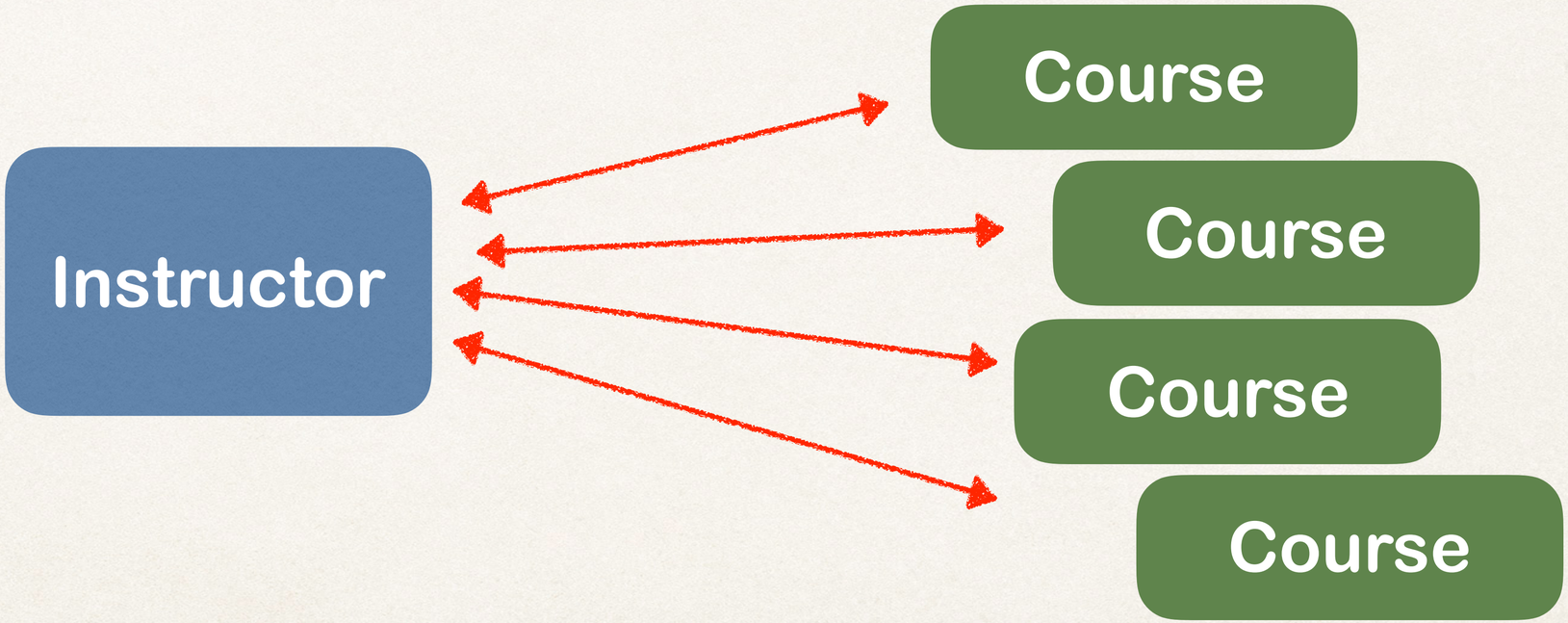
One-to-Many Mapping

- An instructor can have many courses
  - Bi-directional



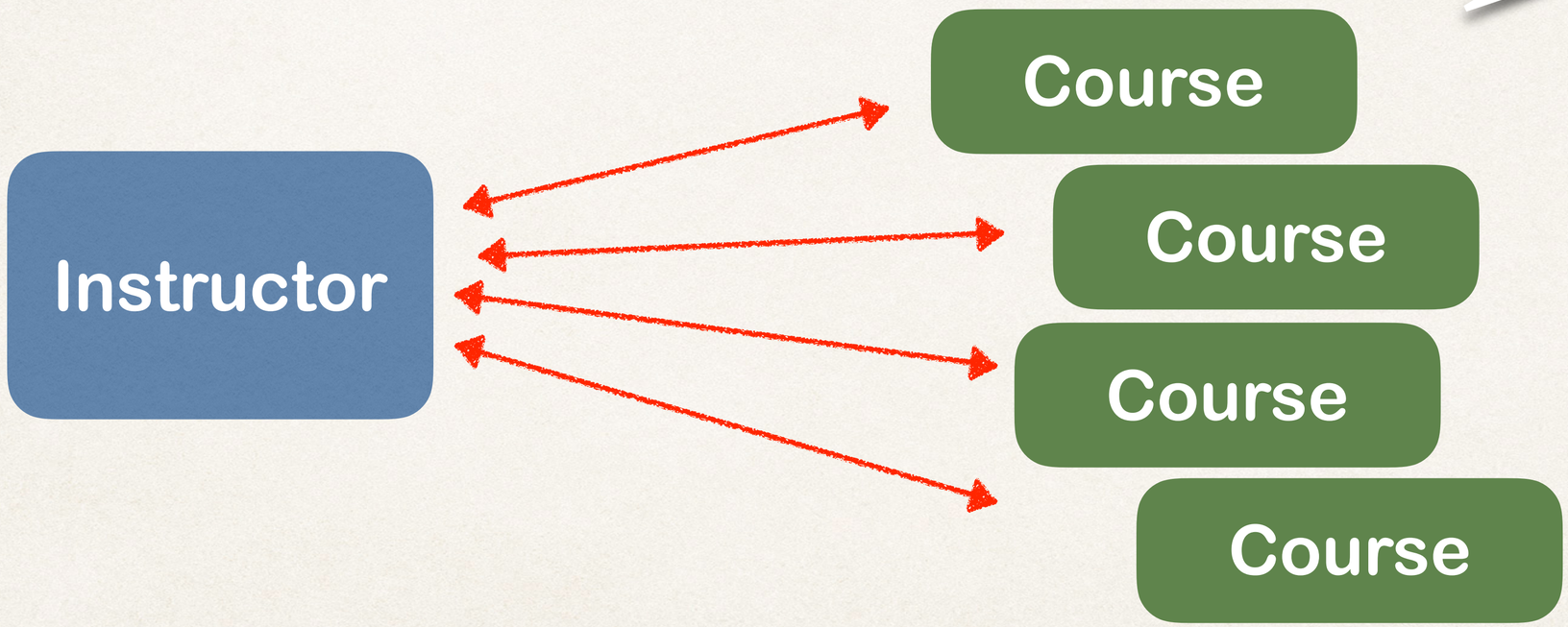
Many-to-One Mapping

- Many courses can have one instructor
  - Inverse / opposite of One-to-Many



Real-World Project Requirement

- If you delete an instructor, DO NOT delete the courses
- If you delete a course, DO NOT delete the instructor



Development Process: One-to-Many

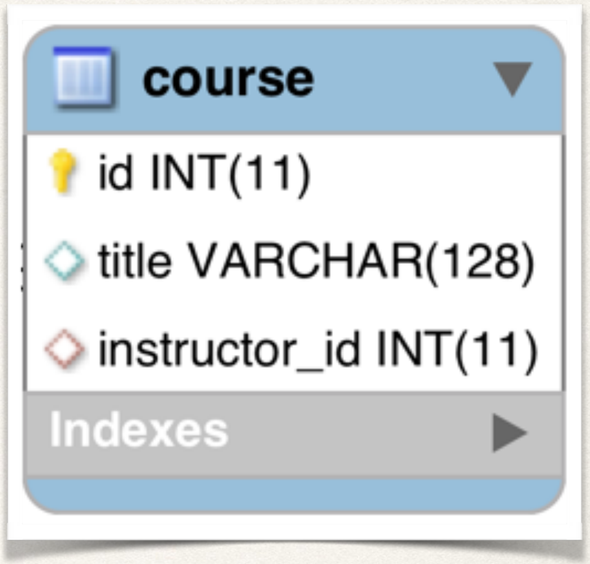
Step-By-Step

1. Prep Work - Define database tables
2. Create Course class
3. Update Instructor class
4. Create Main App

table: course

File: create-db.sql

```
CREATE TABLE `course` (

  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(128) DEFAULT NULL,
  `instructor_id` int(11) DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `TITLE_UNIQUE` (`title`),
    ...
);
```



Prevent duplicate course titles

table: course - foreign key

File: create-db.sql

```
CREATE TABLE `course` (
  ...
  KEY `FK_INSTRUCTOR_idx` (`instructor_id`),
  CONSTRAINT `FK_INSTRUCTOR`
  FOREIGN KEY (`instructor_id`)
  REFERENCES `instructor` (`id`)
  ...
);
```

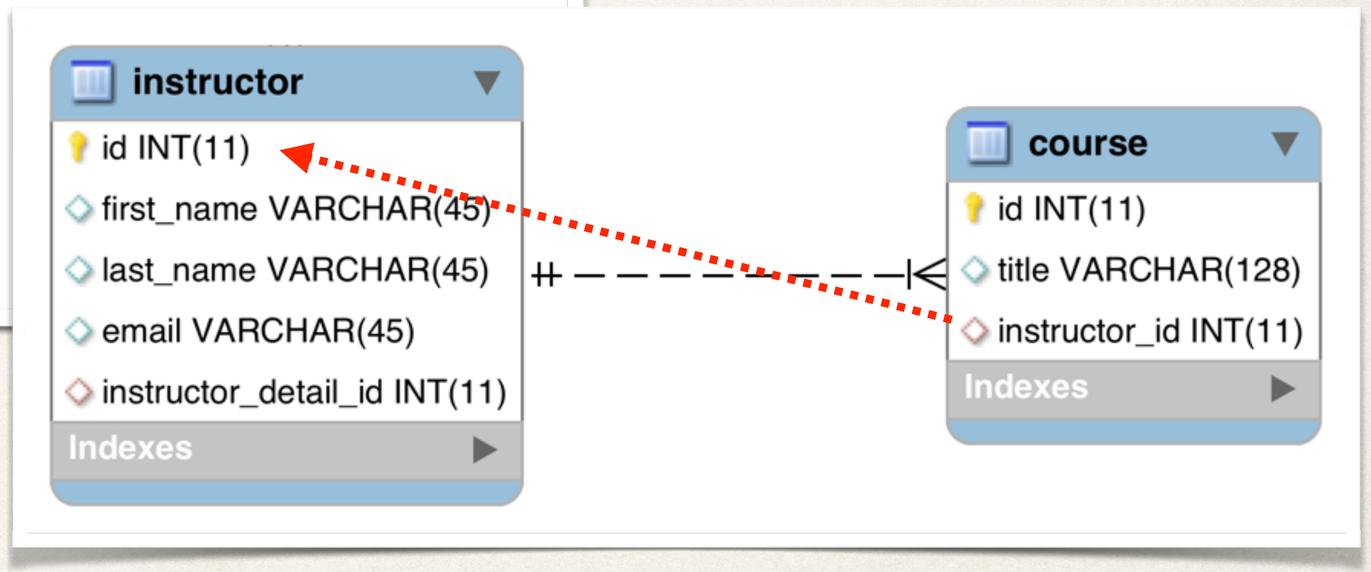
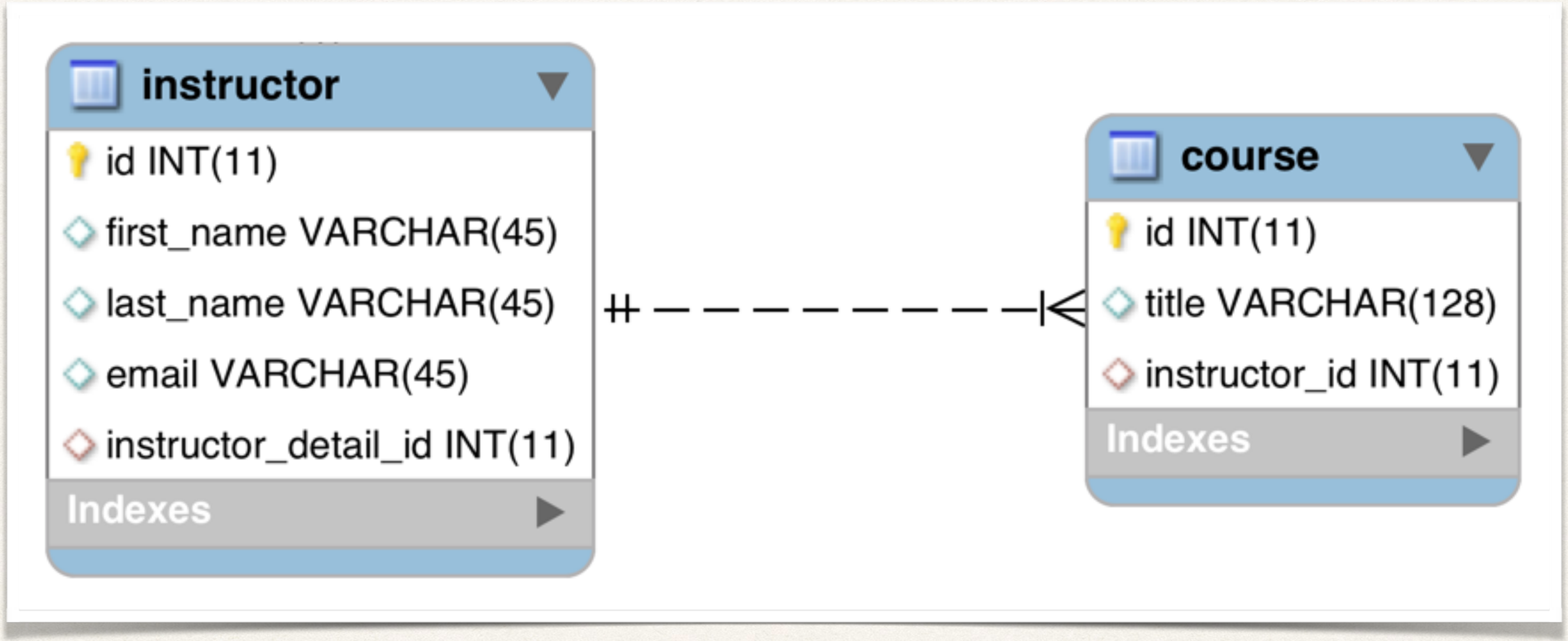
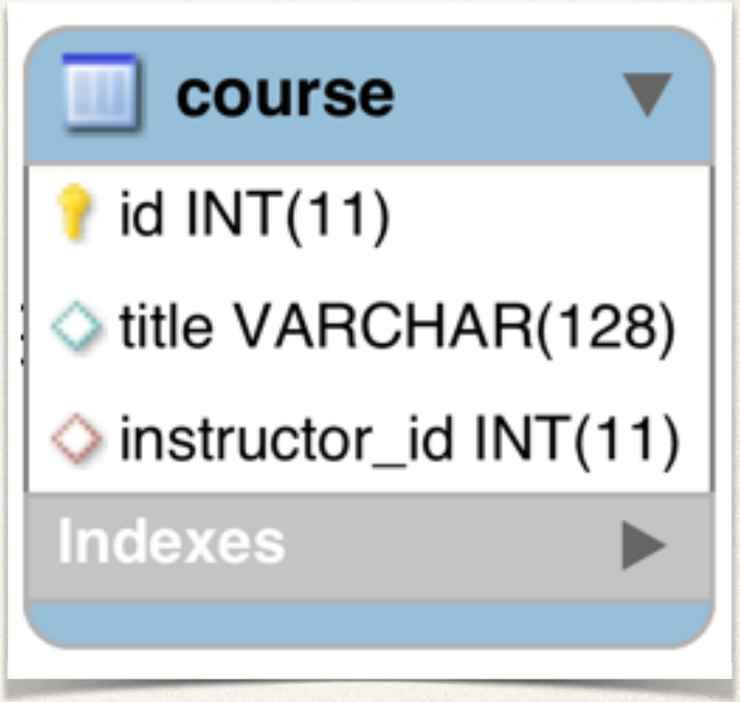


table: instructor - no changes



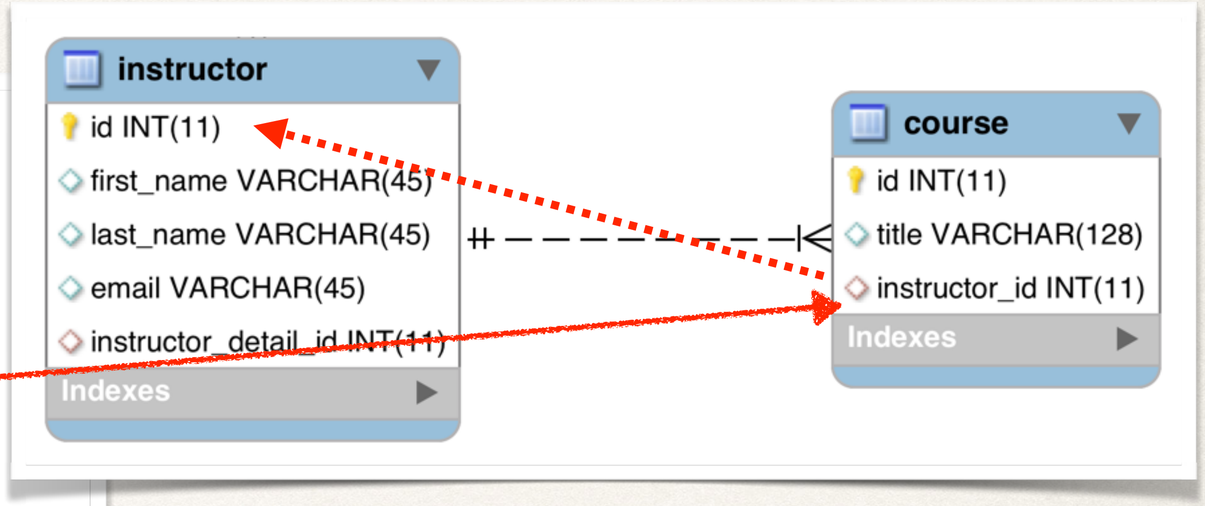
Step 2: Create Course class

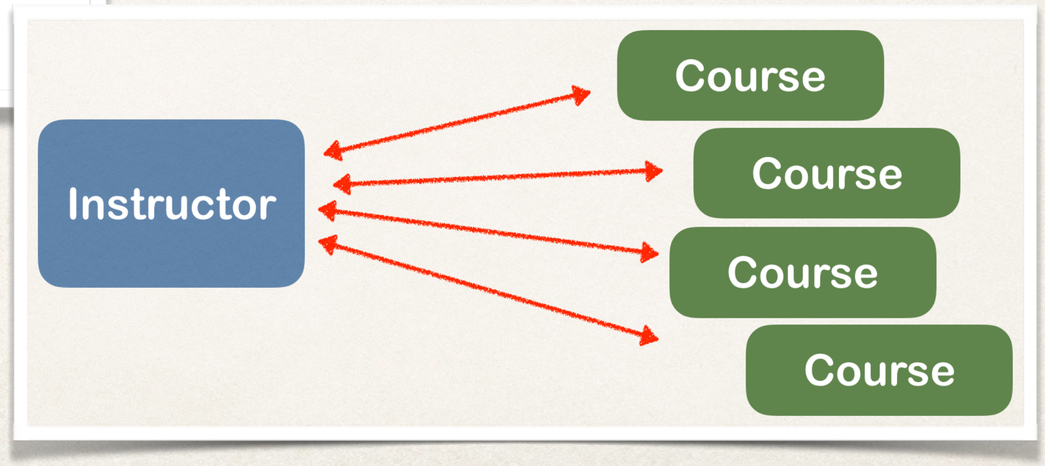
```
@Entity
@Table(name="course")
public class Course {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="title")
    private String title;

    ...
    // constructors, getters / setters
}
```



Step 2: Create Course class - @ManyToOne

```
@Entity
@Table(name="course")
public class Course {
    ...

    @ManyToOne
    @JoinColumn(name="instructor_id")
    private Instructor instructor;

    ...
    // constructors, getters / setters
}
```





Step 3: Update Instructor - reference courses

```
@Entity
@Table(name="instructor")
public class Instructor {
    ...

    private List<Course> courses;

    ...

}
```

Step 3: Update Instructor - reference courses

```
@Entity
@Table(name="instructor")
public class Instructor {
    ...

    private List<Course> courses;

    public List<Course> getCourses() {
        return courses;
    }

    public void setCourses(List<Course> courses) {
        this.courses = courses;
    }
    ...

}
```

Add @OneToMany annotation

```
@Entity
@Table(name="instructor")
public class Instructor {
    ...

    @OneToMany(mappedBy="instructor")
    private List<Course> courses;

    public List<Course> getCourses() {
        return courses;
    }

    public void setCourses(List<Course> courses) {
        this.courses = courses;
    }

    ...

}
```

Refers to “instructor” property in “Course” class

```
public class Instructor {
    ...

    @OneToMany(mappedBy="instructor")
    private List<Course> courses;
```

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

```
public class Course {
    ...

    @ManyToOne
    @JoinColumn(name="instructor_id")
    private Instructor instructor;
```

Add support for Cascading

```
@Entity
@Table(name="instructor")
public class Instructor {
    ...

    @OneToMany(mappedBy="instructor",
                    cascade={CascadeType.PERSIST, CascadeType.MERGE
                        CascadeType.DETACH, CascadeType.REFRESH})
    private List<Course> courses;



    ...

}
```

Do not apply cascading deletes!

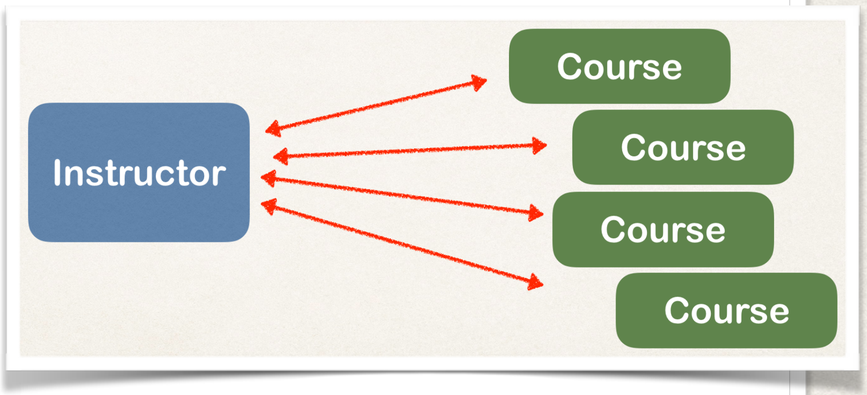
Add support for Cascading

```
@Entity
@Table(name="course")
public class Course {
    ...

    @ManyToOne(cascade={CascadeType.PERSIST, CascadeType.MERGE
                        CascadeType.DETACH, CascadeType.REFRESH})
    @JoinColumn(name="instructor_id")
    private Instructor instructor;

    ...
    // constructors, getters / setters
}
```

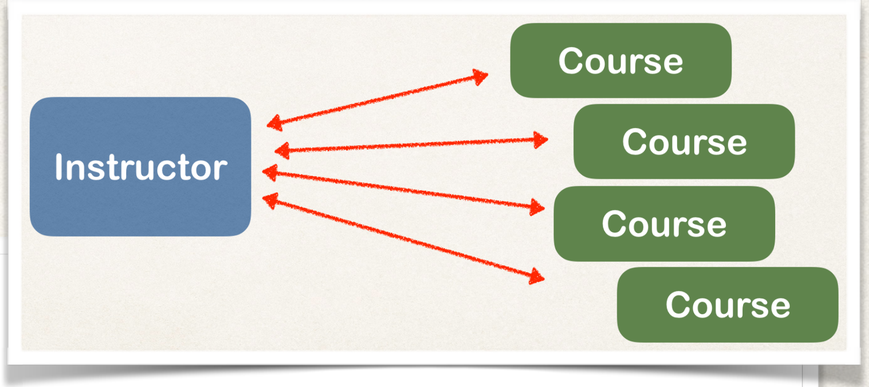
Do not apply cascading deletes!

Add convenience methods for bi-directional

```
@Entity
@Table(name="instructor")
public class Instructor {
    ...
    // add convenience methods for bi-directional relationship

    public void add(Course tempCourse) {

        if (courses == null) {
            courses = new ArrayList<>();
        }

        courses.add(tempCourse);

        tempCourse.setInstructor(this);
    }
    ...
}
```



Step 2: Create Main App




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

        // get the instructor object
        int theId = 1;
        Instructor tempInstructor = session.get(Instructor.class, theId);

        // print instructor
        System.out.println("tempInstructor: " + tempInstructor);

        // print the associated courses
        System.out.println("courses: " + tempInstructor.getCourses());
        ...
}
```