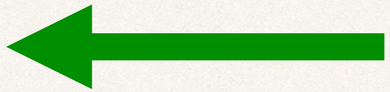
# Inheritance: Joined Table

# Inheritance Mapping Strategies

- Single table
- Table per class
- **Joined table**
- Mapped superclass

# Inheritance Mapping Strategies

- Single table
- Table per class
- **Joined table** 
- Mapped superclass

# Joined Tables

# Joined Tables

- For the inheritance tree, all classes are mapped to a table

# Joined Tables

- For the inheritance tree, all classes are mapped to a table
- Superclass table contains fields common to all subclasses

# Joined Tables

- For the inheritance tree, all classes are mapped to a table
- Superclass table contains fields common to all subclasses
- Subclass tables contain only fields specific to the subclass

# Joined Tables

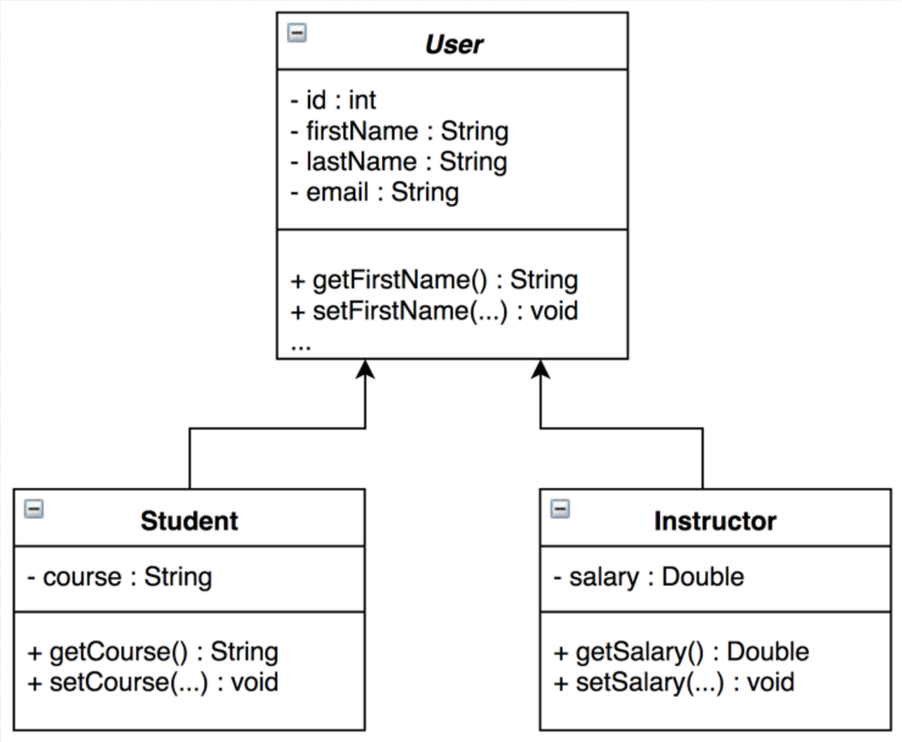
- For the inheritance tree, all classes are mapped to a table
- Superclass table contains fields common to all subclasses
- Subclass tables contain only fields specific to the subclass
- Inheritance is modeled with a foreign key

# Joined Tables

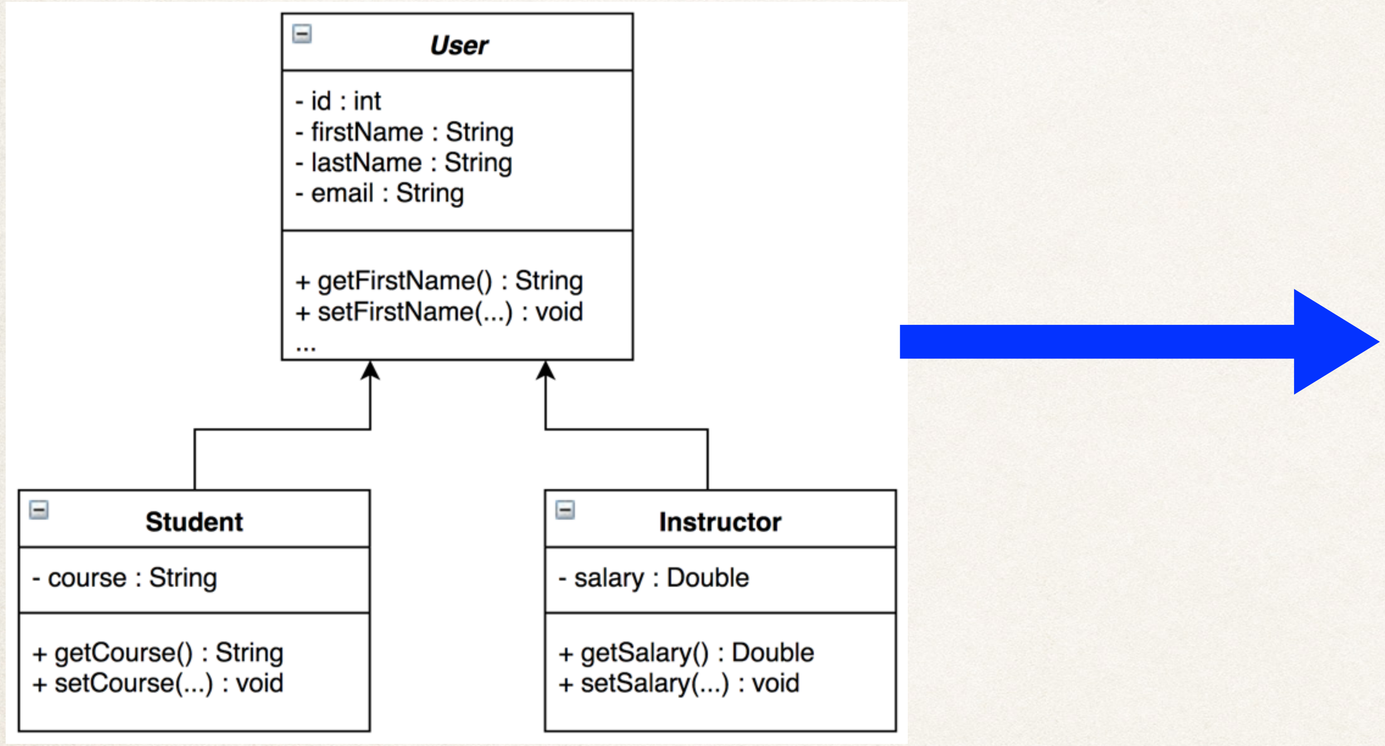
- For the inheritance tree, all classes are mapped to a table
- Superclass table contains fields common to all subclasses
- Subclass tables contain only fields specific to the subclass
- Inheritance is modeled with a foreign key
- Hibernate will join the data based on primary key and foreign key

# Joined Tables

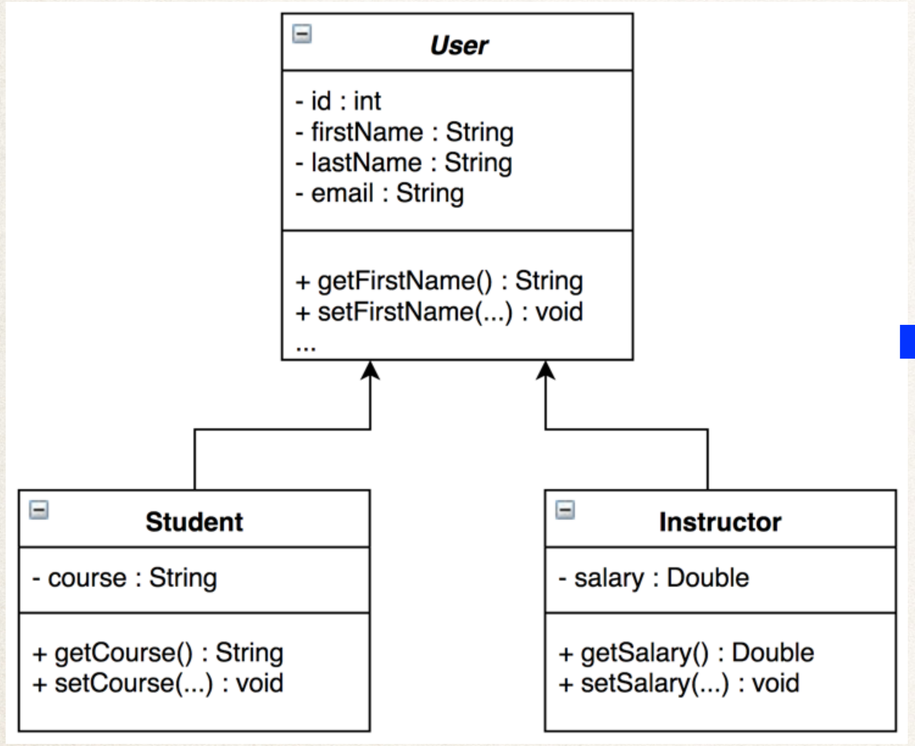
# Joined Tables

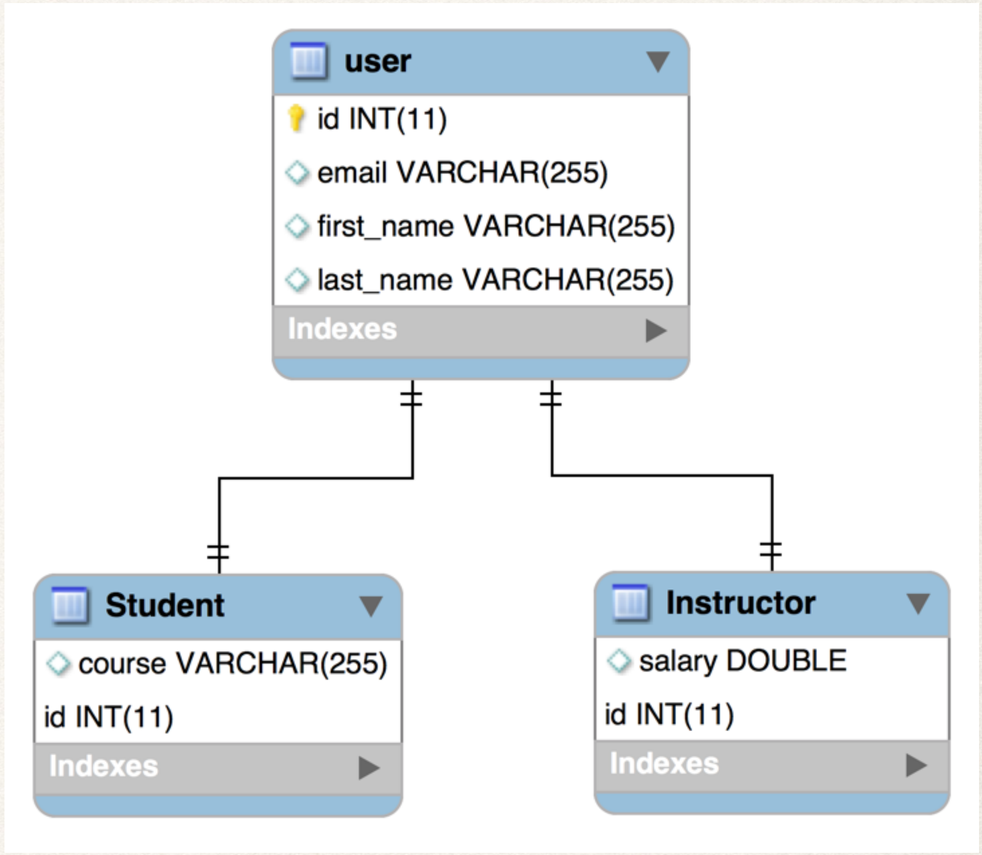


# Joined Tables



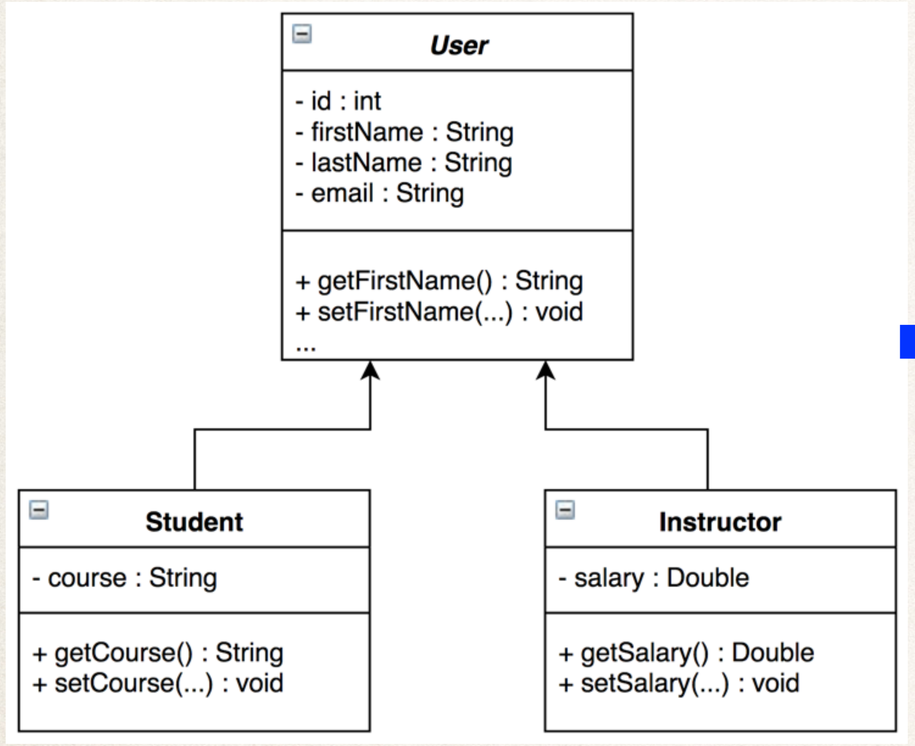
# Joined Tables

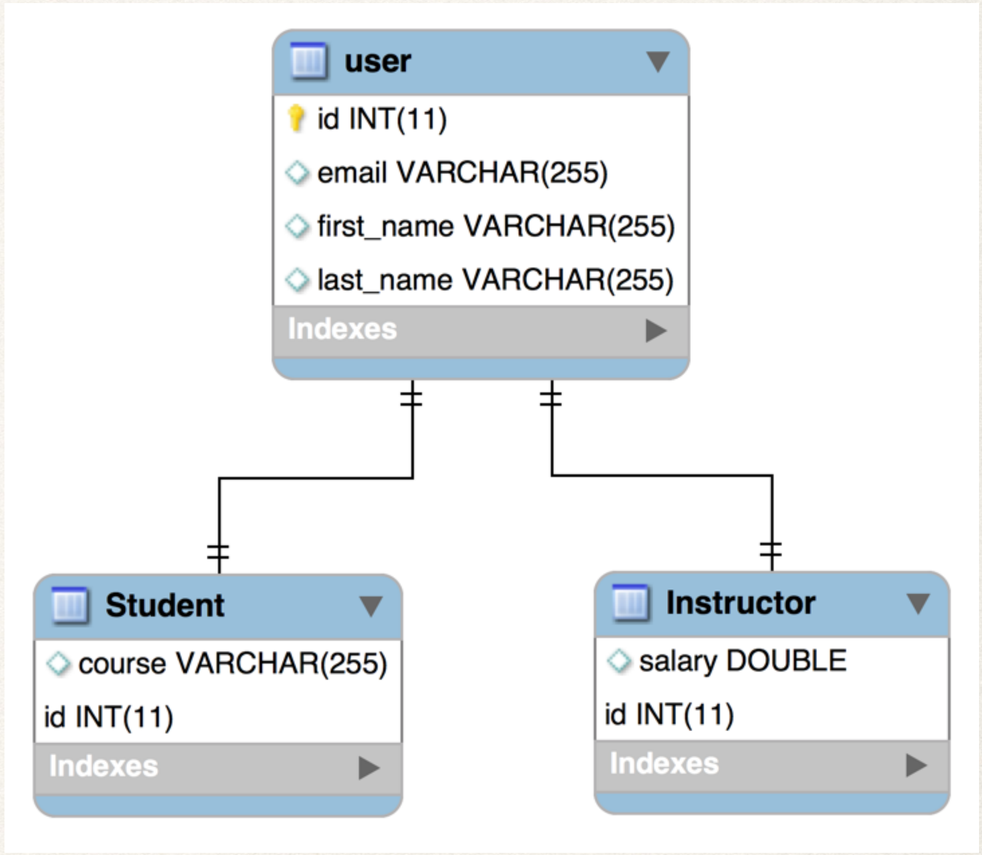


# Joined Tables

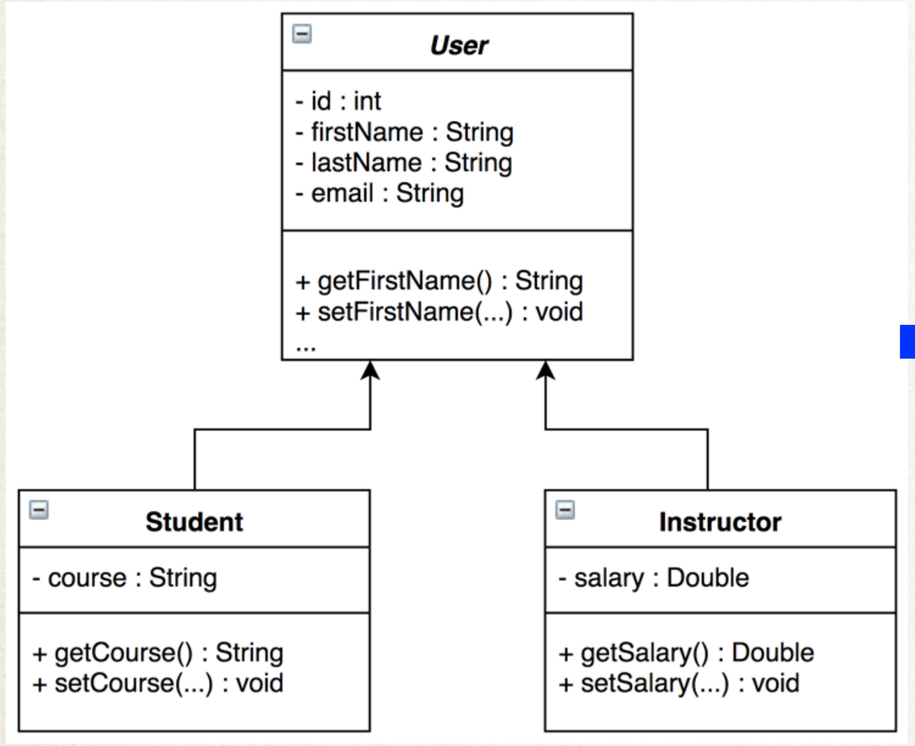
**Inheritance is modeled by joining on primary and foreign keys**

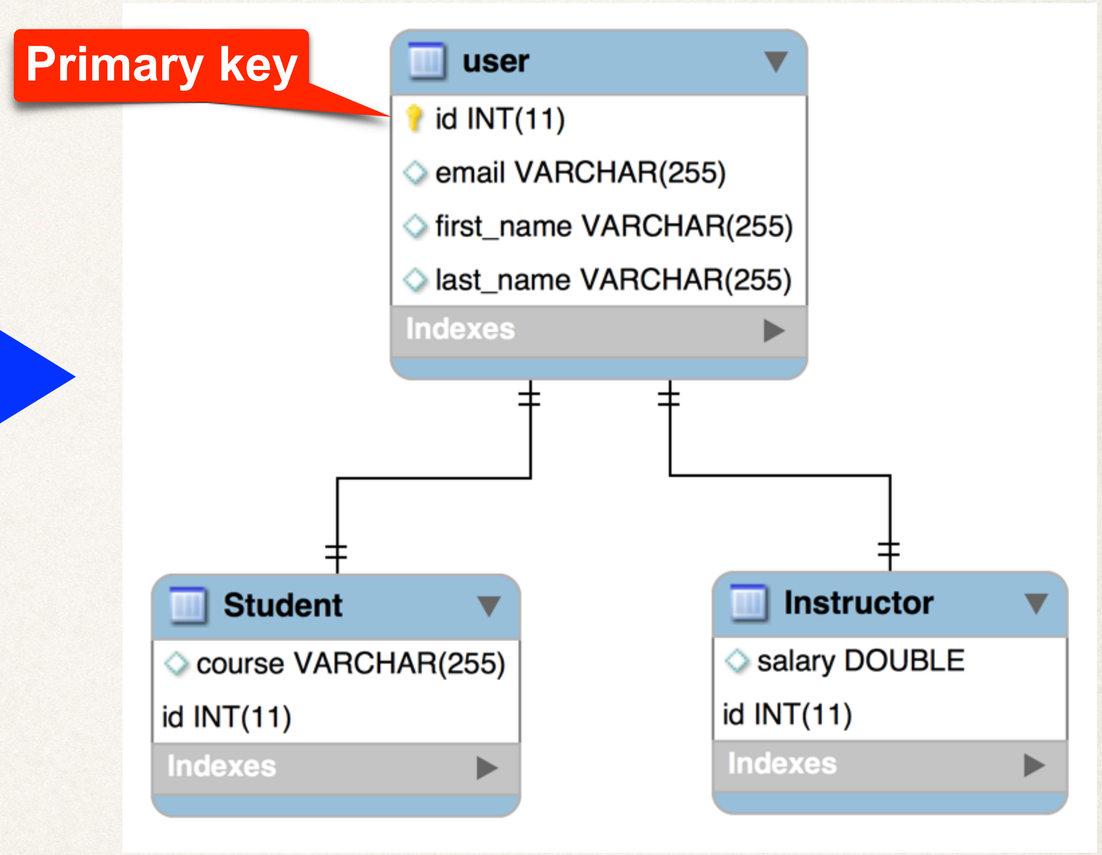


# Joined Tables

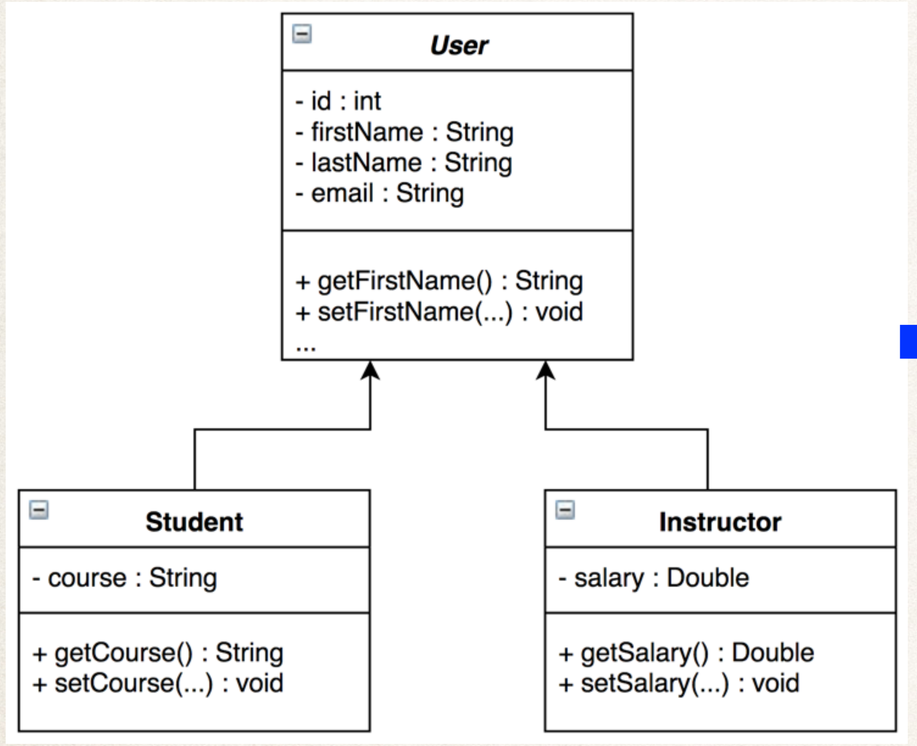
**Inheritance is modeled by joining on primary and foreign keys**

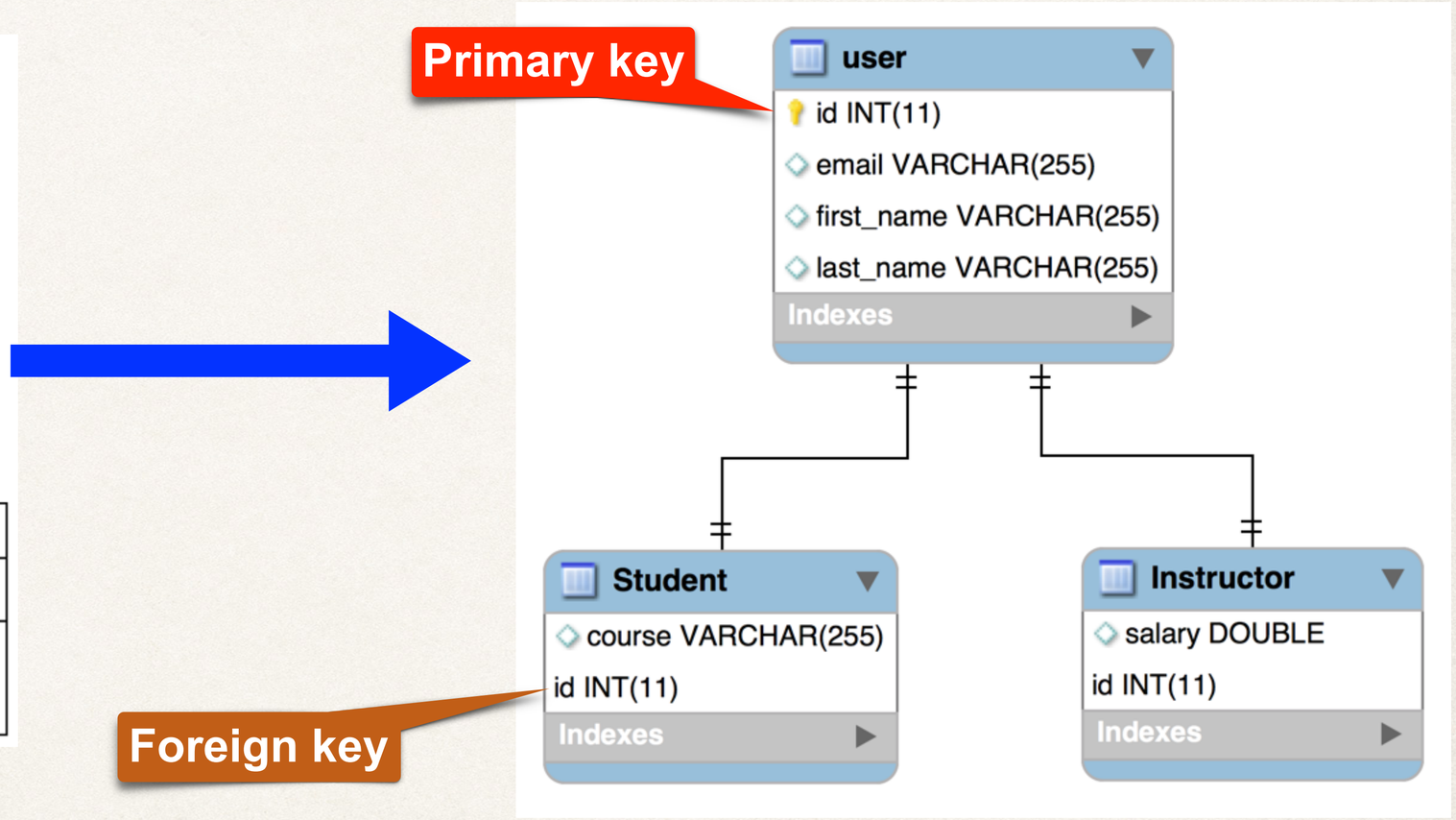


# Joined Tables

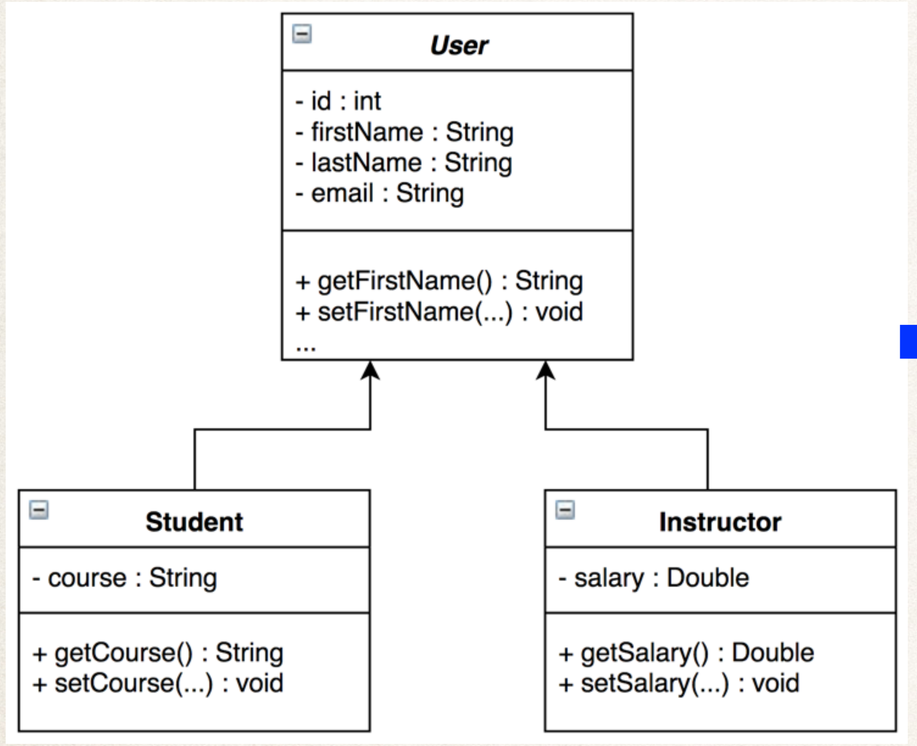
**Inheritance is modeled by joining on primary and foreign keys**

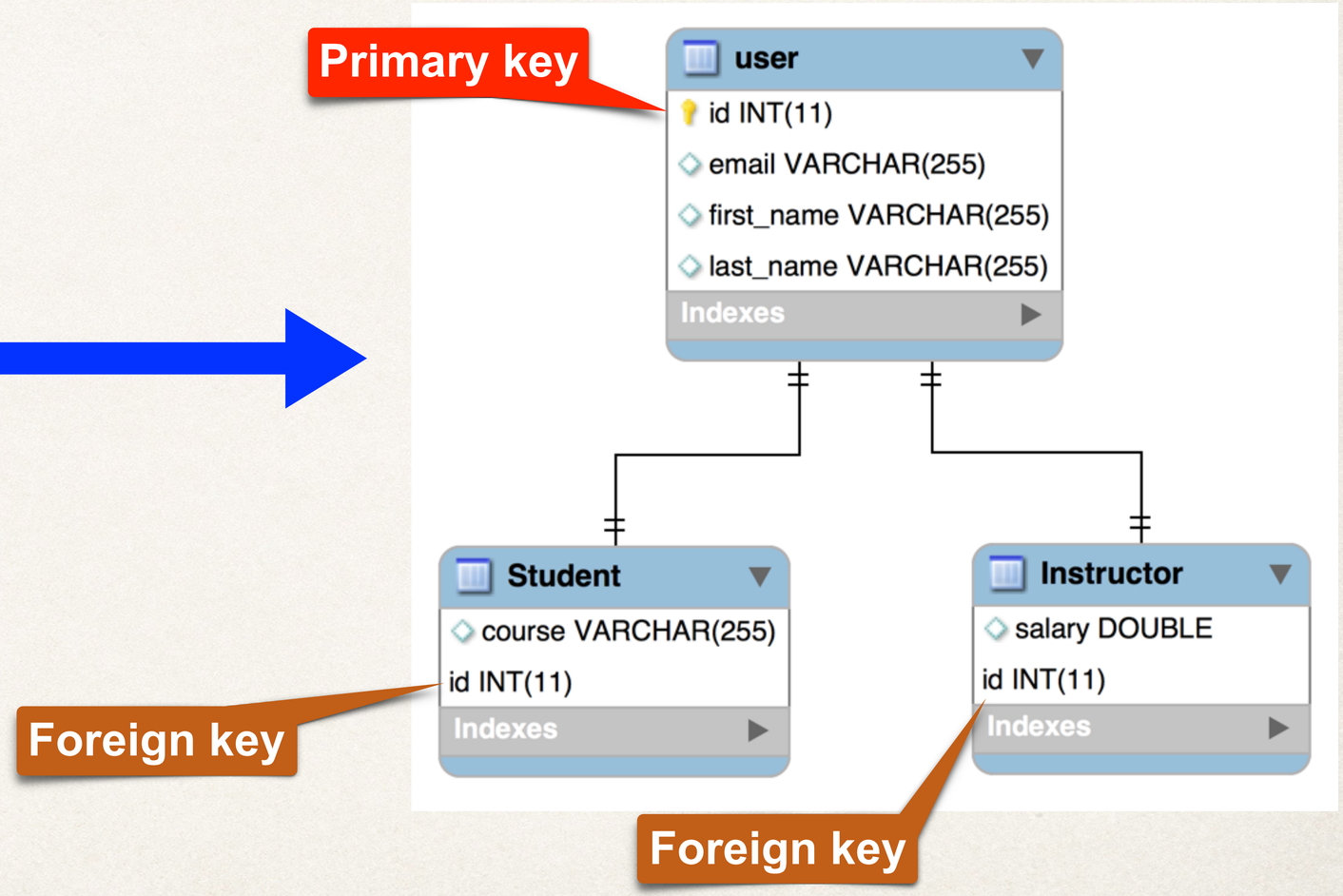


# Joined Tables

**Inheritance is modeled by joining on primary and foreign keys**



# Development Process

Step-By-Step

# Development Process

Step-By-Step

1. In superclass, specify inheritance strategy: `JOINED`

# Development Process

Step-By-Step

1. In superclass, specify inheritance strategy: `JOINED`

2. In superclass, update the ID generation strategy to `IDENTITY`

# Development Process

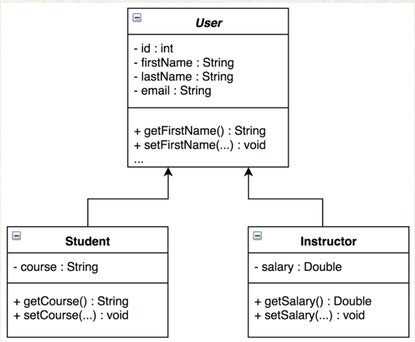
Step-By-Step

1. In superclass, specify inheritance strategy: `JOINED`

2. In superclass, update the ID generation strategy to `IDENTITY`

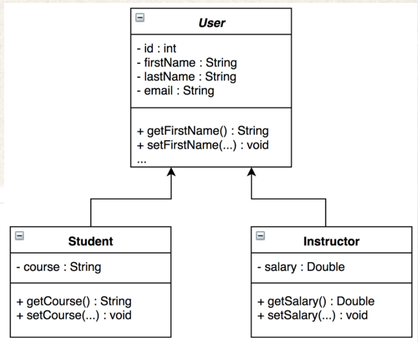
3. Develop main application

# Step 1: Superclass - Inheritance strategy



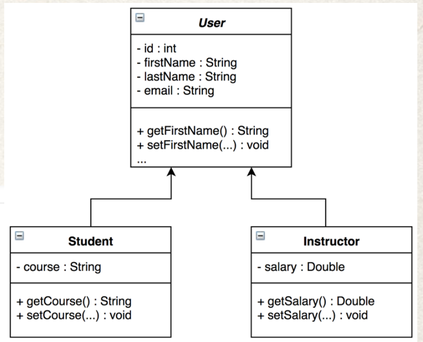
# Step 1: Superclass - Inheritance strategy

```
@Entity
@Table(name="user")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class User {

...

}
```



# Step 1: Superclass - Inheritance strategy

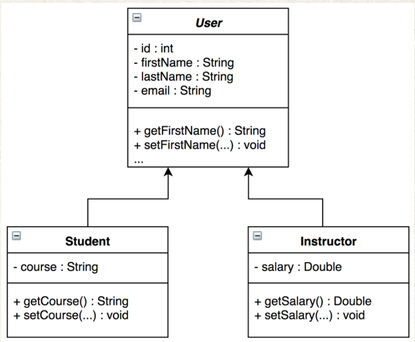
```
@Entity
@Table(name="user")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class User {

...

}
```




**All classes are mapped to tables**

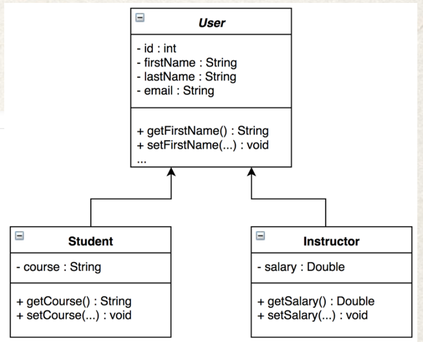
**Inheritance represented with joins on primary and foreign key**

# Step 2: Update ID generation strategy



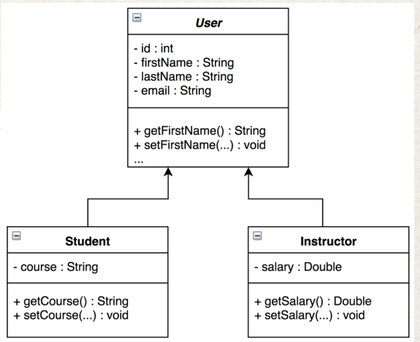
# Step 2: Update ID generation strategy

```
@Entity
@Table(name="user")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class User {

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

}
```



# Step 2: Update ID generation strategy

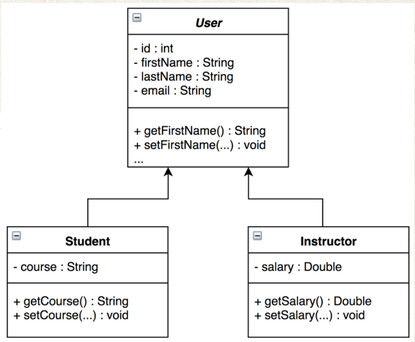
```
@Entity
@Table(name="user")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class User {

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

}
```




**Leverage "auto-increment" feature of the database**

# Step 2: Update ID generation strategy

```
@Entity
@Table(name="user")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class User {

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

}
```




**Leverage "auto-increment" feature of the database**

**Superclass has the id field defined**

**ID primary key is only in one table (no need for sequence table)**

# Step 3: Develop the main application

# Step 3: Develop the main application

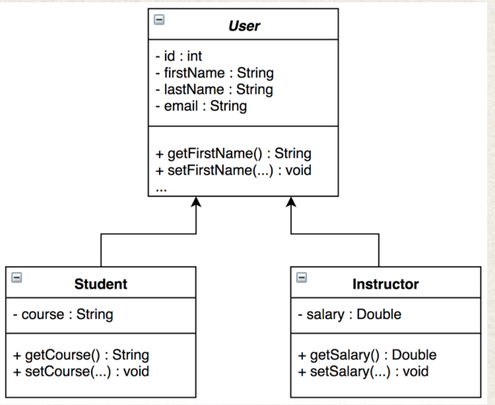
Student's course

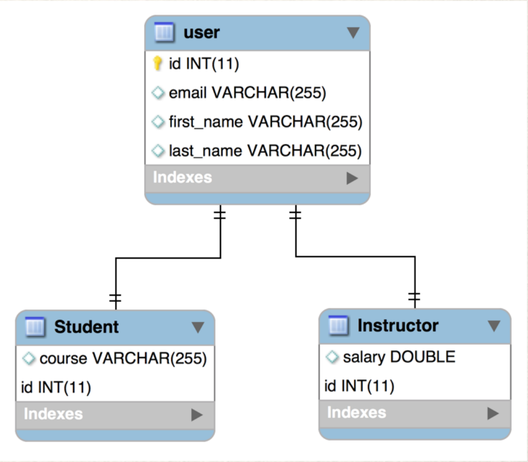
```
// create the objects
Student tempStudent = new Student("Mary", "Public", "mary@luv2code.com", "Hibernate");
Instructor tempInstructor = new Instructor("John", "Doe", "john@luv2code.com", 20000.00);

// start a transaction
session.beginTransaction();

// save the objects
System.out.println("Saving the student and instructor...");
session.save(tempStudent);
session.save(tempInstructor);

// commit the transaction
session.getTransaction().commit();
```

Instructor's salary

All of this code is the same

# Run the App

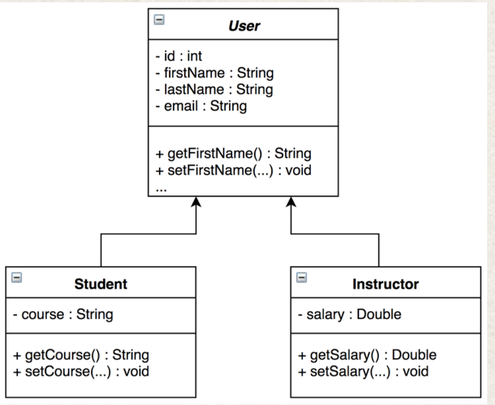


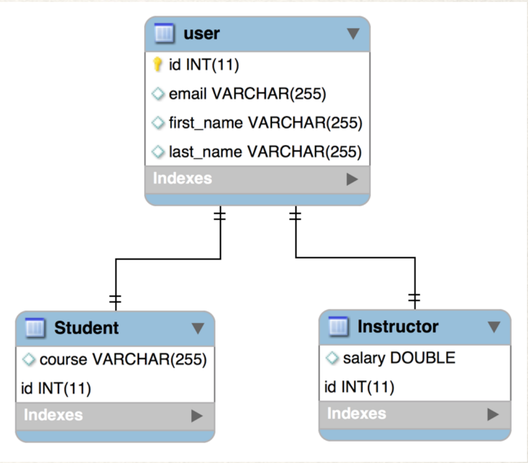



# Run the App

***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```



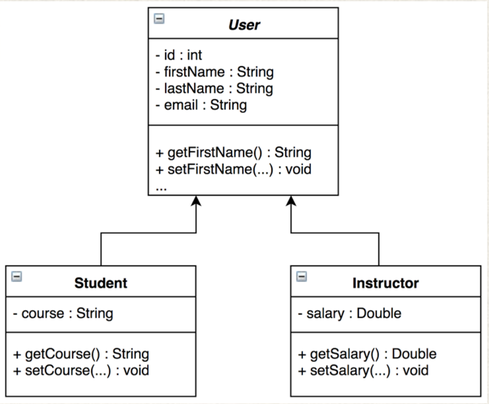
# Run the App

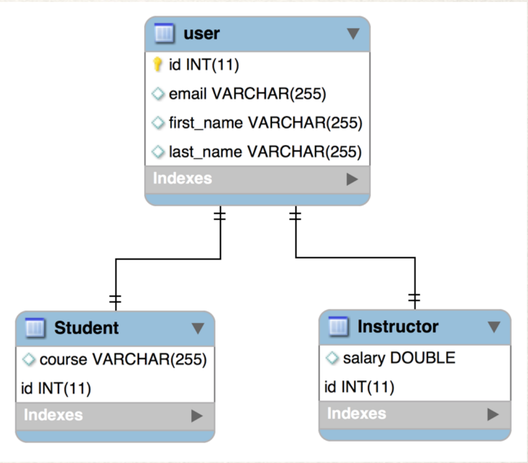
***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

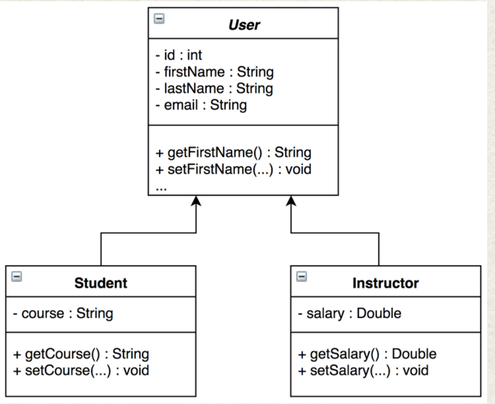
***Table: user***

| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |



# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

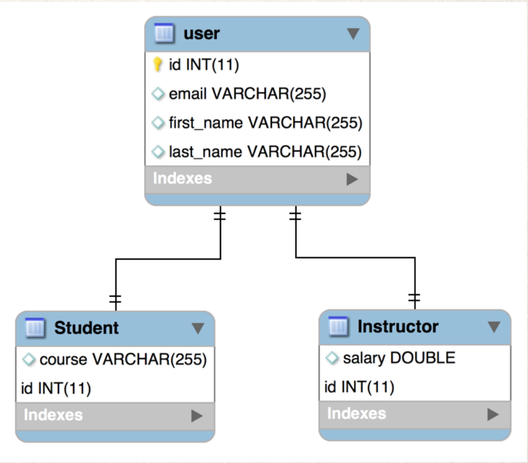
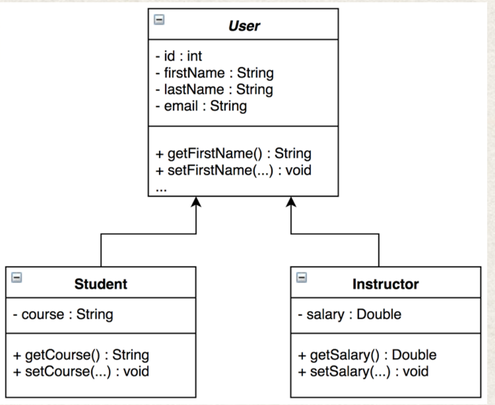



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

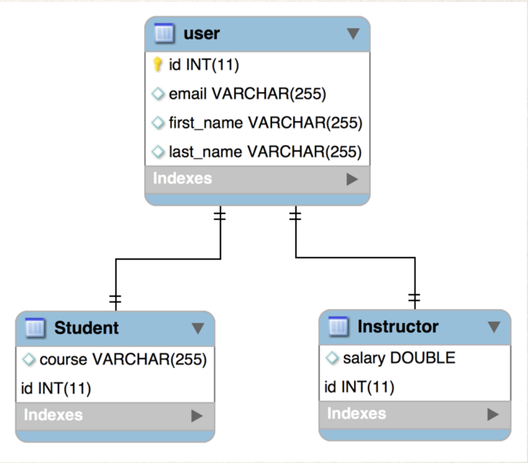



***Table: user***

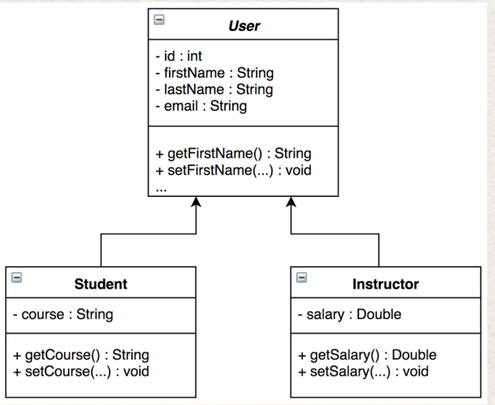
Primary key

| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |

***Table: Student***

| course | id |
|---|---|
| Hibernate | 1 |

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

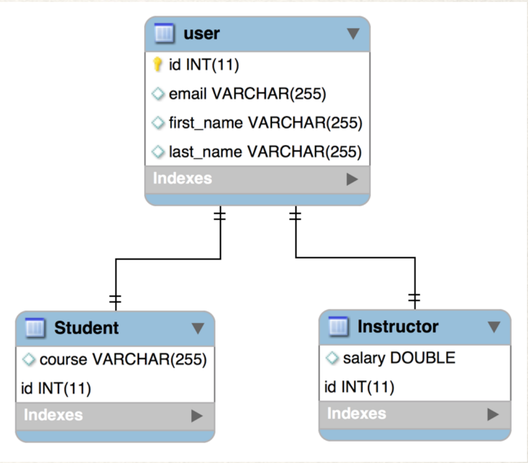



***Table: user***

Primary key

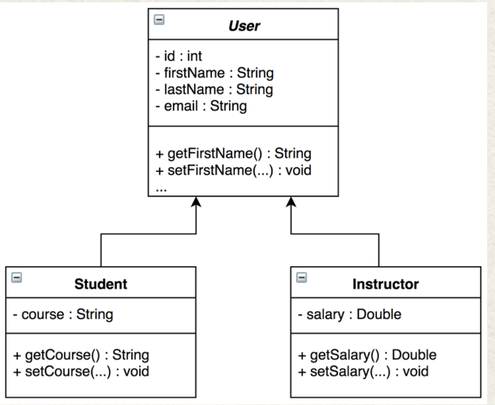
| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |

***Table: Student***

| course | id |
|---|---|
| Hibernate | 1 |

Foreign key

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

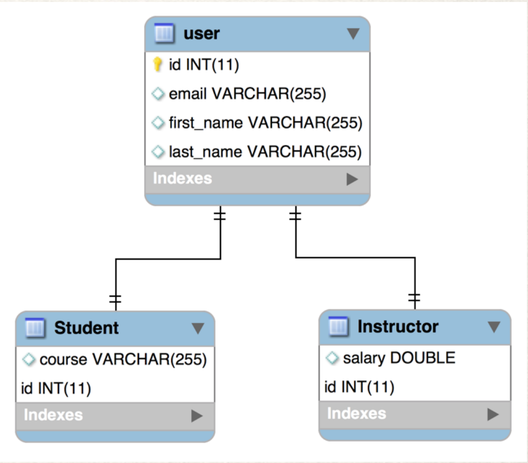



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |

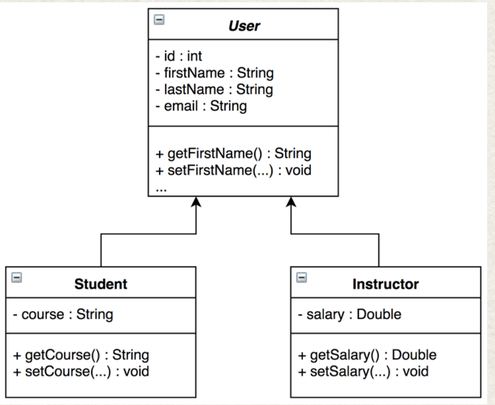
***Join on primary key and foreign key***

***Table: Student***

| course | id |
|---|---|
| Hibernate | 1 |

**Foreign key**

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Student (course, id) values (?, ?)
```

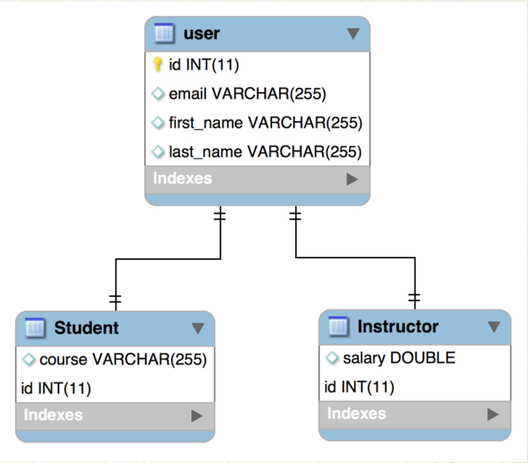



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 1 | mary@luv2code.com | Mary | Public |

***Join on primary key and foreign key***

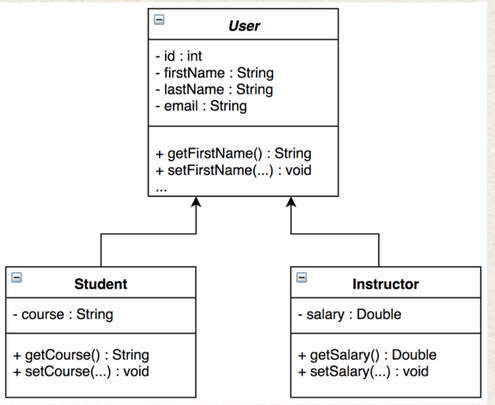
***Table: Student***

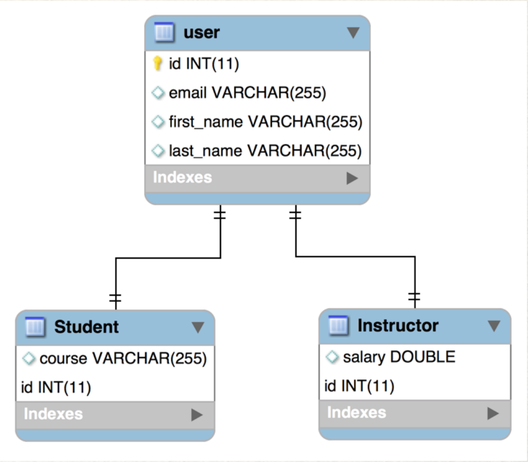
| course | id |
|---|---|
| Hibernate | 1 |

**Foreign key**

**Behind the scenes,**
**Hibernate handles this for you**

# Run the App

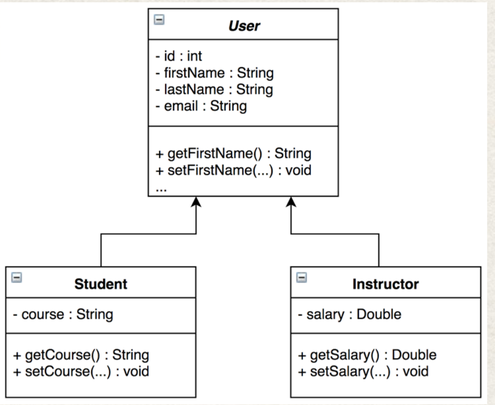


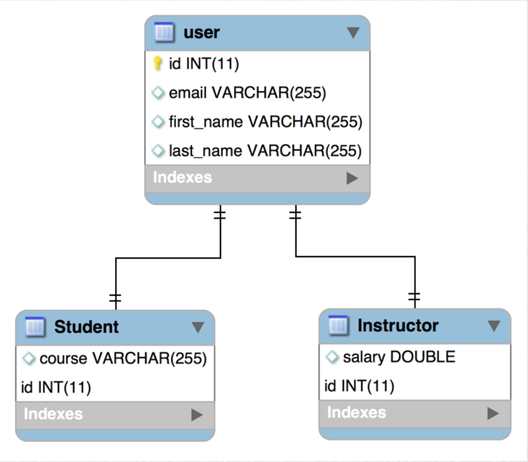



# Run the App

***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```



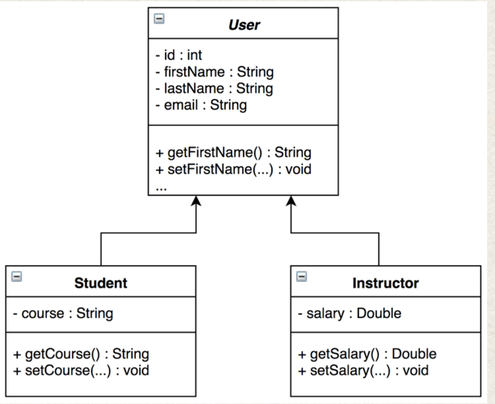
# Run the App

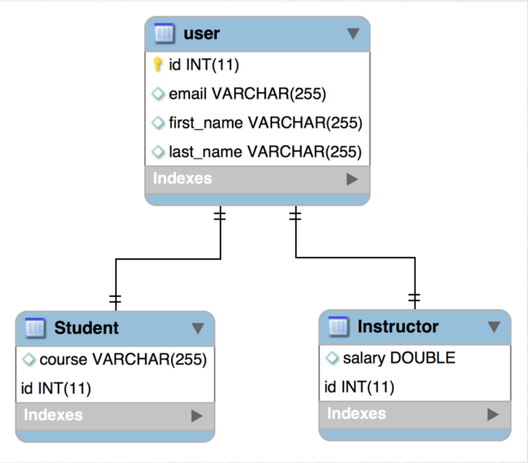
***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

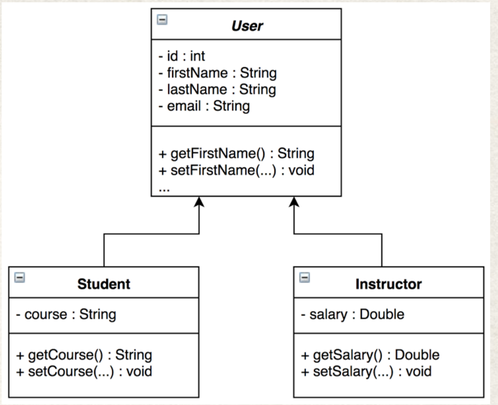
***Table: user***

| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |



# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

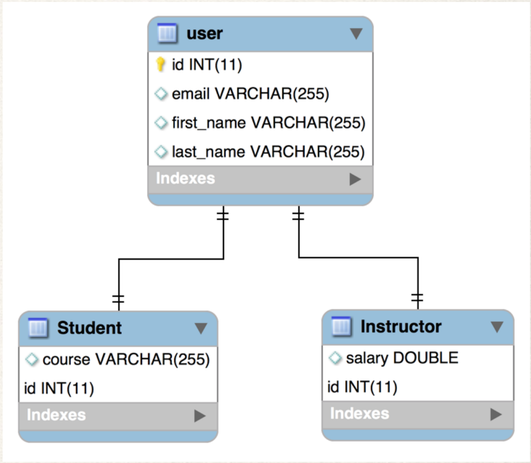
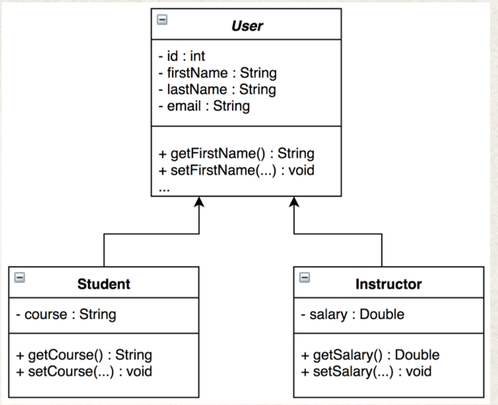



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

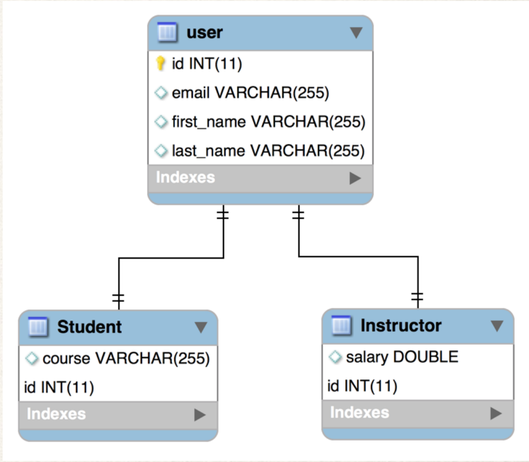



***Table: user***

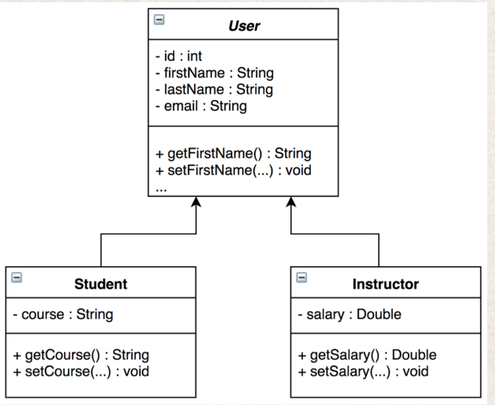
Primary key

| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |

***Table: Instructor***

| salary | id |
|---|---|
| 20000 | 2 |

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

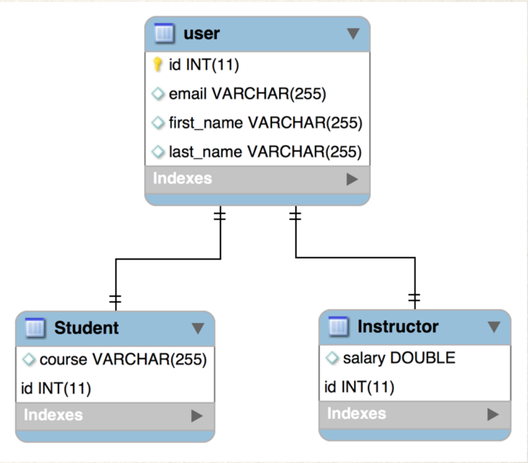



***Table: user***

**Primary key**

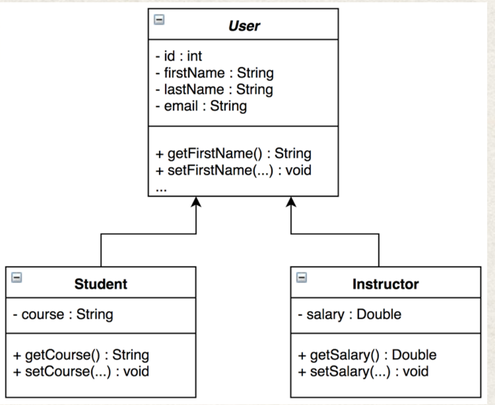
| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |

***Table: Instructor***

| salary | id |
|---|---|
| 20000 | 2 |

**Foreign key**

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

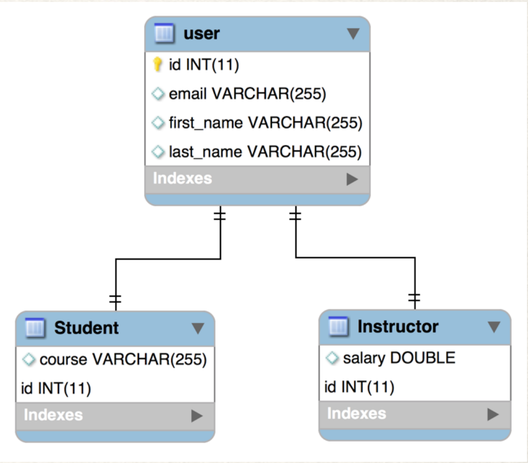



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |

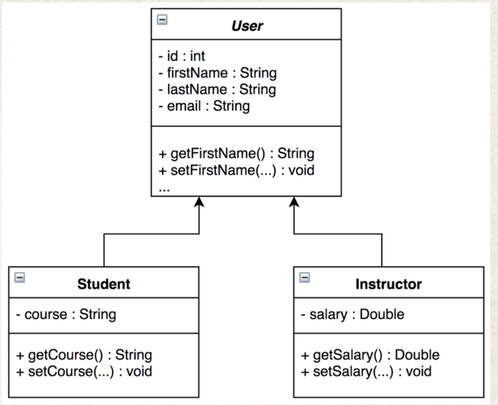
***Join on primary key and foreign key***

***Table: Instructor***

| salary | id |
|---|---|
| 20000 | 2 |

**Foreign key**

# Run the App




***Console output***

```
Hibernate: insert into user (email, first_name, last_name) values (?, ?, ?)
Hibernate: insert into Instructor (salary, id) values (?, ?)
```

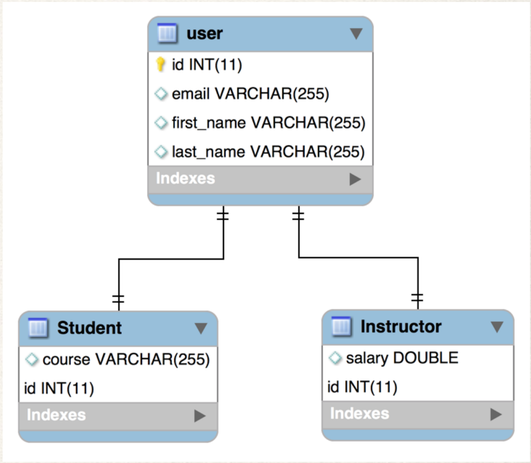



***Table: user***

**Primary key**

| id | email | first_name | last_name |
|---|---|---|---|
| 2 | john@luv2code.com | John | Doe |

***Join on primary key and foreign key***

***Table: Instructor***

| salary | id |
|---|---|
| 20000 | 2 |

**Foreign key**

**Behind the scenes,**
**Hibernate handles this for you**

# Joined Tables

# Joined Tables

- Advantages

# Joined Tables

- Advantages
  - Normalized database model

# Joined Tables

- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields

# Joined Tables

- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields
- Disadvantages

# Joined Tables

- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields

- Disadvantages
  - Slow performance for queries on subclasses (results in many joins)

# Joined Tables

- Advantages
  - Normalized database model
  - No duplicate mapping of inherited fields

- Disadvantages
  - Slow performance for queries on subclasses (results in many joins)
  - Query performance slows down for very deep inheritance trees