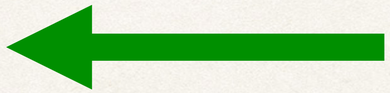
luv2code

Inheritance: Mapped Superclass

Inheritance Mapping Strategies

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

Inheritance Mapping Strategies

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

Mapped Superclass

Mapped Superclass

- For the inheritance tree, subclass is mapped to a table

Mapped Superclass

- For the inheritance tree, subclass is mapped to a table
- Each table has the inherited fields and fields defined in the subclass

Mapped Superclass

- For the inheritance tree, subclass is mapped to a table
- Each table has the inherited fields and fields defined in the subclass

- The mapped superclass is no longer a separate table/entity (no @Entity)

Mapped Superclass

- For the inheritance tree, subclass is mapped to a table
- Each table has the inherited fields and fields defined in the subclass

- The mapped superclass is no longer a separate table/entity (no @Entity)

- Only the subclasses will use @Entity

Mapped Superclass

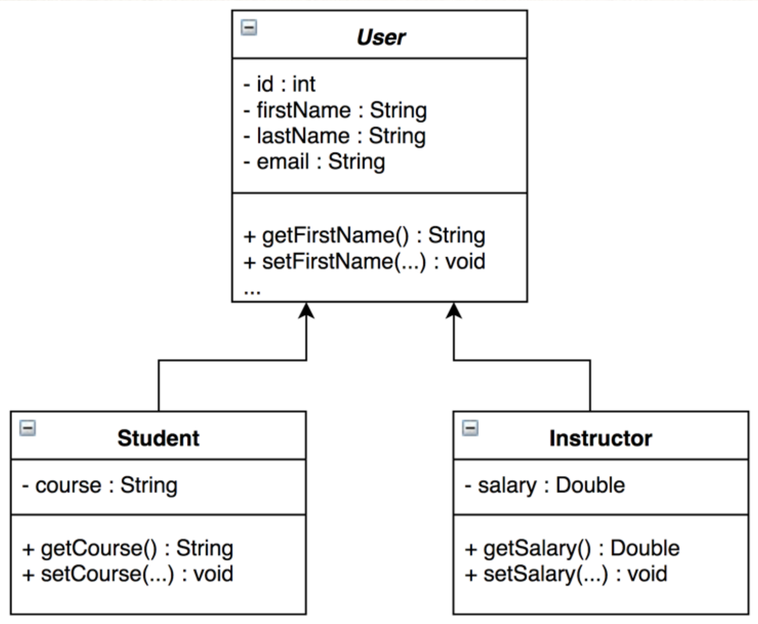
- For the inheritance tree, subclass is mapped to a table
- Each table has the inherited fields and fields defined in the subclass

- The mapped superclass is no longer a separate table/entity (no @Entity)

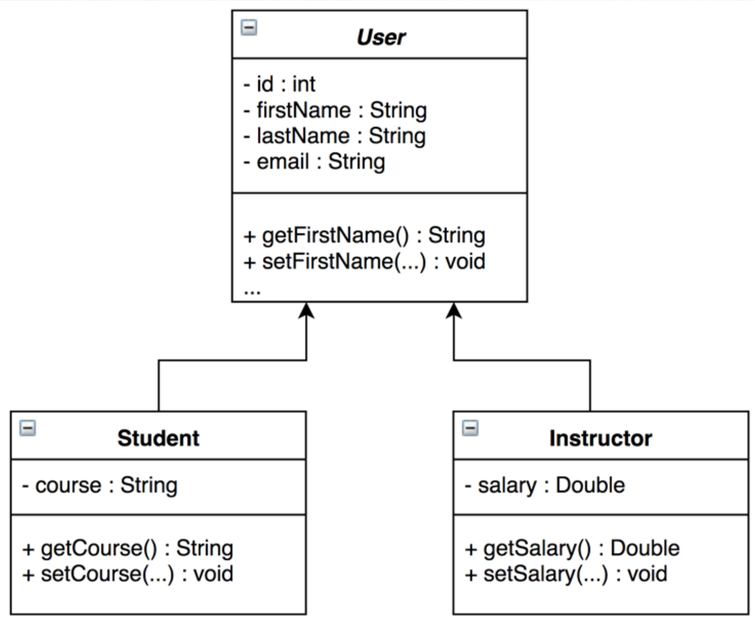
- Only the subclasses will use @Entity
- Only the subclasses will have tables in the database

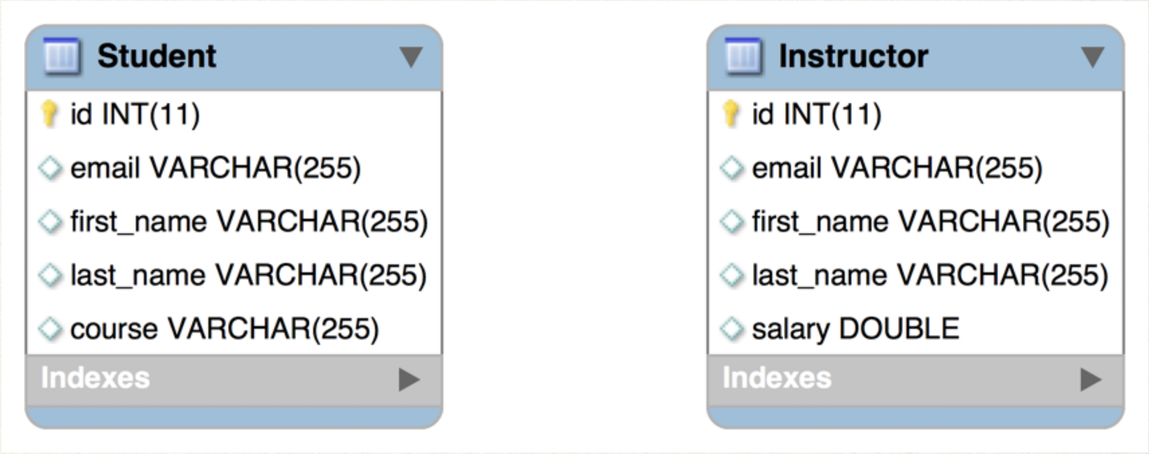
Mapped Superclass

Mapped Superclass

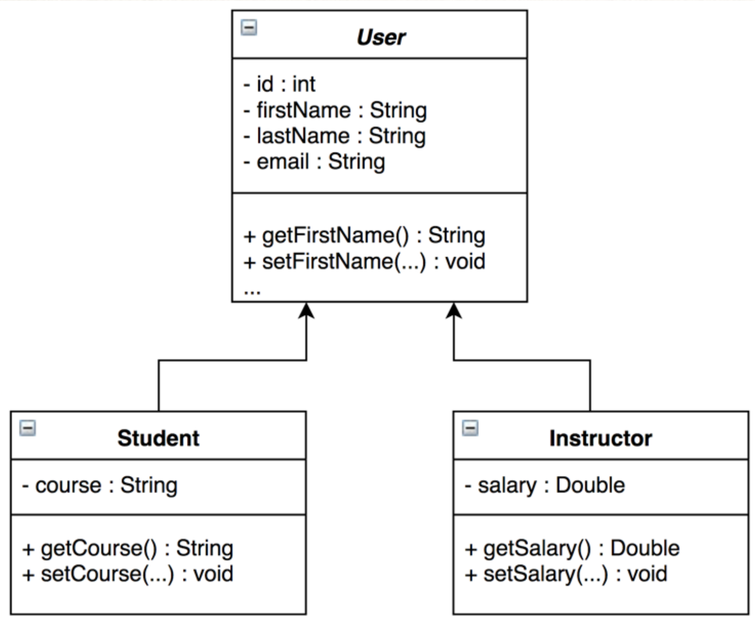


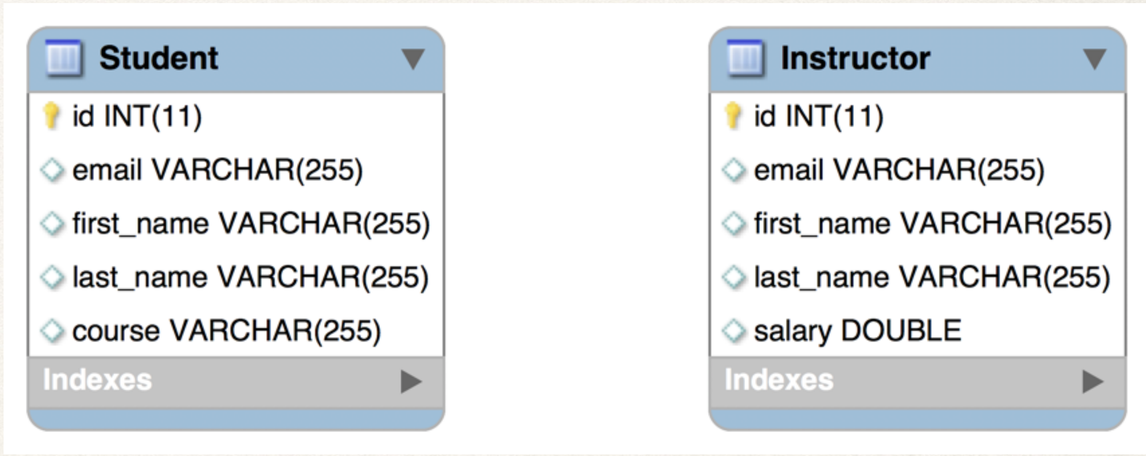
Mapped Superclass





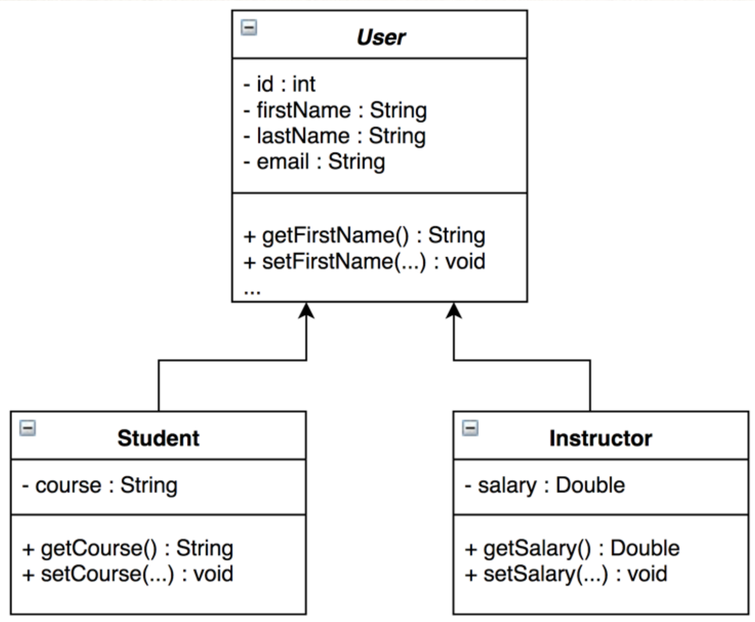
Mapped Superclass



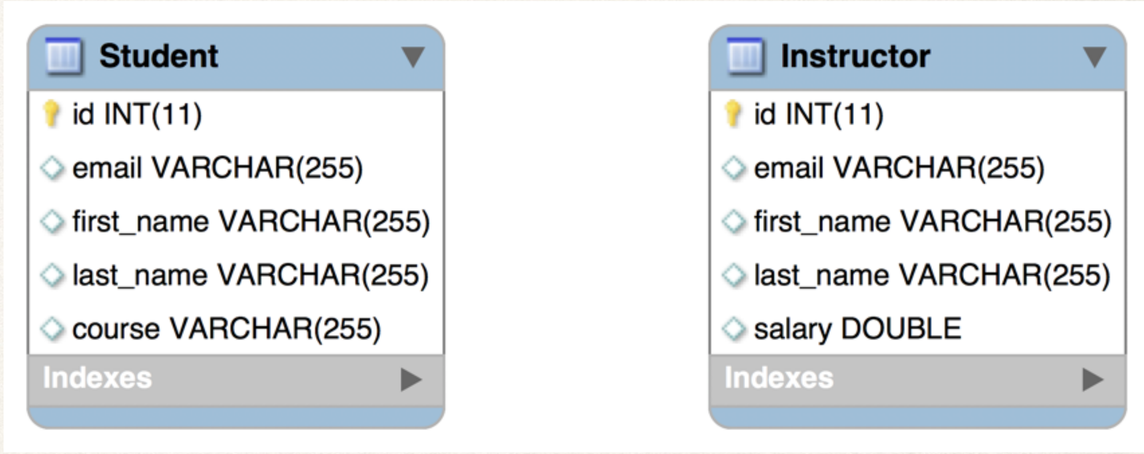


Only subclasses have tables in the database

Mapped Superclass



Note: No table for superclass

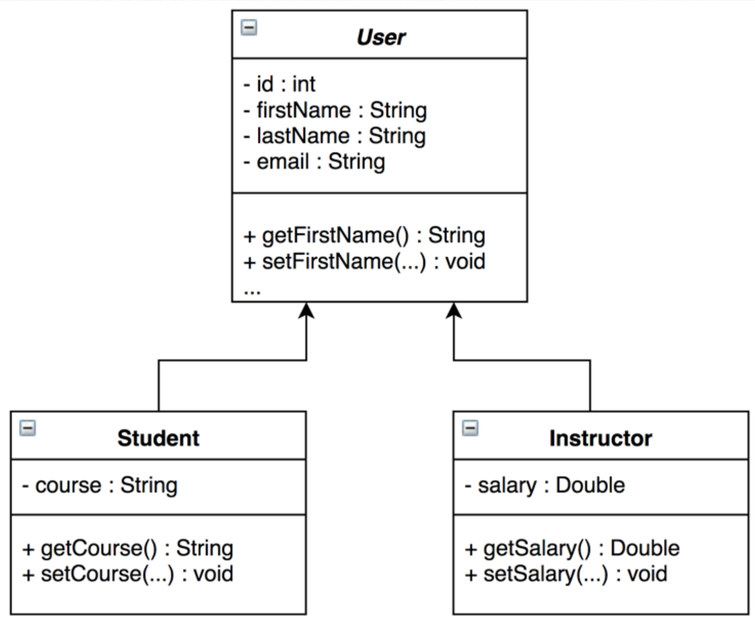


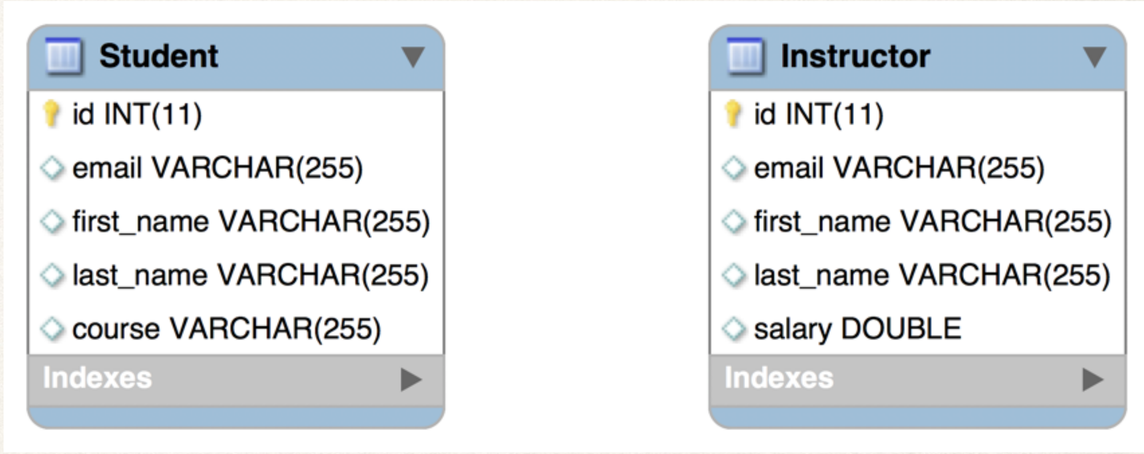
Only subclasses have tables in the database

Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields

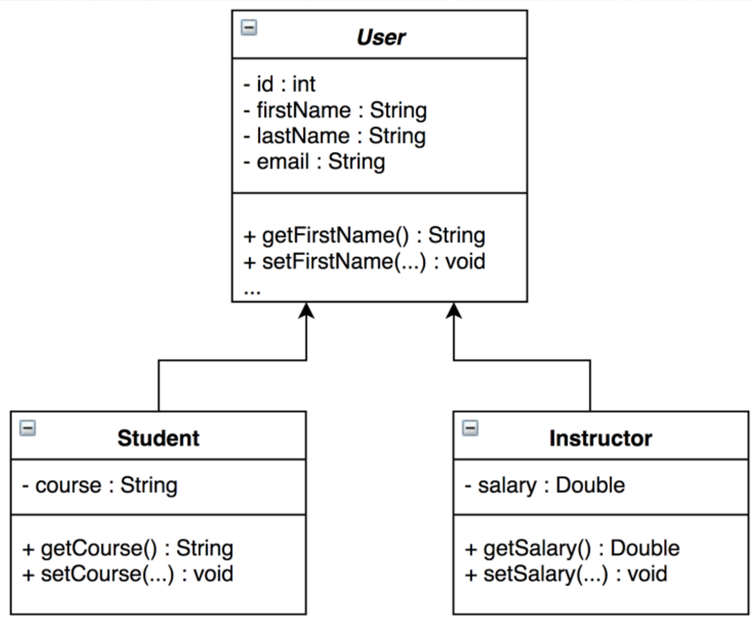


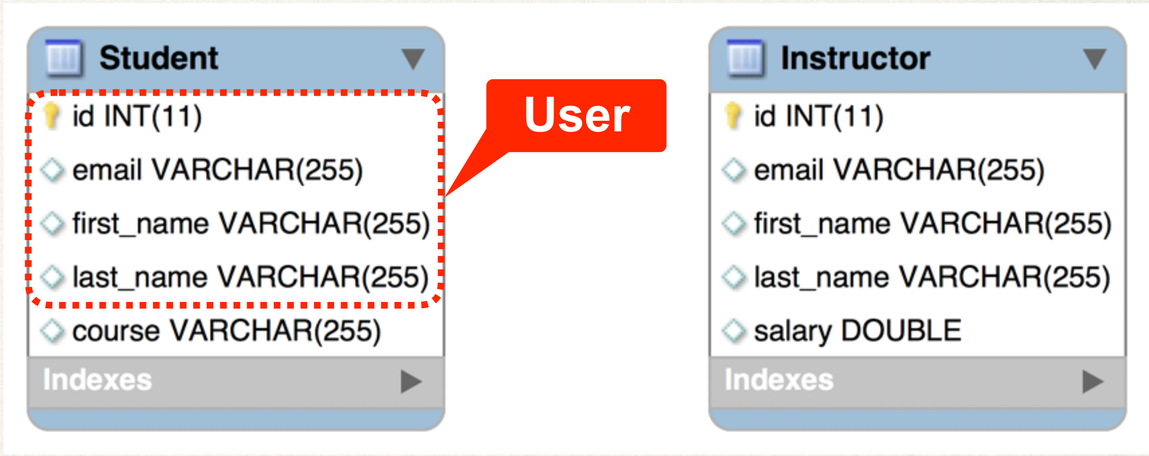


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields

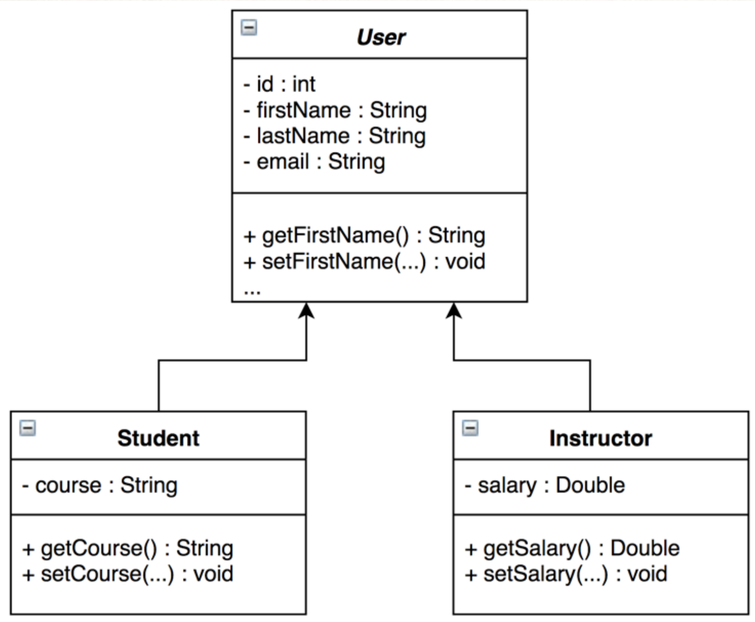


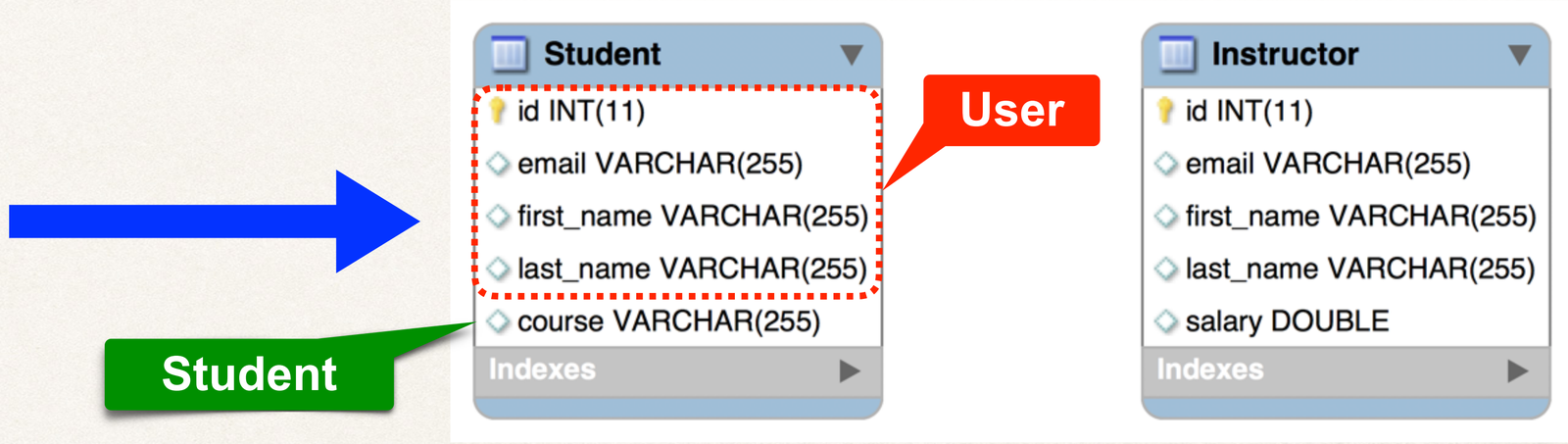


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields

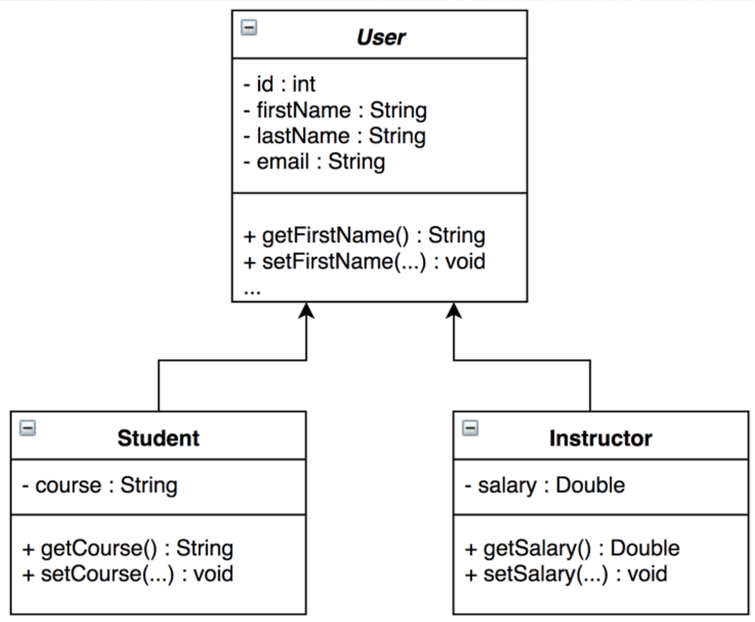


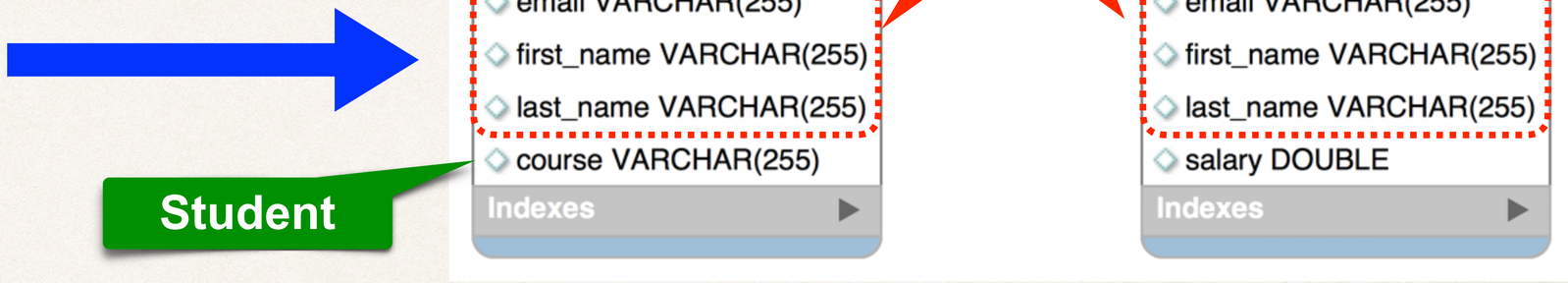


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields

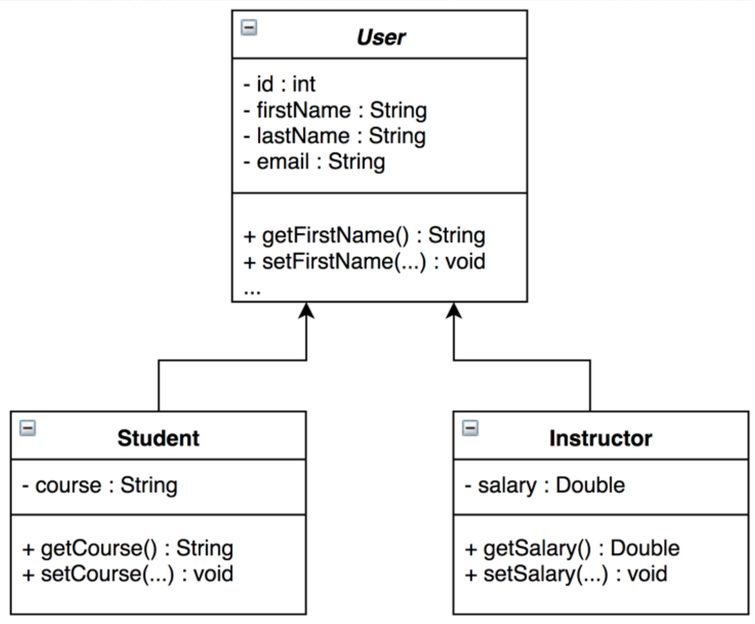


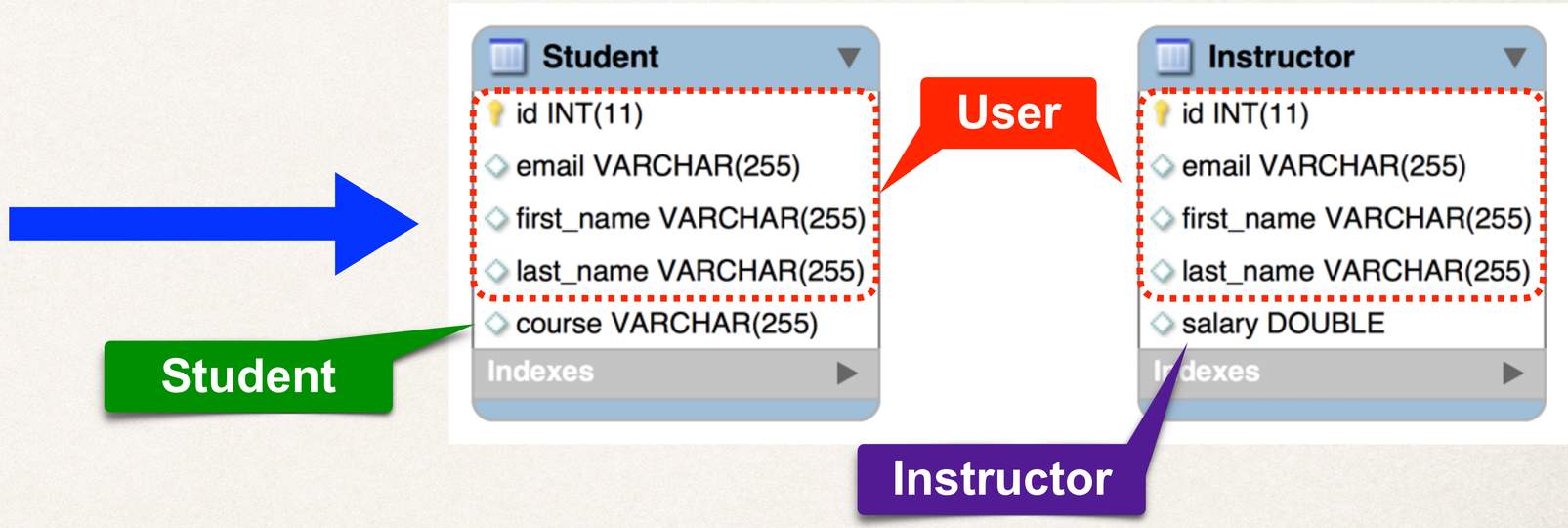


Mapped Superclass

Superclass defines common fields

Subclass table contains superclass fields and subclass fields





Mapped Superclass … looks familiar …

Mapped Superclass … looks familiar …

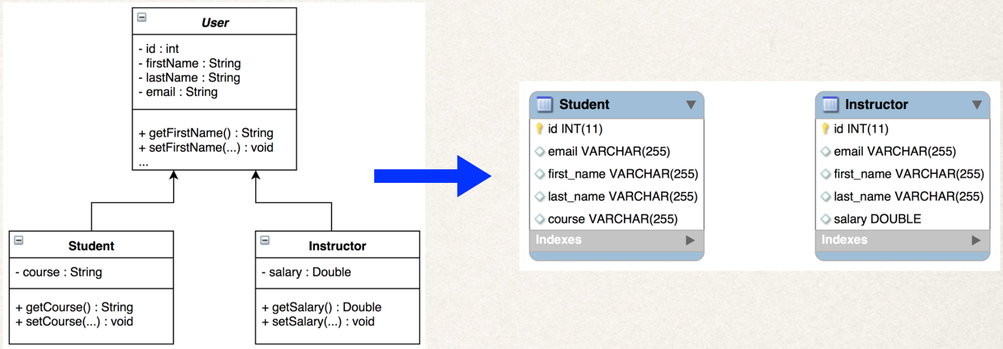
- Similar to **Table per Class** strategy but …

Mapped Superclass … looks familiar …

- Similar to **Table per Class** strategy but …
- **Mapped Superclass**

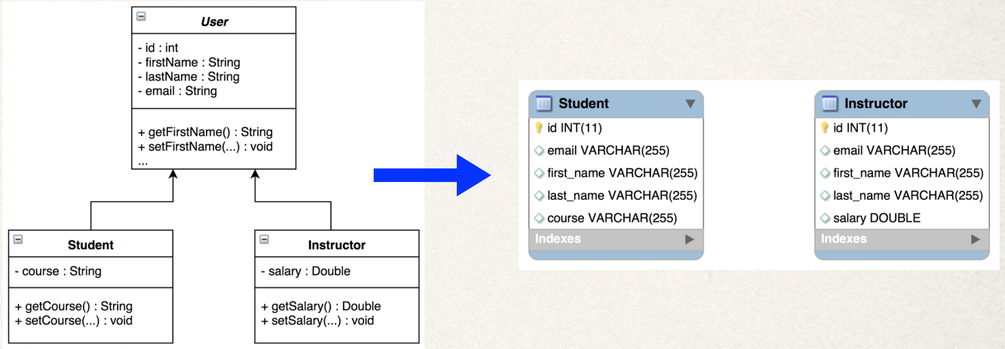
Mapped Superclass … looks familiar …

- Similar to **Table per Class** strategy but …
- **Mapped Superclass**
  - No table for superclass



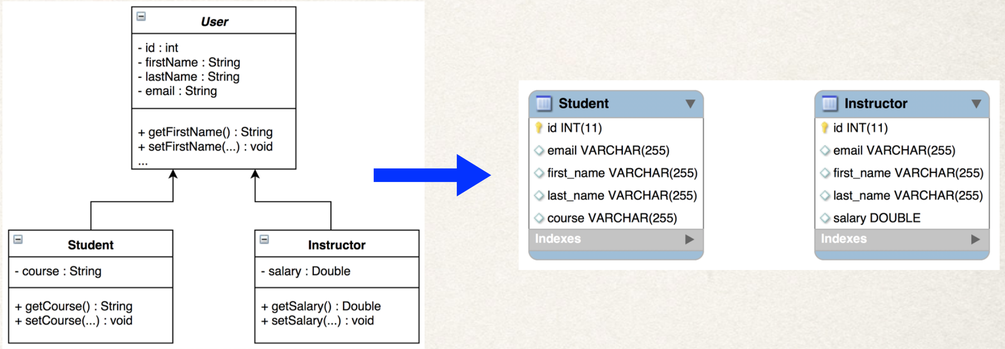
Mapped Superclass … looks familiar …

- Similar to **Table per Class** strategy but …
- **Mapped Superclass**
  - No table for superclass
  - No table joins or inheritance exists in the database schema



Mapped Superclass … looks familiar …

- Similar to **Table per Class** strategy but …
- **Mapped Superclass**
  - No table for superclass
  - No table joins or inheritance exists in the database schema
  - Inheritance only exists in the Java object model



Development Process

Step-By-Step

Development Process

1. In superclass, annotate superclass with **`@MappedSuperclass`**

Development Process

Step-By-Step

1. In superclass, annotate superclass with **`@MappedSuperclass`**

2. In superclass, remove the annotations:

Development Process

1. In superclass, annotate superclass with **`@MappedSuperclass`**

2. In superclass, remove the annotations:

    1. **`@Entity`**

Development Process

1. In superclass, annotate superclass with **`@MappedSuperclass`**

2. In superclass, remove the annotations:
    1. **`@Entity`**
    2. **`@Table`**

Development Process

1. In superclass, annotate superclass with **`@MappedSuperclass`**

2. In superclass, remove the annotations:
    1. **`@Entity`**
    2. **`@Table`**
    3. **`@Inheritance`**

Development Process

Step-By-Step

1. In superclass, annotate superclass with **`@MappedSuperclass`**

2. In superclass, remove the annotations:
    1. **`@Entity`**
    2. **`@Table`**
    3. **`@Inheritance`**

3. In subclasses, use normal Hibernate annotation: **`@Entity`**

Annotation for Inheritance

Annotation for Inheritance

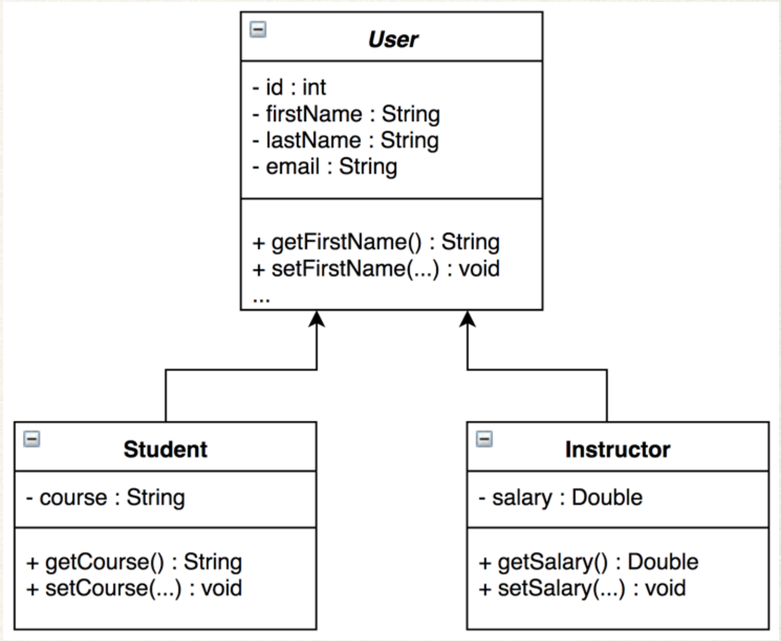
Annotation	Description

Annotation for Inheritance

Annotation	Description
`@MappedSuperclass`	**Designates a class whose mapping is applied to entities that inherit from it.** **A mapped superclass has no separate table defined for it.**

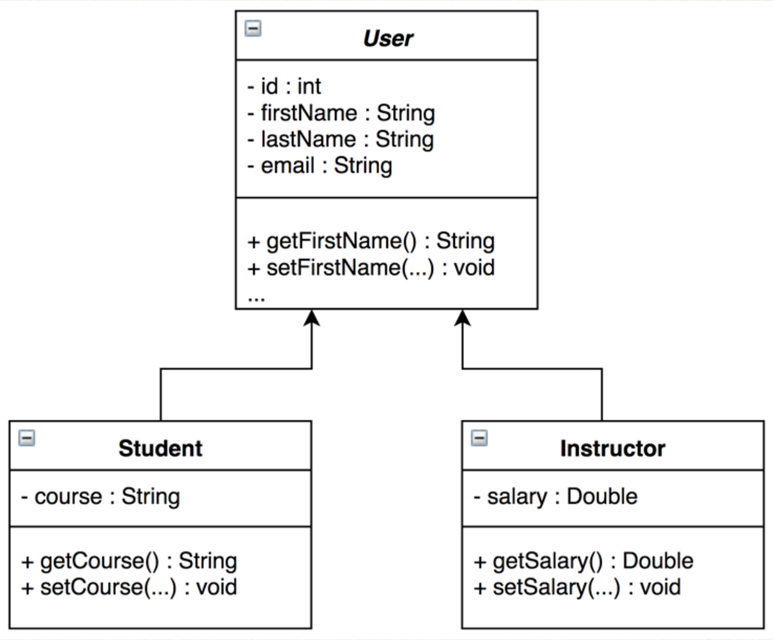
Step 1: Annotate with @MappedSuperclass

Step 1: Annotate with @MappedSuperclass



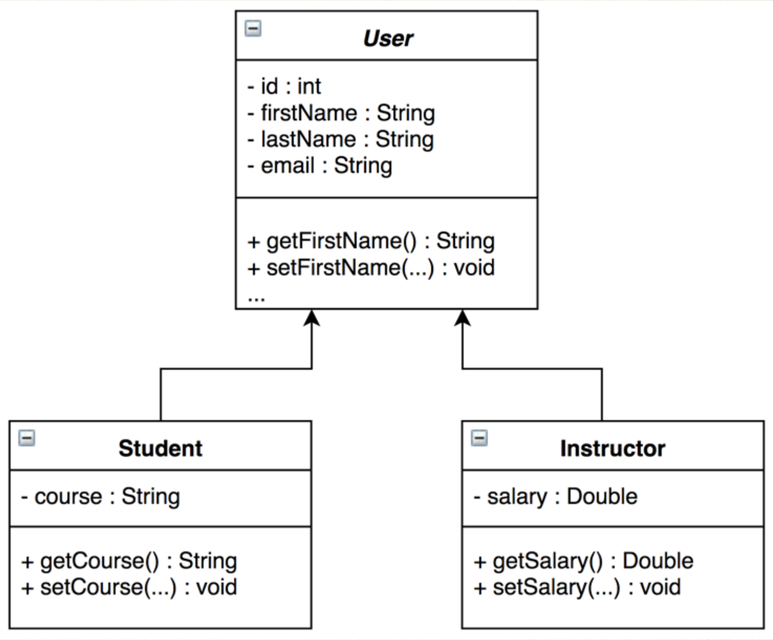
Step 1: Annotate with @MappedSuperclass

```
@MappedSuperclass
public abstract class User {

...

}
```



Step 1: Annotate with @MappedSuperclass

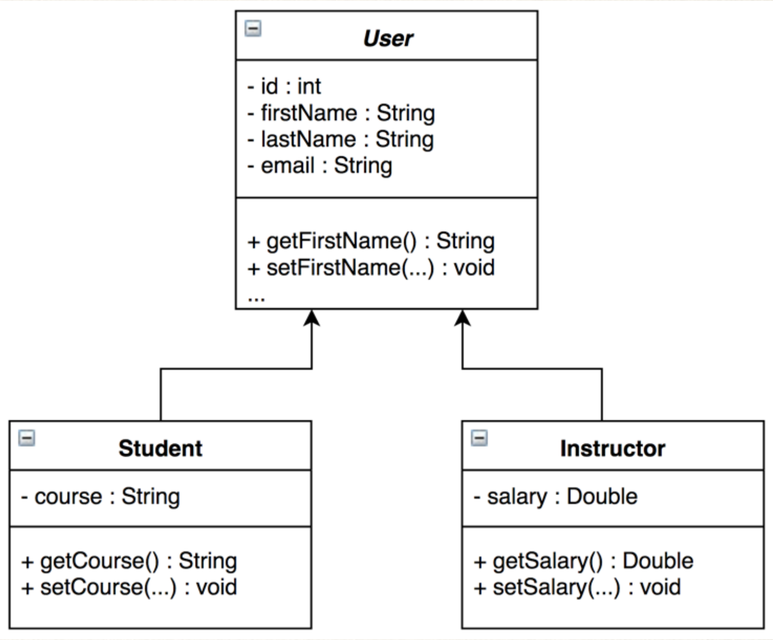
```
@MappedSuperclass
public abstract class User {

...

}
```



Step 1: Annotate with @MappedSuperclass

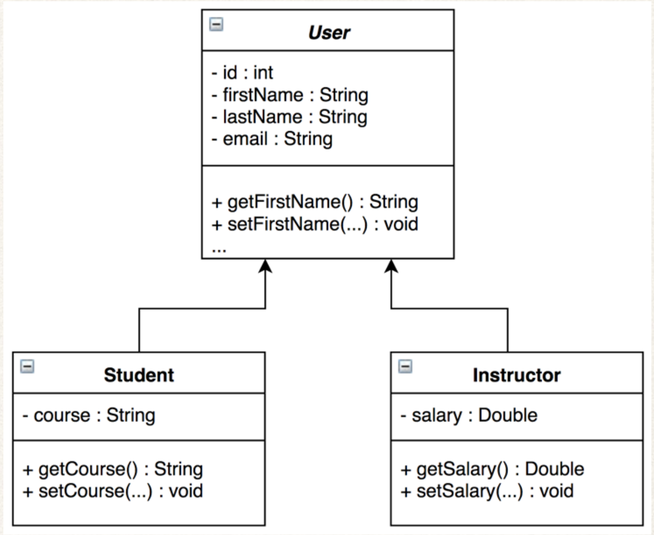
```
@MappedSuperclass
public abstract class User {

...

}
```

Step 2: Remove annotations
@Entity, @Table and @Inheritance



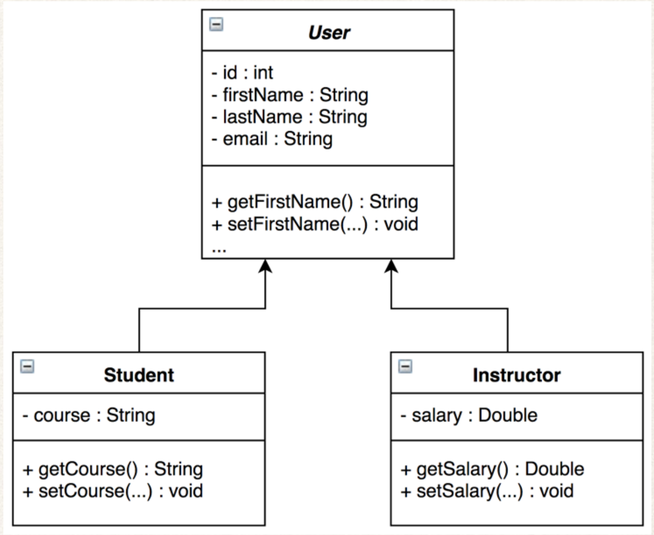
Step 3: Normal Hibernate Annotation: @Entity

Step 3: Normal Hibernate Annotation: @Entity



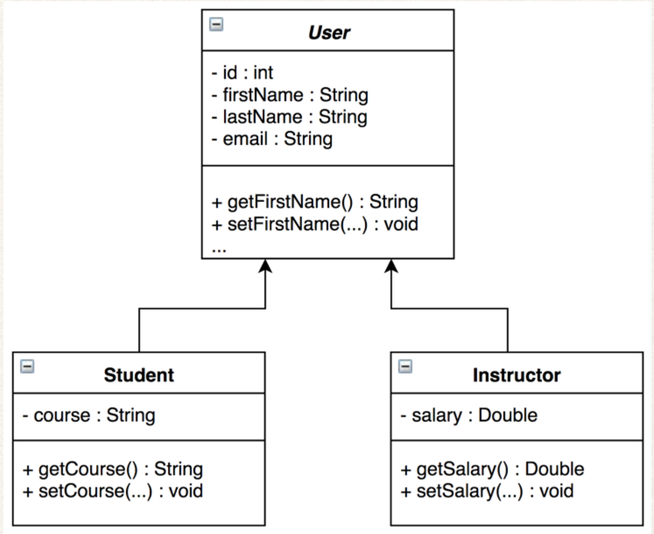
Step 3: Normal Hibernate Annotation: @Entity

```
@Entity
public class Student extends User {
  ...
}
```



Step 3: Normal Hibernate Annotation: @Entity

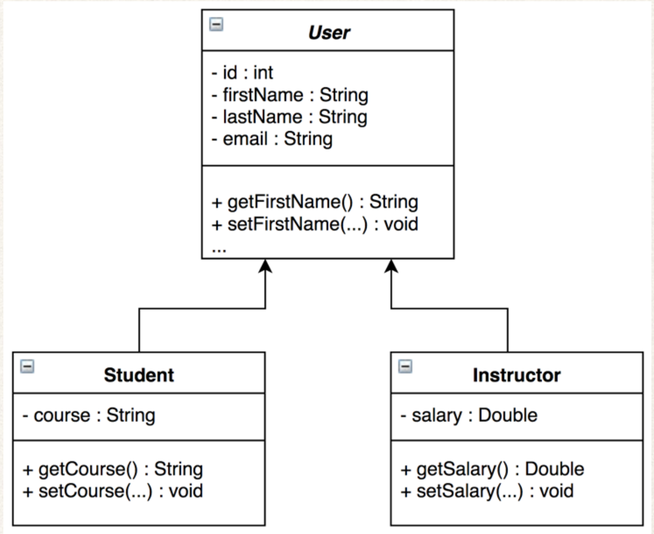
```
@Entity
public class Student extends User {
  ...
}
```




```
@Entity
public class Instructor extends User {
  ...
}
```

Step 3: Normal Hibernate Annotation: @Entity

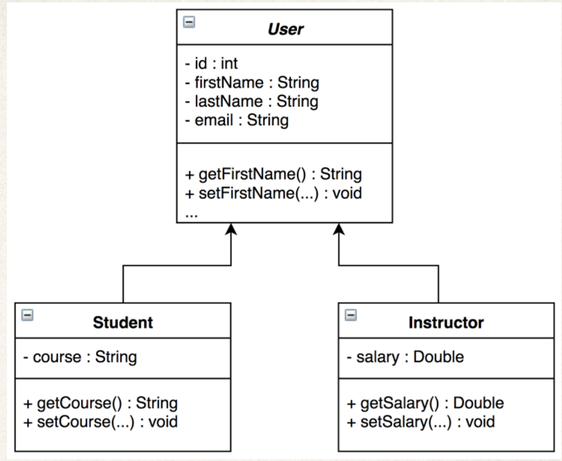
```
@Entity
public class Student extends User {
  ...
}
```

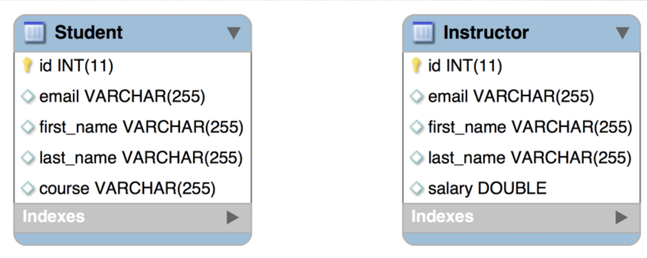



```
@Entity
public class Instructor extends User {
  ...
}
```

Can also customize with @Table, @Column etc…

Run the App

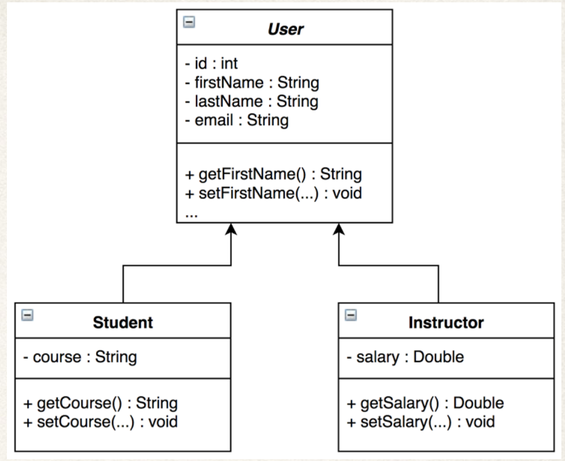


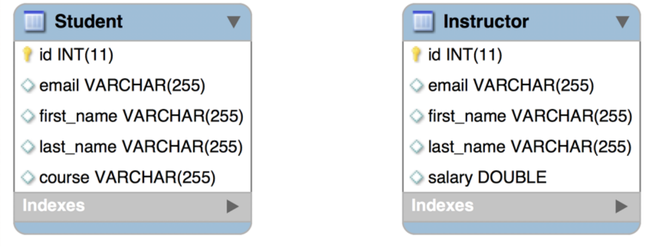


Run the App

Console output

```
Hibernate: insert into Student (email, first_name, last_name, course) values (?, ?, ?, ?)
Hibernate: insert into Instructor (email, first_name, last_name, salary) values (?, ?, ?, ?)
```





Run the App

Console output

```
Hibernate: insert into Student (email, first_name, last_name, course) values (?, ?, ?, ?)
Hibernate: insert into Instructor (email, first_name, last_name, salary) values (?, ?, ?, ?)
```

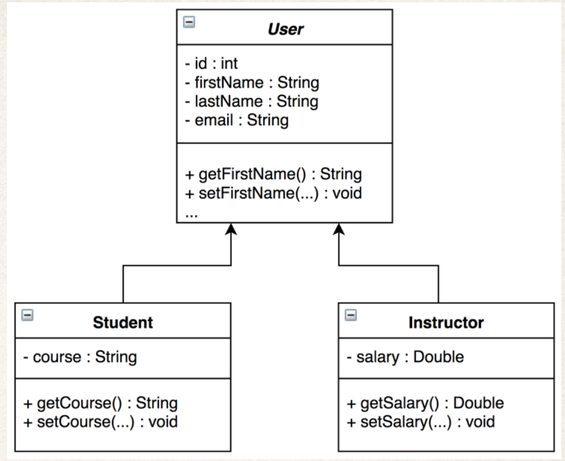
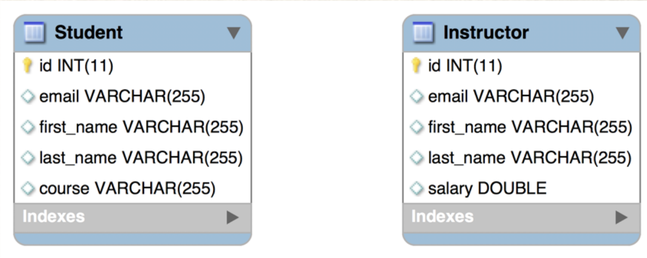
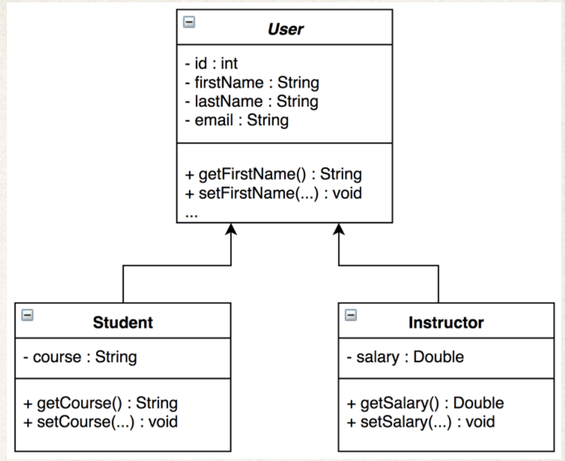


Table: Student

id	email	first_name	last_name	course
1	mary@luv2code.com	Mary	Public	Hibernate



Run the App



Console output

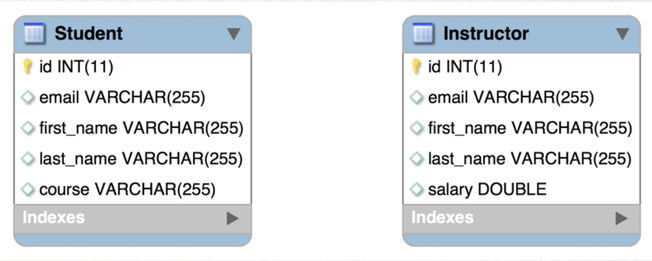
```
Hibernate: insert into Student (email, first_name, last_name, course) values (?, ?, ?, ?)
Hibernate: insert into Instructor (email, first_name, last_name, salary) values (?, ?, ?, ?)
```

Table: Student

id	email	first_name	last_name	course
1	mary@luv2code.com	Mary	Public	Hibernate

Table: Instructor

id	email	first_name	last_name	salary
1	john@luv2code.com	John	Doe	20000



Mapped Superclass

Mapped Superclass

- Advantages

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages
  - For polymorphic queries, need to manually code HQL joins

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages
  - For polymorphic queries, need to manually code HQL joins
  - For example, to query for all **`User`**s, need to join across **`Student`**, **`Instructor`** etc …

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages
  - For polymorphic queries, need to manually code HQL joins
  - For example, to query for all **`User`**s, need to join across **`Student`**, **`Instructor`** etc …
    - Requires manual HQL coding

Mapped Superclass

- Advantages
  - Simple and straight-forward implementation
  - Queries on concrete subclasses perform well (no need for joins)

- Disadvantages
  - For polymorphic queries, need to manually code HQL joins
  - For example, to query for all **`User`**s, need to join across **`Student`**, **`Instructor`** etc …
    - Requires manual HQL coding
    - May result in slower performance (results in many joins)