luv2code

Mapping - Embedded

Refresher on Hibernate Types

Refresher on Hibernate Types

- Hibernate supports two major types

Refresher on Hibernate Types

- Hibernate supports two major types
  - **Value** types: String, Integer, Double, Date, etc …

Refresher on Hibernate Types

- Hibernate supports two major types
  - **Value** types: String, Integer, Double, Date, etc …
  - **Entity** types: Customer, Student, Account, etc …

Refresher on Hibernate Types

- Hibernate supports two major types
  - **Value** types: String, Integer, Double, Date, etc …
  - **Entity** types: Customer, Student, Account, etc …

- **Value** types do not have their own lifecycle and no identifier

Refresher on Hibernate Types

- Hibernate supports two major types
  - **Value** types: String, Integer, Double, Date, etc …
  - **Entity** types: Customer, Student, Account, etc …

- **Value** types do not have their own lifecycle and no identifier
- **Entity** types have their own lifecycle and unique identifier

Categories of Value Types

Categories of Value Types

Category	Examples

Categories of Value Types

Category	Examples
Basic	**String, Integer, Double, Boolean, Date, etc …**

Categories of Value Types

Category	Examples
Basic	**String, Integer, Double, Boolean, Date, etc …**
Embedded	**Address, PhoneNumber *(any custom object)***

Categories of Value Types

Category	Examples
Basic	**String, Integer, Double, Boolean, Date, etc …**
Embedded	**Address, PhoneNumber *(any custom object)***
Collection	**Set, List, Map, etc …** ***We used this earlier with @ElementCollection***

Embedded Value Type

A composition of values

Promotes reuse across other entities

Use Case for Embedded

Use Case for Embedded

- A student will have an **address**

Use Case for Embedded

- A student will have an **address**
  - Composition of fields: street, city, zip code

Use Case for Embedded

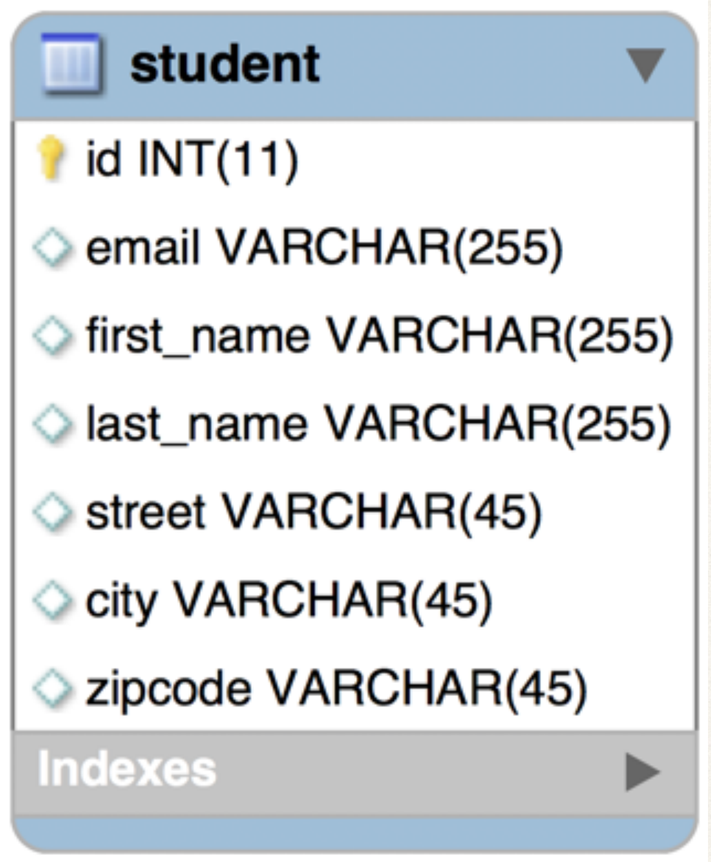
- A student will have an **address**
  - Composition of fields: street, city, zip code

- The **address** fields will be in the **same** database table

Use Case for Embedded

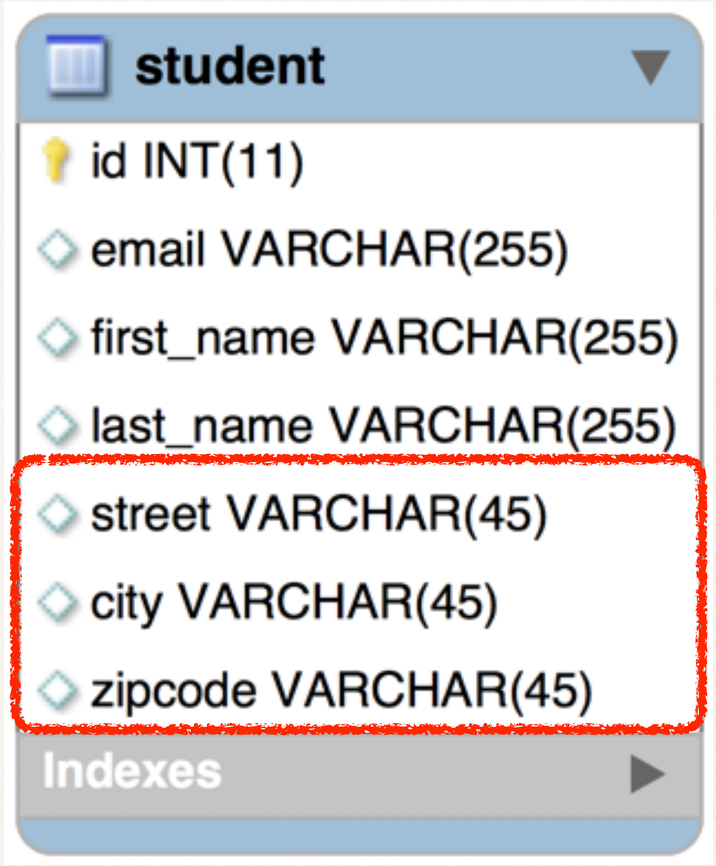
- A student will have an **address**
  - Composition of fields: street, city, zip code
- The **address** fields will be in the **same** database table
- In Java code, model it as an object

Database Diagram



Database Diagram

Address

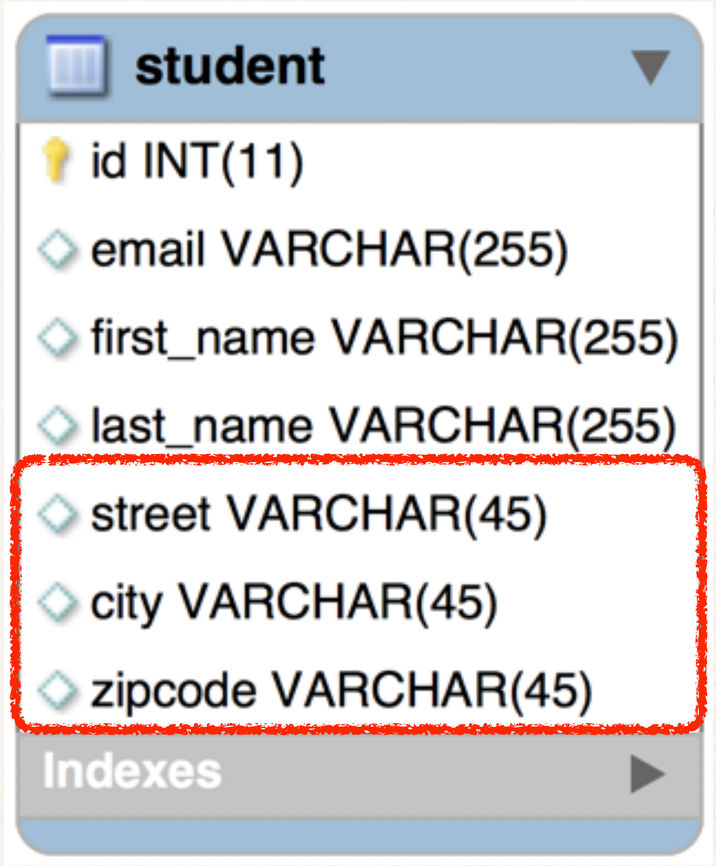


Database Diagram

Address

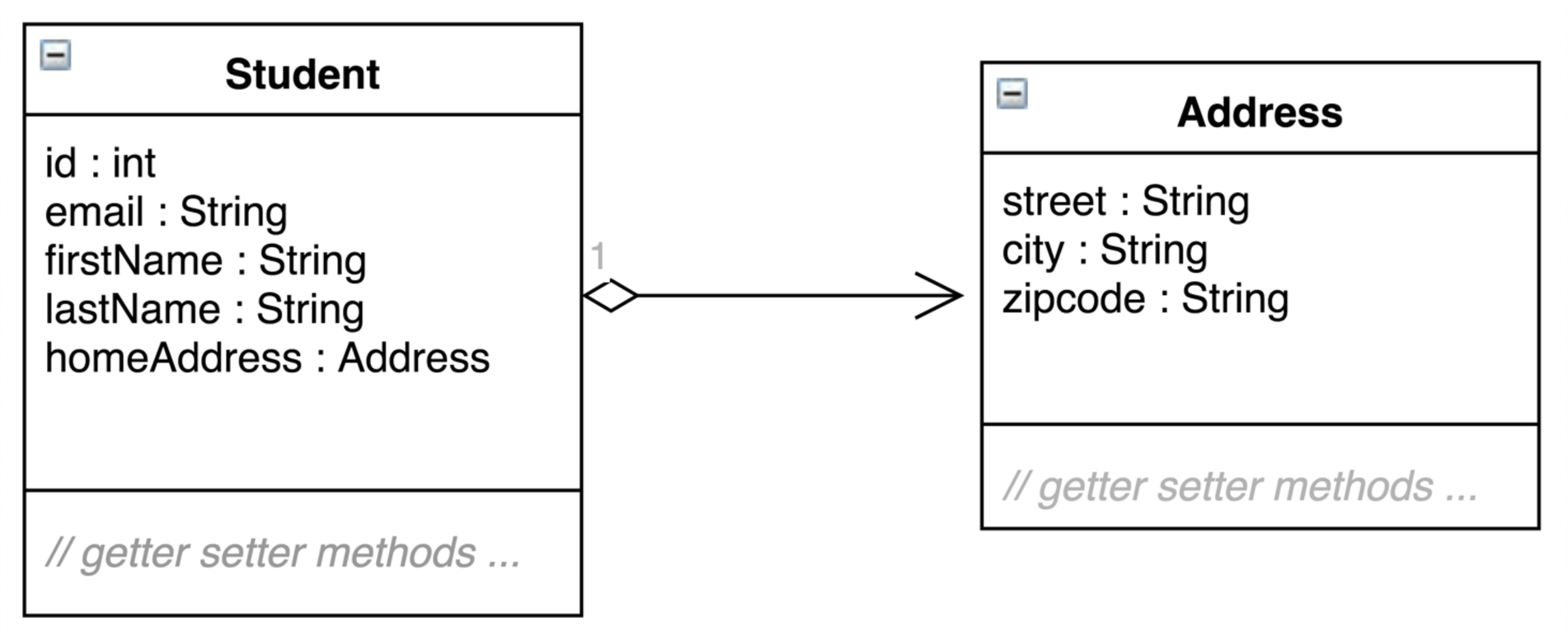
Reuse Address in other parts of our app

For: Student, Customer, Airport etc…

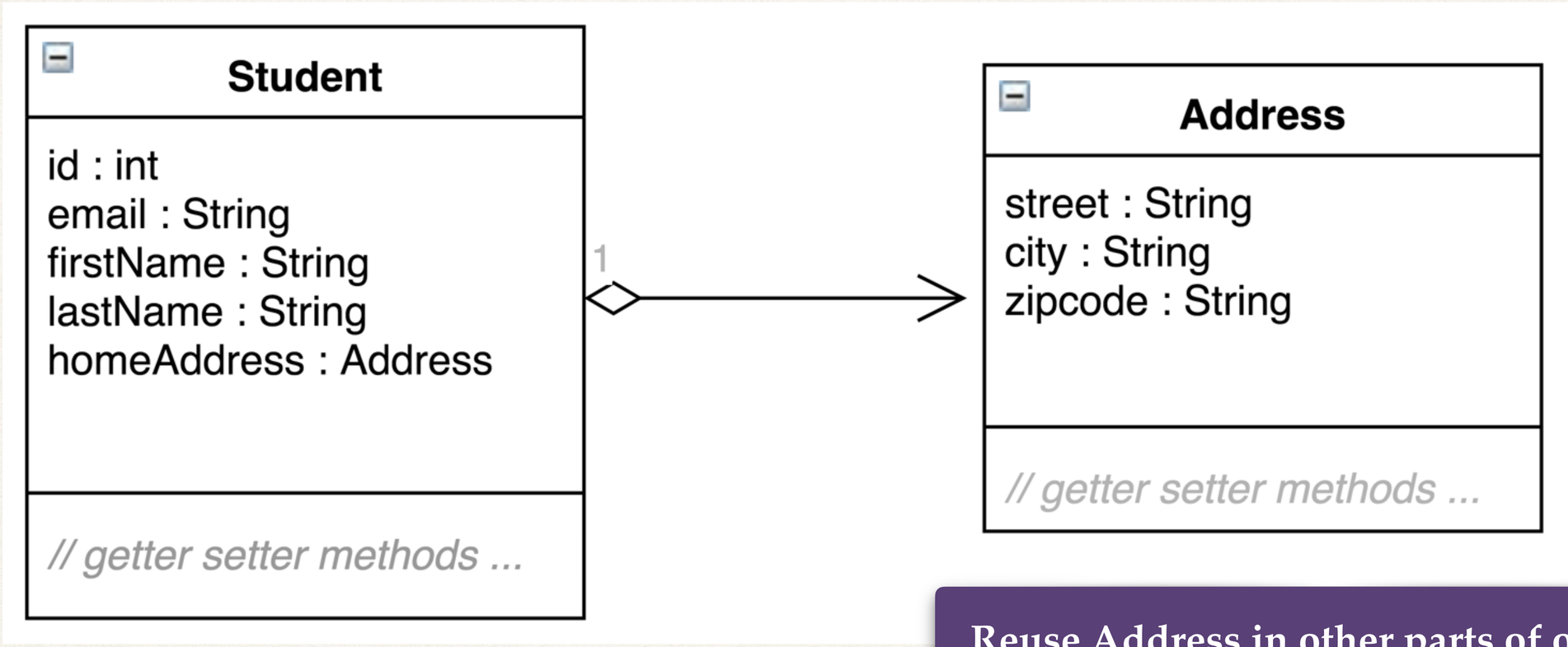


UML Diagram

UML Diagram



UML Diagram



Reuse Address in other parts of our app

For: Student, Customer, Airport etc…

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create database table

Development Process

Step-By-Step

1. Create database table

2. Define the **`Address`** value type

Development Process

Step-By-Step

1. Create database table

2. Define the **`Address`** value type

3. Reference **`Address`** value type in **`Student`** class

Development Process

Step-By-Step

1. Create database table

2. Define the **`Address`** value type

3. Reference **`Address`** value type in **`Student`** class

4. Develop the main application

Step 1: Create database table

Step 1: Create database table

- For ease of development and testing, we'll use auto configuration

```
<property name="hibernate.hbm2ddl.auto">create</property>
```

Annotation for Embedded

Annotation for Embedded

Annotation	Description

Annotation for Embedded

Annotation	Description
`@Embeddable`	**Used to annotate a given class as an Embedded value type. For example: We'll apply this to the `Address` class**

Annotation for Embedded

Annotation	Description
`@Embeddable`	**Used to annotate a given class as an Embedded value type.** **For example: We'll apply this to the `Address` class**
`@Embedded`	**Used to reference an Embedded value type** **For example: `Student` can use this to refer to `Address`** ***The use of @Embedded is optional***

Step 2: Define the Address Value Type

```
public class Address {



}
```

Step 2: Define the Address Value Type

```
@Embeddable
public class Address {




}
```

Declare this as "Embeddable"

Step 2: Define the Address Value Type

```
@Embeddable
public class Address {

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

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

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

  // constructors, getters and setters

}
```

Step 2: Define the Address Value Type

```
@Embeddable
public class Address {

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

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

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

  // constructors, getters and setters

}
```

Normal mapping of fields to database columns

Step 2: Define the Address Value Type

```
@Embeddable
public class Address {

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

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

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

  // constructors, getters and setters

}
```

Normal mapping of fields to database columns

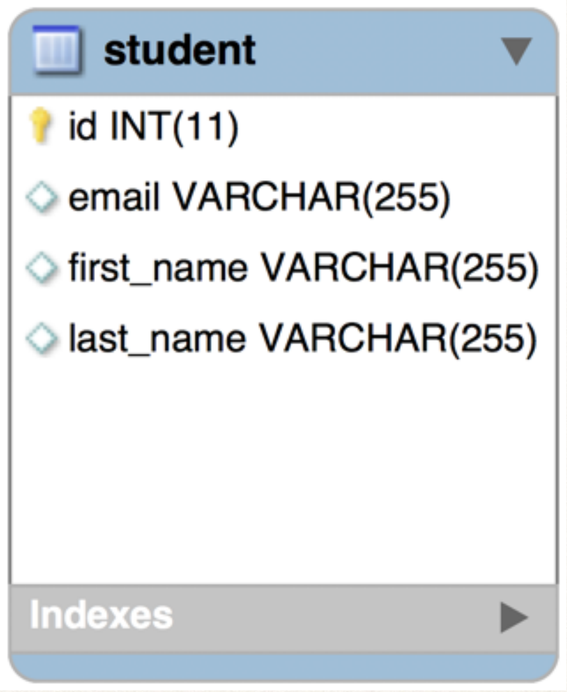
Note: Since this is a value type, there is no identifier (primary key)

Step 3: Reference the Address Value Type

```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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




   …
}
```

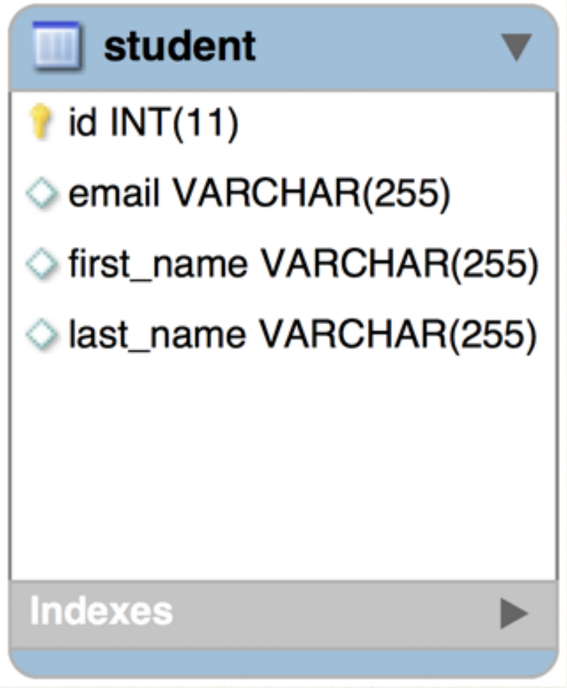
Step 3: Reference the Address Value Type

```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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




   …
}
```



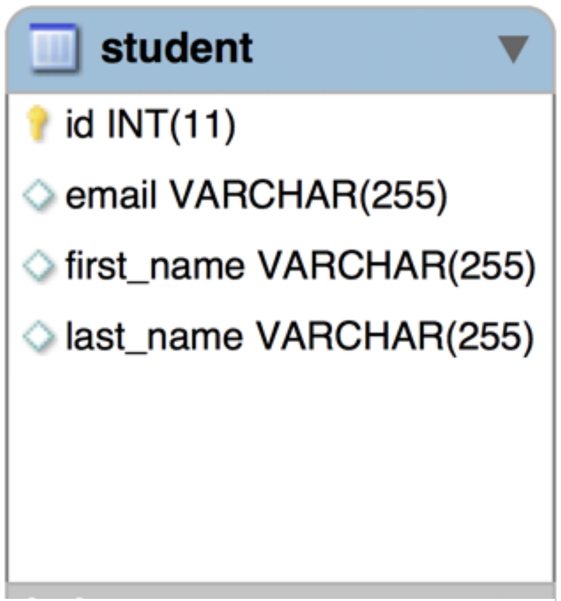
Step 3: Reference the Address Value Type

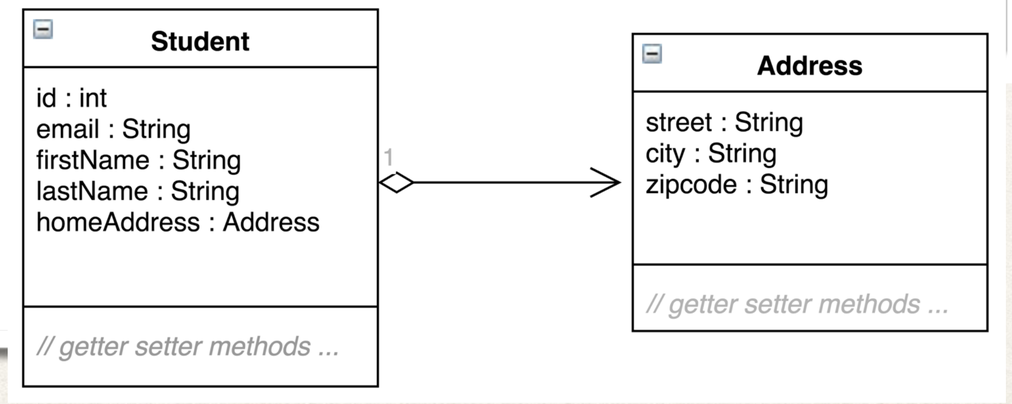
```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```



Step 3: Reference the Address Value Type

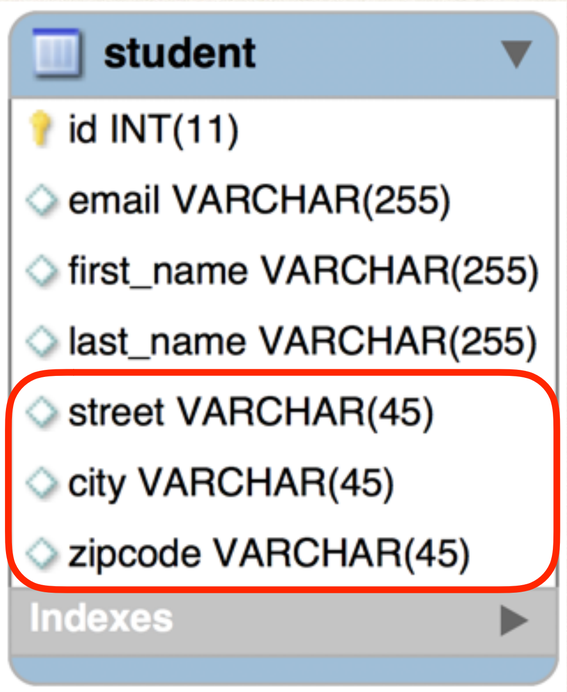
```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```





Step 3: Reference the Address Value Type

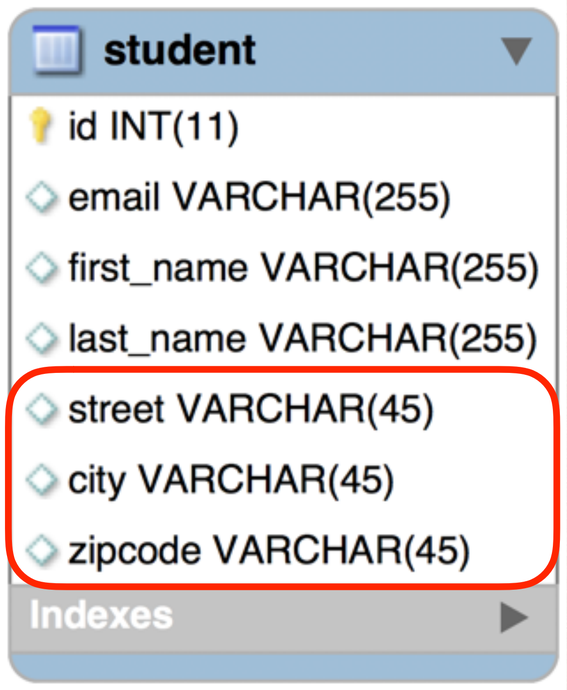
```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```



Step 3: Reference the Address Value Type

```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```

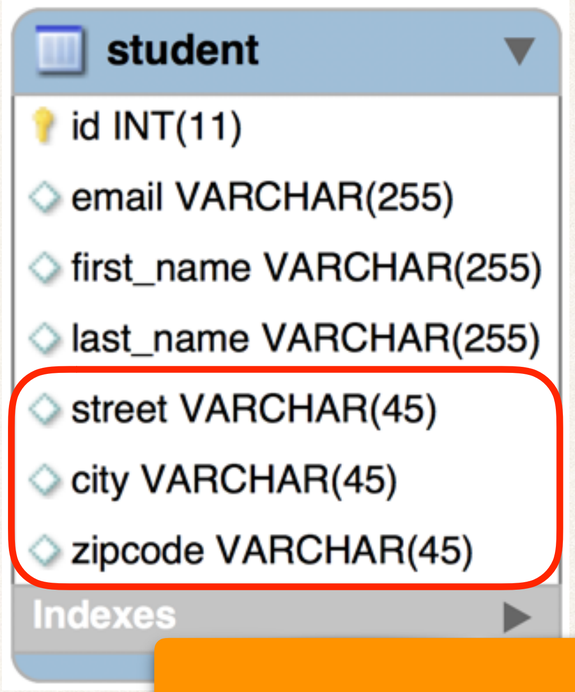
```
@Embeddable
public class Address {

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

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

    @Column(name="zipcode")
    private String zipcode;
```



Step 3: Reference the Address Value Type

```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```

```
@Embeddable
public class Address {

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

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

    @Column(name="zipcode")
    private String zipcode;
```

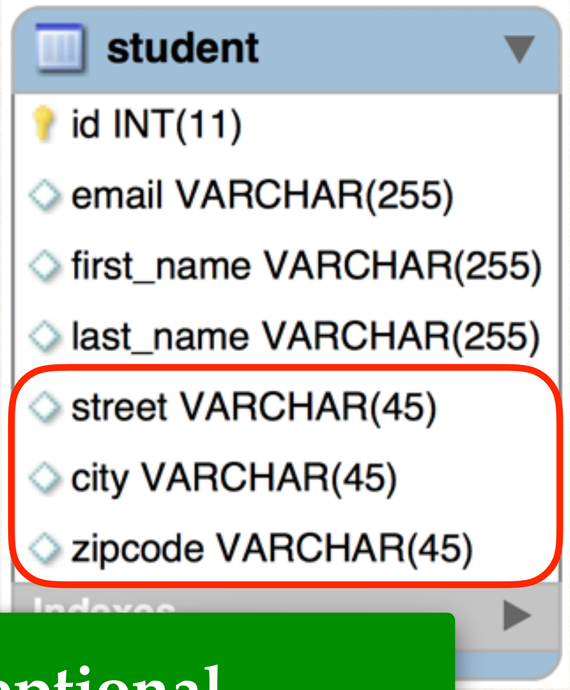


Columns are in the same table as student

Step 3: Reference the Address Value Type

```
@Entity
@Table(name="student")
public class Student {

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

   @Column(name="first_name")
   private String firstName;

   @Column(name="last_name")
   private String lastName;

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

   @Embedded
   private Address homeAddress;

   …
}
```

```
@Embeddable
public class Address {

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

    @Column(name="city")
    private String city;
```



@Embedded is optional
This is inferred because
Address has @Embeddable

Step 4: Develop the main application

Step 4: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
```

Step 4: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");

// create the address object
Address homeAddress = new Address("1 Main St", "Philadelphia", "19103");
```

Step 4: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");

// create the address object
Address homeAddress = new Address("1 Main St", "Philadelphia", "19103");

// start a transaction
session.beginTransaction();
```

Step 4: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");

// create the address object
Address homeAddress = new Address("1 Main St", "Philadelphia", "19103");

// start a transaction
session.beginTransaction();

// save the object
System.out.println("Saving the student and address..");
tempStudent.setHomeAddress(homeAddress);
session.persist(tempStudent);
```

Step 4: Develop the main application

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");

// create the address object
Address homeAddress = new Address("1 Main St", "Philadelphia", "19103");

// start a transaction
session.beginTransaction();

// save the object
System.out.println("Saving the student and address..");
tempStudent.setHomeAddress(homeAddress);
session.persist(tempStudent);
```

Associate the address

Run the App

Run the App

Console output

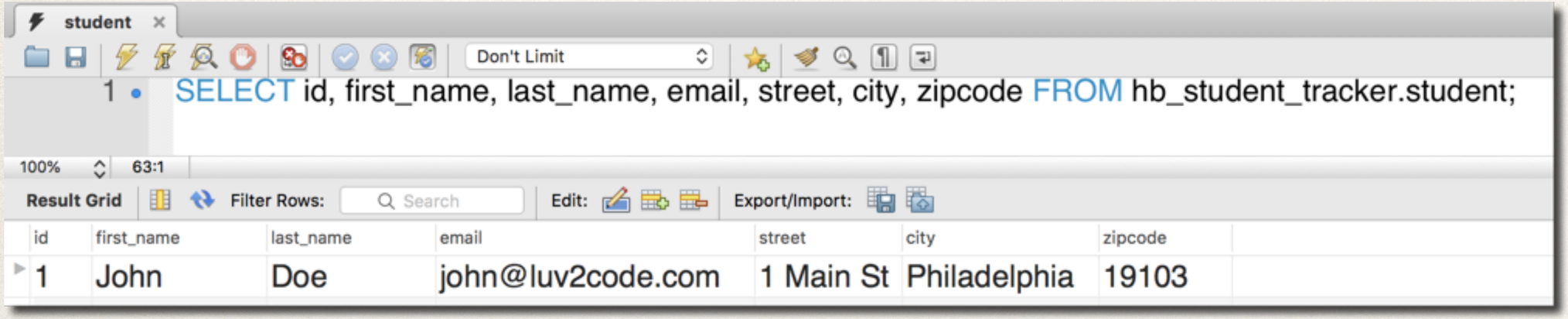
```
Hibernate: insert into student (email, first_name, city, street, zipcode, last_name) values (?, ?, ?, ?, ?, ?)
```

Run the App

Console output

```
Hibernate: insert into student (email, first_name, city, street, zipcode, last_name) values (?, ?, ?, ?, ?, ?)
```

Table: student

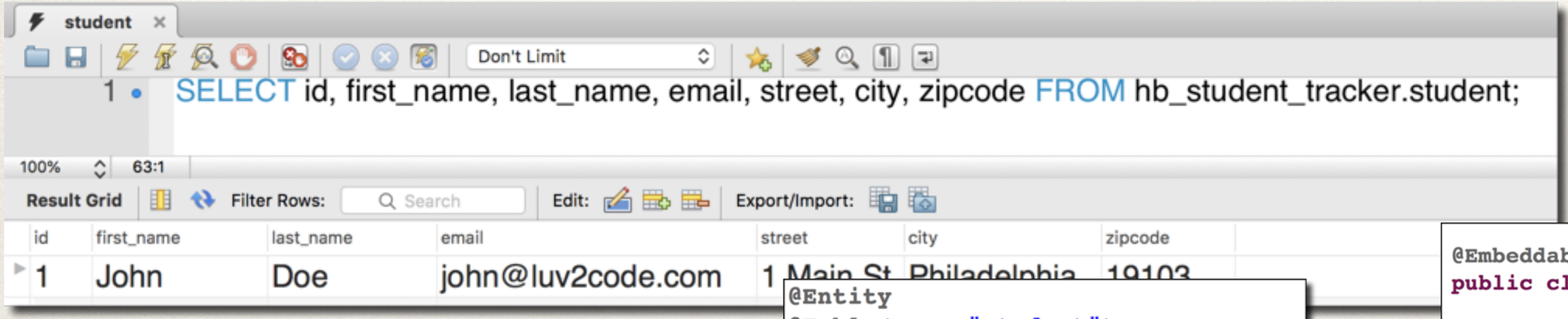


Run the App

Console output

```
Hibernate: insert into student (email, first_name, city, street, zipcode, last_name) values (?, ?, ?, ?, ?, ?)
```

Table: student




```
@Entity
@Table(name="student")
public class Student {

    @Embedded
    private Address homeAddress;
```

```
@Embeddable
public class Address {

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

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

    @Column(name="zipcode")
    private String zipcode;
```