luv2code

Mapping - Sorted Maps

Sorted **Map**

An object that maps unique keys to values

You can control the sorting order

Use Case for Sorted Maps

Use Case for Sorted Maps

- Useful when you want to retrieve sorted data

Use Case for Sorted Maps

- Useful when you want to retrieve sorted data

- Examples

Use Case for Sorted Maps

- Useful when you want to retrieve sorted data
- Examples
  - Map of error codes and error messages

Key	Value
20	Disk Full
30	Syntax Error
40	Account Blocked

Use Case for Sorted Maps

- Useful when you want to retrieve sorted data

- Examples

  - Map of error codes and error messages
  - Map of image file names and image descriptions

Key	Value
photo1.jpg	Photo 1
photo2.jpg	Photo 2
photo3.jpg	Photo 3

Student and Images

Student and Images

- A student will have a ***sorted map*** of images

Student and Images

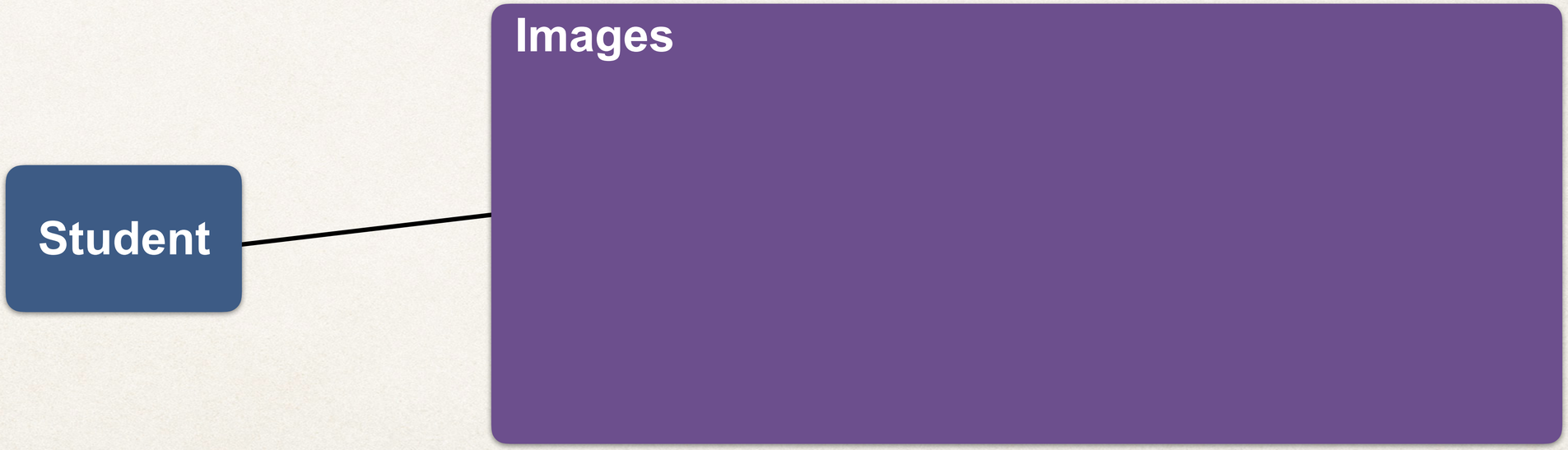
- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***

Student and Images

- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

Student and Images

- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

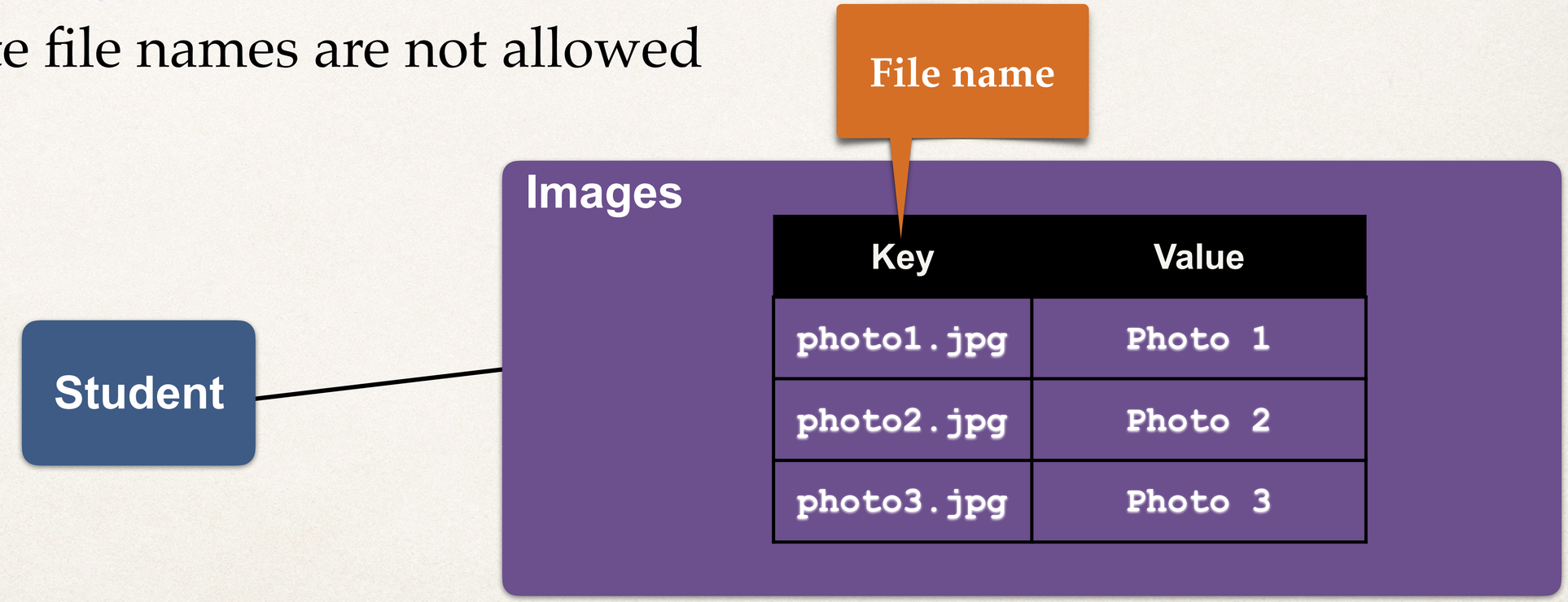


Student and Images

- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

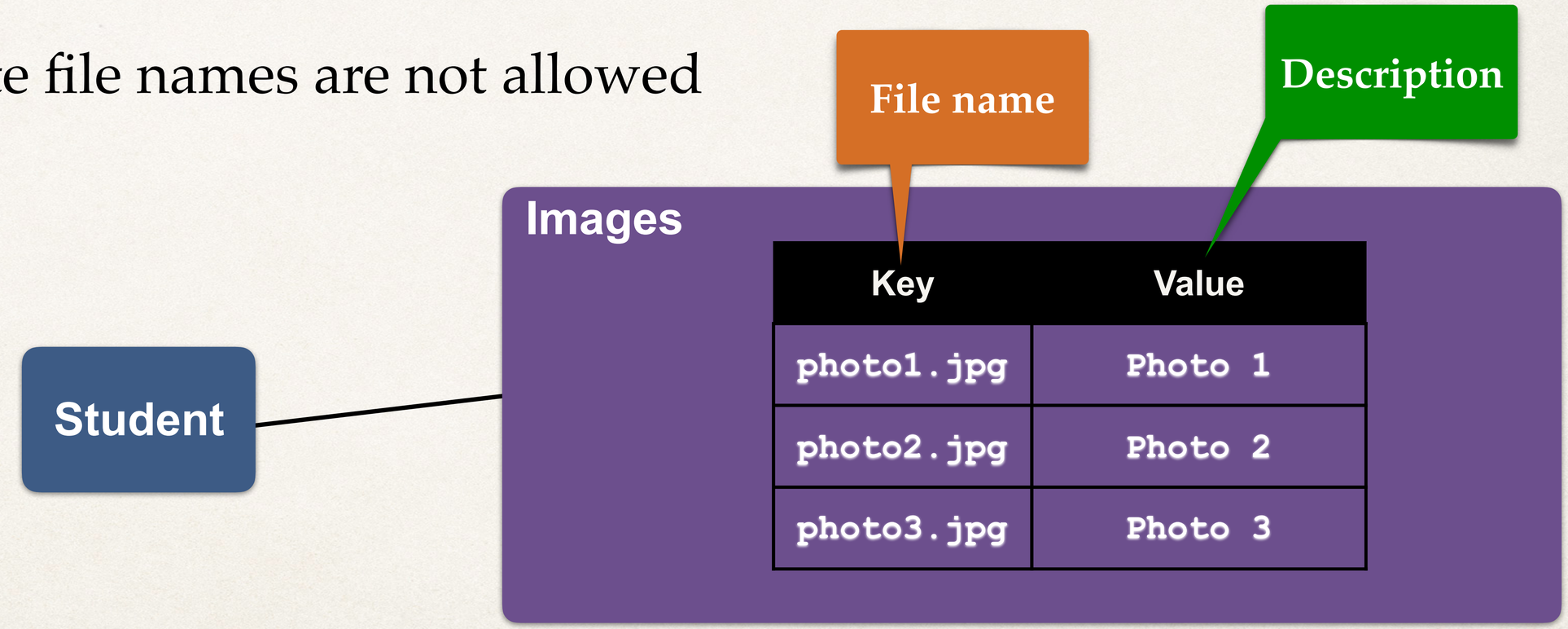
Student and Images

- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed

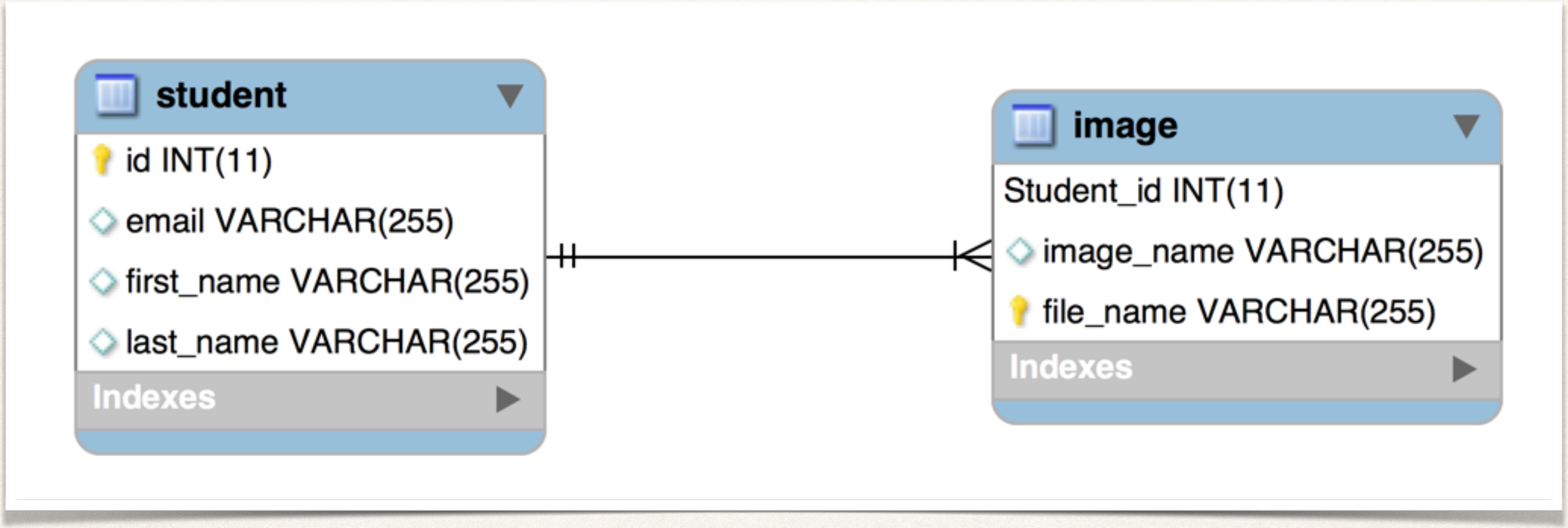


Student and Images

- A student will have a ***sorted map*** of images
  - Each image will have a ***file name*** and a ***description***
  - Duplicate file names are not allowed



Database Diagram



Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create database tables

Development Process

Step-By-Step

1. Create database tables

2. Annotate the Map

Development Process

Step-By-Step

1. Create database tables

2. Annotate the Map

3. Develop the main application

Step 1: Create database tables

Step 1: Create database tables

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

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

Step 2: Annotate the Map

Step 2: Annotate the Map

Step 2: Annotate the Map


```
private SortedMap<String, String> images = new LinkedHashMap<String, String>();
```

Annotation for Ordering

Annotation for Ordering

Annotation	Description

Annotation for Ordering

Annotation	Description
`@OrderBy`	**Specifies the ordering of the elements when a collection is retrieved.**

Annotation for Ordering

Annotation	Description
`@OrderBy`	**Specifies the ordering of the elements when a collection is retrieved.**

Syntax: `@OrderBy("[field name or property name] [ASC | DESC]")`

Annotation for Ordering

Annotation	Description
`@OrderBy`	**Specifies the ordering of the elements when a collection is retrieved.**

Syntax: `@OrderBy("[field name or property name] [ASC | DESC]")`

Example:

```
@OrderBy("file_name DESC")
```

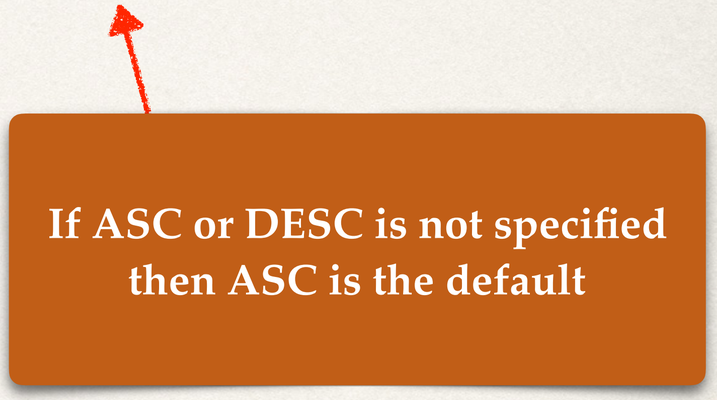
Annotation for Ordering

Annotation	Description
`@OrderBy`	**Specifies the ordering of the elements when a collection is retrieved.**

Syntax: `@OrderBy("[field name or property name] [ASC | DESC]")`

Example:

```
@OrderBy("file_name DESC")
```



Annotation for Ordering

Annotation	Description
`@OrderBy`	**Specifies the ordering of the elements when a collection is retrieved.**

Syntax: `@OrderBy("[field name or property name] [ASC | DESC]")`

Example:

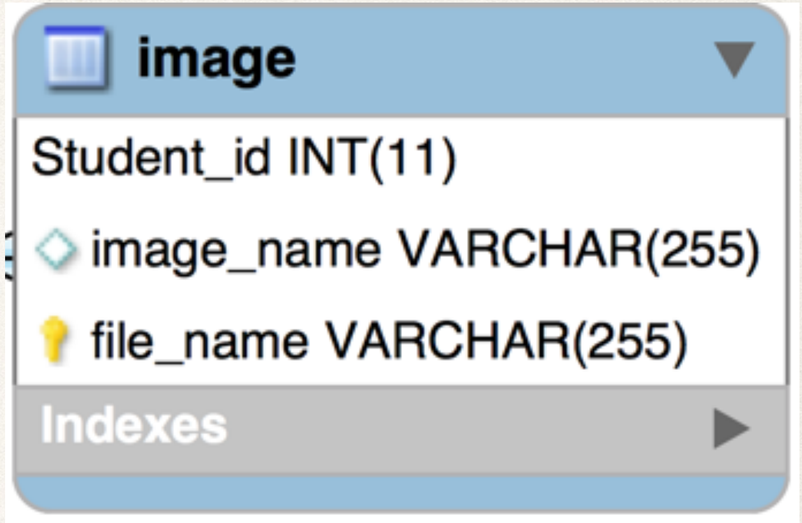
```
@OrderBy("file_name DESC")

@OrderBy
```

If ASC or DESC is not specified then ASC is the default

Defaults to order by map key column, ASC

Step 2: Annotate the Map



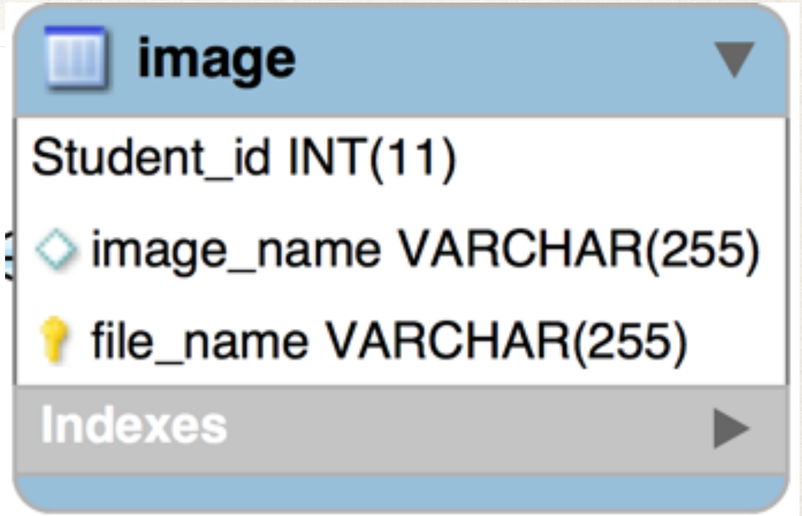
Step 2: Annotate the Map

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

  …

  private SortedMap<String, String> images = new TreeMap<String, String>();

  …
}
```



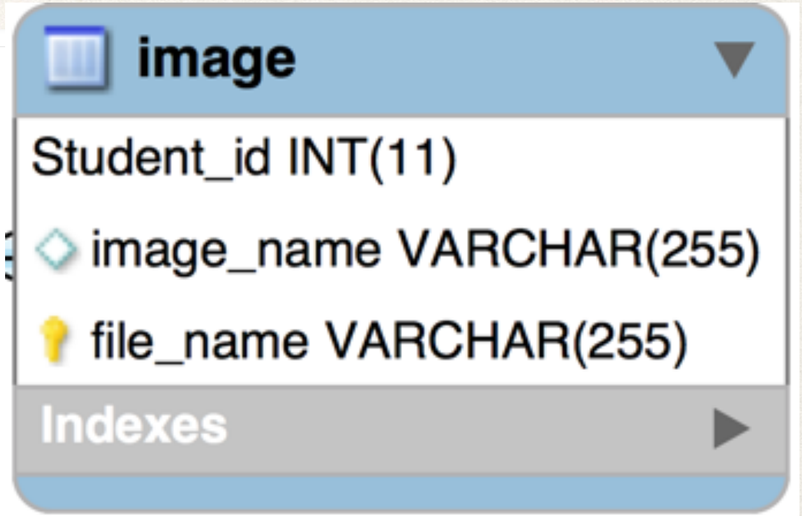
Step 2: Annotate the Map

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

  …

  private SortedMap<String, String> images = new TreeMap<String, String>();

  …
}
```



Use a **Map** to track
image file names and descriptions

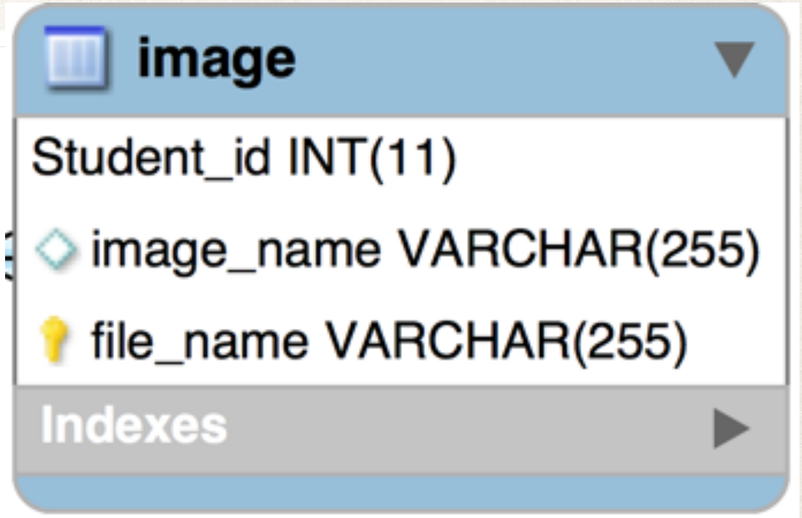
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  @OrderBy
  private SortedMap<String, String> images = new TreeMap<String, String>();


  …
}
```



Step 2: Annotate the Map

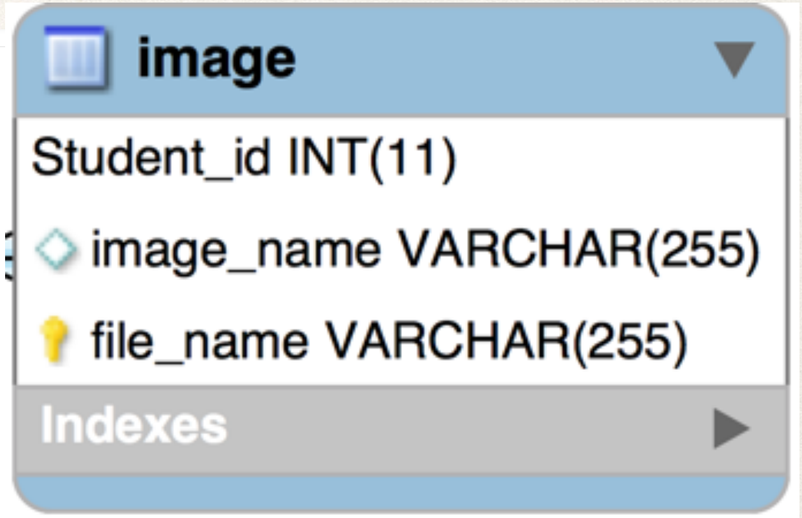
```
@Entity
@Table(name="student")
public class Student {

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  @OrderBy
  private SortedMap<String, String> images = new TreeMap<String, String>();


  …
}
```

Column for map key



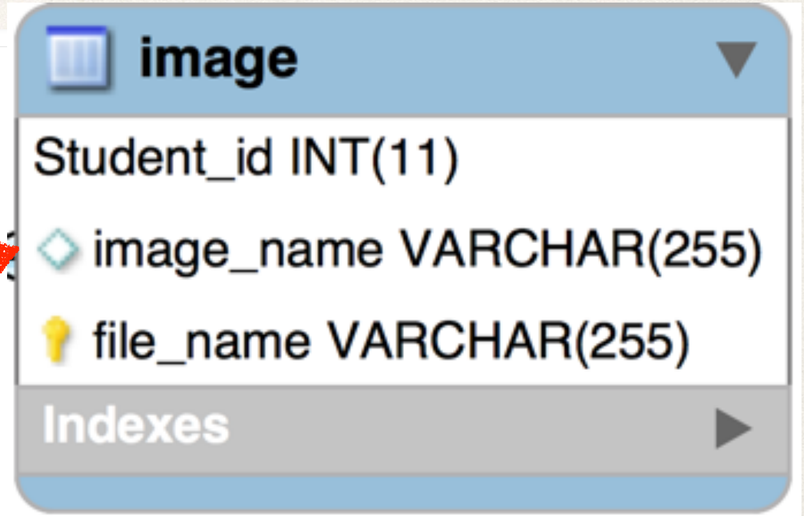
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  @OrderBy
  private SortedMap<String, String> images = new TreeMap<String, String>();


  …
}
```



The column for map value

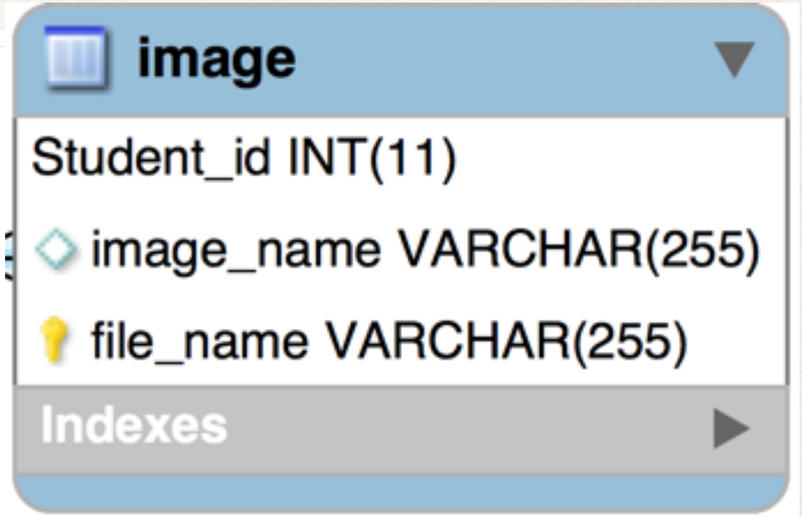
Step 2: Annotate the Map

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

  …

  @ElementCollection
  @CollectionTable(name="image")
  @MapKeyColumn(name="file_name")
  @Column(name="image_name")
  @OrderBy
  private SortedMap<String, String> images = new TreeMap<String, String>();


  …
}
```



Defaults to order by map key column, ASC

Step 3: Develop the main application

Step 3: Develop the main application

CREATE THE IMAGES

Step 3: Develop the main application

CREATE THE IMAGES

```
// create the object
Student tempStudent =  new Student("John", "Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();
```

Get a reference to the images **Map**

Step 3: Develop the main application

CREATE THE IMAGES

Key

Value

```
// create the ob
Student tempStud          Stude          Doe", "john@luv2code.com");
Map<String, String> theImages =  tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");
```

Now, let's add to the images **Map**

Step 3: Develop the main application

CREATE THE IMAGES

Key

Value

```
// create the ob
Student tempStud          Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");
```

File name

Description

Images

Key	Value
photo1.jpg	Photo 1
photo2.jpg	Photo 2
photo3.jpg	Photo 3

Step 3: Develop the main application

CREATE THE IMAGES

Key

Value

```
// create the ob
Student tempStud          Stude          Doe", "john@luv2code.com");
Map<String, String> theImages = tempStudent.getImages();

theImages.put("photo1.jpg", "Photo 1");
theImages.put("photo2.jpg", "Photo 2");
theImages.put("photo3.jpg", "Photo 3");


// start a transaction
session.beginTransaction();

// save the object
System.out.println("Saving the student and images..");
session.persist(tempStudent);

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

Step 3: Develop the main application

Step 3: Develop the main application

RETRIEVE THE IMAGES

Step 3: Develop the main application

RETRIEVE THE IMAGES

```
…
// get the student id
int theId = 1;
Student student = session.get(Student.class, theId);
```

Step 3: Develop the main application

RETRIEVE THE IMAGES

```
…
// get the student id
int theId = 1;
Student student = session.get(Student.class, theId);

// print the student detail
System.out.println("Student details: " + student);
```

Step 3: Develop the main application

RETRIEVE THE IMAGES

```
…
// get the student id
int theId = 1;
Student student = session.get(Student.class, theId);

// print the student detail
System.out.println("Student details: " + student);

// print the associated images
System.out.println("The associated images: " + student.getImages());
…
```

Run the App: Retrieve Student Images

Run the App: Retrieve Student Images

Table: student

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

Table: image

Student_id	file_name	image_name
1	photo1.jpg	Photo 1
1	photo2.jpg	Photo 2
1	photo3.jpg	Photo 3

```
select Student_id, file_name, image_name from image
order by file_name
```

Run the App: Retrieve Student Images

Table: student

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

Table: image

Student_id	file_name	image_name
1	photo1.jpg	Photo 1
1	photo2.jpg	Photo 2
1	photo3.jpg	Photo 3

```
select Student_id, file_name, image_name from image
order by file_name
```

Console output

```
The associated images: {photo1.jpg=Photo 1, photo2.jpg=Photo 2, photo3.jpg=Photo 3}
```

Ascending order, based off Map key column

Run the App: Retrieve Student Images

Table: student

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

Table: image

Student_id	file_name	image_name
1	photo1.jpg	Photo 1
1	photo2.jpg	Photo 2
1	photo3.jpg	Photo 3

```
select Student_id, file_name, image_name from image
order by file_name
```

Console output

```
The associated images: {photo1.jpg=Photo 1, photo2.jpg=Photo 2, photo3.jpg=Photo 3}
```

Ascending order, based off Map key column

```
@ElementCollection
@CollectionTable(name="image")
@MapKeyColumn(name="file_name")
@Column(name="image_name")
@OrderBy
private SortedMap<String, String> images = new TreeMap<String, String>();
```