其他映射

集合映射

- ✓ 在前面的一对多关系和多对多关系中，我们使用Set接口类型作为集合存储多端数据，其实也可以使用List接口类型
- ✓ 在实体类中集合属性类型可以是Set和List接口类型，需要注意不是它们的实现类类型，例如HashSet，或者ArrayList类型
- ✓ 一般Set接口类型对应的映射标签是<set>
- ✓ 一般List接口类型对应的映射标签是<bag>

集合映射示例

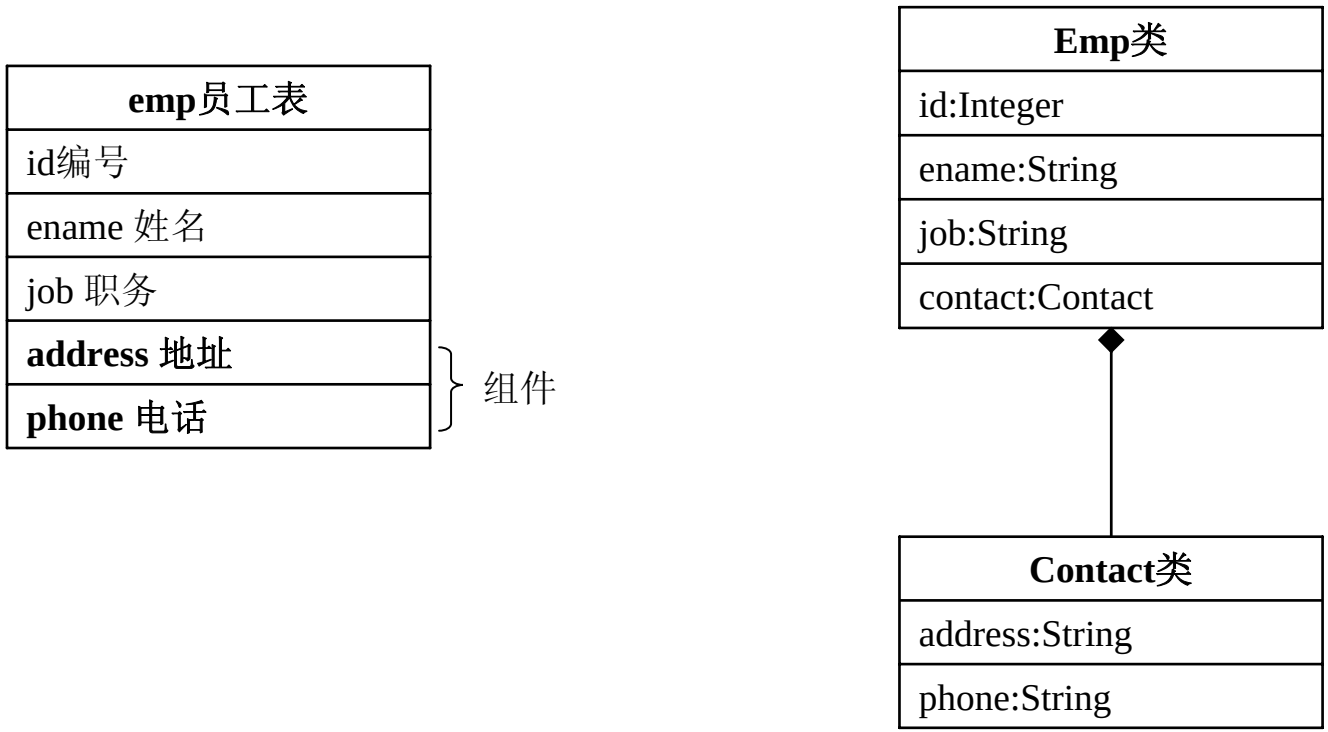
```
public class Dept  {
      private Integer deptno;
      private String dname;
      private List<Emp> emps = new ArrayList<Emp>();
      ……
}

<hibernate-mapping>
   <class name="com.Dept" table="DEPT" schema="HB2">
       <id name="deptno" type="java.lang.Integer">
          <generator class="assigned" />
       </id>
       <property name="dname" type="java.lang.String"/>
       <bag name="emps">
              <key column="deptno"/>
              <one-to-many class="com.Emp"/>
       </bag>
   </class>
</hibernate-mapping>
```

组件映射

- ✓ 组件映射就是把两个表中共同的部门抽象出来，单独形成一个逻辑部分，但和实体的区别是没有OID
- ✓ 优点是可以把粒度变得更细，增强重用
- ✓ 例如一个员工表和学生表，这两个表都有联系方式方面的字段，比如说通信地址，电话等，像通信地址和电话就可以单独抽象出来形成一个逻辑部分，这就是一个组件（component）
- ✓ 组件在数据库中没有对应的表，它是在逻辑上存在的，也没有对应的映射文件，只有一个组件类

组件映射示例

- ✓ 创建一个员工表，把员工的通信地址和电话单独拿出来形成一个组件，然后在员工实体中引用



实体类

组件类com.Contact.java

```
package com;
public class Contact {
    private String address;
    private String phone;
    ……
}
```

员工实体类com.Emp.java

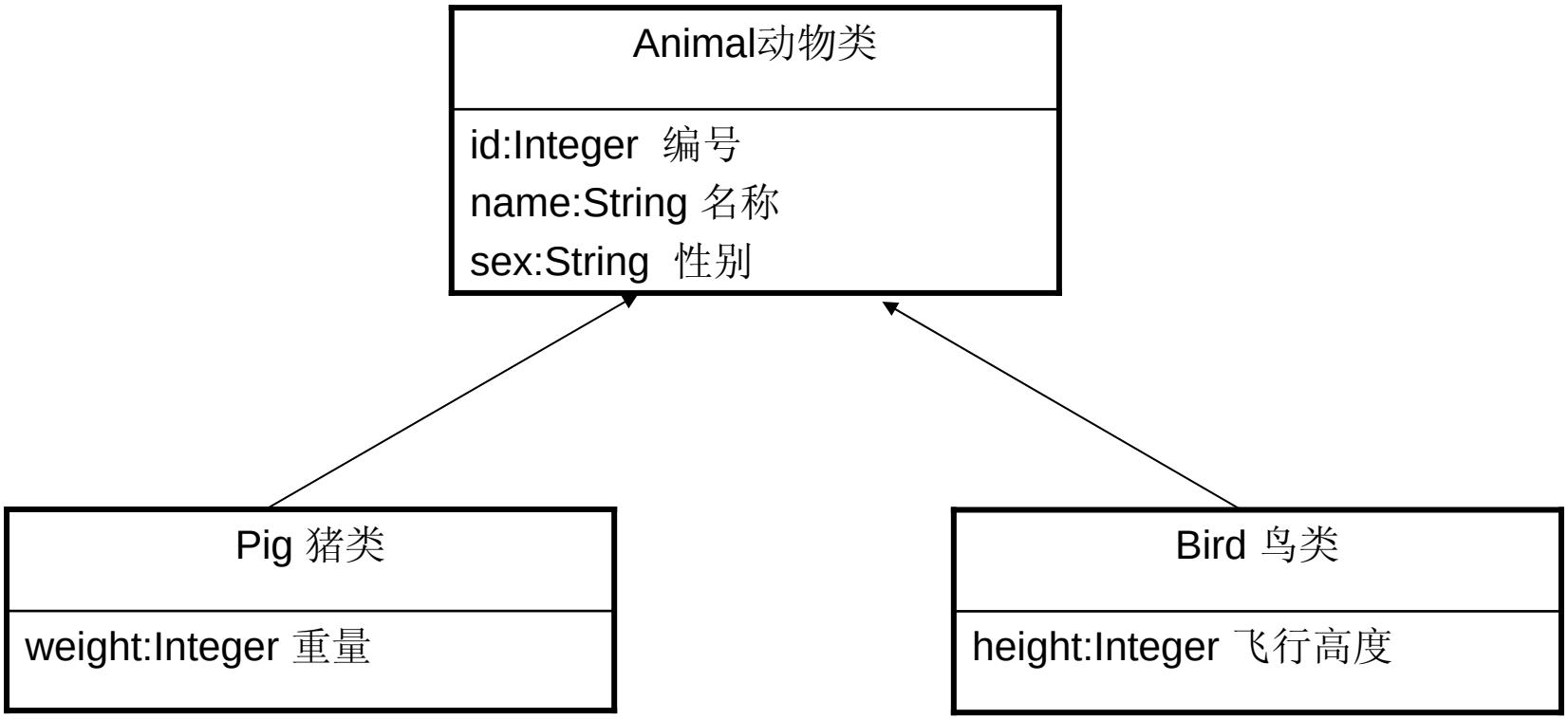
```
package com;
public class Emp {
    private Integer id;
    private String ename;
    private String job;
    private Contact contact;
    ……
}
```

映射文件

```
<hibernate-mapping>
    <class name="com.Emp" table="emp">
     <id name="id">
                        <generator class="increment"/>
     </id>
     <property name="ename"/>
     <property name="job"/>
     <component name="contact" class="com.Contact">
            <property name="address"/>
            <property name="phone"/>
     </component>
    </class>
</hibernate-mapping>
```

继承映射

- ✓ 继承映射是用OR映射来实现对象模型的继承关系



继承映射示例

- ✓ 继承映射有三种方法，但是继承映射并不常用。
- ✓ 这里只介绍一种方法，就是把这一个继承树三个类映射到数据库的一个表中，这三个类总共是5个属性:id,name,sex,weight,height，但实际上表中是6个字段，因为还要增加一个字段标识是哪一个子类,例如type字段
- ✓ 存储表数据示例 ：

id	name	sex	weight	height	type
1	猪猪	man	380		P
2	啄木鸟	woman		3000	B

实体类

创建实体类：Animal父类

```
package com;

public class Animal {
    private Integer id;
    private String name;
    private String sex;
    ……
}
```

创建实体类：Pig子类

```
package com;
public class Pig extends Animal{
    private Integer weight;
    ……
}
```

创建实体类：Bird子类

```
package com;
public class Bird extends Animal {
    private Integer height;
    ……
}
```

映射文件

com/extends.hbm.xml

```
<hibernate-mapping>
   <class name="com.Animal" table="animal">
      <id name="id">
            <generator class="increment"/>
      </id>
      <!-- 设置鉴别子类的字段 -->
      <discriminator column="type" type="java.lang.String"/>
      <property name="name"/>
      <property name="sex"/>
      <!-- 子类：猪 -->
      <subclass name="com.Pig" discriminator-value="P">
            <property name="weight"/>
      </subclass>
      <!-- 子类：鸟 -->
      <subclass name="com.Bird" discriminator-value="B">
            <property name="height"/>
      </subclass>
   </class>
</hibernate-mapping>
```

测试 插入一个猪的数据和一个鸟的数据

```
Pig pig = new Pig();
pig.setName("猪猪");
pig.setSex("man");
pig.setWeight(350);
session.save(pig);

Bird bird = new Bird();
bird.setName("啄木鸟");
bird.setSex("woman");
bird.setHeight(3500);
session.save(bird);

tx.commit();
```

视图映射

- ✓ 在Hibernate3以后支持视图映射
- ✓ 视图中的所有列组成一个复合主键
- ✓ hibernate在映射数据库视图的时候都会生成2个POJO类，一个是复合主键类，另一个引用这个复合主键类
- ✓ 由于主键值不可为空，视图中如果其中的某个字段为空值，可能会导致java抛出NullPointerException异常，所以视图中的列一定要转换为非空值

创建视图示例

- ✓ 创建一个视图dept_view，存储查询所有的部门部门编号，部门名称，部门中的员工工资总额，部门中的员工平均工资

```
create or replace view dept_view
as
select d.department_id,
       d.department_name,
       nvl(sum(e.salary),0) sum_sal,
       nvl(round(avg(e.salary),2),0) avg_sal
from   departments d,
       employees   e
where  d.department_id = e.department_id(+)
group  by d.department_id,d.department_name;
```

复合主键类

```
public class DeptViewId implements java.io.Serializable {
    private Short departmentId;
    private String departmentName;
    private BigDecimal sumSal;
    private BigDecimal avgSal;

    //getter和setter方法
    ……

    //覆盖equals方法
    ……

    //覆盖hashCode方法
    ……
}
```

实体类

```
public class DeptView  {

    private DeptViewId id;

    //getter和setter方法
    ……

}
```

映射文件

```
<hibernate-mapping>
    <class name="com.cwj.view.DeptView" table="DEPT_VIEW">
        <composite-id name="id" class="com.cwj.view.DeptViewId">
            <key-property name="departmentId" type="java.lang.Short">
                <column name="DEPARTMENT_ID" precision="4" scale="0" />
            </key-property>
            <key-property name="departmentName" type="java.lang.String">
                <column name="DEPARTMENT_NAME" length="30" />
            </key-property>
            <key-property name="sumSal" type="java.math.BigDecimal">
                <column name="SUM_SAL" precision="22" scale="0" />
            </key-property>
            <key-property name="avgSal" type="java.math.BigDecimal">
                <column name="AVG_SAL" precision="22" scale="0" />
            </key-property>
        </composite-id>
    </class>
</hibernate-mapping>
```

查询视图测试

```
public class Test {
	public static void main(String[] args) {
		Session session = HibernateSessionFactory.getSession();

		List<DeptView> list = session.createQuery("from DeptView d").list();

		for (DeptView d : list) {
			System.out.println(d.getId().getDepartmentId() + ","
					+ d.getId().getDepartmentName() + ","
					+ d.getId().getSumSal() + "," + d.getId().getAvgSal());

		}

		session.close();
	}
}
```