By: Anil Kumar

# Hibernate Inheritance

There are three inheritance mapping strategies defined in the hibernate:

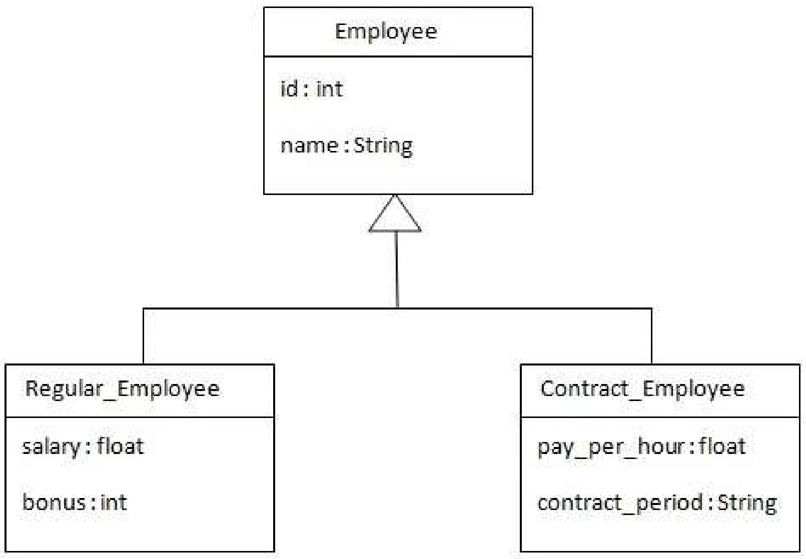
1. Table Per Hierarchy
2. Table Per Concrete class
3. Table Per Subclass

## Table Per Hierarchy

In table per hierarchy mapping, single table is required to map the whole hierarchy, an extra column (known as discriminator column) is added to identify the class. But nullable values are stored in the table.

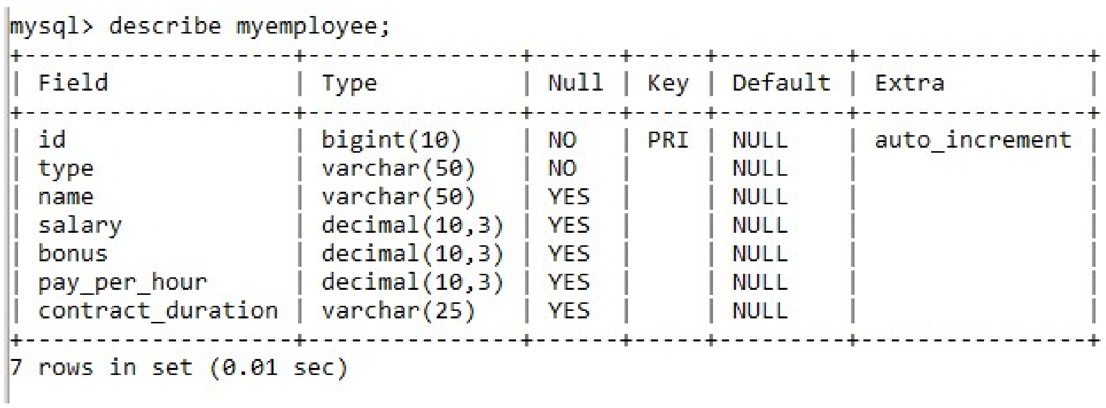
### Hibernate Table per Hierarchy using Annotation:

Let's see the inheritance hierarchy:



There are three classes in this hierarchy. **Employee** is the super class for **Regular_Employee** and **Contract_Employee** classes.

The table structure for this hierarchy is as shown below:



```
mysql> describe myemployee;
+-------------------+---------------+------+-----+---------+----------------+
| Field             | Type          | Null | Key | Default | Extra          |
+-------------------+---------------+------+-----+---------+----------------+
| id                | bigint(10)    | NO   | PRI | NULL    | auto_increment |
| type              | varchar(50)   | NO   |     | NULL    |                |
| name              | varchar(50)   | YES  |     | NULL    |                |
| salary            | decimal(10,3) | YES  |     | NULL    |                |
| bonus             | decimal(10,3) | YES  |     | NULL    |                |
| pay_per_hour      | decimal(10,3) | YES  |     | NULL    |                |
| contract_duration | varchar(25)   | YES  |     | NULL    |                |
+-------------------+---------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

**Super Class MyEmployee**

```
@Entity
@Table(name="myemployee")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="type",discriminatorType=DiscriminatorType.STRIN
G)
@DiscriminatorValue(value="employee")
public class MyEmployee {

        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        @Column(name="id")
        private long id;

        @Column(name="name")
        private String name;
        //setters and getters

}
```

Derive Class RegularEmployee

```
@Entity
@Table(name="myemployee")
@DiscriminatorValue(value="reg_emp")
public class RegularEmployee extends MyEmployee {

        @Column(name="salary")
        private BigDecimal salary;

        @Column(name="bonus")
        private BigDecimal bonus;
        //setters and getters
}
```

## Derive Class ContractEmployee

```
@Entity
@Table(name="myemployee")
@DiscriminatorValue(value="contract_emp")
public class ContractEmployee extends MyEmployee {

        @Column(name="pay_per_hour")
        private BigDecimal pay_per_hour;

        @Column(name="contract_duration")
        private String contract_duration;
        //setters and getters
}
```

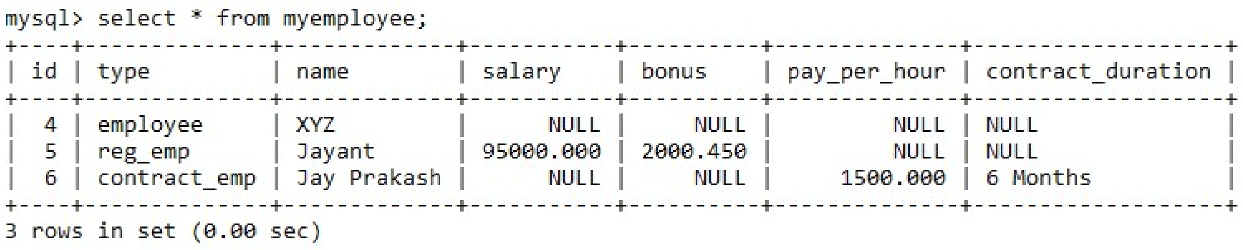
## HibernateInheritanceSingleTableDemo Class

```
public static void main(String[] args) {

                SessionFactory sf=HibernateUtil.getSessionFactory();
                Session session=sf.openSession();
                session.beginTransaction();

                MyEmployee superEmp=new MyEmployee();
                superEmp.setName("XYZ");

                RegularEmployee regEmp=new RegularEmployee();
                regEmp.setName("Jayant");
                BigDecimal _salary = new BigDecimal("95000.00");
                BigDecimal _bonus = new BigDecimal("2000.45");
                regEmp.setSalary(_salary);
                regEmp.setBonus(_bonus);

                ContractEmployee contEmp=new ContractEmployee();
                contEmp.setName("Jay Prakash");
                BigDecimal _pay_perhr = new BigDecimal("1500.00");
                contEmp.setPay_per_hour(_pay_perhr);
                contEmp.setContract_duration("6 Months");

                session.persist(superEmp);
                session.persist(regEmp);
                session.persist(contEmp);

                session.getTransaction().commit();
                session.close();

}
```

OUTPUT:

```
Hibernate: insert into myemployee (name, type) values (?, 'employee')
Hibernate: insert into myemployee (name, bonus, salary, type) values (?, ?, ?, 'reg_emp')
Hibernate: insert into myemployee (name, contract_duration, pay_per_hour, type) values (?, ?, ?, 'contract_emp')
```

Database Table result:


```
mysql> select * from myemployee;
+----+--------------+-------------+-----------+----------+--------------+-------------------+
| id | type         | name        | salary    | bonus    | pay_per_hour | contract_duration |
+----+--------------+-------------+-----------+----------+--------------+-------------------+
|  4 | employee     | XYZ         |      NULL |     NULL |         NULL | NULL              |
|  5 | reg_emp      | Jayant      | 95000.000 | 2000.450 |         NULL | NULL              |
|  6 | contract_emp | Jay Prakash |      NULL |     NULL |     1500.000 | 6 Months          |
+----+--------------+-------------+-----------+----------+--------------+-------------------+
3 rows in set (0.00 sec)
```

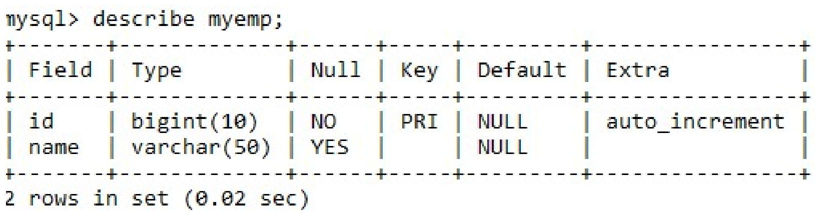
**Query executed by hibernate:**
**MyEmployee myemp=(MyEmployee) session.get(MyEmployee.class, 1L);**

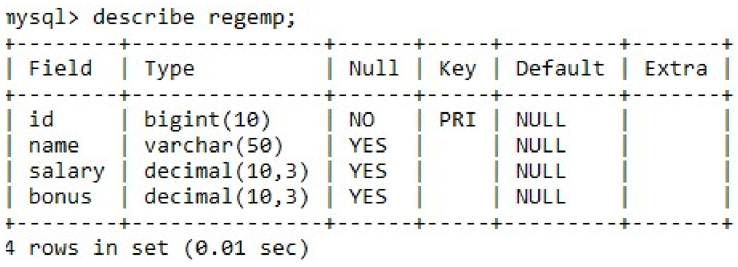
**select** myemployee0_.id as id0_0_, myemployee0_.name as name0_0_, myemployee0_.bonus as bonus0_0_, myemployee0_.salary as salary0_0_, myemployee0_.contract_duration as contract6_0_0_, myemployee0_.pay_per_hour as pay7_0_0_, myemployee0_.type as type0_0_ **from** myemployee myemployee0_ **where** myemployee0_.id=?

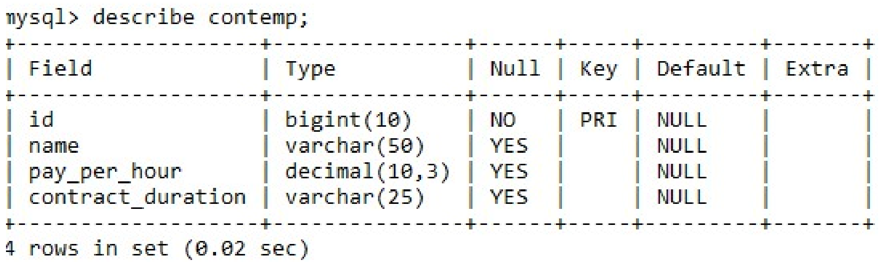
## Table Per Concrete class

In case of table per concrete class, tables are created as per class. But duplicate column is added in subclass tables.

Database tables:


```
mysql> describe myemp;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | bigint(10)  | NO   | PRI | NULL    | auto_increment |
| name  | varchar(50) | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.02 sec)
```


```
mysql> describe regemp;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| id     | bigint(10)    | NO   | PRI | NULL    |       |
| name   | varchar(50)   | YES  |     | NULL    |       |
| salary | decimal(10,3) | YES  |     | NULL    |       |
| bonus  | decimal(10,3) | YES  |     | NULL    |       |
+--------+---------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```


```
mysql> describe contemp;
+-------------------+---------------+------+-----+---------+-------+
| Field             | Type          | Null | Key | Default | Extra |
+-------------------+---------------+------+-----+---------+-------+
| id                | bigint(10)    | NO   | PRI | NULL    |       |
| name              | varchar(50)   | YES  |     | NULL    |       |
| pay_per_hour      | decimal(10,3) | YES  |     | NULL    |       |
| contract_duration | varchar(25)   | YES  |     | NULL    |       |
+-------------------+---------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

**Employee -> myemp ; RegularEmployee -> regemp ; ContractEmployee -> contemp**

In case of Table Per Concrete class, tables are created per class. So there are no nullable values in the table. Disadvantage of this approach is that duplicate columns are created in the subclass tables.

Here, we need to use @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) annotation in the parent class and @AttributeOverrides annotation in the subclasses.

**@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)** specifies that we are using table per concrete class strategy. It should be specified in the parent class only.

**@AttributeOverrides** defines that parent class attributes will be overriden in this class. In table structure, parent class table columns will be added in the subclass table.

**Super Class MyEmployee**

```
@Entity
@Table(name="myemp")
@Inheritance(strategy=InheritanceType.TABLE_PER_CLASS)
public class MyEmployee2 {

        @Id
        @Column(name="id")
        private long id;

        @Column(name="name")
        private String name;
        //setters and getters

}
```

**Derive class RegularEmployee**

```
@Entity
@Table(name="regemp")
@AttributeOverrides({
              @AttributeOverride(name="id",column=@Column(name="id")),
              @AttributeOverride(name="name",column=@Column(name="name"))

})
public class RegularEmployee2 extends MyEmployee2 {

        @Column(name="salary")
        private BigDecimal salary;

        @Column(name="bonus")
        private BigDecimal bonus;
        //setters and getters
}
```

**Derive class ContractEmployee**

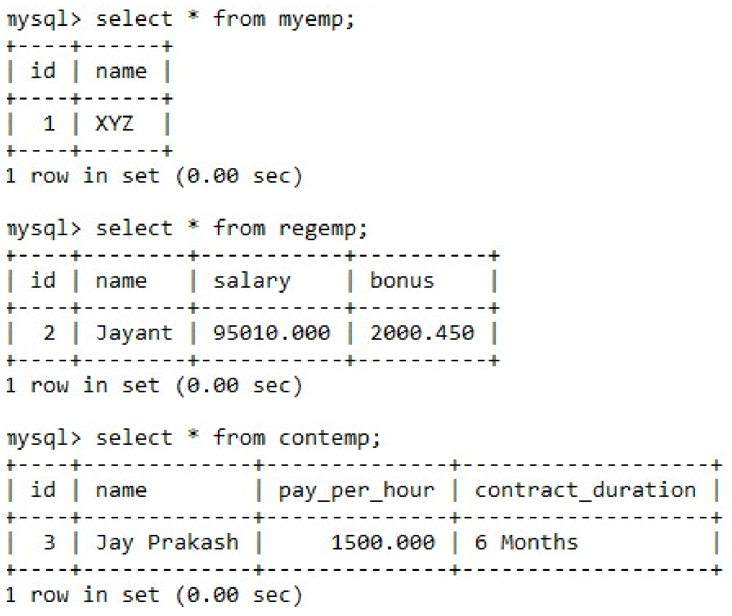
```
@Entity
@Table(name="contemp")
@AttributeOverrides({
        @AttributeOverride(name="id",column=@Column(name="id")),
        @AttributeOverride(name="name",column=@Column(name="name"))
})
public class ContractEmployee2 extends MyEmployee2 {

        @Column(name="pay_per_hour")
        private BigDecimal pay_per_hour;

        @Column(name="contract_duration")
        private String contract_duration;
        //setters and getters
}
```

**HibernateInheritanceTablePerClassDemo class**

```
public static void main(String[] args) {

        SessionFactory sf=HibernateUtil.getSessionFactory();
        Session session=sf.openSession();
        session.beginTransaction();

        MyEmployee2 superEmp=new MyEmployee2();
        superEmp.setId(1);
        superEmp.setName("XYZ");

        RegularEmployee2 regEmp=new RegularEmployee2();
        regEmp.setId(2);
        regEmp.setName("Jayant");
        BigDecimal _salary = new BigDecimal("95010.00");
        BigDecimal _bonus = new BigDecimal("2000.45");
        regEmp.setSalary(_salary);
        regEmp.setBonus(_bonus);

        ContractEmployee2 contEmp=new ContractEmployee2();
        contEmp.setId(3);
        contEmp.setName("Jay Prakash");
        BigDecimal _pay_perhr = new BigDecimal("1500.00");
        contEmp.setPay_per_hour(_pay_perhr);
        contEmp.setContract_duration("6 Months");

        session.persist(superEmp);
        session.persist(regEmp);
        session.persist(contEmp);

        session.getTransaction().commit();
        session.close();
}
```

**OUTPUT:**

Hibernate: insert into myemp (name, id) values (?, ?)
Hibernate: insert into regemp (name, bonus, salary, id) values (?, ?, ?, ?)
Hibernate: insert into contemp (name, contract_duration, pay_per_hour, id) values (?, ?, ?, ?)
Database table result:



```
mysql> select * from myemp;
+----+------+
| id | name |
+----+------+
|  1 | XYZ  |
+----+------+
1 row in set (0.00 sec)

mysql> select * from regemp;
+----+--------+-----------+----------+
| id | name   | salary    | bonus    |
+----+--------+-----------+----------+
|  2 | Jayant | 95010.000 | 2000.450 |
+----+--------+-----------+----------+
1 row in set (0.00 sec)

mysql> select * from contemp;
+----+-------------+--------------+-------------------+
| id | name        | pay_per_hour | contract_duration |
+----+-------------+--------------+-------------------+
|  3 | Jay Prakash |     1500.000 | 6 Months          |
+----+-------------+--------------+-------------------+
1 row in set (0.00 sec)
```

**Query executed by hibernate:**
**MyEmployee2 myemp=(MyEmployee2) session.get(MyEmployee2.class, 1L);**

**select** myemployee0_.id as id0_0_, myemployee0_.name as name0_0_, myemployee0_.bonus as bonus1_0_, myemployee0_.salary as salary1_0_, myemployee0_.contract_duration as contract1_2_0_, myemployee0_.pay_per_hour as pay2_2_0_, myemployee0_.clazz_ as clazz_0_ **from** ( select id, name, null as bonus, null as salary, null as contract_duration, null as pay_per_hour, 0 as clazz_ **from** myemp **union** select id, name, bonus, salary, null as contract_duration, null as pay_per_hour, 1 as clazz_ from regemp **union** select id, name, null as bonus, null as salary, contract_duration, pay_per_hour, 2 as clazz_ from contemp ) myemployee0_ where myemployee0_.id=?

## Table Per Subclass

In this strategy, tables are created as per class but related by foreign key. So there are no duplicate columns.

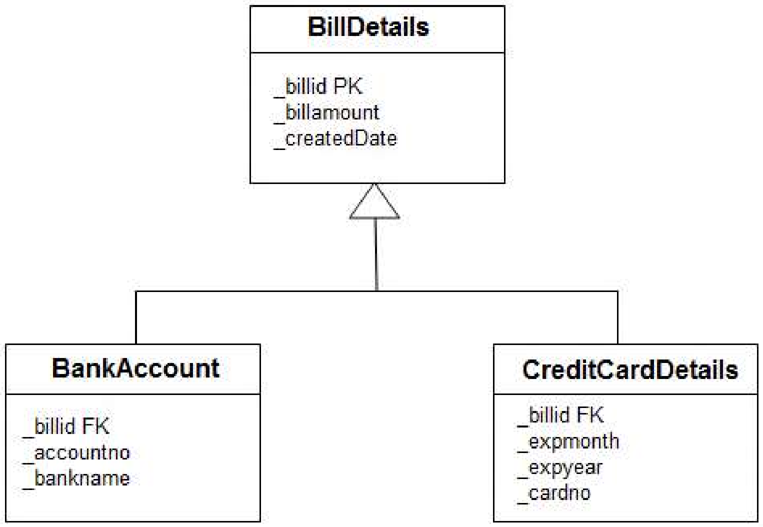
As we have specified earlier, in case of table per subclass strategy, tables are created as per persistent classes but they are related using primary and foreign key. So there will not be duplicate columns in the relation.

We need to specify **@Inheritance(strategy=InheritanceType.JOINED)** in the parent class and **@PrimaryKeyJoinColumn**annotation in the subclasses.

By: Anil Kumar

We have an abstract **BillDetails** class with their implementor are **CreditCardAccount**and **BankAccount** respectively.

**Class Diagram for inheritance:**



Database Tables:

```
mysql> describe BillDetails;
+-------------+---------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra          |
+-------------+---------------+------+-----+---------+----------------+
| billid      | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| Bill_Amt    | decimal(10,2) | YES  |     | NULL    |                |
| createdDate | date          | YES  |     | NULL    |                |
+-------------+---------------+------+-----+---------+----------------+
3 rows in set (0.02 sec)

mysql> describe BankAccount;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| account   | varchar(255) | YES  |     | NULL    |       |
| Bank_Name | varchar(255) | YES  |     | NULL    |       |
| billid    | bigint(20)   | NO   | MUL | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

mysql> describe CreditCardAccount;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| exp_Month   | varchar(255) | YES  |     | NULL    |       |
| exp_Year    | varchar(255) | YES  |     | NULL    |       |
| Card_Number | varchar(255) | YES  |     | NULL    |       |
| billid      | bigint(20)   | NO   | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.14 sec)
```

## Entity class BillDetails

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

        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        @Column(name="billid")
        private long id;

        @Column(name="Bill_Amt")
        private BigDecimal billamount;

        @Column(name="createdDate")
        private Date createdDate;
        //setters and getters
}
```

## Entity class BankAccount

```
@Entity
@Table(name="BankAccount")
@PrimaryKeyJoinColumn(name="billid")
public class BankAccount extends BillDetails {

        @Column(name="account")
        private String accountno;

        @Column(name="Bank_Name")
        private String bankname;
        //setters and getters
}
```

## Entity class CreditCardAccount

```
@Entity
@Table(name="CreditCardAccount")
@PrimaryKeyJoinColumn(name="billid")
public class CreditCardAccount extends BillDetails {
        @Column(name="exp_Month")
        private String expmonth;
        @Column(name="exp_Year")
        private String expyear;
        @Column(name="Card_Number")
        private String cardno;
        //setters and getters
}
```

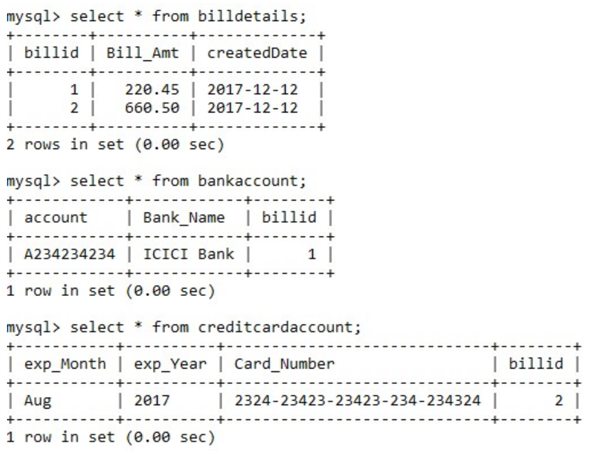
## HibernateInheritanceTablePerSubclassDemo Class:

```
public static void main(String[] args) {
              SessionFactory sf=HibernateUtil.getSessionFactory();
              Session session=sf.openSession();
              session.beginTransaction();


              BankAccount bankAccount1 = new BankAccount();
              bankAccount1.setCreatedDate(new Date());
              bankAccount1.setAccountno("A234234234");
              bankAccount1.setBankname("ICICI Bank");

              BigDecimal billamnt=new  BigDecimal(220.45);
              bankAccount1.setBillamount(billamnt);

              CreditCardAccount cardAccount1 = new CreditCardAccount();
              cardAccount1.setCreatedDate(new Date());
              cardAccount1.setExpmonth("Aug");
              cardAccount1.setExpyear("2017");

              BigDecimal billamnt2=new  BigDecimal(660.50);
              cardAccount1.setBillamount(billamnt2);
              cardAccount1.setCardno("2324-23423-23423-234-234324");

              session.save(bankAccount1);
              session.save(cardAccount1);

              session.getTransaction().commit();
              session.close();
}
```

OUTPUT:

```
Hibernate: insert into BillDetails (Bill_Amt, createdDate) values (?, ?)
Hibernate: insert into BankAccount (account, Bank_Name, billid) values (?, ?, ?)
Hibernate: insert into BillDetails (Bill_Amt, createdDate) values (?, ?)
Hibernate: insert into CreditCardAccount (Card_Number, exp_Month, exp_Year, billid)
values (?, ?, ?, ?)
```

Database Tables:



```
mysql> select * from billdetails;
+--------+----------+-------------+
| billid | Bill_Amt | createdDate |
+--------+----------+-------------+
|      1 |   220.45 | 2017-12-12  |
|      2 |   660.50 | 2017-12-12  |
+--------+----------+-------------+
2 rows in set (0.00 sec)

mysql> select * from bankaccount;
+------------+------------+--------+
| account    | Bank_Name  | billid |
+------------+------------+--------+
| A234234234 | ICICI Bank |      1 |
+------------+------------+--------+
1 row in set (0.00 sec)

mysql> select * from creditcardaccount;
+-----------+----------+-----------------------------+--------+
| exp_Month | exp_Year | Card_Number                 | billid |
+-----------+----------+-----------------------------+--------+
| Aug       | 2017     | 2324-23423-23423-234-234324 |      2 |
+-----------+----------+-----------------------------+--------+
1 row in set (0.00 sec)
```

Fetch the record of BillDetails:

**BillDetails billAccountDetails1 = (BillDetails)session.get(BillDetails.class,1L);**

**Query executed by hibernate:**

**select** billdetail0_.billid as billid0_0_, billdetail0_.Bill_Amt as Bill2_0_0_, billdetail0_.createdDate as createdD3_0_0_, billdetail0_1_.Card_Number as Card1_1_0_, billdetail0_1_.exp_Month as exp2_1_0_, billdetail0_1_.exp_Year as exp3_1_0_, billdetail0_2_.account as account2_0_, billdetail0_2_.Bank_Name as Bank2_2_0_, case when billdetail0_1_.billid is not null then 1 when billdetail0_2_.billid is not null then 2 when billdetail0_.billid is not null then 0 end as clazz_0_ **from** BillDetails billdetail0_ **left outer join** CreditCardAccount billdetail0_1_ on billdetail0_.billid=billdetail0_1_.billid **left outer join** BankAccount billdetail0_2_ on billdetail0_.billid=billdetail0_2_.billid where billdetail0_.billid=?

**Thank You!!!**

**Happy Learning!**