Basic ORM
Bulk Operations
Advanced ORM
 a. Composition Mapping(Granularity)
 b. Collection Mapping
        Indexed Collection   :: List<?>,Map<?,?>
        Non-Indexed Collection :: Set<?>
 c. Inheritance Mapping
        Table Per Class (TBC)
        Table Per Subclass(TBS)
        Table Per Concrete Class(TBCC)
d. Association Mapping
        1---1,1----* | uni-direction
        ---1,----* | bi-direction

Collection Mapping

a. List<?>
  1. @ElementCollection
     @Column(name="")

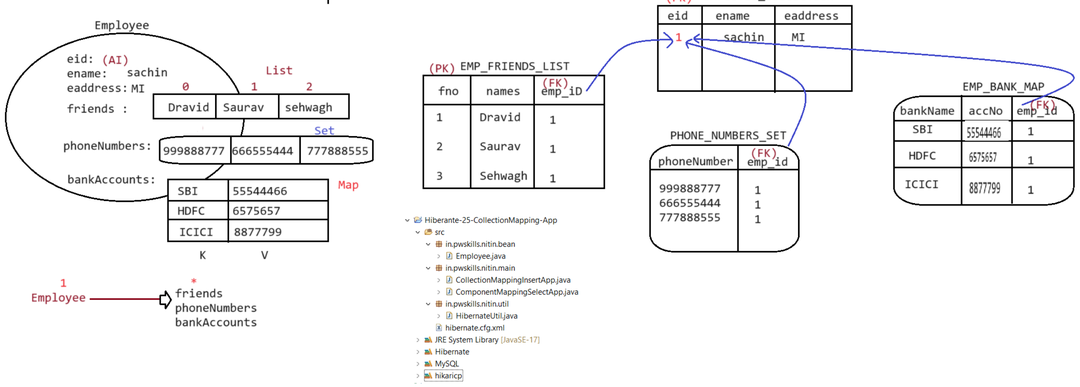
  2. @CollectionTable(name='',
                      joinColumns = {
                                     @JoinColumn(name='',referencedColumn=''
                                    }
)
  3. @OrderColumn(name="")
     @ListIndexBase(value=)

b. Set<?>
1. @ElementCollection
   @Column(name="")

2. @CollectionTable(name='',
                    joinColumns = {
                                   @JoinColumn(name='',referencedColumn=''
                                  }
)

c. Map<?,?>
1. @ElementCollection
   @Column(name="")

2. @CollectionTable(name='',
                    joinColumns = {
                                   @JoinColumn(name='',referencedColumn=''
                                  }
)
3. @MapKeyColumn(name='')



(AI)
(PK) EMPLOYEE_COLLECTION

eid	ename	eaddress
1	sachin	MI

(PK) EMP_FRIENDS_LIST

fno	names	(FK) emp_iD
1	Dravid	1
2	Saurav	1
3	Sehwagh	1

PHONE_NUMBERS_SET

phoneNumber	(FK) emp_id
999888777	1
666555444	1
777888555	1

EMP_BANK_MAP

bankName	accNo	(FK) emp_id
SBI	55544466	1
HDFC	6575657	1
ICICI	8877799	1

inheritance mapping

java
2 entity classes can be kept in inheritance

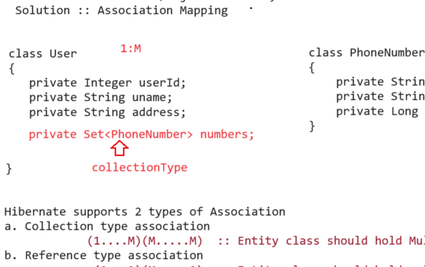
RDBMS
2 tables can't be kept in inheritance

solution:: Inheritance Mapping

1. Table Per class
2. Table Per Subclass
3. Table Per Concreteclass

Table Per Class

```
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "payment_mode",
                     discriminatorType = DiscriminatorType.STRING,
                     length=20)
```




```
@Entity
@DiscriminatorValue
    (value = "card")
```

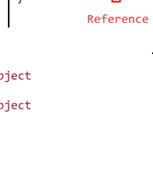
```
@Entity
@DiscriminatorValue
     (value="cheque")
```

PAYMENT_TBCH

(pk) pid	amount	cardNo	cardType	paymentGateway	chequeNo	chequeType	expiryDate	payment_mode
1	8000	1234	debit	visa	—	—	—	card
2	9000	—	—	—	56456	all	10-oct-21	cheque

Payment CardPayment ChequePayment

Retrieval Operations internally used 'mode'



While working with retrieval operation using Parent class we get the details only of Child classes, but not the Parent class.

Limitations
 a. Table will be designed with Huge no of columns
 b. columns will be marked with null values (memory-issue)
 c. dynamically if we link more class, more columns in table
    creates issued in production environment.
 d. lengthy db table not a good practise (not industry standard)

Table Per SUBCLASS

```
@Entity
@Table(name = "PAYMENT_TBSC")
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn
  (name = "payment_mode",
   discriminatorType =DiscriminatorType.STRING,
   length = 20
  )
```



```
@Entity
@Table(name="CARDPAYMENT_TBSC")
@DiscriminatorValue(value = "card")
@PrimaryKeyJoinColumn(name = "payment_id",
                      referencedColumnName = "pid")
```

```
@Table(name="CHEQUEPAYMENT_TBSC")
@Entity
@DiscriminatorValue(value="cheque")
@PrimaryKeyJoinColumn(name = "payment_id",
                      referencedColumnName = "pid")
```

PAYMENT_TBSC

(pk) pid	amount	payment_mode
1	8000	card
2	9000	cheque

CARDPAYMENT_TBSC

cardNo	cardType	paymentGateway	payment_id(fk)
1234	debit	visa	1

CHEQUEPAYMENT_TBSC

chequeNo	chequeType	expiryDate	payment_id(fk)
56456	all	10-oct-21	2

TABLE PER CONCRETE CLASS

```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
```



```
@Entity
@Table(name="CARD_PAYMENT_TBCC")
```

```
@Entity
@Table(name="CHEQUE_PAYMENT_TBCC")
```

CARD_PAYMENT_TBCC

(PK) pid	amount	cardNo	cardType	paymentGateWay
1	8000	1234	debit	visa

CHEQUE_PAYMENT_TBCC

(PK) pid	amount	chequeNo	chequeType	expriyDate
2	90000	56456	all	10-oct-21

Association Mapping

Java
 a. IS-A (Inheritance Mapping)
 b. HAS-A (Composition Mapping)

RDBMS

The relationship b/w 2 classes in java and the relationship b/w 2 tables in java are completely different

There is a mismatch b/w java and RDBM, This mismatch is called "Association MisMatch".
Solution :: Association Mapping

1:M

```
class User
{
   private Integer userId;
   private String uname;
   private String address;
   private Set<PhoneNumber> numbers;
}
```
collectionType

```
class PhoneNumber
{
    private String numberType;
    private String serviceProvider;
    private Long phoneNo;
}
```

1 1

```
class Employee
{
    private Integer eid;
    private String ename;
    private String eaddress;
    AdharInfo adhar;
}
```
Reference Type

```
class AdharInfo
{
   private Long adharNo;
   private String adharType;
   private String adharAddress;
}
```

Hibernate supports 2 types of Association
a. Collection type association
        (1....M)(M.....M) :: Entity class should hold Multiple Associated Object
b. Reference type association
        (1...1)(M.....1) :: Entity class should hold only one Associated Object

Example
  a. 1----1 (Person---->Passport)
  b. 1----M (Person<----> VISA)
  c. M----1 (VISA <------> Person)
            (Phone-No<-----> Person)
            (Employee<----> Department)
  d. M----M (Teacher<---->STUDENT)
            (Patient<----->DOCTOR)

Hibernate supports 2 modes of association Mapping
 a. Uni-Directional
 b. Bi-Directional

Uni-Directional :: We can access child objects from parent objects
                   or
                parent objects from child objects

Bi-Directional :: We can access child objects from parent objects
                  and
                parent objects from child objects

Annotations driven Configuration

1. @OneToOne
2. @OneToMany
3. @ManyToOne
4. @ManyToMany