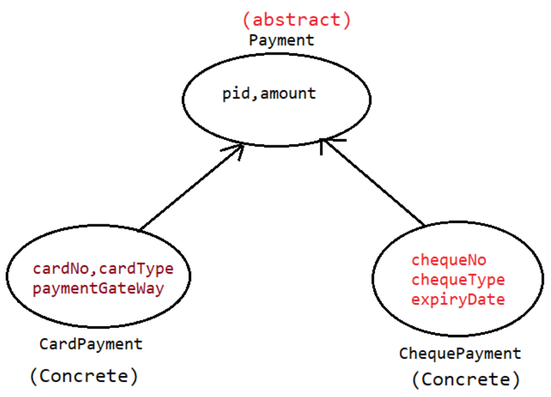
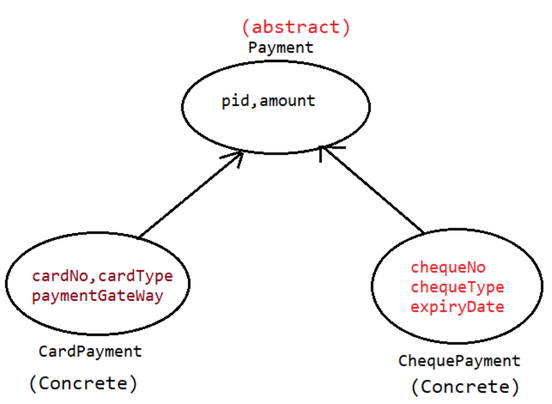
SINGLE_TABLE (DiscrimatorColumn)



Descriminatorcolumn

payment_mode	pid	amount	cardNo	cardType	paymentGateway	chequeNo	chequeType	expriyDate
CARD	1234	8000.5	45678	Debit	visa	*(NULL)*	*(NULL)*	*(NULL)*
CHEQUE	1235	4500.56	*(NULL)*	*(NULL)*	*(NULL)*	546345	self	0023-04-27

TABLE_PER_CLASS (1:1)



Discriminatorcolumn

PAYMENT_TBSC

payment_mode	pid	amount
CARD	1234	8000.5
CHEQUE	1235	4500.56

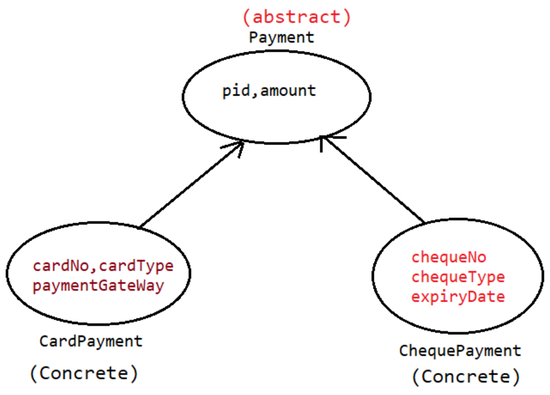
CARDPAYMENT_TBSC (FK)

cardNo	cardType	paymentGateway	payment_id
45678	Debit	visa	1234

(FK)

chequeNo	chequeType	expriyDate	payment_id
546345	self	0023-04-27	1235

TABLE PER CONCRETE CLASS



CHEQUE_PAYMENT_TBCC

pid	amount	chequeNo	chequeType	expriyDate
1235	4500.56	546345	self	0023-04-27

CARDPAYMENT_TBCC

pid	amount	cardNo	cardType	paymentGateway
1234	8000.5	45678	Debit	visa

CollectionMapping (1:M)

```
@ElementCollection
@Column(name="friend_name")
@CollectionTable(name = "EMP_FRNDS", joinColumns = @JoinColumn(name = "emp_id", referencedColumnName = "eid"))
@OrderColumn(name = "friend_no")
@ListIndexBase(value = 1)
private List<String> friendList;
```

0	1	2
saruav	dravid	sehwagh

EMP_FRNDS

friend_no	friend_name	emp_id (fk)
1	saurav	10
2	dravid	10
3	sehwagh	10

EMPLOYEE_COLLECTION

eid	eaddress	ename
10	MI	sachin

```
@ElementCollection
@Column(name="mobile_no")
@CollectionTable(name = "EMP_PHONE", joinColumns = @JoinColumn(name = "emp_id", referencedColumnName = "eid"))
private Set<Long> phones;
```

999888877	777666777	555666777

EMP_PHONE

mobile_no	emp_id
999888877	10
777666777	10
555666777	10

```
@ElementCollection
@Column(name="account_number")
@CollectionTable(name = "EMP_ACCOUNTS", joinColumns = @JoinColumn(name = "emp_id", referencedColumnName = "eid"))
@MapKeyColumn(name = "bankName",length = 10)
private Map<String,Long>bankAccounts;
```

K	V
SBI	112233
HDFC	223344
ICICI	112244

EMP_ACCOUNTS

bankName	account_number	emp_id
SBI	112233	10
HDFC	223344	10
ICICI	112244	10

OR-Mapping
      a. Composition Mapping(HAS-A) ======> @Embeded,@Embedable
      b. Inheritance Mapping(IS-A)
      c. Collection Mapping(List,Set,Map)
      d. Association Mapping
                  a. 1...*   b. *.....1
                  c. 1...1   d. *......*

Inheritance Mapping(IS-A)
==========================
1.@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
2.@Inheritance(strategy = InheritanceType.JOINED)
3.@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
4.@DiscriminatorColumn
5.@DisciriminatorValue
6.@PrimaryKeyJoinColumn

Table per Class(single table)

Inheritance Mapping
----
1. TABLE PER CLASS HIERARCHY:
This design provides single table for all model classes. It will considers all classes variables as column names and takes one extra column “Discriminator” which provides information like “Row related to which model class”.
=> Discriminator column can be int type or String/char type.

For IS-A Relation Mapping enum and Annotation are given as:
enum: InheritanceType
Annotation: @Inheritance

For TABLE PER CLASS HIERATCHY DESIGN CODE:
@Inheritance(strategy= InheritanceType.SINGLE_TYPE)
For single table design, we should also provide extra column ‘name,type and value’ using code:
Annotation.: @DiscriminatorColumn
Enum : DiscriminatorType

Code look like:
@DiscriminatorColumn(name=”obType”,discriminatorType= DiscriminatorType.STRING)

For object value code is:
@DiscriminatorValue(“-----------“)

Code

```
@Entity
@Table(name = "commonTab")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "objType", discriminatorType =
DiscriminatorType.STRING)
@DiscriminatorValue(value = "STD")
public class Student {

}
```

```
@Entity
@DiscriminatorValue(value = "ADD")
public class Address extends Student {

}

@Entity
@DiscriminatorValue(value = "CLS")
public class Classes extends Student{

}
```

TABLE PER SUB CLASS:-
=> In this case hibernate creates table for every child case along with parent class.
=> On saving data, parent data stored at parent table and child data stored at child table.
=> Parent table and child table is connected using PK-FK column. FK column also called as key column.

Code

Payment.java
============

```
@Entity
@DiscriminatorColumn(name = "paymentmode", length = 10)
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Payment implements Serializable {
	@Id
	@GeneratedValue
	private Integer pid;
}
```

CardPayment
===========

```
@Entity
@PrimaryKeyJoinColumn(name = "payment_id",referencedColumnName = "pid")
@DiscriminatorValue("CARD")
public class CardPayment extends Payment {
	private Long cardNo;
	private String cardType;
	private String paymentGatewa
}
```

ChequePayment
=============

```
@Entity
@DiscriminatorValue("cheque")
@PrimaryKeyJoinColumn(name = "payment_id",referencedColumnName = "pid")
public class ChequePayment extends Payment {
	private Integer chequeNo;
	private String chequeType;
	private LocalDate expiryDate;

}
```

TestApp.java
============

```
CardPayment payment = new CardPayment();
      payment.setAmt(45000.0f);
      payment.setCardNo(24567L);
      payment.setCardType("credit");
      payment.setPaymentGateway("VISA");

ChequePayment payment2 = new ChequePayment();
      payment2.setAmt(9000f);
      payment2.setChequeNo(15744);
      payment2.setChequeType("self");
      payment2.setExpiryDate(LocalDate.of(2021, 10, 21));

      session.save(payment);
      session.save(payment2);
```

TABLE PER CONCRETE CLASS
This design creates independent tables for every class in IS-A relation including parent class variable.

```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Payment implements Serializable {

      private static final long serialVersionUID = 1L;

      @Id
      @GeneratedValue
      private Integer pid;
      private float amt;

}

@Entity
public class ChequePayment extends Payment {
      private static final long serialVersionUID = 1L;

      private Integer chequeNo;
      private String chequeType;
      private LocalDate expiryDate;

}

@Entity
public class CardPayment extends Payment {
      private static final long serialVersionUID = 1L;

      private Long cardNo;
      private String cardType;
      private String paymentGateway;

}
```

TestApp.java
============

```
CardPayment payment = new CardPayment();
      payment.setAmt(45000.0f);
      payment.setCardNo(24567L);
      payment.setCardType("credit");
      payment.setPaymentGateway("VISA");
```

```
ChequePayment payment2 = new ChequePayment();
      payment2.setAmt(9000f);
      payment2.setChequeNo(15744);
      payment2.setChequeType("self");
      payment2.setExpiryDate(LocalDate.of(2021, 10, 21));

      session.save(payment);
      session.save(payment2);

Output
create table CardPayment (
       pid integer not null,
        amt float not null,
        cardNo bigint,
        cardType varchar(255),
        paymentGateway varchar(255),
        primary key (pid)
    ) engine=InnoDB

create table ChequePayment (
       pid integer not null,
        amt float not null,
        chequeNo integer,
        chequeType varchar(255),
        expiryDate date,
        primary key (pid)
    ) engine=InnoDB

Note:since the Payment class is abstract and we are not doing any operation,db
table won't be created for Payment class.
     since new table is created for every class,we don't need any discriminator
value.

Collection Mapping in Hibernate:
=> It is all about taking collection type properties with Strings/Wrappers in
Entity class.
=> For every collection property one seperate child table will be created having FK
pointing PK Col of Entity class db table(parent table).
=> The child table for Collection property(List/Set/Map) will have min 2
columns(Set) and max 3 columns(List/Map).
=> Set Collection child table contains
           a. element column  b. foreign key column(no index column)
=> List/Map  collection child table contains
           a. element column  b. foreign key column  c. index column.


Note: Collection are of two types
           a. Indexed Collection     => List/Map
           b. Non-Indexed Collection => Set

On every collection property we must add
      a. @ElementCollection=> Specifying element column name.
      b. @CollectionTable  => Specify child table name and FK column name.
      c. @OrderColumn      => To specify the list index column name.
      d. @MapKeyColumn     => To specify the map index column name.
```