Table per class hierarchy #1 discriminator-value: IP

```
class InsurancePlan {
  int planNo;
  String planName;
  int tenure;
  double premiumAmount;
  double insurredAmount;
  String status;
  // accessors
}
```

```
insurance_plan
plan_no
plan_nm
tenure
premium_amt
insurred_amt
status
copay
coverage_type
disability_coverage_percentage
international_coverage
insurance_plan_type ---------- discriminator-column
```

discriminator-value: MIP #10

```
class MedicalInsurancePlan
extends InsurancePlan{
  int copay;
  String coverageType;
  // accessors
}
```

discriminator-value: AIP #11

```
class AccidentalInsurancePlan extends
InsurancePlan {
  double disabilityCoveragePercentage;
  boolean internationalCoverage;
  // accessors
}
```

plan_no	plan_nm	tenure	premium_amt	insurred_amt	status	copay	coverage_type	disability_coverage_percentage	international_coverage	insurance_plan_type
1	IP1	36	2000	800000	A	NULL	NULL	NULL	NULL	IP
10	MIP1	12	2000	300000	A	10	In-Patient	NULL	NULL	MIP
11	AIP1	24	2500	400000	A	NULL	NULL	50	0	AIP

---

1. map the classes interms of tables
2. evaluate/compute how to perform persistence operations for these entity classes into the database tables
   2.1 SUPERCLASS
    - store
    - retrieve
   2.2 SUBCLASS
    - store
    - retrieve

SUBCLASS
- SAVING
we store all the attributes of data of a Subclass into the Single table (insurance_plan), but while storing the data of an subclass, we store null values for the other columns representing the subclasses of the hierarchy
For eg.. while storing the MedicalInsurancePlan object into the table, we store NULL values for the columns disability_coverage_percentage & international_coverage pertaining to AccidentalInsurancePlan entity class
While Storing any of the entity classes within the hierarchy into the table, additionally we need to store discriminator-column with appropriate value based on the type of the entity of the entity class we are persisting.

```
MedicalInsurancePlan mip = new MedicalInsurancePlan();
mip.setPlanNo(10)
mip.setPlanName("MIP1");
mip.setTenure(12);
mip.setPremiumAmount(2000);
mip.setInsurredAmount(300000);
mip.setStatus("A");
mip.setCopay(10);
mip.setCoverageType("In-Patient");
// discriminator-value=MIP

AccidentalInsurancePlan aip = new AccidentalInsurancePlan();
aip.setPlanNo(11)
aip.setPlanName("AIP1");
aip.setTenure(24);
aip.setPremiumAmount(2500);
aip.setInsurredAmount(400000);
aip.setStatus("A");
aip.setDisabilityCoveragePercentage(50);
aip.setInternationalCoverage(false);
// discriminator-value=AIP
```

SUPERCLASS
SAVING

```
InsurancePlan ip = new InsurancePlan();
ip.setPlanNo(1)
ip.setPlanName("IP1");
ip.setTenure(36);
ip.setPremiumAmount(2000);
ip.setInsurredAmount(800000);
ip.setStatus("A");
// save // discriminator-value=IP
```

The Superclass object will be stored into the SINGLE table "insurance_plan", but for all the columns that belongs to the Subclasses we need to store NULL values, since those values doesnt applies to Superclass object

---

QUERY/READ/RETRIEVE THE OBJECTS

SUB-CLASS:
While fetching the data for a subclass
1. One subclass record cannot be accessed as a reference type of another subclass
2. similary a superclass record cannot be expressed interms of Subclass
So to identify whether a record belongs to that specific subclass or not, while querying the data for a subclass we need use discriminator-value of that subclass to identify whether the record belongs to that subclass or not.

For eg.. while querying the data for MedicalInsurancePlan we need query based on discriminator-column-value pertaining to the entity class as "MIP" if the record carries MIP as the value for that discriminator-column then only the record belongs to that subclass, otherwise we should not fetch the record interms of that Subclass.

```
MedicalInsurancePlan mip = session.get(MedicalInsurancePlan.class, 11);
```

In the above case, even though there exist a record with plan_no = 11 in insurance_plan table, it doesnt belongs to MedicalInsurancePlan, we identify this based on discriminator-value stored for that record and ignore fetching it as below.

```
select * from insurance_plan where plan_no = ? and insurance_plan_type='MIP';
```

In this case the above query returns empty ResultSet so that the get() method returns NULL, which is correct, because that record doesnt belongs to medicalInsurancePlan

```
MedicalInsurancePlan mip = session.get(MedicalInsurancePlan.class, 10);
```

How to fetch the data for a superclass object Type?

```
InsurancePlan ip = session.get(InsurancePlan.class, 1);
```

it looks like for superclass also we can query the data from single_table (insurance_plan) based on id and discriminator-column value.

But to support polymorphic retrieval, the above way of querying the data for superclass it not recommended. Polymorphic retrieval means: any of the subclass objects also can be expressed interms of superclass reference type, which means

we can query MedicalInsurancePlan or AccidentalInsurancePlan object types through InsurancePlan reference type as below

```
InsurancePlan ip = session.get(InsurancePlan.class, 10); = this should return MedicalInsurancePlan object and store in superclass referenceType InsurancePlan.
```

So if we are querying the data for a superclass reference type with discriminator-column value as ="IP" then we cannot fetch subclass types interms of Superclass.

So while querying the data for superclass type, dont use discriminator-column value, just query based on primary key value only

```
select * from insurance_plan where plan_no = ?
```

one the record has been fetched, then look at the discriminator-column value
if the value is
IP = create insurancePlan object and return
MIP = create MedicalInsurancePlan object
AIP = create AccidentalInsurancePlan object

so that we support polymorphic retrieval.