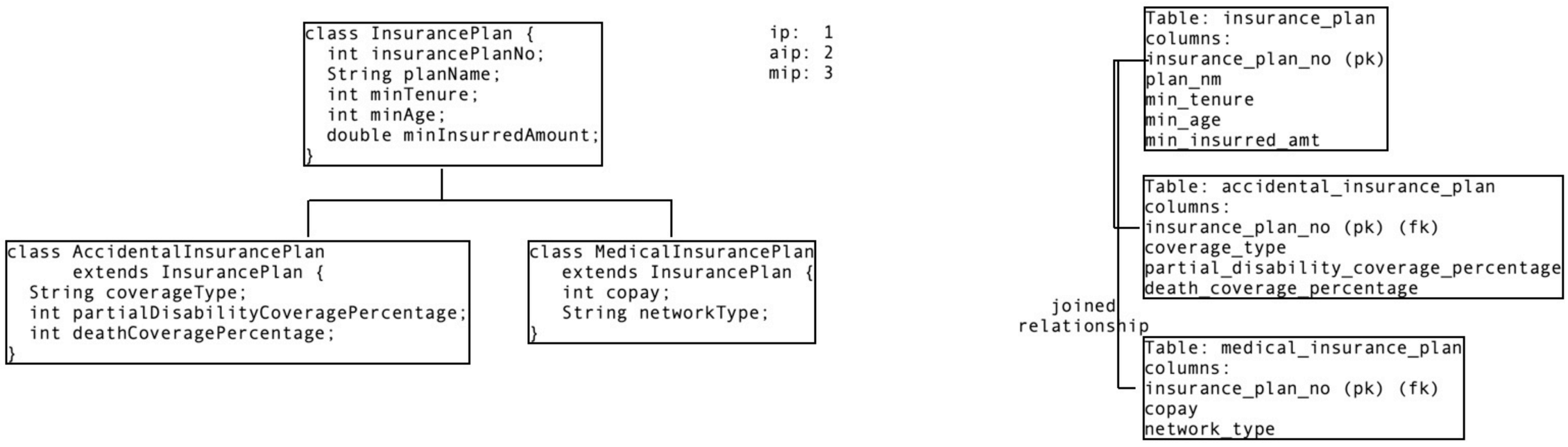
Table per subclass
Here we are creating per each subclass, one sub-class table in joined relationship with super class table. So that we can persist the data of any of the classes in the hierarchy within these tables.



Let us understand how to perform persistence operations for these hierarchy of classes based on the table model we considered.

#1 Persist:
1. Superclass object (InsurancePlan):
directly store the super class object attributes into super class represented table. here InsurancePlan class object into insurance_plan table

2. Subclass object (AccidentalInsurancePlan/MedicalInsurancePlan):
Inorder to store the data of an Subclass object we need to do the following:
  2.1 take the superclass or inherited attributes data from the sub-class object and store them into the super class table
  2.2 then pick the primary key column value of the super class record we inserted
  2.3 store that primary key as an pk and fk in the sub-class represented table along with sub-class attributes

---

#2 Fetch or Get:
1. Subclass object (AccidentalInsurancePlan or MedicalInsurancePlan):
AccidentalInsurancePlan aip = session.get(AccidentalInsurancePlan.class, 2);

While querying the data for a sub-class, we should query the data from the corresponding sub-class table and the super-class table using join query. Only when the record exists within both the tables (sub-class table and super-class table) then only the record belongs to that sub-class object.

So while querying the data we need to perform inner join to fetch the data only when it exists in both the tables.

For eg.. we want to fetch AccidentalInsurancePlan object of id: 2 from the database tables. Here we need to query the data by joining accidental_insurance_plan and insurance_plan using inner join as below.

```
select * from insurance_plan ip inner join accidental_insurance_plan aip on ip.insurance_plan_no = aip.insurance_plan_no where ip.insurance_plan_no = 2;
```

This will returns the data from both the tables which should be wrapped into AccidentalInsurancePlan object.

Let us consider the bad or failure case:
We are fetching MedicalInsurancePlan object with id: 3 as AccidentalInsurancePlan object,
session.get(AccidentalInsurancePlan.class, 3);
since we are querying for AccidentalInsurancePlan class object we still perform an inner join between insurance_plan and accidental_insurance_plan for an insurance_plan_no = 3 that returns 0 records. Even though the record exists with plan_no = 3 in insurance_plan table, it doesnt exist in accidental_insurance_plan table that results in zero record as we are performing inner join.

---

2. Super class object (InsurancePlan):
session.get(InsurancePlan.class, 1);
To support polymorphic retrieval when we are querying the data for Super class object, we should query the data from all the tables in the hierarchy. Then should determine where the record exists.

if it only exists in insurance_plan (SuperClass) table = InsurancePlan (Superclass: object)
if the record exists in only insurance_plan table + accidental_insurance_plan = AccidentalInsurancePlan (object)
if the record exists in only insurance_plan table + medical_insurance_plan = MedicalInsurancePlan (object)