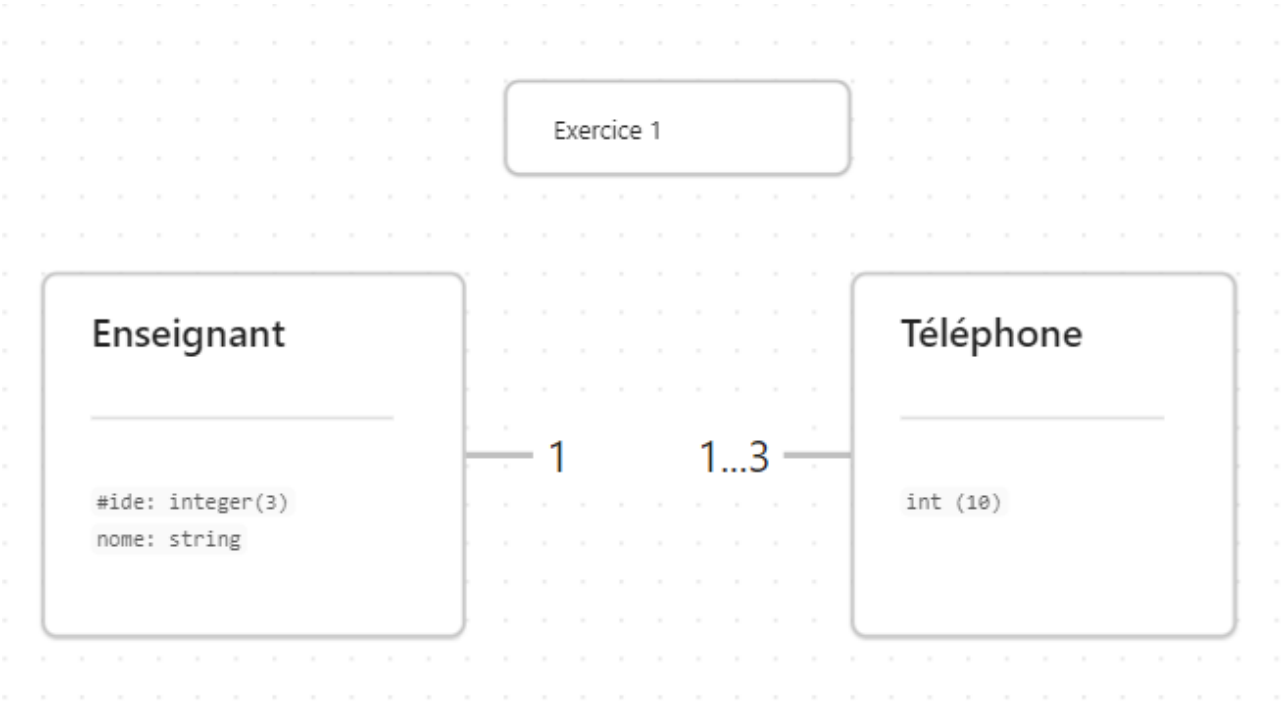
TD 3 : Relationnel – Objet avec Oracle

Alexandre Clenet - Florian Tran
Année 3 - Groupe 1-2

I) Les tables imbriquées

Exercice 1

1. Traduire le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

```
   CREATE OR REPLACE TYPE Tel AS TABLE OF number(10) NULL;
   CREATE TABLE Enseignant (
      ide INTEGER PRIMARY KEY,
      nome VARCHAR(255),
      ltel Tel
   ) NESTED TABLE Tel STORE AS t1
```

3. Insérer dans la table Enseignant les tuples de votre choix permettant de répondre aux questions suivantes.

```
   INSERT INTO Enseignant (ide, nome, ltel) VALUES (1, 'Jean', Tel(1234567890));
   INSERT INTO Enseignant (ide, nome, ltel) VALUES (2, 'Marie', Tel(9876543210,
1234567890, 5555555555));
   INSERT INTO Enseignant (ide, nome, ltel) VALUES (3, 'Pauline', Tel());
   INSERT INTO Enseignant (ide, nome, ltel) VALUES (4, 'Thomas', Tel(1111111111,
2222222222));
```

```
    INSERT INTO Enseignant (ide, nome, ltel) VALUES (5, 'Sophie', Tel(9999999999));
    INSERT INTO Enseignant (ide, nome, ltel) VALUES (6, 'Nicolas',
Tel(7777777777));
```

4. réponder aux questions suivantes :

a) Trouver le(s) numéro(s) de téléphone d'un enseignant dont vous renseignez son identité

```
SELECT ltel FROM Enseignant WHERE nome = 'Thomas';

--Outpout : S5A08B.TEL(1111111111, 2222222222)
```

b) Trouver le nombre de téléphone possédé par enseignant

```
SELECT e.nome, COUNT(*) from Enseignant e,TABLE(e.ltel) GROUP BY e.nome;

--Outpout :
Jean    1
Nicolas 1
Sophie  1
Thomas  2
Marie   3
```

c) Trouver le nombre des enseignants qui possèdent trois numéros de téléphone.

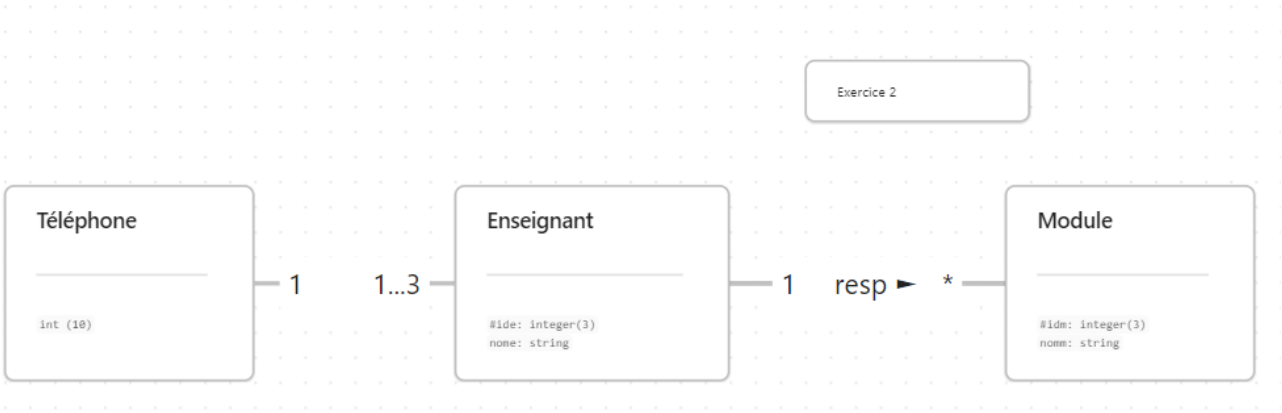
```
SELECT COUNT(*) AS nombre_enseignants
FROM Enseignant
WHERE (SELECT COUNT(*) FROM TABLE(ltel)) = 3;

--Outpout : 1
```

d) Trouver les enseignants qui ne possèdent pas de téléphone

```
SELECT nome
FROM Enseignant
WHERE (SELECT COUNT(*) FROM TABLE(ltel)) = 0;

--Outpout : Pauline
```

Exercice 2

1. Traduire le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

```
CREATE OR REPLACE TYPE Module AS OBJECT (
    idm NUMBER(10),
    nomm VARCHAR2(255)
);
CREATE OR REPLACE TYPE Module_Table AS TABLE OF Module;

DROP TABLE Enseignant;
CREATE TABLE Enseignant (
    ide INTEGER PRIMARY KEY,
    nome VARCHAR2(255),
    ltel Tel,
    modules Module_Table
) NESTED TABLE ltel STORE AS t1 NESTED TABLE modules STORE AS t2;
```

3. Insérer dans la table Enseignant les tuples de votre choix permettant de répondre aux questions suivantes :

```
INSERT INTO Enseignant VALUES (1, 'Martin', Tel(1234567890),
Module_Table(Module(1, 'Système')));
INSERT INTO Enseignant VALUES (2, 'Sophie', Tel(9876543210), NULL);
INSERT INTO Enseignant VALUES (3, 'Dupond', Tel(5555555555), NULL);
INSERT INTO Enseignant VALUES (4, 'Antoine', Tel(1111111111),
Module_Table(Module(2, 'Base de données')));
```

a) Martin a abandonné la responsabilité du module système. Mettre à jour la table enseignant afin de prendre en compte ce changement.

```
UPDATE Enseignant
SET modules = NULL
WHERE nome = 'Martin';
```

b) Sophie et Dupond deux enseignants qui ont pris respectivement la responsabilité du module système et base de données. Ajouter dans la table enseignant les nouvelles responsabilités de ces deux enseignants.

```
UPDATE Enseignant
SET modules = Module_Table(Module(1, 'Système'))
WHERE nome = 'Sophie';

UPDATE Enseignant
SET modules = Module_Table(Module(2, 'Base de données'))
WHERE nome = 'Dupond';
```

c) Un changement au niveau des responsabilités des modules. A partir de maintenant Antoine prend la responsabilité du module gestion à la place de base de données. Mettre à jour la table enseignant.

```
UPDATE Enseignant
SET modules = Module_Table(Module(3, 'Gestion'))
WHERE nome = 'Antoine';
```

d) réponder aux questions suivantes

d1) Pour Martin trouver les noms des modules dont il est responsable

```
SELECT modules
FROM Enseignant
WHERE nome = 'Martin' AND modules IS NOT NULL;

-- Output : Martin n'est responsable d'aucun module après la mise à jour.
```

d2) Trouver les numéros de téléphone de Martin

```
SELECT ltel
FROM Enseignant
WHERE nome = 'Martin';

--Outpout : S5A08B.TEL(1234567890)
```

d3) Trouver les numéros de téléphone du responsable du module système

```
SELECT e.ltel
FROM Enseignant e
WHERE EXISTS (
    SELECT *
    FROM TABLE(e.modules) m
    WHERE m.nomm = 'Système'
```

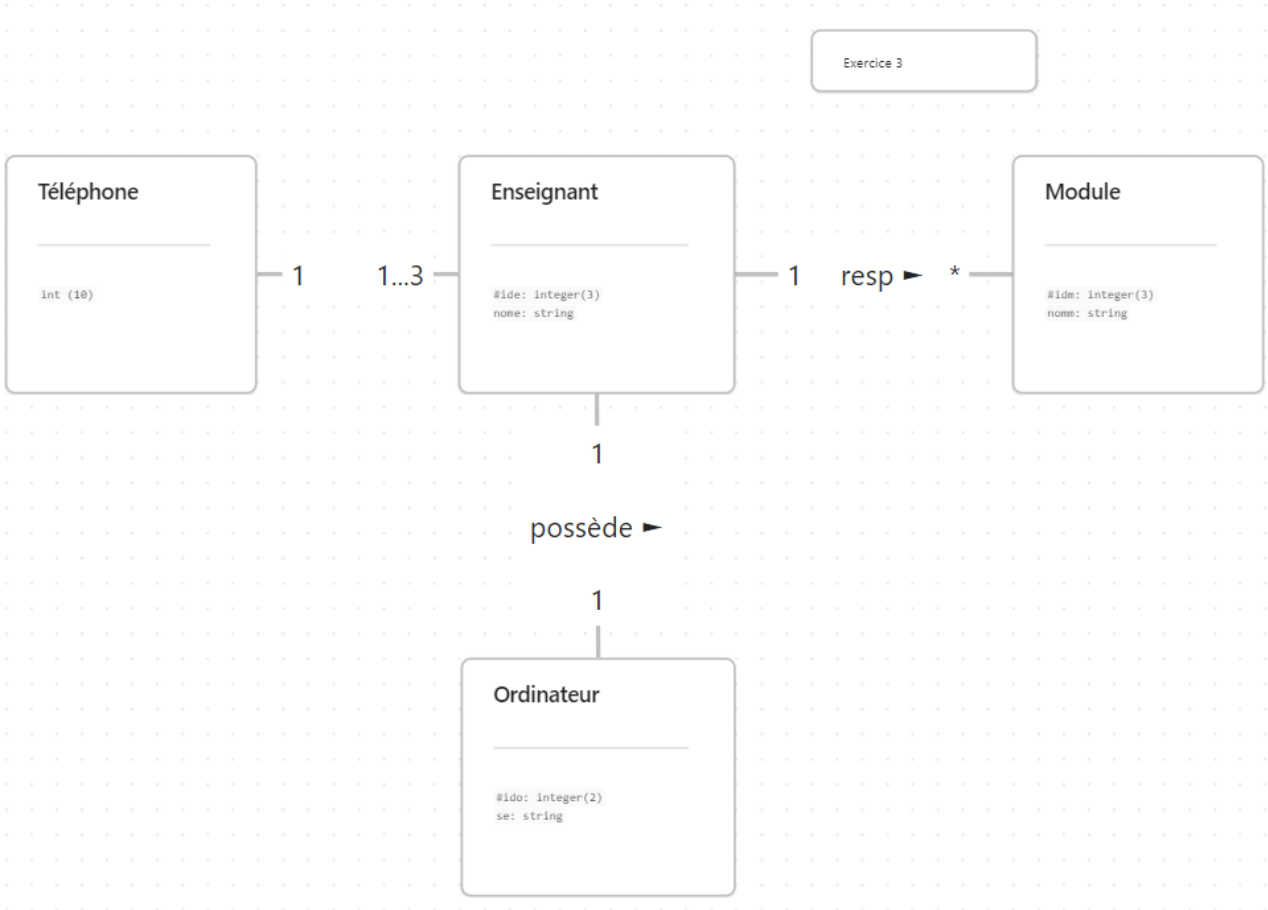
```
);

--Outpout : S5A08B.TEL(9876543210)
```

d4) Trouver les enseignants responsables d'un seul module

```
SELECT nome
FROM Enseignant
WHERE (SELECT COUNT(*) FROM TABLE(modules)) = 1;

--Outpout :
Sophie
Dupond
Antoine
```

Exercice 3

1. Traduire le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

```
CREATE OR REPLACE TYPE Ordinateur AS OBJECT (
    ido NUMBER(10),
    se VARCHAR2(255)
);
```

```
CREATE OR REPLACE TYPE Ordinateur_Table AS TABLE OF Ordinateur;

DROP TABLE Enseignant;
CREATE TABLE Enseignant (
    ide INTEGER PRIMARY KEY,
    nome VARCHAR2(255),
    ltel Tel,
    modules Module_Table,
    ordinateur Ordinateur_Table
) NESTED TABLE ltel STORE AS t1 NESTED TABLE modules STORE AS t2 NESTED TABLE
ordinateur STORE AS t3;
```

3. Insérer plusieurs tuples dans la table enseignant.

```
INSERT INTO Enseignant VALUES (1, 'Martin', Tel(1234567890),
Module_Table(Module(1, 'Système')), Ordinateur_Table(Ordinateur(101, 'Windows')));
INSERT INTO Enseignant VALUES (2, 'Sophie', Tel(9876543210),
Module_Table(Module(2, 'Base de données')), Ordinateur_Table(Ordinateur(102,
'Linux')));
INSERT INTO Enseignant VALUES (3, 'Dupond', Tel(5555555555),
Module_Table(Module(3, 'Réseaux')), Ordinateur_Table(Ordinateur(103, 'Windows')));
```

4. réponder aux questions suivantes : a) Trouver le se de Martin et son numéro de téléphone

```
SELECT e.nome, o.se AS systeme_exploitation, e.ltel
FROM Enseignant e, TABLE(e.ordinateur) o, TABLE(e.ltel) tel
WHERE e.nome = 'Martin';

--Outpout : Martin  Windows S5A08B.TEL(1234567890)
```

b) Afficher les responsables des modules utilisant windows comme système d'exploitation.

```
SELECT e.nome
FROM Enseignant e
WHERE EXISTS (
    SELECT *
    FROM TABLE(e.ordinateur) o
    WHERE o.se = 'Windows'
) AND e.modules IS NOT NULL;

--Outpout :
Martin
Dupond
```

c) Afficher le nombre d'enseignants utilisant le système windows

```
SELECT COUNT(*) AS nombre_enseignants_windows
FROM Enseignant e
WHERE EXISTS (
    SELECT 1
    FROM TABLE(e.ordinateur) o
    WHERE o.se = 'Windows'
);

--Outpout : 2
```

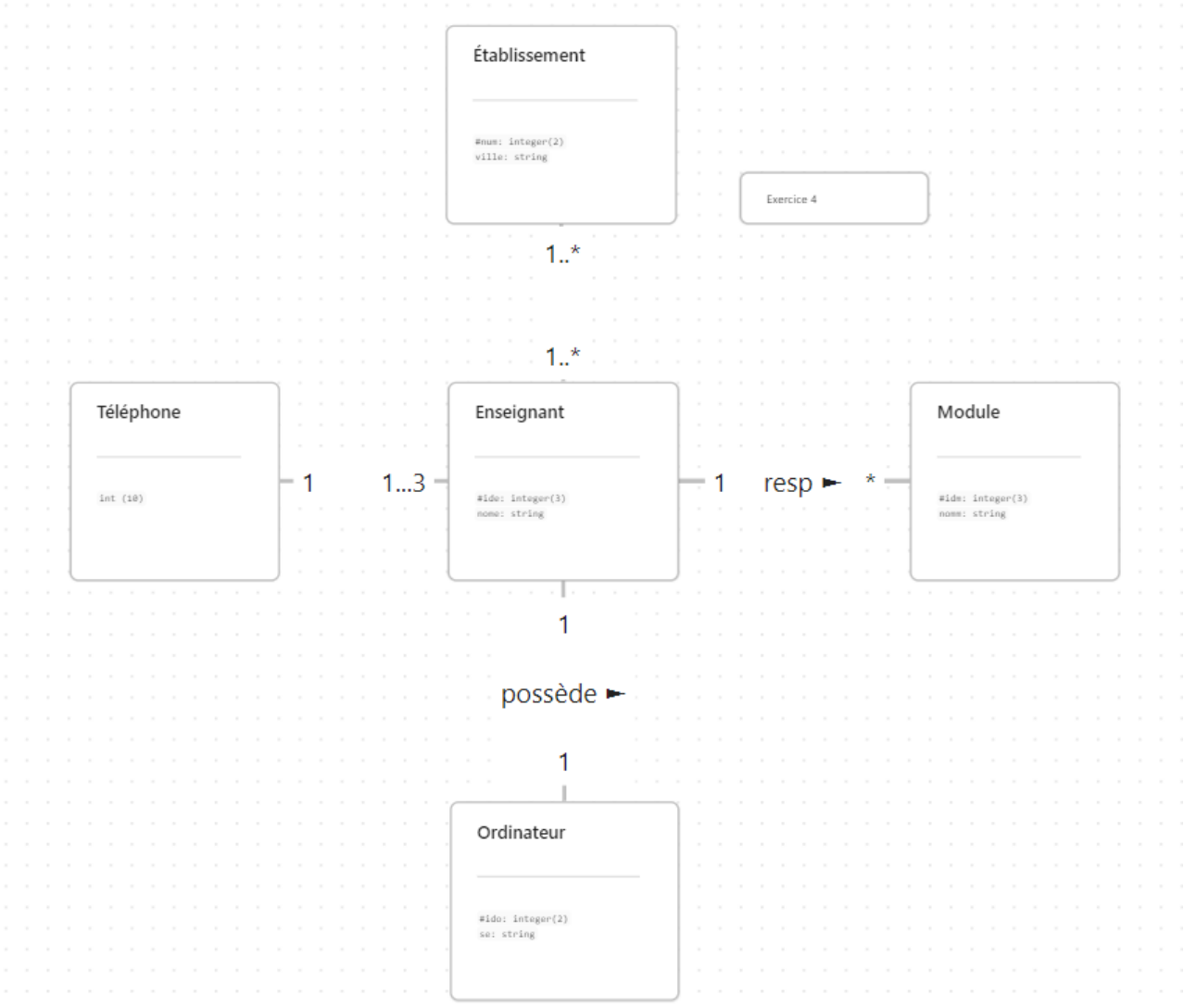
d) Afficher les n-uplets de la table Enseignant

```
SELECT * FROM Enseignant;

-- Output :
Martin  | 1234567890    | Module_Table(Module(1, 'Système')) |
Ordinateur_Table(Ordinateur(101, 'Windows'))
Sophie  | 9876543210    | Module_Table(Module(2, 'Base de données')) |
Ordinateur_Table(Ordinateur(102, 'Linux'))
Dupond  | 5555555555    | Module_Table(Module(3, 'Réseaux')) |
Ordinateur_Table(Ordinateur(103, 'Windows'))
```

Exercice 4

1. Compléter le schéma UML de l'exercice précédent en tenant compte des nouvelles règles

2. Traduire le schéma UML en schéma logique Relationnel-Objet.



3. Implémentez le schéma logique avec Oracle. Il faut penser à ajouter les contraintes d'intégrité (clés primaires et étrangères) suite à l'introduction de la table établissement.

```
CREATE TABLE Etablissement (
    num INTEGER PRIMARY KEY,
    ville VARCHAR2(255) NOT NULL
);

CREATE OR REPLACE TYPE Intervention AS OBJECT (
    idi INTEGER,
    etablissement INTEGER,
    duree NUMBER(10)
);
CREATE OR REPLACE TYPE Intervention_Table AS TABLE OF Intervention;

CREATE TABLE Enseignant (
    ide INTEGER PRIMARY KEY,
    nome VARCHAR2(255),
    ltel Tel,
    modules Module_Table,
    ordinateur Ordinateur_Table,
    intervention Intervention_Table
```

```
) NESTED TABLE ltel STORE AS t1 NESTED TABLE modules STORE AS t2 NESTED TABLE
ordinateur STORE AS t3 NESTED TABLE intervention STORE AS t4;
```

4. Insérer des tuples dans les tables selon votre choix de manière à répondre aux questions suivantes. Vérifier toutes les contraintes d'intégrités

```
INSERT INTO Etablissement VALUES (1, 'Paris');
INSERT INTO Etablissement VALUES (2, 'Lyon');

INSERT INTO Enseignant VALUES (
    1,
    'Martin',
    Tel(1234567890),
    Module_Table(Module(1, 'Système')),
    Ordinateur_Table(Ordinateur(101, 'Windows')),
    Intervention_Table(Intervention(1, 1, 20))
);
INSERT INTO Enseignant VALUES (
    2,
    'Sophie',
    Tel(9876543210),
    Module_Table(Module(2, 'Base de données')),
    Ordinateur_Table(Ordinateur(102, 'Linux')),
    Intervention_Table(Intervention(2, 2, 25))
);
INSERT INTO Enseignant VALUES (
    3,
    'Dupond',
    Tel(5555555555),
    Module_Table(Module(3, 'Réseaux')),
    Ordinateur_Table(Ordinateur(103, 'Windows')),
    Intervention_Table(Intervention(3, 1, 30), Intervention(4, 2, 15))
);
```

5. réponder aux questions suivantes :

a) Trouver les systèmes d'exploitation (se) utilisés par des enseignants qui travaillent plus de 25h par semaine.

```
SELECT e.nome, o.se
FROM Enseignant e,
Table(e.ordinateur) o,
Table(e.intervention) i
WHERE i.duree > 25;

--Outpout : Dupond  Windows
```

b) Afficher les numéros de téléphones de l'enseignant responsable du module bd et travaillant dans un établissement situé à Paris.

```
INSERT INTO Enseignant VALUES (
    4,
    'Alex',
    Tel(4566515458),
    Module_Table(Module(3, 'Base de données')),
    Ordinateur_Table(Ordinateur(103, 'Windows')),
    Intervention_Table(Intervention(5, 1, 20))
);

SELECT e.ltel
FROM Enseignant e,
Table(e.intervention) i,
Table(e.modules) m,
Etablissement et
WHERE i.etablissement = et.num AND m.nomm = 'Base de données' AND et.ville =
'Paris';

--Outpout : S5A08B.TEL(4566515458)
```

c) Trouver le nombre d'enseignants qui intervient dans chaque établissement

```
SELECT e.etablissement, COUNT(DISTINCT e.nome) AS nombre_enseignants
FROM (
    SELECT et.ville AS etablissement, e.nome
    FROM Enseignant e,
    TABLE(e.intervention) i,
    Etablissement et
    WHERE i.etablissement = et.num
) e
GROUP BY e.etablissement;

--Outpout :
Lyon    2
Paris   3
```

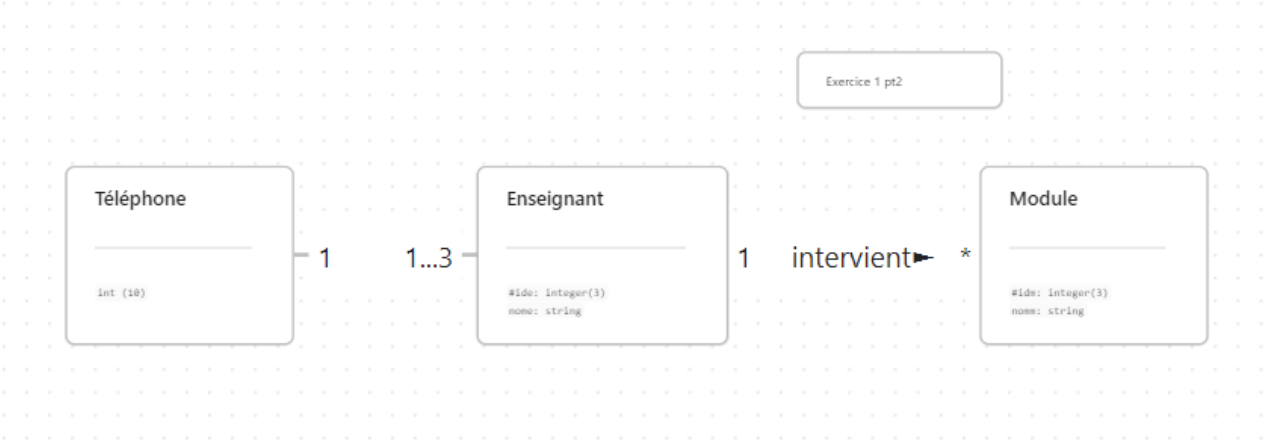
d) Afficher les enqeignant enseignant et pour chacun les établissements dans lesquels il inervient ainsi que la durée

```
SELECT e.nome, et.ville, i.duree
FROM Enseignant e,Etablissement et,
Table(e.intervention) i
where et.num = i.etablissement;

--Outpout :
Martin  Paris   20
Sophie  Lyon    25
Dupond  Paris   30
Dupond  Lyon    15
```

II) Les tables d'objets

Exercice 1

1. Traduiser le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

```
CREATE OR REPLACE TYPE Tel AS TABLE OF number(10) NULL;

CREATE OR REPLACE TYPE Module AS OBJECT (
    idm NUMBER(10),
    nomm VARCHAR2(255)
);

create or replace TYPE Enseignant AS OBJECT(
    ide NUMBER(10),
    nome VARCHAR2(255),
    ltel Tel,
    refmodule REF Module
);

drop table tModule;
CREATE TABLE tModule OF Module (
    idm PRIMARY KEY,
    nomm NOT NULL
);

drop table tTel;
CREATE TABLE tTel OF Tel;

drop table tEnseignant;
CREATE TABLE tEnseignant OF Enseignant(
    ide PRIMARY KEY,
    nome NOT NULL,
    refmodule SCOPE IS tModule
)NESTED TABLE ltel STORE AS tTel;
```

3. Insérer plusieurs enseignants, modules et ordinateurs dans la base de données afin de répondre aux intérrogations de la question suivante. Vous insérez les données en utilisant des requêtes SQL ou avec des programmes PL/SQL.

```
INSERT INTO tModule VALUES (1, 'Base de données');
INSERT INTO tModule VALUES (2, 'Système');
INSERT INTO tEnseignant VALUES (101, 'Martin', Tel(1234567890), (SELECT REF(m)
FROM tModule m WHERE m.idm = 1));
INSERT INTO tEnseignant VALUES (102, 'Alice', Tel(9876543210), (SELECT REF(m) FROM
tModule m WHERE m.idm = 2));
```

4. réponder aux questions suivantes :

a) Trouver les modules enseignés par Martin

```
SELECT e.refmodule.nomm
FROM tEnseignant e
WHERE e.nome = 'Martin';

--Outpout : Base de données
```

b) Trouver le nom de l'enseignant qui intervient dans le module base de données.

```
SELECT e.nome
FROM tEnseignant e
WHERE e.refmodule.nomm = 'Base de données';

--Outpout : Martin
```

c) Trouver le nombre d'intervenants dans chacun des modules

```
SELECT e.refmodule.nomm, COUNT(e.ide) AS nombre_intervenants
FROM tEnseignant e
GROUP BY e.refmodule.nomm;

--Outpout :
Système 1
Base de données 1
```

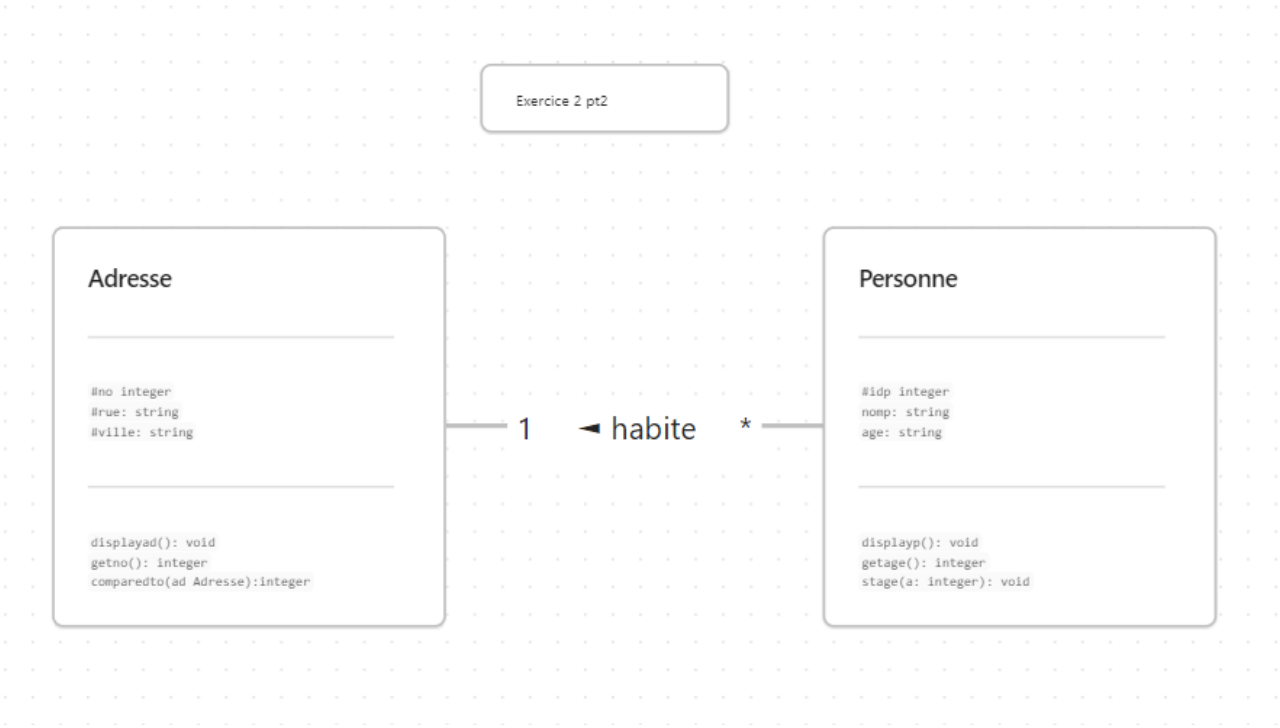
d) Trouver les numéros de téléphones des intervenants du module système

```
SELECT e.ltel
FROM tEnseignant e
WHERE e.refmodule.nomm = 'Système';
```

```
--Outpout : S5A08B.TEL(9876543210)
```

Exercice 2 : méthodes dans les tables d'objets

1. Traduire le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle. La méthode getage() sera implémentée avec l'option MAP.

```
CREATE OR REPLACE TYPE Adresse AS OBJECT (
    no   NUMBER(10),
    rue VARCHAR2(255),
    ville VARCHAR2(255),
    MEMBER FUNCTION displayad RETURN varchar2,
    MEMBER FUNCTION getno RETURN number,
    MEMBER FUNCTION compareto(ad IN Adresse) RETURN number
);

CREATE OR REPLACE TYPE BODY Adresse AS
    MEMBER FUNCTION displayad RETURN varchar2 IS
    BEGIN
        RETURN no || ' ' || rue || ', ' || ville;
    END displayad;

    MEMBER FUNCTION getno RETURN number IS
    BEGIN
        RETURN no;
    END getno;

    MEMBER FUNCTION compareto(ad IN Adresse) RETURN number IS
    BEGIN
        IF no = ad.no AND rue = ad.rue AND ville = ad.ville THEN
```

```
            RETURN 0;
        ELSIF no > ad.no OR (no = ad.no AND rue > ad.rue) OR (no = ad.no AND rue =
ad.rue AND ville > ad.ville) THEN
            RETURN 1;
        ELSE
            RETURN -1;
        END IF;
    END compareto;
END;

CREATE OR REPLACE TYPE Personne AS OBJECT(
    idp NUMBER(10),
    nomp VARCHAR2(255),
    age  NUMBER(10),
    refAdresse REF Adresse,
    MEMBER FUNCTION displayp RETURN varchar2,
    MEMBER FUNCTION getage RETURN number,
    MEMBER FUNCTION setage(a IN NUMBER) RETURN number
);

CREATE OR REPLACE TYPE BODY Personne AS
    MEMBER FUNCTION displayp RETURN varchar2 IS
    BEGIN
        RETURN idp || ': ' || nomp || ', ' || age || ' years old, Address: ' ||
refAdresse.displayad();
    END displayp;

    MEMBER FUNCTION getage RETURN number IS
    BEGIN
        RETURN age;
    END getage;

    MEMBER FUNCTION setage(a IN OUT NUMBER) RETURN number IS
    BEGIN
        age := a;
        RETURN age;
    END setage;
END;

DROP TABLE tAdresse;
CREATE TABLE tAdresse OF Adresse (
    PRIMARY KEY(no,rue,ville)
);

DROP TABLE tPersonne;
CREATE TABLE tPersonne OF Personne(
    idp PRIMARY KEY,
    nomp NOT NULL,
    age NOT NULL,
    refAdresse SCOPE IS tAdresse
);
```

3. Ecrire un programme PL/SQL qui permet de a) Créer un objet de type Adresse (new adresse (...)), afficher l'adresse sans passer par la procédure displayAd()

```
DECLARE
    myAddress Adresse := Adresse(123, 'Rue de la Paix', 'Paris');
BEGIN
    DBMS_OUTPUT.PUT_LINE('Address: ' || myAddress.no || ' ' || myAddress.rue || ',
' || myAddress.ville);
END;
/

--Outpout : Address: 123 Rue de la Paix, Paris
```

b) Créer deux objets de type Adresse, comparer ces objets entre eux et afficher l'objet le plus grand

```
DECLARE
    address1 Adresse := Adresse(123, 'Rue A', 'Ville A');
    address2 Adresse := Adresse(456, 'Rue B', 'Ville B');
    result NUMBER;
BEGIN
    result := address1.compareto(address2);
    IF result = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Adresses égales');
    ELSIF result = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Adresse 1 plus grande');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Adresse 2 plus grande');
    END IF;
END;
/

--Outpout : Adresse 2 plus grande
```

c) Créer une personne et afficher ses propriétés avec son adresse en utilisant la méthode displayP()

```


--Outpout :
```

d) Afficher l'identifiant de la personne le plus jeune parmi les personnes suivantes : pers1 : <1,'Dupont',30,NULL>; pers2 : <5,'Martin',22,NULL>

```
DECLARE
    -- Creating instances of Personne
    pers1 Personne := Personne(1, 'Dupont', 30, NULL);
    pers2 Personne := Personne(5, 'Martin', 22, NULL);
    youngestId NUMBER;
```

```
BEGIN
    IF pers1.getage < pers2.getage THEN
        youngestId := pers1.idp;
    ELSE
        youngestId := pers2.idp;
    END IF;
    DBMS_OUTPUT.PUT_LINE('L ID du plus jeune: ' || youngestId);
END;
/

--Outpout : L ID du plus jeune: 5
```

e) Créer un objet de type personne : <1,'Dupond', 23,NULL>, modifier l'age de Dupont à 34 (au lieu de 23) et afficher le résultat

```
--Outpout :
```

4. Insérer plusieurs personnes dans la base de données en précisant leur adresse

```
INSERT INTO tAdresse VALUES (10, 'Rue de la Paix', 'Paris');
INSERT INTO tAdresse VALUES (62, 'Rue du Faubourg Saint-Honoré', 'Paris');
INSERT INTO tAdresse VALUES (3, 'Avenue des Champs-Élysées', 'Paris');

INSERT INTO tPersonne VALUES (1, 'Jean Dupont', 28, (SELECT REF(a) FROM tAdresse a
WHERE a.no = 10));
INSERT INTO tPersonne VALUES (2, 'Marie Leclerc', 35, (SELECT REF(a) FROM tAdresse
a WHERE a.no = 62));
INSERT INTO tPersonne VALUES (3, 'Pierre Lambert', 40, (SELECT REF(a) FROM
tAdresse a WHERE a.no = 3));
```

5. réponder aux questions suivantes avec des requête :

a) Affichez la ville de la personne n° 1 (deux versions)

```
SELECT p.refadresse.ville AS ville
FROM tPersonne p
WHERE p.idp = 1;

--Outpout : Paris

SELECT a.ville AS ville_version_2
FROM tPersonne p
JOIN tAdresse a ON p.refAdresse IS NOT NULL AND p.refAdresse = REF(a)
WHERE p.idp = 1;

--Outpout : Paris
```

b) Afficher l'adresse de Pierre Lambert en utilisant la méthode displayAd() dans un programme PL/SQL

```
DECLARE
    v_address VARCHAR2(255);
BEGIN
    SELECT p.refAdresse.displayad()
    INTO v_address
    FROM tPersonne p
    WHERE p.nomp = 'Pierre Lambert' AND ROWNUM = 1;

    DBMS_OUTPUT.PUT_LINE('Pierre Lambert habite ' || v_address);
END;
/

--Outpout : Pierre Lambert habite 3 Avenue des Champs-Élysées, Paris
```

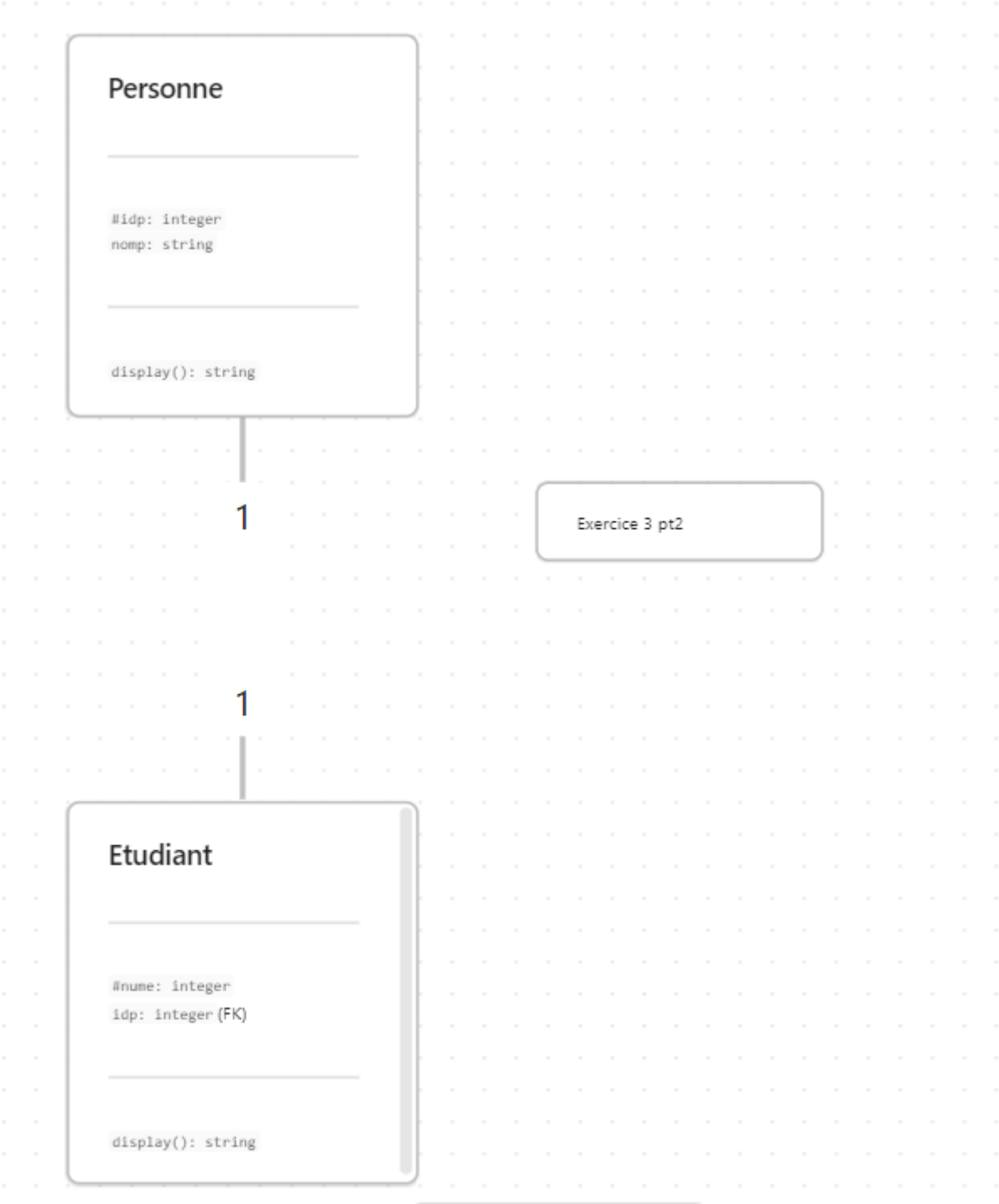
c) Ajuter une personne avec adresse NULL :<4,'Martin',35,NULL>. Martin habite maintenant à l'adresse 3 rue Garenne à Nantes. Mettre à jour la base de données

```
INSERT INTO tPersonne VALUES (4, 'Martin', 35, NULL);
INSERT INTO tAdresse VALUES (3, 'Rue Garenne', 'Nantes');

UPDATE tPersonne p
SET p.refAdresse = (SELECT REF(a) FROM tAdresse a WHERE a.no = 3 AND a.rue = 'Rue
Garenne')
WHERE p.idp = 4;

SELECT p.refadresse.displayad() AS Adresse_de_Martin
FROM tPersonne p
WHERE p.nomp = 'Martin';

--Outpout : 3 Rue Garenne, Nantes
```

Exercice 3

Cas I) l'héritage entre Etudiant et Personne est implémentée sous forme d'une clé étrangère reliant Etudiant à Personne

1. Traduiser le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

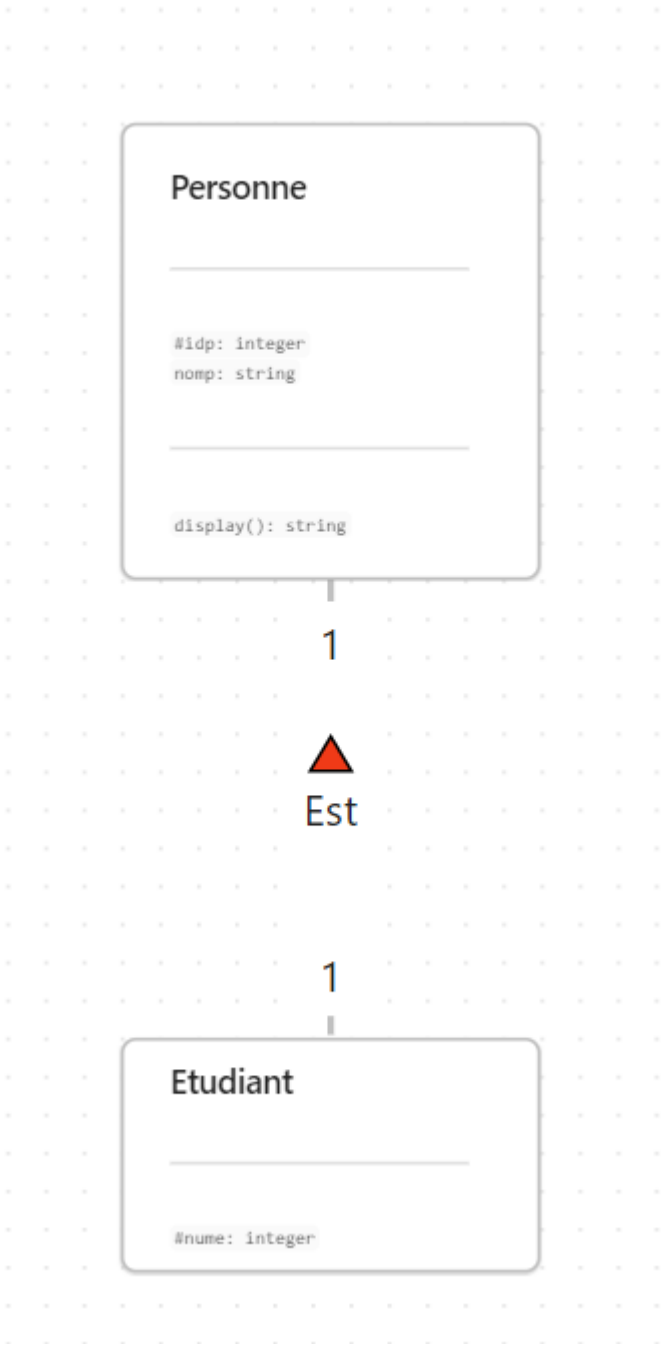
```
CREATE TABLE Personne (
    idp NUMBER PRIMARY KEY,
    nomp VARCHAR2(255) NOT NULL
);

CREATE TABLE Etudiant (
    nume NUMBER PRIMARY KEY,
    idp NUMBER NOT NULL,
    CONSTRAINT fk_personne FOREIGN KEY (idp) REFERENCES Personne(idp)
);
```

3. Ecrire un programme PL/SQL permettant de créer un étudiant et d'afficher toutes ses propriétés

```
DECLARE
    v_idp NUMBER;
    v_nomp VARCHAR2(255);
    v_nume NUMBER;
BEGIN
    v_idp := 1;
    v_nomp := 'Martin';
    INSERT INTO Personne (idp, nomp) VALUES (v_idp, v_nomp);
    v_nume := 84552;
    INSERT INTO Etudiant (nume, idp) VALUES (v_nume, v_idp);

    -- Display properties of the student
    DBMS_OUTPUT.PUT_LINE('Etudiant ID: ' || v_nume);
    DBMS_OUTPUT.PUT_LINE('Personne ID: ' || v_idp);
    DBMS_OUTPUT.PUT_LINE('Nom: ' || v_nomp);
END;
/

--Outpout :
Etudiant ID: 84552
Personne ID: 1
Nom: Martin
```

Cas II) l'héritage entre Etudiant et Personne est implémentée sous forme d'une référence reliant Etudiant à Personne

1. Traduiser le schéma UML en schéma logique Relationnel-Objet



2. Implémenter le schéma logique avec Oracle

```
CREATE OR REPLACE TYPE Personne AS OBJECT(
    idp NUMBER(10),
    nomp VARCHAR2(255)
);

CREATE OR REPLACE TYPE Etudiant AS OBJECT(
nume NUMBER ,
refPersonne REF Personne
);

CREATE TABLE tPersonne OF Personne(
    idp PRIMARY KEY,
    nomp NOT NULL
```

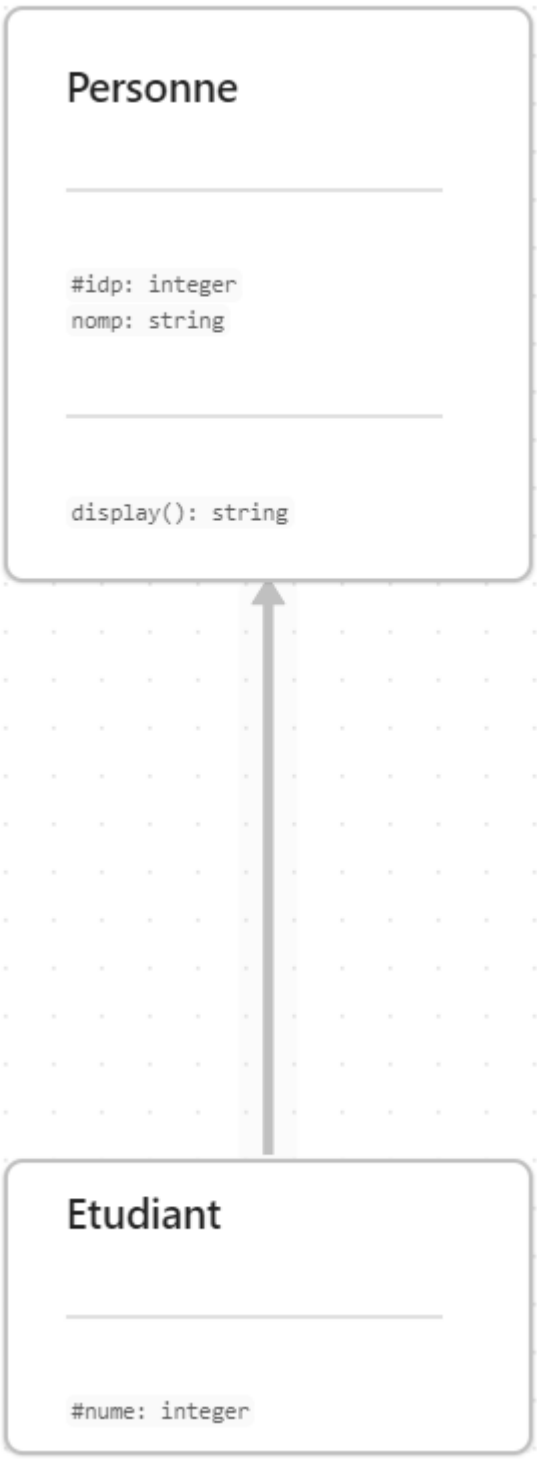
```
);

CREATE TABLE tEtudiant OF Etudiant(
    nume PRIMARY KEY,
    refPersonne SCOPE IS tPersonne
);
```

3. Ecrire une requête SQL permettant d'afficher toutes les propriétés d'un étudiant

```
INSERT INTO tPersonne VALUES (Personne(1, 'John Doe'));
INSERT INTO tEtudiant VALUES (Etudiant(202358, (SELECT REF(p) FROM tPersonne p
WHERE p.idp = 1)));

SELECT e.nume, e.refpersonne.idp AS idp, e.refpersonne.nomp AS nomp
FROM tEtudiant e
WHERE e.nume = 202358;
--Outpout : 202358  1   John Doe
```

Cas III) l'héritage entre Etudiant et Personne est implémentée comme étant un héritage de type permettant à la classe Etudiant d'hériter les propriétés et les comportement de la classe Personne.

a) Traduire le schéma UML en schéma logique Relationnel-Objet



b) Implémenter le schéma logique avec Oracle

```
CREATE OR REPLACE TYPE Personne AS OBJECT(
    idp NUMBER(10),
    nomp VARCHAR2(255),
    MEMBER PROCEDURE display
)NOT FINAL;

CREATE or REPLACE TYPE BODY Personne AS
    MEMBER PROCEDURE display IS
    BEGIN
        dbms_output.put_line('Personne: ' || idp || ', ' || nomp);
```

```
    END;
END;

CREATE TYPE Etudiant UNDER Personne (
    nume NUMBER ,
    OVERRIDING MEMBER PROCEDURE display
);

CREATE or REPLACE TYPE BODY Etudiant AS
    OVERRIDING MEMBER PROCEDURE display IS
    BEGIN
        dbms_output.put_line('Etudiant: ' || idp || ', ' || nomp || ', ' || nume);
    END;
END;

CREATE TABLE PersonneTable OF Personne;
CREATE TABLE EtudiantTable OF Etudiant;
```

c) Insérer des instances dans la classe Etudiant

```
INSERT INTO EtudiantTable VALUES (1, 'John', 101);
INSERT INTO EtudiantTable VALUES (2, 'Jane', 102);
```

d) On utilise maintenant le polymorphisme dans le modèle objet pour afficher toutes les propriétés des étudiants

```
--Outpout :
```

e) Insérer le tuple : ❤️,'Kilian'> dans la table personne (Kilian n'est pas inscrit comme étudiant) ; écrire une requête permettant d'afficher les noms des personnes qui ne sont pas reconnues comme étudiants dans la base de données.

```
INSERT INTO PersonneTable VALUES (3, 'Kilian');
SELECT nomp FROM PersonneTable MINUS SELECT nomp FROM EtudiantTable;
--Outpout : Kilian
```

f) Ecrire une requête SQL permettant d'afficher les noms de toutes les personnes avec leur identifiant

```
SELECT idp, nomp FROM PersonneTable p
UNION
SELECT idp, nomp FROM EtudiantTable e;
--Outpout :
1   John
```

```
2   Jane
3   Kilian
```

g) Kilian est désormais inscrit comme étudiant avec le numéro étudiant 30. Ecrire un programme PL/SQL afin de tenir compte de cette nouvelle information.

```
DECLARE
BEGIN
    DELETE FROM PersonneTable WHERE idp=3 ;
    INSERT INTO EtudiantTable VALUES (3, 'Kilian', 30);
END;
/
```

h) Kilian n'est plus étudiant. Ecrire un programme PL/SQL pour mettre à jour la base de données.

```
DECLARE
BEGIN
    DELETE FROM EtudiantTable WHERE nume=30 ;
    INSERT INTO PersonneTable VALUES (3, 'Kilian');
END;
/
```