1. Solución.

Un Centro de enseñanza de secundaria desea informatizar mediante Oracle Orientado a Objetos los siguientes enunciados:

- Creación de un tipo de objetos "MiembroEscolar":

```
CREATE OR REPLACE TYPE MiembroEscolar AS OBJECT (
        codigo          INTEGER,
        dni             VARCHAR2(10),
        nombre          VARCHAR2(30),
        apellidos       VARCHAR2(30),
        sexo            VARCHAR2(1),
        fecha_nac       DATE
) NOT FINAL;
/
```

- Creación, como tipo heradado de "MiembroEscolar", el tipo de objeto "Profesor":

```
CREATE OR REPLACE TYPE Profesor UNDER MiembroEscolar (
        especialidad VARCHAR2(20),
        antiguedad INTEGER,

        CONSTRUCTOR FUNCTION Profesor(codigo INTEGER, nombre VARCHAR2,
        apellido1 VARCHAR2, apellido2 VARCHAR2, especialidad VARCHAR2)
        RETURN SELF AS RESULT,

        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2
);
/
```

- Creación de un método constructor para el tipo de objetos Profesor, en el que se indiquen como parámetros el código, nombre, primer apellido, segundo apellido y especialidad. Este método debe asignar al atributo apellidos los datos de primer apellido y segundo apellido que se han pasado como parámetros, uniéndolos con un espacio entre ellos:

```
CREATE OR REPLACE TYPE BODY Profesor AS
        CONSTRUCTOR FUNCTION Profesor(codigo INTEGER, nombre VARCHAR2, apellido1
        VARCHAR2, apellido2 VARCHAR2, especialidad VARCHAR2)
        RETURN SELF AS RESULT IS
        BEGIN
                SELF.codigo := codigo;
                SELF.nombre := nombre;
                SELF.apellidos := CONCAT(apellido1, apellido2);
                SELF.especialidad := especialidad; antiguedad := 1;
                RETURN;
        END;

-- Creación del método getNombreCompleto para el tipo de objetos Profesor
        MEMBER FUNCTION getNombreCompleto RETURN VARCHAR2 IS
        BEGIN
                RETURN (apellidos || ' ' || nombre);
        END getNombreCompleto;
END;
/
```

BD07 - ENUNCIADO TAREA ESCUELA

- Creación del tipo de objeto "Curso":

```
CREATE OR REPLACE TYPE Cursos AS OBJECT (
        codigo          INTEGER,
        nombre          VARCHAR2(20),
        refProfe REF Profesor,
        max_Alumn       INTEGER,
        fecha_Inic      DATE,
        fecha_Fin       DATE,
        num_Horas       INTEGER,

        MAP MEMBER FUNCTION ordenarCursos RETURN VARCHAR2
);
/
```

- Creación de un método MAP llamado "ordenarCursos" para el tipo "Cursos". Este método debe retornar el nombre completo del profesor al que hace referencia cada curso. Para obtener el nombre has de utilizar el método "getNombreCompleto" que se ha creado previamente.

```
CREATE OR REPLACE TYPE BODY Cursos AS
        MAP MEMBER FUNCTION ordenarCursos RETURN VARCHAR2 IS
                unProfesor Profesor;
        BEGIN
                SELECT DEREF(refProfe) INTO unProfesor FROM Dual;
                RETURN (unProfesor.getNombreCompleto());
        END ordenarCursos;
END;
/
```

- Creación de una colección VARRAY llamada "ListaCursos" en la que se podrán almacenar hasta un máximo de 10 objetos "Cursos":

```
CREATE OR REPLACE TYPE ListaCursos IS VARRAY( 10 ) OF Cursos;
/
```

- Creación, como tipo heredado de "MiembroEscolar", el tipo de objeto "Alumno":

```
CREATE OR REPLACE TYPE Alumno UNDER MiembroEscolar (
        cursoAlumno Cursos
);
/
```

- Creación de una tabla Profesorado de objetos Profesor:

```
CREATE TABLE Profesorado OF Profesor;
```

- Agregación de dos objetos Profesor. El segundo con el método Constructor.

```
INSERT INTO Profesorado VALUES ( Profesor( 2, '51083099F', 'MARIA LUISA', 'FABRE BERDUN', 'F',
'31/03/1975', 'TECNOLOGIA', 4) );

INSERT INTO Profesorado VALUES (Profesor(3, 'JAVIER', 'JIMENEZ', 'HERNANDO', 'LENGUA'));
```

- Creación de una tabla Alumnado de objetos Alumno:

```
CREATE TABLE Alumnado OF Alumno;
```

BD07 - ENUNCIADO TAREA ESCUELA

```
DECLARE
        curso1 Cursos;
        curso2 Cursos;
        unAlumno Alumno;
        listaCursos1 ListaCursos;
        refUnProfesor REF Profesor;
BEGIN
        -- Guardamos en una instancia listaCursos1 de dicha lista, dos cursos
        select ref(p) into RefUnProfesor FROM Profesorado p WHERE codigo = 3;
        curso1 := Cursos(1, 'Curso 1', RefUnProfesor, 20, '1/6/2011', '30/6/2011', 30);
        select ref(p) into RefUnProfesor FROM Profesorado p WHERE dni = '51083099F';
        curso2 := Cursos(2, 'Curso 2', RefUnProfesor, 20, '1/6/2011', '30/6/2011', 30);
        listaCursos1 := ListaCursos(curso1, curso2);

        -- Insertamos en dicha tabla las siguientes filas ...
        INSERT INTO Alumnado VALUES (Alumno(100, '76401092Z', 'MANUEL', 'SUAREZ IBAÑEZ', 'M',
                '30/6/1990', curso1));
        INSERT INTO Alumnado VALUES (Alumno(102, '6915588V', 'MILAGROSA', 'DIAZ PEREZ', 'F',
                '28/10/1984', listaCursos1(2)));

-- Obtenenos de la tabla Alumnado el alumno que tiene el código 100 y lo asignamos a una variable unAlumno
        SELECT VALUE(a) INTO unAlumno FROM Alumnado a WHERE codigo = 100;

-- Modificamos el código del alumno guardado en esa variable unAlumno asignando el valor 101, y su curso debe
-- ser el segundo que se había creado anteriormente, e insertamos ese alumno en la tabla Alumnado
        unAlumno.codigo := 101;
        unAlumno.cursoAlumno := curso2;
        INSERT INTO Alumnado VALUES (unAlumno);
END;
/
```

- Realizamos una consulta de la tabla Alumnado ordenada por cursoAlumno para comprobar el funcionamiento del método MAP.

```
SELECT * FROM Alumnado ORDER BY cursoAlumno;
```