Trabajando con objetos y collections desde PL/SQL.

Constructores definidos por el usuario:

```
CREATE OR REPLACE TYPE rectangle AS OBJECT
(
-- The type has 3 attributes.
length NUMBER,
width NUMBER,
area NUMBER,
-- Define a constructor that has only 2 parameters.
CONSTRUCTOR FUNCTION rectangle(length NUMBER, width NUMBER)
RETURN SELF AS RESULT
);
/


CREATE OR REPLACE TYPE BODY rectangle AS
CONSTRUCTOR FUNCTION rectangle(length NUMBER, width NUMBER)
RETURN SELF AS RESULT
AS
BEGIN
SELF.length := length;
SELF.width := width;
-- We compute the area rather than accepting it as a parameter.
SELF.area := length * width;
RETURN;
END;
END;
/
```

Creando objetos `rectangle` desde un bloque anónimo PL

```
DECLARE
r1 rectangle;
r2 rectangle;
--podemos podemos declarer e inicializar en un solo paso
r3 rectangle:= rectangle(30,40);
BEGIN
-- Podemos seguir usando el constructor por defecto (fijaros que NEW es optativo).
r1 := NEW rectangle(10,20,200);
-- o pedmos usar el constructor definido por el usuario
r2 := rectangle(10,20);
END;
/
```

Inicializar collections . Uso ampliado de `EXTEND`

`EXTEND`	añade una instancia nula
`EXTEND(n)`	añade `n` instancias nulas
`EXTEND(n,m)`	añade `n` copias de la instancia `m`

Ejemplo para inicializar un varray de 512 posiciones

```
Create type arrayType is varray(512) of number;


declare
l_data arrayType;
begin
        l_data := arrayType();
        l_data.extend(512);
        for i in 1 .. 512
        loop
            l_data(i) := i;
       dbms_output.put_line(l_data(i));
        end loop;
end;
/
```

Forzar un error definido por el usuario

```
CREATE or REPLACE PROCEDURE mayorquecinco (num integer)
IS
   horas_actuales NUMBER;
BEGIN
   if num  <=5  then
      RAISE_APPLICATION_ERROR(-20010,'El numero tiene que ser mayor que cinco');
   else
dbms_output.put_line(num);
   end if;
End mayorquecinco;
/
```

Probar el procedimiento anterior en un bloque PL

```
BEGIN
Mayorquecinco(4);
EXCEPTION
   WHEN OTHERS THEN
      dbms_output.put_line(SQLERRM);
END;
/
```

Uso de argumentos `OUT` en un procedimiento

```
CREATE or REPLACE PROCEDURE probarout (num  out integer)
IS
BEGIN
   num:=5;
End probarout;
/

Declare
       num integer;
begin
       dbms_output.put_line('Primera llamada:'||num);
       probarout(num);
       dbms_output.put_line('Segunda llamada:'||num);
end;
/
```