-- This is an introduction to objects in Oracle.

create type address_type as object(
    street varchar2(10),
    zip number(5))
/

create type person_t as object (
    name      varchar2(10),
    address   address_type, -- SEE ABOVE
    birthyear number,
    member    function age(empid in number) return number,
    -- ^^ method header
    pragma    restrict_references(age, WNDS) -- This means
        -- Don't let it change the DB!!
)  
/

-- Write No Database State.
-- Note: You can only use a method in a SELECT statement if it is WNDS.
-- There is also RNDS. That means: No queries allowed.
-- The method implementation comes at the end.

create table employees ( 
    person   person_t, -- defined above
    employeeid  number(4)
)
/

insert into employees values 
    (person_t('Jones',  
              address_type('123 Main', 07102),  
              1971), 
     4412)
/

insert into employees values 
    (person_t('Smith',  
             address_type('43 Elm', 07112), 
             1945), 
     4413)
/

-- The following shows you your objects.

select * from user_objects
/
-- Now we can define select statements for an object.

```
select * from employees
/
```

```
select c.person.name --table/column/field in object
from employees c
/
```

```
select c.person.address.street -- table/column/field/nested field
from employees c
/
```

```
select c.person.address -- table/column/field
from employees c
/
```

```
select c.person.name, c.person.address.street
from employees c
where c.person.name = 'Jones'
/
```

-- Now write the method!
-- The following works only in 2014.
-- At the end is a "year safe" method.

```
create or replace type body person_t as
    member function age(empid in number)
        return number is
            diff employees.person.birthyear%type; -- The variable "diff"
                -- is the same type as the
                -- column birthyear in
                -- table employees !!
            cursor joe is
                (select e.person.birthyear
                    from employees e
                    where e.employeeid = empid);
            begin
                open joe;
                fetch joe into diff; -- Get birthyear
                diff := 2014 - diff; -- Compute age
                close joe;
                return diff; -- Return age
            end;
end;
/
```

-- Now use the method in a select statement.
select c.employeeid, c.person.birthyear,
       c.person.age(c.employeeid) from employees c
/

select c.person.age(4413) from employees c
/

select c.person.age(4413) from employees c where c.employeeid=4413
/

select c.person.age(4412) from employees c where c.employeeid=4412
/

select c.person.age(4411) from employees c where c.employeeid=4411
/

select unique c.person.age(4413) from employees c
/
select distinct c.person.age(4413) from employees c
/

-- We need the unique, because really, this is executed
-- once for every row of the table...

-- If that seems strange, remember what this does:
-- select 3*4 from employees;

-- We have achieved what we wanted.
-- We have written a method.
-- Used in a select statement.

select c.employeeid, c.person.age(c.employeeid) from employees c   -- shows all correctly
/

-- ------------------------

-- Now inheritance

-- Here is a class.
-- It is meant to describe people at a university.

create type parenttype as object
  ( name        varchar2(50),
    description varchar2(1000),
    id          number(2)
  )
NOT INSTANTIABLE NOT FINAL;
-- NOT FINAL means it can have children. It's not a leaf.
-- NOT INSTANTIABLE means that you cannot create instances.

-- If you leave out the NOT FINAL you get a compiler error. Because if you cannot create instances AND you cannot create children the type is useless.
-- If you leave out the NOT INSTANTIABLE, that compiles correctly. Because you CAN create children for a type that has instances.

-- Here is a subclass.

create type childtype UNDER parenttype 
(department varchar2(20)) NOT FINAL
/

-- Note: overriding is not permitted in inheritance

[First surprise. When I created a grandchild type I got an error message. And YET it created the grandchild type.]

create type grandchildtype UNDER childtype 
(specialization varchar2(20))
/

-- Here is a table with subclass data.
create table inheri (oneobj childtype)

-- Now we insert data.

insert into inheri values (childtype('tom','teacher',12, 'cs')
)
/

create table inherit2 (oneobj grandchildtype)
/

insert into inherit2 values (grandchildtype('joe','teacher',12, 'cs', 'AI')
)
/

select * from inheri
/
select * from inherit2
/

You can use constructors in WHERE of SELECT
You can use constructors in SET and in WHERE of UPDATE
You can use triggers with objects in tables.
-- Object constructor in where clause

select c.person.name, c.person.address, c.person.zip
from employees c
where c.person.address = ADDRESS_TYPE('123 Main', 7102);

-- Update of an object column

update employees c
set c.person.address = ADDRESS_TYPE('123 Main', 8202)
where c.person.address = ADDRESS_TYPE('49 Elm', 7102)
/

-- Trigger on object column

create or replace trigger jennifer
after update on employees
for each row
begin
  dbms_output.put_line('Here I am after changing');
end;
/

-- This update works together with the above:

update employees c
set c.person.address = ADDRESS_TYPE('68 Mull', 9202)
where c.person.address = ADDRESS_TYPE('123 Main', 7102)

-- Multiple inheritance is not possible

-- The above method is not "year safe". Now comes the year-safe version.

select sysdate from dual
/

select to_char(sysdate, 'YYYY') from dual
/
select to_char(sysdate, 'YYYYMMDD') from dual
/
select to_char(sysdate, 'YYYY-MM-DD') from dual
/
select to_char(sysdate, 'HH-MI-SS') from dual
/
begin
  dbms_output.put_line( to_char(sysdate, 'YYYY'));
end;
/

For more look here:

http://www.techonthenet.com/oracle/functions/to_date.php

Don't really need a second cursor!!! Because only one value is returned.

create or replace type body person_t as
  member function age(empid in number)
    return number is
      diff number;
      nowyear varchar2(4);
      cursor joe is
        (select e.person.birthyear -- I had a wrong column.. no error msg
         from employees e
         where e.employeeid = empid);
    begin
      select to_char(sysdate, 'YYYY') into nowyear from dual;
      open joe;
      fetch joe into diff;
      close joe;
      diff := nowyear - diff;
      return diff;
    end;
end;
end;
/

=================================
drop table inherit
/
drop table inherit2
/
drop type grandchildtype
/
drop type childtype
/
drop type parenttype