```
SQL> Create or Replace Package ENROLL is
  2  Function func_validate_snum (
  3    p_snum Students.snum%type) return varchar2;
  4  Function func_validate_callnum (
  5    p_callnum SchClasses.callnum%type) return varchar2;
  6  Procedure AddMe (
  7    p_snum Students.snum%type, p_callnum SchClasses.callnum%type);
  8  End ENROLL;
  9  /

Package created.

SQL> Create or replace Package Body ENROLL is
  2  Function func_validate_snum (
  3    p_snum Students.snum%type) return varchar2 is
  4    v_count number;
  5    begin
  6        select count(snum) into v_count from students where snum=p_snum;
  7        If v_count > 0 Then
  8          return null;
  9        Else
 10          return 'student #' || p_snum || ' is invalid';
 11        End If;
 12    end;
 13  Function func_validate_callnum (
 14    p_callnum SchClasses.callnum%type) return varchar2 is
 15    v_count number;
 16    begin
 17        -- count callnum
 18        select count(callnum) into v_count from schclasses where callNum=p_callnum;
 19        -- check if that class exists in the schclasses table
 20        If v_count > 0 Then
 21          return null;
 22        Else
 23          return 'call #' || p_callnum || ' is invalid';
 24        End If;
 25    end;
 26  Procedure Check_Capacity (
 27    p_snum IN Students.snum%type, p_callnum IN SchClasses.callnum%type,
 28    p_error_msg OUT Varchar2) is
 29     v_capacity number;
 30         v_registered number;
 31    begin
 32        select capacity into v_capacity from SchClasses where callNum=p_callnum;
 33        Select count(Snum) into v_registered from enrollments where callnum=p_callnum;
 34        If v_registered < v_capacity Then
 35          p_error_msg := null;
 36        Else
 37          p_error_msg := 'the class was full';
 38        End If;
 39    end;
 40  Procedure Check_Unit_Limit (
 41    p_snum IN Students.snum%type, p_callnum IN SchClasses.callnum%type,
 42    p_error_msg OUT Varchar2) is
 43     v_unit number;
 44         v_Unit_registered number;
 45    begin
```

```
46        select CRHR into v_unit from courses c, schclasses s where callnum=p_callnum AND
s.dept=c.dept AND s.cnum=c.cnum;
47        select sum(CRHR) into v_Unit_registered from courses c, schclasses s,
enrollments e where e.snum=p_snum
48        AND e.callnum = s.callnum AND s.dept = c.dept AND s.cnum = c.cnum;
49        -- make deciscion
50        If v_unit + v_Unit_registered <= 15 Then
51          p_error_msg := null;
52        Else
53          p_error_msg := '15 units exceeded';
54        End If;
55    end;
56  Procedure AddMe (
57    p_snum Students.snum%type, p_callnum SchClasses.callnum%type) is
58    v_valid_snum_msg varchar2(50);
59    v_valid_callnum_msg varchar2(50);
60    v_valid_capacity_msg varchar2(50);
61    v_valid_limit_msg varchar2(50);
62    v_error_msg varchar2(100);
63    begin
64        v_valid_snum_msg := func_validate_snum(p_snum);
65        v_valid_callnum_msg := func_validate_callnum(p_callnum);
66        IF v_valid_snum_msg is null AND v_valid_callnum_msg is null then
67          Check_Capacity(p_snum, p_callnum, v_valid_capacity_msg);
68          Check_Unit_Limit(p_snum, p_callnum, v_valid_limit_msg);
69          IF v_valid_capacity_msg is null Then
70            if v_valid_limit_msg is null then
71              Insert into enrollments values (p_snum, p_callnum, null);
72              v_error_msg := null;
73              commit;
74            else
75              v_error_msg := v_valid_limit_msg;
76            end if;
77          Else
78            if v_valid_limit_msg is null then
79              v_error_msg := v_valid_capacity_msg;
80              -- commit;
81            else
82              v_error_msg :=  v_valid_capacity_msg || ', and ' || v_valid_limit_msg;
83            end if;
84          End If;
85        ELSE
86          If v_valid_snum_msg is not null Then
87            if v_valid_callnum_msg is not null then
88              v_error_msg := 'Both ' || p_snum || ' and ' || p_callnum || ' are invalid';
89            else
90              v_error_msg := v_valid_snum_msg;
91            end if;
92          Else
93            v_error_msg := v_valid_callnum_msg;
94          End If;
95        END IF;
96        -- FINALLY, PRINT MESSAGE
97        IF v_error_msg is null THEN
98           dbms_output.put_line('Enrolled Successfully');
99        ELSE
100          dbms_output.put_line('Enrollment Errors: ' || v_error_msg);
```

```
101        END IF;
102     end;
103  End ENROLL;
104  /

Package body created.

SQL> show error;
No errors.
SQL> pause;

SQL> Exec ENROLL.AddMe(103, 10110);
Enrollment Errors: the class was full, and 15 units exceeded

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(114, 10112);
Enrollment Errors: Both 114 and 10112 are invalid

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(103, 10177);
Enrollment Errors: call #10177 is invalid

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(114, 10110); Enrollment
Errors: student #114 is invalid

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(104, 10125);
Enrolled Successfully

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(105, 10160);
Enrollment Errors: the class was full

PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(101, 10140);
Enrollment Errors: 15 units exceeded


PL/SQL procedure successfully completed.

SQL> Exec ENROLL.AddMe(101, 10160);
Enrollment Errors: the class was full, and 15 units exceeded

PL/SQL procedure successfully completed.

SQL> Spool off;
```