```
with ARRAY_OP,
     RECORD_OP,
     VARIANT_RECORD_OP,
     ACCESS_OP;

separate (EXECUTION_UNIT)
package body OBJECT_OP is

    package ENUM_OP is

        procedure TYPE_DECL (VISIBILITY : CODE.VISIBILITY;
                             LIMITATION : CODE.LIMITATION;
                             COMPLETE   : CODE.DECL_STATE);

        procedure VAR_DECL (VISIBILITY : CODE.VISIBILITY);

        procedure DISPATCH (OPERATOR : CODE.OPERATOR);

    end ENUM_OP;


    package INT_OP is

        procedure TYPE_DECL (VISIBILITY : CODE.VISIBILITY;
                             LIMITATION : CODE.LIMITATION;
                             COMPLETE   : CODE.DECL_STATE);

        procedure VAR_DECL (VISIBILITY : CODE.VISIBILITY);

        procedure DISPATCH (OPERATOR : CODE.OPERATOR);

    end INT_OP;


--    package FLOAT_OP is

--      procedure TYPE_DECL (VISIBILITY : CODE.VISIBILITY;
--                           LIMITATION : CODE.LIMITATION;
--                           COMPLETE   : CODE.DECL_STATE);

--      procedure VAR_DECL (VISIBILITY  : CODE.VISIBILITY);

--      procedure DISPATCH (OPERATOR    : CODE.OPERATOR);

--    end FLOAT_OP;
```

```
procedure TYPE_DECL (CLASS      : CODE.OPERAND_CLASS;
                     VISIBILITY : CODE.VISIBILITY;
                     LIMITATION : CODE.LIMITATION;
                     COMPLETE   : CODE.DECL_STATE)  is
begin
    case CLASS is
        when CODE.ENUM_CLASS     =>
            ENUM_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.INT_CLASS      =>
            INT_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.PACKAGE_CLASS =>
            MODULE_OP.TYPE_DECL (STACKS.PACKAGE_VAR, VISIBILITY,
                                 COMPLETE);

        when CODE.TASK_CLASS     =>
            MODULE_OP.TYPE_DECL (STACKS.TASK_VAR, VISIBILITY,
                                 COMPLETE);

        when CODE.RECORD_CLASS =>
            RECORD_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.VARIANT_RECORD_CLASS =>
            VARIANT_RECORD_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.ACCESS_CLASS =>
            ACCESS_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.COLLECTION_CLASS =>
            ACCESS_OP.DECL_COLLECTION (VISIBILITY);

        when CODE.ARRAY_CLASS    =>
            ARRAY_OP.TYPE_DECL (VISIBILITY, LIMITATION, COMPLETE);

        when CODE.ENTRY_CLASS  | CODE.FAMILY_CLASS | CODE.DELAY_CLASS |
             CODE.ACCEPT_CLASS | CODE.SELECT_CLASS =>
            TASKING_OP.DECL_CONSTRUCT (CLASS);

        when others => null; -- to be added later;
    end case;
end TYPE_DECL;
```

```
procedure VAR_DECL (CLASS      : CODE.OPERAND_CLASS;
                    VISIBILITY : CODE.VISIBILITY;
                    PARAM      : CODE.PARAMETERIZATION;
                    ALLOC      : CODE.DATA_ALLOCATION) is
begin
    case CLASS is
        when CODE.ENUM_CLASS          =>
             ENUM_OP.VAR_DECL (VISIBILITY);

        when CODE.INT_CLASS =>
             INT_OP.VAR_DECL (VISIBILITY);

        when CODE.PACKAGE_CLASS | CODE.TASK_CLASS    =>
             MODULE_OP.VAR_DECL (CLASS, VISIBILITY, ALLOC);

        when CODE.RECORD_CLASS        =>
             RECORD_OP.VAR_DECL (VISIBILITY, ALLOC);

        when CODE.VARIANT_RECORD_CLASS       =>
             VARIANT_RECORD_OP.VAR_DECL (VISIBILITY, PARAM, ALLOC);

        when CODE.ARRAY_CLASS         =>
             ARRAY_OP.VAR_DECL (VISIBILITY, PARAM, ALLOC);

        when CODE.ACCESS_CLASS        =>
             ACCESS_OP.VAR_DECL  (VISIBILITY);

        when others => null; -- not there yet
    end case;
end VAR_DECL;
```

```
    procedure DISPATCH (OPERAND     : CODE.OPERAND_CLASS;
                        OPERATOR    : CODE.OPERATOR;
                        FIELD       : CODE.FIELD_INDEX;
                        FIELD_KIND  : CODE.FIELD_SORT)    is
    begin
        case OPERAND is
            when CODE.ENUM_CLASS        =>
                ENUM_OP.DISPATCH (OPERATOR);

            when CODE.INT_CLASS         =>
                INT_OP.DISPATCH (OPERATOR);

            when CODE.RECORD_CLASS      =>
                RECORD_OP.DISPATCH (OPERATOR, FIELD);

            when CODE.VARIANT_RECORD_CLASS =>
                VARIANT_RECORD_OP.DISPATCH (OPERATOR, FIELD, FIELD_KIND);

            when CODE.ACCESS_CLASS      =>
                ACCESS_OP.DISPATCH (OPERATOR);

            when CODE.ARRAY_CLASS       =>
                ARRAY_OP.DISPATCH  (OPERATOR);

            when CODE.PACKAGE_CLASS | CODE.TASK_CLASS =>
                MODULE_OP.DISPATCH (OPERAND, OPERATOR, FIELD);

            when CODE.ENTRY_CLASS   | CODE.FAMILY_CLASS | CODE.DELAY_CLASS |
                 CODE.ACCEPT_CLASS  | CODE.SELECT_CLASS =>
                TASKING_OP.DISPATCH (OPERAND, OPERATOR, FIELD);

            when CODE.EXCEPTION_CLASS =>
                ERROR.DISPATCH (OPERATOR);

            when others =>
                ERROR.ILLEGAL_INSTRUCTION;
        end case;
    end DISPATCH;


    package body ENUM_OP              is separate;
    package body INT_OP               is separate;
--    package body FLOAT_OP           is separate;

end OBJECT_OP;
```