Introduction:

Procedure oriented / structured programming / modular programming languages ---> C, PASCAL, COBOL --> Importance is given to Functions

Object oriented --> ABAP, C++, JAVA, .Net

---> Importance is given to securing the data

Features of OOPS Oriented Programming Languages

1. **Encapsulation** --> binding data and member functions into single unit (class)

2. **Data abstraction / Data hiding** --> hiding the implementation details

3. **Inheritance** ---> reusability (inheriting the components of parent class into child class)

4. **Polymorphism**--> many forms behavior

Procedure followed for Basic ABAP Report development:

1. Declare variables, work areas, internal tables...

2. Generate selection screen for reading user input (parameters/select-options)

3. Validate the user input

4. Execute select queries to read data from db into work area / internal table

5. Process the work area/ internal table to show the content (data) to the user

Types Declaration Vs class declaration

types : begin of <type name>, (ty_emp) similar to -------> class declaration

field 1, empno components

field 2, ename

---

end of <type name>. endclass.

ty_emp-empno = 1. (invalid)

data wa_emp1 type ty_emp. similar to --> object1 based on class

wa_emp1-empno = 1.

wa_emp1-ename = 'Raju'.

data wa_emp2 type ty_emp. --> object2 based on class

wa_emp2-empno = 4.

wa_emp2-ename = 'Ramesh'.

Class ---> is a User defined data type which is a collection of components

C++ class ---> data members + member functions

Java class --> instance variables + methods

ABAP class ---> attributes + methods + events + interfaces + aliases + macros

ABAP ---> 2 types of classes

1. Local class--> local to an object (program) ---> ABAP editor (se38)

2. Global class --> global to all objects ----> class builder tool (se24)

---> stored inside class pool

Procedure for creating local class:

1. Definition of class

Syntax:

```
class  <class name>   definition  [public] [protected] [private]
                                  [deferred]  [final]  [load] [abstract].
    declaration of components.
endclass.
```

Note: If the class definition contains method declarations, then we need to implement the class

2. Implementation of class

Syntax:

```
class <class name> implementation.
    implementation of methods.
endclass.
```

Note: Class definition and Class implementation doesn't allocate any memory, it only provides template of the class

Instantiate the class ---> creating the object for the class--> memory will be allocated

Object creation --> 2 steps

1. Declare reference (alias) for the class

Syntax:

```
data  <ref.name>  type ref to  <class name>.
```

Note: reference doesn't allocate any memory, it only provides alias.

2. Create object based on reference

Syntax:

create object <ref.name>. --> Memory will be allocated based on components of class

Attribute: is like a variable which can be used to store the data.

Instance Attributes:

In local classes they are declared by using the keyword “Data”.

They are specific to object i.e. for every object, separate memory will be allocated for each instance attribute.

They can be accessed only by using the Object.

Static Attributes:

In local classes they are declared by using the keyword “class-data”.

They are also called as “Class Variables / Class attributes”.

They are not specific to any object.

The memory for static attributes will be allocated whenever the class is loaded in to the memory i.e. memory will be allocated only once which will be shared by all the objects of the class.

They can be accessed either by using the object or by using the class name.

Constant Attributes:

In Local classes, they are declared by using the keyword ‘constants’. They must be initialized at the time of declaration itself. The value of the constant attribute remains same throughout the program.

They are not specific to any object.

The memory for them will be allocated whenever the class is loaded in to the memory i.e. memory will be allocated only once which will be shared by all the objects of the class.

They can be accessed either by using the object or by using the class name.

Note: We go for instance attributes whenever we need to maintain unique (different) values for each object. We go for static attributes, whenever we need to maintain a common (same) value for all the objects of the class. We go for constant attributes, whenever we need to maintain a fixed value for all the objects of the class.

Methods:

- A method is a set of statements which is implemented only once and can be called any no. of times.
- It is similar to subroutine / function module.

Subroutines: - 2 sections ---> definition ---> form...endform

Calling ---> perform <subroutine>.

Subroutine Parameters → keywords used → using, changing, tables

F.M's: 2 sections ---> definition --> function..endfunction (source code tab)

calling ---> call function <f.m>

Function Module Parameters → keywords used → importing, exporting, changing, tables

Local class methods: 3 steps

Declaration (Method prototype declaration) --> Inside Class definition

Implementation --> Inside Class Implementation

Calling --> outside class / inside other method implementation

Method Parameters:

Types of parameters: importing, exporting, changing and returning.

By Default, Importing parameters are obligatory.

We can use the keyword 'optional' as part of local class method parameters to declare it as optional parameter and in case of global class methods, select the checkbox 'optional'

Exporting parameters are always optional.

Procedure for using Local class methods:

1. Declaration

syntax:

methods/class-methods <method name> [parameters list].

2. Implementation

syntax:

method <method name>.

statements.

endmethod.

3. Calling

syntax 1:

call method <method name> [parameters list]

syntax 2:

<method name>([parameters list]).

Method Returning Values:

- In case of ABAP a method can return any no. of values. The no. of return values depends on no of Exporting/changing Parameters.

Returning Parameters:

- In general, a method can return any no. of parameters.
- To restrict a method to return exactly one value, we must use returning parameter.
- If a method contains returning parameter it cannot contain exporting /changing parameters at the same time or vice versa (prior EHP7)
- A method can contain only one returning parameter
- Returning parameter is always passed by value.
- Returning parameters are always optional.

Instance methods:

In local classes they are declared by using the keyword “Methods”.

They can be accessed only by using the object.

They can access any kind of components of the class (instance / static / constant / types).

Static methods:

In local classes they are declared by using the keyword “class-Methods”.

They can be accessed either by using the object or class name.

They can access only static components, constant attributes and types attributes. i.e they cannot access instance components.

Note: In Local classes, the sequence of visibility sections (access specifiers) for the class components should be in the order of public, protected, private sections.

Note: It is recommended to define and implement the local class at the beginning of executable program and explicitly handle the event 'start-of-selection' after the class implementation to indicate the starting point of program execution.

Me keyword:

"Me" keyword refers to current object in execution i.e. as part of every instance method execution (runtime), an implicitly created object (created by SAP) will be available and it refers to the object which is currently executing the method.

Note: If a class contains both attributes and methods, it is recommended to declare attributes under protected/private sections and methods under public sections.

Since we cannot access protected/private components outside the class, we need to access them through one of the public method and call the public method outside the class.

Friend keyword:

In General outside the class, A Object can access only public components of the class directly. i.e protected and private components of the class cannot be accessed outside the class using an object of a class.

In order to enable the objects of the class to access all components of the class outside the class irrespective of the visibility, then we can go for Friend classes.

In local classes "Friend" is a keyword used as part of class definition to consider another class as friend.

Deferred Keyword:

As part of declaring local friend classes, we need to forward declare the friend classes by using **"Deferred" keyword.** This deferred keyword indicates that the class definition is provided somewhere else in the program and not at the beginning of the program.

Note: In Global classes, we declare friend classes as part of 'FRIENDS' tab.

Load keyword:

It is used to load the global classes or global interfaces explicitly in the executable program. This is mandatory before the release 6.20 for accessing static components of the global class or global interface before instantiating them.

From 6.20, it is not required, but in case, if we get any compilation error while accessing the static components / any other components of global class or global interface, then we can load the global class or global interface explicitly from the class library by using load keyword.

Constructor:

- It is special method used for initializing the attributes of the class i.e. whenever an object is created, the attributes should be initialized with some initial values.
- It is special because it is executed/called automatically whenever an object is created (instance const) or whenever a class is loaded in the memory (static const).
- They are always declared in public section.
- They never return any value.
- There are two types of constructors

  1. Instance constructor.

2. Static constructor.

Instance constructor

- They are declared by using the keyword “Constructor”.
- They can contain only importing parameters and exceptions.
- It is specific to object i.e. whenever a new object is created SAP executes “Instance constructor”.
- Instance constructor is executed only once in the life time of every object.

Static constructor

- It is declared by using keyword “class_constructor”.
- It cannot contain any parameters or exceptions.
- It is not specific to any object.
- It is **executed only once in the life time of every class**. I.e. it is executed in either of the following cases.
- **Case 1:** When we access any static components of the class before creating any objects for the class. (or)
- **Case 2:** when we create first object of the class before accessing any static components of the class.

Note:

If a class contains static and instance Constructor and if we instantiate first object before accessing any static components, than SAP first executes Static Constructor and then the Instance Constructor will be executed on behalf of that first object, from the second object onwards only instance constructor will be executed.

If an instance constructor contains any mandatory importing parameters, we must pass the values to those parameters while creating objects itself.

INHERITANCE

It is a process of using the properties (components) of one class inside other class.

The main aim of Inheritance is Reusability of the components i.e. a component declared in one class is accessible inside other class.

In Inheritance, two classes are involved (super class/base class **and** sub class/derived class).

The class which gives properties is called as super class / base class.

The class which takes the properties is called as sub-class/derived class.

Only Public and Protected components can be inherited i.e. Private Components cannot be inherited.

Types of inheritance:

1. **SINGLE INHERITANCE:** A Class acquiring properties from only one super class.
2. **MULTIPLE INHERITANCE:** A class acquiring properties from more than one entity (interface).

   Note: In ABAP, Multiple inheritance can be implemented using the combination of class and interfaces.

3. **MULTILEVEL INHERITANCE:** A class acquiring the properties from another sub-class.

   Inheriting from – is a keyword used as part of local class definition for inheriting another class.

   A class declared as Final cannot be inherited i.e. it cannot have subclass. In case of local classes, we use the keyword 'final' as part of class definition to declare a final class.

Polymorphism

Poly -> many,

Morph-> forms,

Ism-> behaviour

It is a process of making an entity behaving in multiple forms.

In this case, the entity is a method

Example:

Method overloading, Method Overriding.

Method Overloading:

It is similar to function overloading in C++.

It is a process of overloading the same method by passing different Number and different types of parameters.

Eg: methods m1.

Methods m1 importing x type i.

Methods m1 importing x type c.

Methods m1 importing x type l y type i.

ABAP does not support method overloading.

Method Overriding:

- If a subclass redefines a super class method it is called as Method Overriding.

- Whenever a local subclass wants to redefine the super class method, the sub class as to re-declare super class method by using **"redefinition"** keyword.
- Whenever the super class method is redefined in subclasses, we cannot change the visibility / category.
- Only public and protected instance methods can be redefined.
- Whenever a subclass redefines super class method, it is always recommended to call the super class method implementation in the subclass and this is done by using "super" keyword.
- **"Super"** keyword is always used in subclass method implementations (inside redefined super class method) to call the super class method implementation.
- A class declared as final cannot be inherited.
- Static methods cannot be redefined
- Instance public / Instance protected methods declared as final can be inherited but cannot be redefined.
- A static public/static protected methods cannot be declared as final because by default, static methods cannot be redefined.

Hierarchy of constructor execution-scenario 1

- If a super class contains static and instance constructor and subclass without any constructors, and if we instantiate first object of super class/sub class before accessing any static components of super/sub class ,then SAP first executes static constructor of super class and then instance constructor of super class and from second object of super class/sub class only instance constructor of super class will be executed.

Hierarchy of constructor execution-scenario 2

- If a super class contains static and instance constructor and subclass only with static constructor, and if we instantiate first object of sub class before

accessing any static components of super/sub class and also before creating any objects for super class, then SAP first executes static constructors from super class to sub class and then instance constructor of super class and from second object onwards, only instance constructor of super class will be executed.

Hierarchy of constructor execution-scenario 3

- If a super class contains static and instance Constructor and subclass also with static and instance constructor, then it is mandatory for sub class instance constructor to call the super class instance constructor explicitly by using super keyword.

- In this case, if we instantiate first object of sub class before accessing any static components of super/sub class and before creating any objects for super class, then SAP first executes static constructors from super class to sub class (top to bottom) and then instance constructors from sub class to super class (bottom to top) and from second object of sub class onwards, only instance constructors will be executed from sub class to super class.

Hierarchy of constructor execution-scenario 4

If a super class contains instance constructor with mandatory parameters and sub class also contains instance constructor with mandatory parameters and whenever we instantiate the sub class, make sure that you pass values for the parameters of sub class instance constructors and from sub class instance constructor implementation we need to call super class instance constructor by passing values for parameters of super class instance constructor.

Hierarchy of constructor execution-scenario 5

If a super class contains instance constructor with mandatory parameters and sub class without any instance constructor and whenever we instantiate the sub class, make sure that you pass values for the parameters of super class instance constructors while instantiating sub class object.