and classes in detail. Here, we will learn about the accompanying features of object orientation in ABAP.

Inheritance in SAP ABAP

- Inheritance means to derive code functionality from one class to another
- It means defining a class in terms of another (parent) class
- This feature of object orientation promotes reusability of classes
- We can use inheritance while writing a class with which to derive a parent class, by mentioning the name of parent class in the definition of derived class
- Hence, the class that is inherited is called parent or base class or super class, and the class inheriting the base class is the child class or derived class or subclass
- Through inheritance, an object from derived class can obtain characteristics of objects from base class
- The keyword to use for inheritance is 'INHERITING FROM', just beside the class definition

SYNTAX FOR DEFINING CLASS:

```
CLASS <subclass_name> DEFINITION INHERITING FROM <superclass_name>
```

EXAMPLE

```
CLASS Z_Dog DEFINITION INHERITING FROM Z_Animal
```

ABAP PROGRAM EXAMPLE

```
Report ZDATAFLAIR_INHERITANCE.
CLASS DataflairParent Definition.
PUBLIC Section.
Data: df_public(25) Value 'This is from parent class'.
Methods: DataflairParentM.
ENDCLASS.

CLASS DataflairChild Definition Inheriting From DataflairParent.
PUBLIC Section.
Methods: DataflairChildM
```

```
12.
13.  CLASS DataflairParent
Implementation.
14.  Method DataflairParentM.
15.  Write /: df_public.
16.  EndMethod. ENDCLASS.
17.
18.  CLASS DataflairChild
Implementation.
19.  Method DataflairChildM.
20.  Skip.
21.  Write /: 'This is from child
class', df_public.
22.  EndMethod.
23.  ENDCLASS.
24.
25.  Start-of-selection.
26.  Data: DataflairParent Type Ref To
DataflairParent,
27.  DataflairChild Type Ref To
DataflairChild.
28.  Create Object: DataflairParent,
DataflairChild.
29.  Call Method:
DataflairParent→DataflairParentM,
30.  child→DataflairChildM.
```

OUTPUT:

```
1.  This is from parent class
2.  This is from child class
3.  This is from parent class
```

ABAP Access Control and Inheritance

- We know that a class can be defined as public, private, protected
- So when a subclass inherits a superclass, there are certain rules that govern how the class can access objects and data of superclass
- The following table shows whether a derived class has access to base class, based on whether the base class is defined as public/protected/private:

ACCESS	SAME CLASS	DERIVED CLASS	OUTSIDE (NON-DERIVED CLASS)
PUBLIC	Yes	Yes	Available

PROTECTED	Yes	No	No

The above table is explained as follows –

1. Public Inheritance:

- Public data and members of superclass become public data and members of subclass
- Protected members of superclass become protected members of subclass
- Private members of superclass cannot be accessed by subclass

2. Private Inheritance:

- Public members of superclass become private members of subclass
- Protected members of superclass become private members of subclass
- Private members of superclass cannot be accessed by subclass

3. Protected Inheritance:

- Public members of superclass become protected members of subclass
- Protected members of superclass become protected members of subclass
- Private members of superclass cannot be accessed by subclass

Redefining Methods in Sub Class in SAP ABAP

- We can redefine methods of superclass in subclass
- We usually do this so that we can have subclass-specific methods
- However, we must keep the section of redefinition of the method the same as the parent method
- We only need to use the name of the inherited method and can access its components using ‘super’ reference

Encapsulation in ABAP

- Encapsulation in object orientation means wrapping data and functions together
- This hides the data and function from the outside world, thus promoting data hiding
- Data hiding means obstructing the view of private data and functions from unwanted third parties, and data abstraction means showing users only what they need to see
- In ABAP, we can do encapsulation via access methods – public, private, protected.
- We can also perform encapsulation via interfaces (which are similar to classes, the only difference is that we do not implement anything in an interface, it has to be done via class inheriting the interface, as shown in example below)

EXAMPLE OF ABAP ENCAPSULATION VIA INTERFACE

```
1.  Report ZDATAFLAIR_ENCAPSULATION.
2.  Interface df_interface.
3.     Data text1 Type char35.
4.     Methods Dataflairmethod1.
5.  EndInterface.
6.
7.  CLASS DataflairClass1 Definition.
8.     PUBLIC Section.
9.        Interfaces df_interface.
10. ENDCLASS.
11.
12. CLASS DataflairClass2 Definition.
13.    PUBLIC Section.
14.       Interfaces df_interface.
15. ENDCLASS.
16.
17. CLASS DataflairClass1 Implementati
18.    Method df_interface~Dataflairme
19.       df_interface~text1 = 'Datafl
Interface method'.
20.       Write / df_interface~text1.
21.    EndMethod.
22. ENDCLASS.
23.
24. CLASS DataflairClass2 Implementati
25.    Method df_interface~Dataflairme
26.       df_interface~text1 = 'Datafl
Interface method'.
27.       Write / df_interface~text1.
28.    EndMethod.
29. ENDCLASS.
30.
31. Start-Of-Selection.
32.    Data: DataflairObject1 Type Ref
```

```
33.         DataflairObject2 Type Ref To
DataflairClass2.
34.
35.     Create Object: DataflairObject1
DataflairObject2.
36.     CALL Method:
DataflairObject1→df_interface~Dataf
37.
DataflairObject2→df_interface~Dataf
```

OUTPUT:

```
1.  DataflairClass 1 Interface method
2.  DataflairClass 2 Interface method
```

Polymorphism in SAP ABAP

- It means using one thing for different operations

- It occurs usually in inheritance – for e.g. redefining methods which we saw under the concept of inheritance
- Polymorphism means redefining methods to either overload them or override them
- Overloading methods means when we use the same method name but use different parameters
- Overriding methods means when we use the same method name and parameters, but the two methods are related via inheritance relationship

ABAP Polymorphism Example(Redefining methods)

```
1.  Report ZDATAFLAIR_POLYMORPHISM.
2.  CLASS df_class_02 Definition
    Abstract.
3.  PUBLIC Section.
4.  Methods: prgm_type Abstract,
5.  dfapproach1 Abstract.
6.  ENDCLASS.
7.
8.  CLASS df_class_02 Definition
9.  Inheriting From df_class_01.
10. PUBLIC Section.
11. Methods: prgm_type Redefinition,
12. approach1 Redefinition.
13. ENDCLASS.
```

```
15.  CLASS class_procedural
Implementation.
16.  Method prgm_type.
17.  Write: 'An apple'.
18.
19.  EndMethod. Method approach1.
20.  Write: 'a fruit'.
21.
22.  EndMethod. ENDCLASS.
23.  CLASS class_OO Definition
24.  Inheriting From class_prgm.
25.  PUBLIC Section.
26.  Methods: prgm_type Redefinition,
27.  approach1 Redefinition.
28.  ENDCLASS.
29.
30.  CLASS class_OO Implementation.
31.  Method prgm_type.
32.  Write: 'An onion'.
33.  EndMethod.
34.
35.  Method dfapproach1 .
36.  Write: 'a vegetable'.
37.  EndMethod.
38.  ENDCLASS.
39.
40.  CLASS class_type_approach
Definition.
41.  PUBLIC Section.
42.  CLASS-METHODS:
43.  start Importing df_class_01_prgm
44.  Type Ref To class_prgm.
45.  ENDCLASS.
46.
47.  CLASS class_type_approach
IMPLEMENTATION.
48.  Method start.
49.  CALL Method
df_class_01→prgm_type.
```

```
52.  CALL Method
df_class_01→approach1.
53.  EndMethod.
54.  ENDCLASS.
55.
56.  Start-Of-Selection.
57.  Data: df_class_01 Type Ref To
class_procedural,
58.  df_class_02 Type Ref To class_OO.
59.
60.  Create Object df_class_01.
61.  Create Object df_class_02.
62.  CALL Method
class_type_approach⇒start
63.  Exporting
64.
65.  class1_prgm = df_class_01.
66.  New-Line.
67.  CALL Method
class_type_approach⇒start
68.  Exporting
69.  class1_prgm = df_class_02.
```

OUTPUT:

```
1.  An apple is a fruit
2.  An onion is a vegetable
```

Summary

Thus, in this tutorial we learnt more about the concepts of object orientation – inheritance, polymorphism, encapsulation – which form the core of reusability, data hiding and data

abstraction properties of object-oriented programming, especially in ABAP.

Your opinion matters

Please write your valuable feedback about DataFlair on **Google**

Tags: Encapsulation in SAP ABAP | Inheritance in SAP ABAP | Polymorphism in SAP ABAP | SAP ABAP OOPS

1 RESPONSE

Comments 1 Pingbacks 0

pragyan sahoo March 18, 2023 at 10:04 am

i think here the table maintained here for PUBLIC ,PRIVATE AND PROTETED member accessibility needs a bot of modification.

The Private members of the class is not visible to Derived class where as the protected members is visible

Reply

LEAVE A REPLY

Comment *

Name * **Email** *

Post Comment

Trending Courses

Trending Data

Trending Tutorials

Free C++ Certification Course
Free DSA Certification Course [Hindi]
Free Web Development Certification Course [Hindi]
Free Java Certification Course
Free Scala Certification Course
Free Angular Certification Course
Free Java Certification Course [Hindi]
Free JavaScript Certification Course [Hindi]

Trending Data

Science Courses

Free Python Certification Course
Free Machine Learning Certification Course [Hindi]
Free NumPy Certification Course [Hindi]
Free SciPy Certification Course [Hindi]
Free TensorFlow

Data Science Tutorials
Machine Learning Tutorials
Python Tutorials
TensorFlow Tutorials
Pandas Tutorials
SQL Tutorials
Cassandra Tutorials
MongoDB Tutorials
IoT Tutorials
R Tutorials
SAS Tutorials
AI Tutorials
Tableau Tutorials
PowerBI Tutorials
QlikView Tutorials
Qlik Sense Tutorials
SAP HANA Tutorials
Big Data Tutorials
Hadoop Tutorials
Spark Tutorials
Flink Tutorials
Kafka Tutorials

Java Tutorials
Spring Tutorials
Django Tutorials
JavaScript Tutorials
Angular JS Tutorials
Android Tutorials
Data Mining Tutorials
Cloud Tutorials
AWS Tutorials
Blockchain Tutorials
Salesforce Tutorials

Free Matplotlib Certification Course [Hindi]
Free Pandas Certification Course [Hindi]
Free Python Project Training Course [Hindi]
Free Python Certification Course [Hindi]

Trending Big Data

Courses

Free Big Data - Hadoop Certification Course
Free Apache Spark - Scala Certification Course
Free Apache Flink Certification Course
Free Apache Kafka Certification Course
Free Big Data Hadoop + Spark Certification Course
Free AWS Course for AWS Certified Cloud Practitioner
Free UPSC Prelims Mock Test Series