资深SAP专家扛鼎之作
为SAP ABAP开发提供解决方案

孙东文 郭欢 张岩 等编著

SAP

ABAP开发详解与高端应用

SAP

ABAP

Detailed Development Technology - High-end Application

机械工业出版社
CHINA MACHINE PRESS

SAP ABAP 开发详解与高端应用

孙东文　郭　欢　张　岩　等编著

机械工业出版社

本书主要介绍了ABAP面向对象编程、ALV列表、接口、增强及ABAP开发人员求职面试的相关技术问题。ABAP面向对象编程部分由浅入深地讲解了面向对象的相关概念，并从实际应用出发，举例说明了ABAP编程所涉及的本地对象和全局对象的编辑实现及应用效果；ALV列表部分全面地介绍了ALV的种类以及各种ALV的实现方式，并详述了面向对象ALV列表及ALV树形列表的编辑实现及应用效果；接口部分系统地介绍了各类接口的原理及实现方法，并详述了IDOC、RFC、BAPI的相关概念及原理；增强部分对增强的升级及不同时期的各代增强的原理、查找办法、实现方式作了详细阐述，并举例说明了每一代增强的编辑实现及应用效果；ABAP面试问题及答案部分提供了较为全面的面试问题集，为求职者打开方便之门。

本书提供了大量配套资源及实例源码，深入剖析了SAP NetWeaver架构的关键技术，是Java等开发人员深入学习SAP系统的必备指南，还适用于SAP技术人员和Java\.NET平台下SAP接口程序的开发人员。

图书在版编目（CIP）数据

SAP ABAP开发详解与高端应用 / 孙东文等编著. —北京：机械工业出版社，2015.6
ISBN 978-7-111-50126-8

Ⅰ. ①S… Ⅱ. ①孙… Ⅲ. ①企业管理－应用软件－软件开发
Ⅳ. ①F270.7

中国版本图书馆CIP数据核字（2015）第114323号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策划编辑：丁 诚 责任校对：张艳霞
责任编辑：丁 诚 周 萌
责任印制：李 洋

三河市宏达印刷有限公司印刷

2015年7月第1版·第1次印刷
184mm×260mm·23.5印张·579千字
0001－3000册
标准书号：ISBN 978-7-111-50126-8
定价：69.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：（010）88361066
读者购书热线：（010）68326294
（010）88379203

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
教育服务网：www.cmpedu.com
金 书 网：www.golden-book.com

前　言

随着 SAP 软件在中国企业的广泛应用，SAP 实施及运维的 IT 服务行业也异军突起，ASP 相关技术也得到了广泛的应用，这其中不乏 SAP 开发的高端技术。

作者根据 SAP 官方教材和项目经验对 ABAP 开发技术做了一下分类，将 ABAP 面向对象编程、ALV 的面向对象应用、接口和增强归作为高级应用技术纳入本书。

以往的书籍仅对 ABAP 高级应用部分做原理剖析，能够提及对面向对象、接口及增强实现的图书是少之又少。本书特点在于对各技术原理、实施前提、实施步骤、实施结果作了详尽的说明，更有 ABAP 求职者相关基础技术及高端技术的面试题，以帮助那些做技术提升的 SAP 开发人员。

学习 SAP 高级开发技术，开发者不仅需要提高自身水平（例如，从面向过程的编程到面向对象编程的转变，从单一开发语言的编程到不同语言之间编程的转变及其之间数据关系映射的了解），也需要熟悉 SAP 相关操作（例如，远程 RFC 的 TCP 链接的创建和调试），更需要 SAP 开发者对各种技术实施后最终体现的结果的认同（例如，如何应用创建的 RFC 链接，高端技术实施应用以后有什么样的效果，是否和需求相吻合等）。

参与本书编写的有孙东文（笔名：东方先生）、郭娟、郭欢、张岩，由于水平有限，书中难免有疏漏和不足，请读者批评指正。

编　者

目　录

第 1 章　ABAP 面向对象编程

SAP 系统从 4.0 版本开始对 ABAP 语言进行了扩充，增加了 ABAP 类和对象功能。也就是说，可以使用类的所有特性，如封装、多态、接口和继承等，使复杂的应用变得更容易控制且简单化，并且 SAP 要求 ABAP 语言严格向上兼容。

ABAP 类语言跟其他面向对象的语言一样，有类和接口，类又包括属性、方法和事件。

一、面向对象

■ 对象（Object）

对象是包含部件并提供服务的程序代码段，其中部件部分代表该对象的属性，而所提供的服务则被称为方法。

■ 类

类是对象的定义，通过属性和方法的封装来描绘对象。类是对象的模板，相应地，一个对象的类型与它的类相同。一个类是一个对象的抽象描述，它是构建一个对象所需要的一系列指令。对象的属性和方法由类的部件定义，类描述了对象的框架和行为。

■ 类和对象的关系

从技术角度看，对象是程序中类的实例，在运行环境中根据类的定义生成。类和对象是一对多的关系，即可以创建某个类的多个对象，而每个对象均是独立的，拥有代表自身状态的数据值集。

■ 全局类和本地类

全局类是在 Class Builder（事务码：SE24）中定义的类，其被存储在类池中的程序类型为接口池（J）和类别库（K）定义在 Include 程序中的本地类也可作为全局类。所有 ABAP 程序都可以访问并使用全局类。本地类是在 ABAP 程序中定义的类，它和本地接口只能用于定义本地类的程序中，应用时系统首先查找指定名字的本地类，如果没有找到，再查找全局类。除了有效性问题，全局类和本地类在使用时没有区别。

二、名词解释

1．抽象（abstraction）

抽象是指对于一个过程或者一件制品的某些细节有目的地隐藏，以便把其他方面、细节或者结构表达得更加清楚。

抽象是控制复杂性时最重要的工具。人们通常使用一些简单的工具来建立、理解和管理复杂的系统。其中最重要的技术称为“抽象”。

2．封装

封装就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。

3. 继承

继承可以使用现有类的所有功能，并在无需重新编写原有类的情况下对这些功能进行扩展。

4. 多态

多态允许将父对象设置成为与一个或多个其子对象相等的技术，赋值之后，父对象就可以根据当前赋值使它的子对象以不同的方式运作。简单地说，就是允许将子类类型的指针赋值给父类类型的指针。

1.1 面向对象的基本要素

1.1.1 封装

对象可以对其内部资源的可见性进行限定，每个对象都有一个接口，决定了其他对象如何与之交互。

基类包括多个派生类，但该基类是作为模板出现，并不需要有任何对象作为实例。

本地类由 ABAP 源代码组成，封装在 CLASS ... ENDCLASS 中。一个完整的类定义包括声明部和执行部，声明中的<class>部分是一个语句块。

语法：
```
CLASS <class> DEFINITION.
  ...
ENDCLASS.
```

它包含类所有部件（属性、方法、事件）的声明。当定义一个本地类时，声明部分属于全局 Program 数据，所以必须把它放在 Program 的开始。

如果在类的声明部声明方法，必须也为它写一个执行部。这是由更多的语句块组成的。

语法：
```
CLASS <class> IMPLEMENTATION.
  ...
ENDCLASS.
```

在类的执行部分包含了所有类方法的执行，本地类的执行部是一个处理块。那些不在一个处理块中的后面的代码是不能访问的。

1. 声明为抽象类（Abstract Class）

语法：
```
CLASS <class> DEFINTION ABSTRACT.
  METHODS meth ABSTRACT...
ENDCLASS.
```

不可用 CREATE OBJECT 语句创建类对象，只作为派生类模板；抽象方法不可在类本身实现，而要在其派生出的非抽象类中实现。

注：抽象方法的类必须为抽象类,不能在类中实现；在派生类实现中，使用 REDEFINTION 对类方法重新定义，重新定义过程中不可使用 SUPER。

2．最终类和最终方法

语法： CLASS <class> DEFINITION FINAL.
METHODS meth FINAL...
ENDCLASS.

最终类（Final Class）是不能被继承的类，最终方法是不可重新定义的方法，不一定只出现在最终类中，但最终类中的所有方法都是最终的，而最终类中无需指明 FINAL。

注：类的种别决定该类的实例可以被所有用户创建，只能被其本身或派生类创建或只能通过其自身方法创建，ABAP 常用类型如下：

- Final：全局类不能在程序中被继承，即最终类。
- Abstract：只能被继承不能被实例化。
- For Testing：ABAP 单元的测试类。

最终类可同时为抽象类，但只能包含静态成员；最终方法则不可以同时为抽象方法。

1.1.2 多态

相同名称的方法在不同的类中呈现出不同的行为，当一个相同的方法在不同的类中有不同的实现就是多态。在 ABAP 中，多态可以通过继承和接口实现方法。

多态性是指相同的用户操作在不同类中有不同的实现形式，在继承中，多态性是通过方法重载及引用变量实现的，即子类可重新定义并以不同的方式实现基类中的公有或保护方法，此时，基类中的方法需为实例方法。派生类声明部分需进行 METHOD meth REDEFINTION，该方法必须和基类中的方法具有相同接口，但可通过不同的代码实现。

当访问基类时，将采取基类的原始方法实现；而访问派生类时，则采用新的派生类方法实现，原有基类方法将被屏蔽，包括通过自身引用 me-> meth，不过可使用 SUPER 来指定其基类 CALL METHOD super->meth。

注：只有在方法重载实现代码内部才能使用 SUPER。最终方法不能在子类中重定义，最终类没有子类。

1.1.3 继承

继承可以定义多个类之间的执行关系，可以从已存在类的基础上建立新的派生类，派生类可以继承原有类的方法与属性，或者添加新的类成员；高级类的运行方式也可能得到调整和扩展。

继承允许从一个类中引出新的类。新的类叫子类（Sub Class），被引出的类叫父类（Super Class）。

1．继承的声明

语法： CLASS <class> DEFINITION INHERITING FROM <superclass>.
……（省略完整代码）
ENDCLASS.

在子类中，父类的类型为 Public 和 Protected 的组件才是可见的，并且继承只能是单继承。所有 ABAP 对象的根类是 OBJECT。

2．方法的重定义

语法： CLASS <class> DEFINITION INHERITING FROM <superclass>.
METHODS meth REDEFINITION...
ENDCLASS.

方法的重用使用声明 REDEFINITION 在 METHODS 中定义，但是不能改变它的接口。用 ME->代替自身的对象变量，用 SUPER->代替父类的对象变量。

在类定义时，使用 INHERTING FROM 附加项可以指定各个子类之间的继承关系。ABAP 中继承为单一继承，即一对多。

3．继承过程中各个成员的可见性

- 派生类中的公有成员以及基类的公有成员，都可以通过"->"在类外部获得。
- 派生类的被保护成员以及基类的被保护成员，都不可以通过"->"在类外部获得，但可在派生类内部使用。
- 派生类中的私有成员只包括其本身私有部分定义的成员，只能在派生类内部使用。

继承类和基类的公有成员和被保护成员享有共同的命名空间，而私有成员则可在不同类之间出现重名情况。基类应用变量可以指向其继承类对象，反之则会被视为错误的类型转换。

1.1.4 面向对象的成员

在类中能够使用 TYPE 语句来定义自己的 ABAP 数据类型，类型不是实例相关的，对类中的所有对象只存在一次。

1．成员类型

ABAP 中可以定义三种不同的成员类型，分别是属性、方法和事件，各成员的可见性以及生存周期在类定义的同时被确定。

（1）属性

属性是在类内部的数据对象，用于描述类的状态。属性可以由多个数据类型组成，对象的说明由属性内容定义。一种类型的属性是关联变量，它允许建立和定位对象。关联变量能够在类内部定义，也允许从类中访问对象。

实例属性

实例属性的内容定义了专用实例，可以用 DATA 语句声明。

静态属性

静态属性的内容定义了类的声明，对类的所有实例有效。静态属性对每个类存在一次，可以用 CLASS-DATA 语句来声明它们，它们在类的全部运行时间都可以被访问。

常量

常量是特殊的静态属性，在声明时设置值，然后就不能再修改。声明使用 CONSTANTS 语句。

（2）方法

方法是对象行为的实现部分，可以将其视为是类中的过程，用于访问或修改对象状态。

方法的定义和参数接口与函数模块相同，可以在声明部定义。

语法： METHODS <meth> IMPORTING.. [VALUE（]<ii>[）] TYPE type [OPTIONAL]..
EXPORTING.. [VALUE（]<ei>[）] TYPE type [OPTIONAL]..
CHANGING.. [VALUE（]<ci>[）] TYPE type [OPTIONAL]..
RETURNING VALUE（<r>）
EXCEPTIONS.. <ei>..

然后在执行部实现，语法如下。

语法： METHOD <meth>.
...
ENDMETHOD.

本地数据类型和方法的定义与其他 ABAP 程序相同，可以用语句 CALL METHOD 来调用。

语法： CALL METHOD <meth> EXPORTING... <ii> =.<f i>...
IMPORTING... <ei> =.<g i>...
CHANGING ... <ci> =.<f i>...
RECEIVING r = h
EXCEPTIONS... <ei> = rc i...

参数解释：

- Importing：指定一个或多个输入参数。
- Exporting：指定一个或多个输出参数。
- Changing：指定一个或多个输入/输出参数。
- Value：指定参数传递形式，分为值传递和引用传递两种形式。默认为引用传递，如果使用 Value 选项则为值传递。
- Type：指定参数类型，每一个参数都必须有指定的类型。
- Optional：指定该参数为可选时，程序将使用该参数类型的初始值进行传递，或者是使用 Default 选项后指定的默认值。
- Returning：该选项可以替代 Importing 和 Exporting。

实例方法

可以用 METHODS 语句声明实例方法，它们可以访问类中的所有属性，能够触发所有的事件。

静态方法

可以使用 CLASS-METHODS 语句声明静态方法，它们只能够访问静态属性和触发静态事件。

构造方法（类构建器）

同普通方法一样，可以通过 CALL METHOD 调用，当建立一个对象（CONSTRUCTOR）或第一次访问类的部件时（CLASS_CONSTRUCTOR），有两个叫 CONSTRUCTOR 和 CLASS_CONSTRUCTOR 的专用方法能够自动被调用。

（1）构造方法

- 构造方法是运行时环境自动调用的一种方法，用于为对象设定一个初始化状态，不能

在程序运行过程中由程序代码调用。与其他方法类似，构造方法也存在静态和实例两种形式。

- 构造方法不一定需要在类中定义，系统通常会自动生成一个默认的构造方法，将对象状态设置为初始值。
- 构造方法中不包含任何形式的输出参数，其功能只是定义对象状态，而不是对其进行改变。
- 构造方法在 Create Object 语句中被调用，根据实例构造方法的定义，该语句本身也可以包含输出参数 Exporting 和异常返回值 Exceptions 选项。
- 类构造方法属于静态方法，只能在程序中被调用一次，即第一次使用该类时被调用，该方法也必须在公有部分声明，其名称必须为 Class_Constructor。
- 类构造方法不包含接口，而且在其中只能访问类的静态属性。

（2）析构方法

在某些面向对象语言中还存在析构方法的概念，该方法在对象删除过程中被调用，目前 ABAP 对象中尚未提供该方法的实现形式。

每个类都有一个隐含的方法，在 Java 或 C++中叫构造函数；在一般语言中，构造函数和类名相同，但是在 ABAP 中，有一个保留的名字，叫做 Constructor。当调用语句 Create Object 创建对象的时候，这个函数会被执行。

构造函数 Constructor 可以分为实例 Constructor 以及静态 Constructor。对于实例 Constructor（声明的时候使用 Methods），可以有 Import 参数以及 Exceptions，而静态 Constructor（声明的时候使用 Class-Methods），没有参数传递。

注意：静态构造函数一定会在以下动作之前执行。

- 创建对象，Create Object。
- 访问静态属性，Class=>Attribute。
- 调用静态方法，Call Method_Class=>Method。
- 注册一个静态事件句柄。

（3）事件

用于一个类对象发布其状态的改变，因而其他对象可以捕获该方法并做出响应。方法是类的内部处理过程，定义了一个对象的行为。它可以访问所有类的属性，也可以修改对象的数据内容，还可以有一个参数接口，用户可以通过带参数调用，接收返回值。私有属性只能通过方法类修改。

对象或者类能够在其他对象或类中通过事件来触发事件处理的方法。在一个一般的方法调用中，一个方法可以被许多用户调用。当一个事件被触发，一系列事件处理方法可以被调用。在运行之前，触发器和处理方法之间的联系没有被建立。在一般的方法调用中，调用的 Program 决定了它希望调用的方法，这些方法必须存在。根据事件，处理方法确定它希望响应的事件。对所有的事件不需要注册处理方法。

类的事件可以在类的方法中通过使用 RAISE EVENT 语句触发。可以通过使用字句 FOR EVENT <evt> OF <class>声明同一个或不同的类的方法作为一个事件处理的方法。

事件有一个与方法相似的参数接口，但是只有输出参数时这些参数通过触发器（RAISE

EVENT 语句）传递给事件处理方法，它把它们作为输入参数接收。

触发器和处理方法之间的联系通过 SET HANDLER 语句动态建立。触发器和处理方法可以是对象或类，这取决于使用的是实例还是静态事件或静态处理方法。当一个事件被触发，相关事件处理方法在所有注册的处理类中执行。

实例事件

可以用 EVENTS 语句声明实例事件。一个实例事件只能在实例方法中被触发。

静态事件

可以用 CLASS-EVENTS 语句声明静态事件，所有的方法（静态或实例）都可以触发静态事件。静态事件是只能在静态方法中被触发的事件类型。

1）触发器和处理事件

在 ABAP 对象中，触发器和处理一个事件意味着特定的方法作为触发器和触发事件执行。对于其他方法的相应处理器，这意味着当事件触发时处理方法被执行。

在 ABAP 对象中处理事件有以下两种方法。

- 在声明部声明事件。
- 使用它的一个方法触发事件。

事件声明

声明实例事件语法如下所示。

语法： EVENTS <evt> EXPORTING... VALUE(<e_i>) TYPE type [OPTIONAL]..

声明静态事件语法如下所示。

语法： CLASS-EVENTS <evt>...

两个语句都有同样的语法，EXPORTING 字句用于指定传递给事件处理的参数。参数一般是值传递。实例事件一般包含默认参数 SENDER，它具有与事件声明的接口或类型相关的类型。

触发事件

实例事件可以被类中的所有方法触发，静态事件可以被静态方法触发。

语法： RAISE EVENT <evt> EXPORTING... <e_i> = <f_i>...

对每个没有被定义为正式参数的<e_i>，必须在 EXPORTING 字句中传递一个相应的实际参数<f_i>。自定义的 ME 自动被传递给默认参数 SENDER。

2）事件处理

事件处理使用指定的方法，处理事件方法的要求有以下两个。

- 被定义成为事件的事件处理方法。
- 在执行时为事件注册。

事件处理方法的声明

所有的类可以包含针对其他类事件的处理方法。当然，也可以在同一个类中定义处理方法。实例方法的声明语句如下。

语法： METHODS <meth> FOR EVENT <evt> OF <cif> IMPORTING.. <e_i>..

静态方法使用 CLASS-METHODS 来代替 METHODS。<evt>是在类或接口中的一个事件声明。

一个处理方法的接口只能包含在事件<evt>声明中定义的正式参数中。参数属性也由事件使用。处理方法不一定需要使用 RAISE EVENT 语句所传递的所有参数，如果希望使用默认参数 SENDER，必须在接口中包含。这个参数允许一个实例事件处理访问触发器，例如，允许它返回结果。

一旦在类中定义了事件处理方法，这意味着类的实例或类本身在理论上能够处理在一个方法中触发的事件<evt>。

注册事件处理方法

为一个事件处理方法响应事件，必须在运行触发器时确定哪个用于响应。通过以下语句实现。

语法： SET HANDLER... <h_i>... [FOR]...

它将处理方法清单和相应的触发方法联系起来。事件有四个不同的方法：

- 在类中声明的实例事件。
- 在接口中声明的实例事件。
- 在类中声明的静态事件。
- 在接口中声明的静态事件。

针对以上四种情况，SET HANDLER 语句语法和作用是不同的。

对于实例事件，可以使用 FOR 字句来指定实例需要注册的处理。也可以指定一单个实例作为触发器，使用关联变量<ref>。

语法： SET HANDLER... <h_i>...FOR <ref>.

或者可以为所有能够触发事件的实例注册处理方法。

语法： SET HANDLER... <h_i>...FOR ALL INSTANCES.

然后，注册器为在注册处理时还没有建立的触发实例提供事件。可以使用 FOR 字句定义静态事件。

语法： SET HANDLER... <h_i>...

注册器自动提供给整个类，或者给所有实现含静态事件接口的类。对于接口，注册器还应用于那些在动态运行中直到处理程序的类。

3）事件处理的生命期

在 RAISE EVENT 后，所有注册的处理方法在下一个语句处理之前被执行（同步事件处理）。如果一个处理方法自己触发事件，则它的处理方法在初始处理方法之前。为避免无穷的递归，事件处理一般只嵌套 64 层。

处理方法根据它们被定义的顺序执行，一旦事件处理被动态注册，则不能认为它们在特定的顺序执行，相反，必须认为所有的事件处理会同步执行。

注：在 ABAP 中，接口需要另外实施（其实就是在类中实施），同时又独立于类。在接口

中只有声明部分，而没有具体的实施部分。

- 接口必须在类的公共部分列出（Public Section）。
- 接口中的操作在类中实现，并且作为类的方法；所有接口中的方法必须在类的实现部分实施。
- 在接口中定义的属性、事件、常量、类型等在声明它的类中自动可见。
- 实施接口的方法：Interface～Component Name。

2．成员的可见性

成员的可见性是基于封装原则确定的。

（1）PUBLIC SECTION

公有部分定义的类成员可以被所有对象使用，包括类方法及其派生类中定义的方法。这些组件构成了类的外界接口部分。

（2）PROTECTED SECTION

保护部分中定义的类成员只能被类及其派生类中的方法使用，外部不可见，这些组件连同公有组件构成了类和派生类对象之间的接口。在 4.5B 中，继承是无效的，所以 Protected 范围与 Private 相同。

（3）PRIVATE SECTION

类成员只能被类本身的方法使用。

注：

公有属性（Public Attributes）定义在 Public Section 中，其内容可以在类的外部看到或者修改。尽量减少公共属性，其原因可以参考面向对象语言的特性——封装。尽可能在类的内部修改属性，留给外部的只有接口，避免外部直接操作类里面的属性（或称数据）。

只能在类的内部访问或者修改的数据，定义在 Private Section 里面。

在类中，每个静态属性都是唯一的。属性的值在所有类的实例中是相同的。用 class-data 声明。

保护属性（Protected Attributes）定义在 Protected Section 里面，其作用是更改区域里面的属性或者方法，不能被外部调用，但是在它的子类中是可见的。也就是说，其子类可以访问 Protected Section 中的数据，这和 Private Section 是不同的。

Public Method、Private Method、Static Method、Protected Method 及其对应的属性同上。

1）使用对象的一般步骤

- 定义类的声明与方法实现。
- 使用 DATA 语句中的 TYPE REF TO 选项参照类类型声明引用变量。
- 使用 CREAT OBJECT 语句创建对象。
- 通过-> 或=>运算符访问对象或类组件。

2）访问对象组件

对象是 ABAP 程序所获得内存空间里的一个类的实例。首先在 DATA 声明里用 TYPE REF TO <类>声明 ABAP 数据类型，然后用 CREATE OBJECT 创建类的实例。也可以用 MOVE 对兼容的对象之间赋值，以后就可以使用。

3）访问内容语法格式

- 一个对象的实例属性或静态属性 oref-> attr。
- 类的静态属性 class=>attr。
- 在类内部访问自身实例属性或静态属性 me->attr 或 attr。
- 对象的实例属性或静态方法 CALL METHOD oref->meth。
- 类的静态方法 CALL METHOD class=>meth。
- 在类内部访问自身实例方法或静态方法 CALL METHOD me->attr 或 CALL METHOD attr。

1.1.5 接口

当有些类具有相同的功能却有不同的行为时，就产生了接口的概念。接口这样描述自己："对于实现所有类，看起来都应该像现在这个样子"。因此，采用了一个特定接口，所有代码都知道对于那个接口可能会调用什么方法。所以常把接口用于建立类和类之间的一个"协议"。有些面向对象的程序设计语言采用了一个名为"Protocol"（协议）的关键字，它做的便是与接口相同的事情。接口只有声明部分，并不提供实现的细节。

1．接口的定义与实现

对象引用包括类引用和接口引用，通过 ABAP 类的实例及引用可构建各种商业应用模块及其中的元素。

接口是一个独立结构，可以在其中定义一些成员并在具体类中实现，其作用是对类中已定义的成员进行扩展。实现接口后，其成员将成为公有成员，但类可自行对接口中的方法以自身特定形式实现。

1）定义接口：与类的定义相似，接口作为独立 R/3 Repository 的对象在 Class Builder 中定义，也可以依附于程序在 ABAP 程序中定义（本地类）。

2）实现接口：接口没有自己的实例，因而不需要进行实现，其实现通过具体类进行。

注： 接口 intf 中的成员 icomp 在类内部实现过程以名称 inf～imeth 的形式出现。

接口的实现只能出现在公有部分，接口中定义的所有组件都将被添加为该类的公有成员。在类的实现部分须包含所有的接口方法。

一个接口可被任意多个不同类实现，该接口中定义的成员集在各个类中名称相同，然而实现方法不同，也就体现了多态性。若一个类中除了接口之外没有定义任何类自身的公有成员方法，则接口就成为了该类的全部"外部接口"。

2．接口的定义方法

接口跟类一样，也有全局接口和本地接口。

1）全局接口的定义由类定义工具实现（事务码 SE24）。

2）本地接口的定义

本地接口封装在 INTERFACE...ENDINTERFACE 中，包含所有的部件（属性、方法、事件）。所有的部件不能包含可视部分，并且自动会被声明为 PUBLIC。

```
INTERFACE <intf>.
```