Class Inheritance

Inheritance allows grouping of similar classes in a general-to-specific hierarchy.

The parent class is a general representation and contains those attributes and methods common to the 'family.'

The child class(es) is/are a more specific instance of the parent with added or redefined attributes and/or methods.

Often described as a "is a" relationship.

Parent class = superclass
Child class = subclass
Class Inheritance Example

- **document**
  - doc_number
  - date
  - doc_type
  + get_doc_number
  + set_doc_number
  ...
  + display_doc

- **po_document**
  - vendor_id
  - company_code
  - delivery_date
  + get_vendor_id
  + set_vendor_id
  ...

- **sales_document**
  - customer_id
  - company_code
  - fulfill_by_date
  + get_customer_id
  + set_customer_id
  ...

Class inheritance principles

Common attributes and methods are defined in the superclass. Items defined in the superclass are inherited in the subclass(es) and considered to be a part of their definition.

- Avoids redundant definition.
- Promotes "programming by difference"—subclass only defines those things that are different.

ABAP supports only singular inheritance, so a subclass can have only one parent superclass.

In comparing the superclass and the subclass, which is the more specific class?
**ABAP Inheritance Syntax**

In the child class, add **INHERITING FROM parent** to the first line of the class definition.

```
CLASS po_document DEFINITION INHERITING FROM document.
```

The inheriting class cannot delete anything from the parent, but it can redefine method implementations.

When redefining a method’s implementation, the **interface** (parameter passing) and the **visibility** (public, private) must remain unchanged.

In the child class' **definition** the method is noted as a redefinition.

Since the interface is already defined in the parent, it is not listed again (as it cannot be changed)

```
CLASS po_document DEFINITION.
  METHODS display_doc REDEFINITION.
```

---

**ABAP Inheritance Syntax**

To call a method of the parent (such as when you want to call the original method as part of a method's redefinition), use **super->**

Constructor methods have special rules.

A subclass may use the superclass constructor as it is, or the subclass may create a new constructor with a new interface.

The subclass constructor does not use REDEFINITION.

If the subclass constructor wishes call the superclass constructor it may do so as follows:

```
  super->constructor ( params ).
```

The calling of the superclass constructor must be the **first thing** done in the subclass constructor.

Given the above, the constructor is the only method in ABAP objects that is **overloaded**. (Two active methods sharing the same name.)
Let's Practice

Let's Practice
Inherited element visibility

A **private component** (attribute or method) of a superclass cannot be referenced by the subclass. These elements can only be accessed if made available by the superclass through public methods. This is done to ensure security of data.

Attributes and methods can be designated as **PROTECTED**.

PROTECTED resources are defined in the PROTECTED SECTION of a class definition.

Protected resources are visible within the class and its children.

Similar to saying something is public to a class and its descendants, but private to everyone else.

In the definition of an Object, the sections must appear in the order: PUBLIC SECTION, PROTECTED SECTION, and PRIVATE SECTION.

Static elements

A class shares static attributes with its children in the same fashion as other attributes.

A class shares static methods with its children, but those methods cannot be redefined.
**Narrowing casts**

A data object that is a reference to a superclass can be assigned a subclass object at runtime using a **narrowing cast**.

The subclass contains all the methods and attributes of the superclass so all expected elements are present.

This data object can only access the methods and attributes available in the subclass through inheritance.

When methods are invoked, if the method has been REDEFINED, the REDEFINED version is used.

The above is called **polymorphism**.

Value? Example: Create internal table based on the superclass and children of various types can be stored. Assumption: only inherited methods needed.

Example: method defined to take in reference to superclass data object. Subclass data object can be used.

---

**Widening casts**

A data object that is a reference to a subclass can be assigned a superclass object at runtime using a **widening cast**.

Typically used when a narrowing cast was previously used and you wish to restore the object back to its true identity.

After the widening cast, the methods of the subclass will once again be available.

Done using the syntax **MOVE superclass ?TO subclass**.

The object moved should have been created as a subclass object or method calls on the new object will result in an error.
Narrowing, Widening Summary

Narrowing Cast
When Document Reference is used to refer to this object, can only use methods and attributes defined for Documents. I lose some abilities. (Have narrower ability.)

Widening Cast
When P.O. Document Reference is used to refer to this object, I can use methods and attributes defined for P.O. Documents. (Have wider ability.) However, this is only valid if the object I’m assigned is actually a P.O Document formerly cast using a Narrowing Cast.

Let's Practice
Let's Practice

Class Builder

Just as TYPES creates local data types, so too does CLASS define local classes. We must use the ABAP Dictionary to define classes that are reusable across programs.

The Class Builder (SE24) is a special maintenance tool that allows the creation of global classes within the ABAP Dictionary.

Start the Class Builder and supply the name of the class to be created (respecting the namespace).

Specify "Usual ABAP Class". Remove "Final" as it means the Class will not be able to be extended to subclasses.

Use Attributes tab to specify Attributes.

Use Methods tab to specify Methods.

Button for constructor and parameters

Double click to define method behavior.