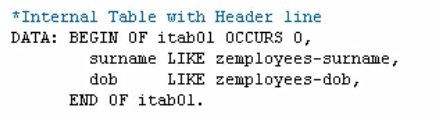
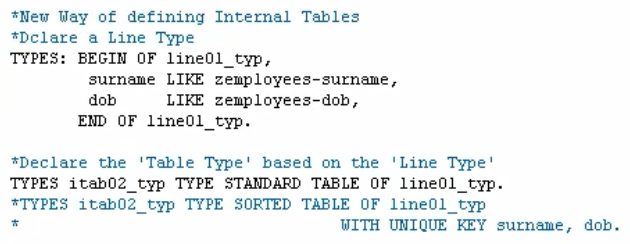
Now when we declare internal tables, we must define the start of the table and the end of the table.



we must have an occurs addition. Followed by a numeric. Now the OCCURS clause tells SAP we are declaring an internal table, and we are declaring an internal table with zero records. So, OCCURS means this table is going to be made up of several records, but we are going to define zero to begin with. And then let the table expand as we fill it with data.

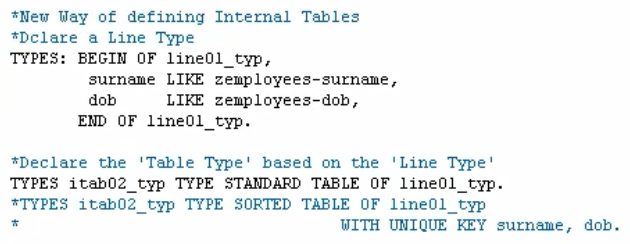


there are two types of declared end tables. The old style, where we use header records, and the new style where we use a separate work header. But just by the fact that we're using this OCCURS clause, we're telling the system we are going to create an internal table with a header record.

If you want to move five fields from table one to five fields from table two, and both tables have same field names, we can just write one line of code, using this move-corresponding statement. Which will automatically search for any field names that are identical, and make sure the data contained in the table one fields are moved to the corresponding fields for table two. it doesn't matter if these two different tables have a different structure. What I mean is the fields can be in a completely different order, the move-corresponding works it out. It matches the actual name of the field together and keep your field names the same.

And of course, there's no occurs edition, because we're not defining an internal table at this point. We are just defining a line of a table.

Now, once we have defined our line, the next step is to define our table type. we're going to create a type of table. So, we're just defining the structure of it.



Now before moving on from this statement, let's also consider how we can create a sorted table type instead of a standard table type. Well, we can do this by a, we can add SORTED TABLE OF line01\_typ.

And then, we specify our table key. So, because our table is made of both surname and date of birth, we can then say, WITH UNIQUE KEY, and then we add the key.

now we have defined the line type, and then we have defined the table type that is based on the line type. But all these are just type definitions, we've just created the blueprints to make up a table.



We use the data statement again, then now we declare our internal table name. So, because we've already defined our line type and our table type, all we need to do now is use the data statement to say we're going to create a new table called itab02 that is based on the table type we declared, itab02\_typ.



And when declaring internal tables this way, we have the option of whether to use a header line or not. If you do want to use a header line, you must specify this explicitly in the data statement.

But remember, I'm saying don't do this. Because if you do, you'll not be able to use this style of table in ABAP objects.



The last thing we need to do is define our work area. And we will use this work area in conjunction with the table we've just defined.

Now before we put the code in, just note that the work area is not part of the internal table. We have declared the complete internal table now in the previous step. But we declared our work area to work in conjunction with the table. Meaning we move data to our work area, we work with it, and then we update our table from the work area. Or we read data from our table into our work area to allow us to work with it. The work area is separate. And because it is separate, if we have multiple internal tables with the same structure, we can also use one work area to read and write data to and from multiple tables.