

REPORT ZYNY\_CLASS\_09.  
\* Interfaces  
  
INTERFACE intf\_speed.  
  METHODS: writespeed.  
ENDINTERFACE.                    "intf\_speed  
  
\*----------------------------------------------------------------------\*  
\*       CLASS train DEFINITION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS train DEFINITION.  
  PUBLIC SECTION.  
    INTERFACES intf\_speed.  
    ALIASES writespeed FOR intf\_speed~writespeed.  
    METHODS: gofaster.  
  PROTECTED SECTION.  
    DATA: speed TYPE i.  
ENDCLASS.                    "train DEFINITION  
  
\*----------------------------------------------------------------------\*  
\*       CLASS train IMPLEMENTATION  
\*----------------------------------------------------------------------\*  
\*  
\*----------------------------------------------------------------------\*  
CLASS train IMPLEMENTATION.  
  METHOD gofaster.  
    speed = speed + 5.  
  ENDMETHOD.                    "gofaster  
  METHOD intf\_speed~writespeed.            "Still have to use the full name here  
    WRITE: / 'The train Speed is: ', speed LEFT-JUSTIFIED.  
  ENDMETHOD.                    "intf\_speed~writespeed  
ENDCLASS.                    "train IMPLEMENTATION  
  
\* Ou prgram starts here  
  
DATA mytrain TYPE REF TO train.  
  
START-OF-SELECTION.  
  
  CREATE OBJECT mytrain.  
  mytrain->gofaster( ).  
  mytrain->intf\_speed~writespeed( ). " though this will give some warning  
  mytrain->writespeed( ).              " Now using the Alias name  
  "error, can't use this syntax before, now it is working  
  "both syntax can be used

This lesson is going to focus on how we use alias names within classes and interfaces.

Now, as you have seen, when we refer to components of an interface in a class, we make use of the component selector and you can see that on the screen here just at the end.

INFP components like and.

Now, Sappi have introduced a really nice way for us to refer to interface components without the need to use the components selector.

We can create what's called an alias name.

And the syntax for this is just this.

We use the word aliases, then specify the name and we use the addition for and then specify the actual

interface and component.

This gives us the option of assigning alias names when implementing interfaces in the declaration part

of our class or when an interface is composed during the interface definition itself.

Now, when we declare an A-list name in a class, we must make sure we place them inside the visibility

section of the class, which in turn determines how the alias name is seen outside the class.

Now, alias names are a pretty simple concept, but a great benefit of using them in a class definition

or when an interface is defined is that as long as calling programs use the alias names, the class

definition or interface declaration can change and not affect any programs that use the alias names.

It's all so to the calling program.

The interface of the class has stayed exactly the same, which means we're not going to break other

people's programs when we change our classes.

Let's have a look at some code and see how we can implement this.

So they have op ed.

He is the old program that we had.

And let's just change this to use alias names, so I will scroll to the top and all I need to do is

find our definition for our class, which is here.

And then under the interfaces.

This is where I'm going to create an alias, so I use the key word aliases.

And what I'm going to do is create an alias for the right speed method, and you can call your alias

whatever you want.

But just to show you that we can do this, I'm going to use exactly the same name as the method.

So I'm going to say right.

Speed oops.

Possibile right.

And then I specify for.

And that's where I put the interface that I'm creating the alias for.

So our interface is interface speed.

So I just keep that in and then we have to use the component selector and specify the method, which

is right speed.

And that's it is pretty simple, we created an alias now so that when we're writing code to instantiate

an object of this train class and we want to call the method right speed, we don't have to use the

interface component selector and then the method name to call it.

We can just refer to our class dash greater than right speed.

Now, we're not done just yet.

Let's go down to the implementation section, because here we define the method, right, speed using

our component selector.

Well.

This has to stay just the same.

I'm not going to change anything, but I wanted to highlight to you that we don't just by making the

entry here, give us the freedom to use the alias name still within our class implementation and definition.

No, we still have to use our component selected just as normal.

It's the calling programs that can make use of the alias names, not the rest of our class.

So let me out a comment here.

We still have to use the full name here, but where we can make use of Daley's name is now within our

calling program here, our original code uses the component selector.

But what I'll do, I'll copy that and comment this.

I want to show now how we have to do is just like before, just when we define the method in Santa Clause

in the old way, before I introduced interfaces we can call our right speed method just like this.

Because we've implemented the alias that gives us this lovely, nice shortcut again.

So think about it, that means our calling program or a program that uses a class that has aliases built

in, doesn't have to worry about trying to figure out the correct interface with a component selector,

a method name.

It just needs to know, hey, I've got a right speed method and that's all you need to call.

There's no need to use your component selectors.

And then for the person that created the class and interface, they then have the knowledge that if

they do change the class or interface in any way with regards to this right speed method, as long as

they keep that same alias name, they're not going to break any culling programs when they make code

modifications.

So let me check our code.

Everything is good, I'll give it Inactivates.

Fantastic.

Give it a run again, Jotham.

It's not really worth debugging because there isn't much to see.

It's just to show you we can make use of alias names instead of the component selectors.

Let's move on.