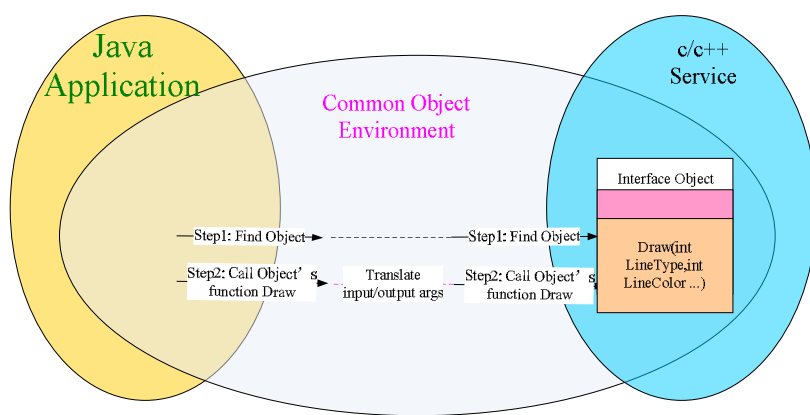




Multi-Language Programming : More Complicated Function Call Between Languages

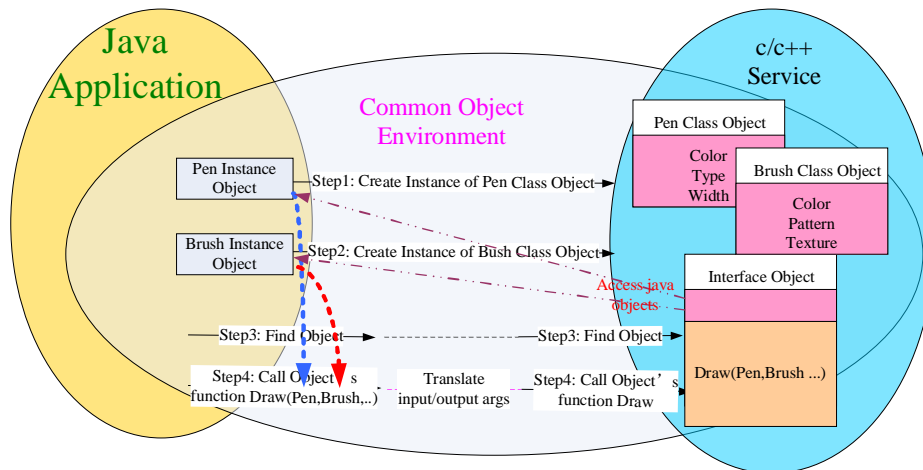
Components or libraries provide some functions. Other modules may access these functions through their interface. In some cases, the interface functions may be simple. The caller constructs input arguments, calls the function, and gets the return value. These type interfaces are easy to define and no more restrictions. But in other cases, interface functions may be more complicated. Let's use an example to explain in detail.

Supposing there is a c/c++ component. It provides a function to draw and fill a circle into memory pixel buffer. If the component uses a simple interface, it should take parameters as input arguments, such as line type, color, width, fill color, fill pattern, fill bitmap, etc. We define an interface object contains this function in common object environment. The function may be called by other languages, for example java.



In this case, there are some disadvantages. First, interface function may have many input arguments, and for each call, these arguments should be input although their values do not change. Second, if a new parameter is added, all related functions must be changed. This will increase the burden of software maintenance.

In general, we often define two other objects: pen and brush objects. The draw function takes pen and brush object as input arguments. The pen object has attributes color, type and width, and brush object has attributes color, pattern, image, etc. We also define an interface object contains the draw function in common object environment. As result, there are three objects defined by c/c++ language in environment, which are pen, brush, and object containing the draw function. For other languages using the function, for example java, it should create instances of the three interface class objects. And the instances can be accessed by c/c++ and java language base on the support of the environment.



In this case, the function call is more complicated. The caller needs to create instances of objects defined in c/c++ languages. The instances can be accessed by the two languages. They are used as input arguments of draw function. Without the support of common object environment, it is difficult to realize in this way, especially to support the call from applications programmed with multiple different languages.

It is clear that the c/c++ service can be call by any other languages. Is it easy and concise? You can compare it with traditional method using JNI interface.