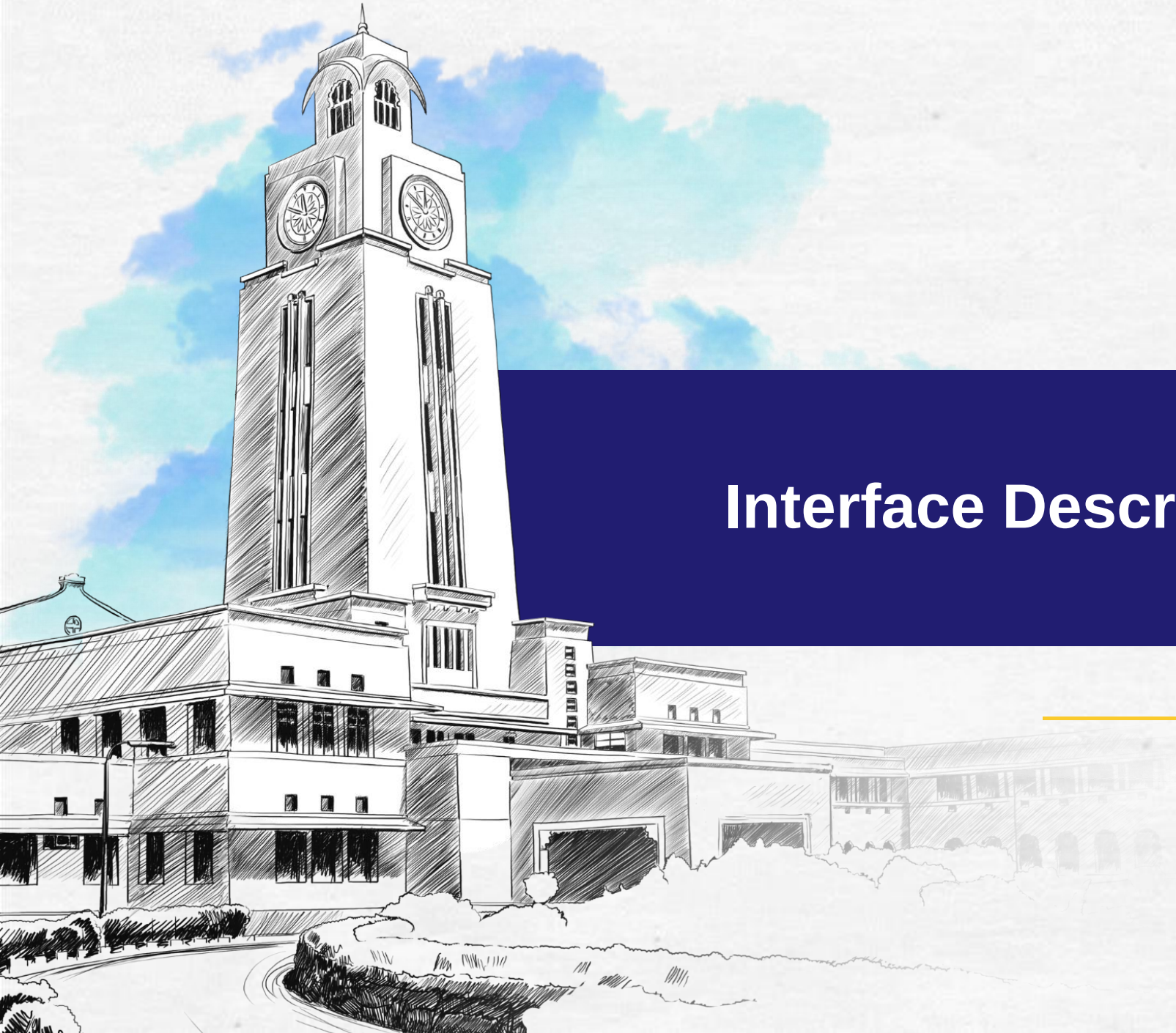


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Interface Description Language (IDL)

Srikanth Gunturu

Guest Faculty
BITS, WILP

In this segment

CORBA IDL

- IDL Overview
  - Parameters
  - Modules
  - Exceptions
  - Structs and Arrays
  - Sequences
  - Attributes
- IDL to Java bindings

Interface Description Language (IDL)

Overview

- Defines methods in a fashion similar to interfaces in Java
- Implementation is internal to the server (and transparent to client)
- Data types supported:
  - String
  - Integer
    - short – 16 bit (signed / unsigned)
    - long – 32 bit (signed / unsigned)
    - long long – 64 bit (signed / unsigned)
  - Octet – 8 bit (similar to byte)
- No keyword restrictions for naming (as IDL is language neutral), however there may be compiling issues depending on languages involved
- Enums – Order not guaranteed

```
interface echo {
  string echostring (in string the_echostring);
};
```

```
interface echo {
  string echoshort (inout short the_echoshort);
};
```

```
interface if {
  while switch (in for the_data);
}
```

```
enum Money {euro, dollar, pound, yen};
```

Interface Description Language (IDL)

Parameters, Modules and Exceptions

- Parameters
  - 'in' is input – from client to server
  - 'out' is output – from server to client
  - 'inout' – same parameter sent from client is used for output (similar to pass by reference)
- Modules – Groups multiple interfaces together, with hierarchical naming (like Java packages)
- Exceptions
  - System Defined
    - COMM_FAILURE
    - MARSHAL
    - BAD_PARAM
    - OBJECT_NOT_EXIST
    - TRASNIENT
    - UNKNOWN
  - User defined

pass by value
Pass by ref

```
interface echo {
  string echostring (inout string the_echostring);
};
```

out

```
module echomodule {
  interface echo {
    string echostring (in string the_echostring);
  };
};
```

echomodule.echo

```
interface echo {
  exception Bad_Message{};
  string echostring (in string the_echostring) raises (Bad_Message);
};
```

Interface Description Language (IDL)

Structs, Arrays and Sequences as Params

- Structs as params
  - Can be used as in, out and inout

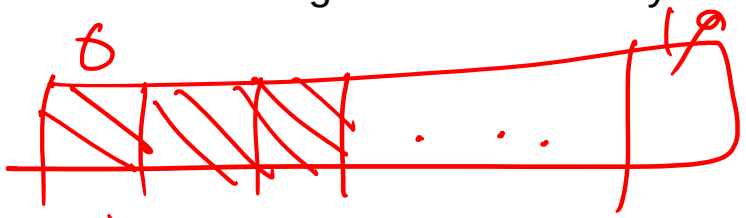
```
struct theData {
  long firstvalue;
  string secondvalue;
};

typedef theData StructType;
```

```
interface sending_stuff {
  string sendvalue (in StructType mystruct);
};
```

```
interface sending_stuff {
  StructType  sendvalue (in string mystring);
};
```

- Arrays as params
  - No guarantee on array indexing




```
typedef short ArrayType[20];
const short MAX=20;

interface sending_stuff {
  string sendvalue (in ArrayType myarray, in short size);
};
```

- Sequence as params
  - Sends only the data present in the sequence at the time of method call

```
typedef sequence<long> SequenceType;
interface sending_stuff {
  string sendvalue (in SequenceType mysequence);
};
```

Interface Description Language (IDL)

Attributes

- Attribute gets/sets a particular variable on the servant
- Example:
  - A remote procedure named the_value with no parameters passed, that returns a Short value
  - A remote procedure named the_value with a Short value passed as a parameter, that does not return any values (has a void value as the return value)
- Not recommended to use as behavior is too dependent on network and language implementations.

```
interface sending_stuff {
   attribute short the_value;
};
```

Interface Description Language (IDL)

CORBA IDL to Java bindings

- Module in IDL maps to package in Java
- CORBA IDL types map to most Java types
  - boolean to Boolean
  - octet to Byte
  - short to short
  - long to int
  - long long to long
  - string to string
  - sequences and arrays to arrays
  - float to float
  - double to double
  - enums, structs, unions and exceptions to Java classes
- For user defined types in IDL, Java generates:
  - Helper class – manages read/write to CORBA I/O streams
  - Holder class – Implements out/inout parameters as a wrapper class, to hold their value (by reference) and pass it to Helper class streams

Interface Description Language (IDL)

Example code – in

- IDL definition

```
typedef short ArrayType[20];
const short MAX=20;

interface sending_stuff {
  string sendvalue (in ArrayType myarray, in short size);
};
```

- Java code (servant)

```
public class myreceiver extends sending_stuffPOA
{
    public String sendvalue (short[] myarray, short size)
    {
      int the_real_size;
      the_real_size=size;
      if (the_real_size > MAX.value)
         the_real_size=MAX.value;
      for(int i=0;i<the_real_size;i++)
      {
          System.out.println("myarray["+i+"] is "+myarray[i]);
      }
      String mymsg="got here";
      return mymsg;
    }
}
```

```
public interface MAX
{
  public static final short value = (short)(20);
}
```

No memory management involved!

Interface Description Language (IDL)

Example code – inout

- IDL definition

```
typedef long ArrayType[20];
const long MAX=20;

interface sending_stuff {
        string sendvalue (inout ArrayType myarray, in long size);
};
```

- Java code (servant)

```
public class myreceiver extends sending_stuffPOA
{
    public String sendvalue (ArrayTypeHolder myarray, int size)
    {
      int the_real_size;
      the_real_size=size;
      if (the_real_size > MAX.value)
        the_real_size=MAX.value;
      for(int i=0;i<the_real_size;i++)
      {
        System.out.println("Original value of myarray["+i+"] is "+myarray.value[i]);
        myarray.value[i]=myarray.value[i]*10; // Multiply each myarray value times 10
        System.out.println("Value to pass back to client of myarray["+i+"] is "+myarray.value[i]);
      }
      String mymsg="Hello there from servant to client";
      return mymsg;
    }
}
```

myarray[i]

myarray.Value[i]

Thank You!

In our next session:

CORBA Addressing