Appendix: Java Implementations using Orbacus

Compiling the IDL

jidl --package hello Hello.idl

For every construct in the IDL file that maps to a Java class or interface, a separate class file is generated. Directories are automatically created for those IDL constructs that map to a Java package (e.g., a module).

This command generates several Java source files on which the actual implementation will be based:

- Hello.java
- HelloHelper.java
- HelloHolder.java
- HelloOperations.java
- HelloPOA.java
- _HelloStub.java

Servant Class

The servant class Hello_impl must inherit from the generated class HelloPOA .

// Hello_impl.java

```
package hello;

public class Hello_impl extends HelloPOA {
        public void say_hello()  {
                System.out.println("Hello World!");
        }
}
```

Server Class

The Server class holds the server's main() and run() methods:

// Server.java

```
package hello;

public class Server {
        public static void main(String args[]) {
                //Properties  necessary to use the Orbacus ORB instead of the JDK’s ORB.
                java.util.Properties props = System.getProperties();
                props.put("org.omg.CORBA.ORBClass",  "com.ooc.OBServer.ORB");
                props.put("org.omg.CORBA.ORBSingletonClass","com.ooc.CORBA.ORBSingleton");
```

//Server.java- main() (ctd.)

```
int status = 0; //exit status
org.omg.CORBA.ORB orb = null;
try {
        //initialize the ORB
        orb = org.omg.CORBA.ORB.init(args, props);
        status = run(orb);
}
catch(Exception ex) { ex.printStackTrace(); 25 status = 1; }



if(orb != null) { //If the ORB was successfully created, destroy it to free resources.
        try { orb.destroy();  }
        catch(Exception ex) {
                ex.printStackTrace();

//The exit status is returned. If there was no error, 0 is returned, or 1otherwise.
                status = 1;
        }
}
}
```

//Server.java (ctd.)

```
    static int run(org.omg.CORBA.ORB orb) throws org.omg.CORBA.UserException {
     //Get a reference to the Root POA, and use it to obtain a reference to its POA Manager.
       org.omg.PortableServer.POA rootPOA =
            org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
       org.omg.PortableServer.POAManager manager =  rootPOA.the_POAManager();


  //Create a servant of type Hello_impl, which is used to incarnate a CORBA object.
       Hello_impl helloImpl = new Hello_impl();
       Hello hello = helloImpl._this(orb);

        try { //The object reference is “stringified” and written to a file.
                String ref = orb.object_to_string(hello);
                String refFile = "Hello.ref";
                java.io.PrintWriter out = new java.io.PrintWriter(new java.io.FileOutputStream(refFile));
                out.println(ref);
                out.close();
            } catch(java.io.IOException ex) {
                  ex.printStackTrace();
                  return 1;
            }

//The server enters its event loop to receive incoming requests.
       manager.activate();
       orb.run();
       return 0;
      }
} //end Sever.java
```

Client Implementation

```
//Client.java
package hello;

public class Client {
        public static void main(String args[]) {
                ... // Same as for the server
        }

        static int run(org.omg.CORBA.ORB orb) {
                org.omg.CORBA.Object obj = null;

                try { //The stringified object reference is read and converted to an object.
                        String refFile = "Hello.ref";
                        java.io.BufferedReader in =
                                new java.io.BufferedReader( new java.io.FileReader(refFile));
                        String ref = in.readLine();
                        obj = orb.string_to_object(ref);
                } catch(java.io.IOException ex) {
                                                ex.printStackTrace();
                                                return 1;
                                                }
                Hello hello = HelloHelper.narrow(obj); //“Narrowed” to a reference to a Hello object.
                hello.say_hello(); //Invoke the say_hello method
                return 0;
        }
} //end Client.java
```

Compiling the Application

-Ensure that your CLASSPATH environment variable includes the current working directory as well as the Orbacus for Java classes (i.e the OB.jar file):

- **UNIX**: CLASSPATH=.:your_orbacus_directory/lib/OB.jar:$CLASSPATH
  export CLASSPATH
- **Windows:** set CLASSPATH=.;your_orbacus_directory\lib\OBE.jar;%CLASSPATH%

-Use ***javac*** to compile the implementation classes and the classes generated by the Orbacus IDL-to-Java translator.

javac hello/*.java

Running the Application

1. Start the 'Hello World' Java server: ***java hello.Server***
2. Start the 'Hello World' Java client: ***java hello.Client***

Note: make sure that your CLASSPATH environment variable includes the OBE.jar file.
You may use a C++ server together with a Java client (or vice versa).