E.4 Mapping IDL to Java

- Mapping of IDL data types to Java
- Mapping of object interfaces to Java

1 General Remarks

- Mapping of IDL types to Java interfaces and classes
- Latest language mapping document from June 1999
- Contains mapping for POA interfaces

Important goals:

- Portable stubs
  - Stubs may be loaded over the network
  - Stubs must work with any ORB that is locally installed – independent of the ORB core
  - Interface between stubs and ORB core fixed to guarantee exchangeability
- Reverse mapping of Java to IDL should be possible

- Identifiers from IDL source are used unchanged in Java
  - In case of clashes prepend _ (underscore)
2 Primitive Types

- **Integer numbers**
  - `short` becomes `short`
  - `unsigned short` becomes `short`
  - `long` becomes `int`
  - `unsigned long` becomes `int`
  - `long long` becomes `long`
  - `unsigned long long` becomes `long`
  - Large `unsigned` values become negative in Java!

- **Floating point numbers**
  - `float` becomes `float`
  - `double` becomes `double`
  - `long double` no mapping yet

2 Primitive Types (2)

- **Characters**
  - `char` becomes `char`
  - `wchar` becomes `char`
  - Because Java `char` is superset of IDL `char` marshalling may cause `CORBA::DATA_CONVERSION` exception

- **boolean**
  - Stays `boolean`

- **octet**
  - Becomes `byte`

- **any**
  - Class `org.omg.CORBA.Any`
  - `insert` and `extract` methods for primitive types in class `Any`
  - For other types `insert` and `extract` methods in Helper class of this type
3 Helper Classes

- One Helper class for each IDL type (here *name*)

```java
public class nameHelper {
    public static void insert( org.omg.CORBA.Any a, Name t ) {
        ...
    }
    public static name extract( org.omg.CORBA.Any a ) {
        ...
    }
    public static org.omg.CORBA.TypeCode type() {
        ...
    }
    public static String id() {
        ...
    }
    public static name read( org.omg.CORBA.portable.InputStream istream ) {
        ...
    }
    public static void write( org.omg.CORBA.portable.OutputStream ostream, name value ) {
        ...
    }
    // only for interface helpers
    public static name narrow( org.omg.CORBA.Object obj ) {
        ...
    }
}
```

3 Helper Classes (2)

- **extract and insert**
  - Methods to extract and insert this type from/into an *any* object

- **type and id**
  - Inquiry of type code and type identification (Repository ID) for this type

- **read and write**
  - Methods for marshalling and de-marshalling in portable stubs

- **narrow**
  - Only exists in Helpers for *interfaces*
  - Modification of the visible interface of an object reference - Casting
4 Holder Classes

- Java only has call-by-value semantics (Object references cannot be manipulated!)

- `out`- and `inout` parameters need call-by-reference

- Encapsulation of parameters in a Holder object (here for type `name`)

```java
final public class nameHolder
  implements org.omg.CORBA.portable.Streamable {

  public name value;

  public nameHolder() {...}
  public nameHolder( name initial ) {...}

  public void _read( org.omg.CORBA.portable.InputStream i )
          {...}
  public void _write(
          org.omg.CORBA.portable.OutputStream o ) {...}

  public org.omg.CORBA.TypeCode _type() {...}
}
```

5 IDL Entity

- Empty marker interface

- Inherited by all IDL generated interfaces and classes

- Declaration:

```java
package org.omg.CORBA.portable;

public interface IDLEntity
  extends java.io.Serializable
{
}
```
6 Modules

- IDL:
  
  ```
  module name {
    Declarations
  };
  ```

- Mapping to Java packages

- Java:
  ```
  package name;
  ```

- Mapping of module CORBA to package `org.omg.CORBA`

7 Type Declarations

- IDL:
  ```
  typedef existing_type alias;
  ```

- Java:
  - Only Holder and Helper class for `alias`
  - For the type itself mapping of `existing_type` has to be used

- Example:
  ```
  // IDL
  typedef long IDNumber;

  // Java
  public class IDNumberHelper {...}
  final public class IDNumberHolder
    implements org.omg.CORBA.portable.Streamable {
    public int value;
    ...
  }
  ```
8 Structures

- IDL:

```java
struct name {
    Declarations of structure elements
};
```

- Mapping to a `public final` class with Helper and Holder class
  - Elements become `public` variables
  - Empty constructor and constructor that takes values for all variables

- Java:

```java
public final class name
    implements org.omg.CORBA.portable.IDLEntity
{
    Mapping for structure elements as public variables
    public name() {}
    public name( Mapping_for_structure_elements ) {...}
}
```

8 Structures (2)

- Example:

```java
// IDL
struct Example {
    float value;
    char currency;
};

// Java
final public class Example
    implements org.omg.CORBA.portable.IDLEntity
{
    public float value;
    public char currency;
    public Example() {}
    public Example( float value, char currency ) {
        this.value = value;
        this.currency = currency;
    }
}
```
8 Nested Structures

- **Example:**

```plaintext
class Outer {  
    class Inner {  
        char foo;  
    }  
    fooBar;  
};
```

- **Inner** is mapped to class **Inner** with Helper and Holder class in a sub-package named **OuterPackage**

```plaintext
Outer.java:  
final class Outer {  
    class Inner {  
        char foo;  
    }  
    fooBar;  
}  

OuterPackage/Inner.java:  
package OuterPackage;  
final class Inner {  
    char foo;  
}
```

9 Unions

- **IDL:**

```plaintext
union name switch( switch_type ) {  
    case switch_constant: Declaration  
    ...  
    default: Declaration  
};
```

- **Mapping to a public final class with Helper and Holder class**
  - Access method for switch type (discriminator) and all types of the Union
  - Access method for default discriminator

- **Java:**

```plaintext
public final class name  
    implements org.omg.CORBA.portable.IDLEntity  
{
    public name() {}  
    public Mapping_for_switch_type discriminator() {...}  
    public Mapping_for_union_element union_element_name() {...} //get  
    public union_element_name( Mapping_for_union_element ) {...} //set  
}
9 Unions

Example:

```
// IDL
union Example switch( long ) {
    case 1: long l;
    case 2: float f;
};

// Java
final public class Example
    implements org.omg.CORBA.portable.IDLEntity
{
    private java.lang.Object _object;
    private int _disc;
    private int _defdisc = -2147483648;
    public Example() {}
    public int discriminator() { return _disc; }
    public int l() {...} // get l
    public void l( int value ) {...} // set l
    public float f() {...} // get f
    public void f( float value ) {...} // set f
    public void _default() {...} // set to impossible discr.
}
```

10 Enumerations

IDL:

```
enum name {
    value1, value2, ...
};
```

Mapping to a `public final` class with Helper and Holder class

- Enumeration values are mapped to Integer values (Identifier `_value1, ...`)
- And to static instances within the enumeration class

Java:

```
public final class name
    implements org.omg.CORBA.portable.IDLEntity
{
    public static final int _value1 = int_value1;
    public static final name value1 = new name( _value1 );
    ...
    private final int _value;
    private name( int value ) { this._value = value; }
    public int value() { return _value; }
    public static name from_int( int value ) {...};
```
10 Enumerations

- Example:

```java
// IDL
eenum Color { GREEN, RED, BLUE };

// Java
final public class Color
    implements org.omg.CORBA.portable.IDLEntity {
    final public static int _GREEN = 0;
    final public static int _RED = 1;
    final public static int _BLUE = 2;
    final public static Color GREEN = new Color( _GREEN );
    final public static Color RED = new Color( _RED );
    final public static Color BLUE = new Color( _BLUE );

    private int _value;
    private Color( int value ) { this._value = value; }
    public int value() { return _value; }
    public static Color from_int( int value ) {
        switch( value ) {
            ...
        }
    }
}
```

11 Arrays

- IDL:

```
typedef element_type name[positive_constant] [positive_constant]...;
```

- Mapping to Java Arrays und `nameHelper` and `nameHolder` class
  - Array elements have the type that arises from mapping `element_type`

- Example:

```java
// IDL
typedef long Matrix[3][3];

// Java
public class MatrixHelper {...}
final public class MatrixHolder
    implements org.omg.CORBA.portable.Streamable {
    public int[][] value;
    ...
}
12 Sequences

IDL:

```idl
typedef sequence<element_type> name; // unbounded
typedef sequence<element_type, positive_constant> name; // bounded
```

■ Mapping the same as for single dimension arrays
  ◆ Length check for bounded sequences will only be done while marshalling

Example:

```idl
// IDL
typedef sequence<long> Longs;

// Java
public class LongsHelper {...}
final public class LongsHolder
    implements org.omg.CORBA.portable.Streamable {
    public int[] value;
    ...
}
```

13 Strings

IDL:

```idl
typedef string name; // unbounded
typedef string<positive_constant> name; // bounded
typedef wstring name; // unbounded
typedef wstring<positive_constant> name; // bounded
```

■ Mapping to `java.lang.String`
  ◆ Exceptions during marshalling when length is exceeded or characters cannot be mapped to CORBA `char`

Example:

```idl
// IDL
typedef string<80> Name;

// Java
public class NameHelper {...}
final public class NameHolder
    implements org.omg.CORBA.portable.Streamable {
    public java.lang.String value;
    ...
}
```
14 Fixed-Point Numbers

- **IDL:**
  ```
  typedef fixed<positive_constant, scaling_constant> name;
  ```
- **Mapping to** `java.math.BigDecimal`
- **Helper class:** `nameHelper`
- **Holder class:** `org.omg.CORBA.FixedHolder`

15 Constants

- **Symbolic name for special values**
- **IDL:**
  ```
  const type name = constant_expression;
  ```
- **Mapping of local constants in IDL interfaces**
  - **final public static** variables in Java interface
  - **Example:**

    ```
    // IDL
    interface Example {
      const Color WARNING = RED;
    };

    // Java
    public interface Example ... {
      final public static Color WARNING = (Color) Color.RED;
      ...
    }
    ```
15 Constants (2)

- Mapping of constants outside an IDL interface
  - Class of its own with name of constant and local value `value`
  - Example:
    ```
    // IDL
    module Example {
        const Color WARNING = RED;
    }

    // Java
    package Example;
    public interface WARNING {
        final public static Color value = (Color) Color.RED;
    }
    ```

16 Interfaces

- IDL:
  ```
  interface name {
      Declaration of attributes and operations (as well as types and exceptions)
  }
  ```

- Mapping to:
  - `public` Java interface `nameOperations`
  - `public` Java interface `name`
  - `nameHelper` and `nameHolder` class
  - Stub and Skeleton class
16 Interfaces (2)

Java:

```java
public interface nameOperations {
    Mapping for attributes and operations
}
public interface name extends org.omg.CORBA.Object,
    nameOperations, org.omg.CORBA.portable.IDLEntity {}

final public class nameHolder
    implements org.omg.CORBA.portable.Streamable {...
public class nameHelper {...}
```

16 Interfaces – Inheritance

IDL:

```idl
interface name : inherited_interface1, inherited_interface2, ...
    Declaration of additional attributes and operations
};
```

Mapping to multiple inheritance of Java interfaces

Java:

```java
public interface nameOperations
    extends inherited_interface1Operations,
    inherited_interface2Operations, ...
{
    Mapping for additional attributes and operations
}
public interface name
    extends inherited_interface1, inherited_interface2, ...
    nameOperations, org.omg.CORBA.portable.IDLEntity {}
final public class nameHolder
    implements org.omg.CORBA.portable.Streamable {...
public class nameHelper {...}
```
16 Interfaces – Attributes

**IDL:**

```idl
attribute type name; // read & write
readonly attribute type name; // read-only
```

**Mapping to a pair of access methods**

**Java:**

```java
public Mapping_for_type name(); // get attribute
public void name( Mapping_for_type ); // set attribute (not if read-only)
```

**Example:**

```idl
interface Account {
    readonly attribute float balance;
}
```

```java
public interface AccountOperations { 
    public float balance(); 
}
```

16 Interfaces – Operations

**IDL:**

```idl
return_type name( parameter_list ) raises( exception_list );
```

**Mapping to methods in the Java interface**

**Java:**

```java
public Mapping_for_return_type name( Mapping_for_parameter_list )
    throws Mapping_for_exception_list;
```
## 16 Interfaces – Parameter Transmission

- **IDL:**
  
  ```idl
  ( copy_direction1 type1 name1, copy_direction2 type2 name2, ... )
  ```

- **Mapping of parameter types depends on copy direction**
  
  - **in** to `Mapping_for_type`
  - **out** and **inout** to `typeHolder`

- **Example:**
  
  ```idl
  interface Account {
    void makeWithdrawal( in float sum, out float newBalance );
  }
  ```

  ```java
  public interface AccountOperations {
    public void makeWithdrawal( float sum, FloatHolder newBalance );
  }
  ```

---

## 16 Interfaces

- **Example:**
  
  ```idl
  module Bank {
    interface Account {
      void withdraw(in double amount);
      void deposit(in double amount);
      void transfer(inout Account src, in double amount);
      readonly attribute double balance;
    }
  }
  ```

  ```java
  package Bank;
  public interface AccountOperations {
    public void withdraw(double amount);
    public void deposit(double amount);
    public void transfer(AccountHolder src, double amount);
    public double balance();
  }
  ```

  ```java
  public interface Account extends org.omg.CORBA.Object,
  AccountOperations, org.omg.CORBA.portable.IDLEntity {
  }```
17 Exceptions

- IDL:

```idl
exception name {
  Declarations of data elements
};
```

- Mapping to `final public` class in the following class hierarchy:

```
java.lang.Exception

org.omg.CORBA.UserException  java.lang.RuntimeException

name

org.omg.CORBA.SystemException

CORBA System Exceptions
```

17 Exceptions (2)

- Mapping of local exceptions from IDL interface `name` to `final public` class in package `namePackage`

- Example:

```java
// IDL
interface Account {
  exception Overdraft { float howMuch };
  void withdraw( in double amount )raises( Overdraft );
};

// Java
public interface AccountOperations {
  public void withdraw( double amount )
      throws AccountPackage.Overdraft;
}

public interface Account extends ... {}
package AccountPackage;
final public class Overdraft extends org.omg.CORBA.UserException {
  public float howMuch;
  ...
}
```
### Exceptions (3)

- In addition: Creation of Helper and Holder class for exception
- Mapping of CORBA System Exceptions to `final public` Subclasses of `org.omg.CORBA.SystemException`

```java
package org.omg.CORBA;

abstract public class SystemException extends java.lang.RuntimeException {
    public int minor;
    public CompletionStatus completed;
    protected SystemException(String reason, int minor,
                                 CompletionStatus status) {
        super(reason); this.minor = minor;
        this.status = status;
    }
}

// CORBA::UNKNOWN
final public class UNKNOWN extends org.omg.CORBA.SystemException {
    public UNKNOWN() ...  
    public UNKNOWN(String reason) ...
    ...
}
```

### Stubs

- Client only has a Java reference to a local proxy object
  - Stub object
  - Stub class is automatically generated from the IDL description
  - Stub objects are transparent to the user – automatically created and destroyed by the CORBA system

Class hierarchy for IDL interface `module::name`
19 Java Mapping Summary

- For each IDL there are two classes
  - A Holder class for out and inout parameter passing
  - A Helper class for marshalling and for insertion and extraction into/from any objects

- Primitive types mapped to Java types (Caution: no one-to-one relation)

- IDL arrays and sequences are Java arrays

- For other constructed types there are special classes

- IDL interfaces mapped to Java interfaces

- IDL exceptions are special Java exception classes

- Clients have Java references to stub objects

E.5 Java Client

- Missing pieces for a complete client

- How to get the first reference to a CORBA object

- Example "Hello World!"
1 CORBA Pseudo Objects

- No real CORBA objects
  - Make sense only locally
  - Not remotely accessible

- Description of interface in Pseudo IDL (PIDL)
  - Syntax like IDL
  - Language mapping can define special mapping for each pseudo interface

- Examples:
  - `CORBA::Object` Features of CORBA object references
  - `CORBA::ORB` Interface to ORB features
  - `PortableServer::POA` Interface to the Portable Object Adaptor

- Recently more and more Pseudo Interfaces have been defined as local interfaces

2 Object References – CORBA::Object

- Operations that a client can invoke on each CORBA object:

```plaintext
module CORBA {
  interface Object {
    // PIDL
    InterfaceDef get_interface();
    boolean is_nil();
    Object duplicate();
    void release();
    boolean is_a( in string logical_type_id );
    boolean non_existent();
    boolean is_equivalent( in Object other_object );
    unsigned long hash( in unsigned long maximum );
    Status create_request( in Context ctx,
                           in Identifier operation,
                           in NVList arg_list,
                           inout NamedValue result,
                           out Request request,
                           in Flags req_flags );

    ...
  }
};
```
2 Object References – CORBA::Object

- **InterfaceDef get_interface()**
  - Returns an interface description (from Interface Repository) for this object
  - Usually used in connection with the Dynamic Invocation Interface (DII)

- **boolean is_a( in string logical_type_id )**
  - Checks whether the object implements the given interface
  - Interface Repository ID as a string, e.g. IDL:Bank/Account:1.0

- **Object duplicate()**
  - void release()
  - Copying and deleting of object references
  - Reference counting only locally in the client
  - Object implementation will not be informed

- **boolean is_nil()**
  - Checks whether this is a valid object reference

- **boolean non_existent()**
  - Checks whether there is an implementation for this object

- **unsigned long hash( in unsigned long maximum )**
  - Hash to distinguish object references

- **boolean is_equivalent( in Object other_object )**
  - Checks whether two references point to the same CORBA object
  - Caution: only best-effort semantics
    - true: references point to the same object
    - false: references probably point to different objects

- **Status create_request( ... )** Create a DII request
Java mapping to interface `org.omg.CORBA.Object`

```java
package org.omg.CORBA;

public interface Object {
    boolean _is_a( String Identifier );
    boolean _is_equivalent( Object that );
    boolean _non_existent();
    int _hash( int maximum );
    org.omg.CORBA.Object _duplicate();
    void _release();
    ImplementationDef _get_implementation();
    InterfaceDef _get_interface();
    ...}
```

duplicate and release really not necessary
◆ Java uses built-in Garbage Collection instead of reference counting

Caution: Simple `Object` means in every package `java.lang.Object`!

Common operations of the ORB

```java
module CORBA {  
    interface ORB {  // PIDL
        string object_to_string( in Object obj );
        Object string_to_object( in string str );

        typedef string ObjectId;
        typedef sequence<ObjectId> ObjectIdList;
        exception InvalidName {};
        ObjectIdList list_initial_services();
        Object resolve_initial_references(
            in ObjectId identifier ) raises (InvalidName);
        ...
    }
};
```
4 ORB Interface – CORBA::ORB

- `string object_to_string( in Object obj )`
- `Object string_to_object( in string str )`
  - Conversion of object references into unique strings and vice versa
  - String format: `IOR:00202020...`

- `ObjectIdList list_initial_services()`
  - List of services the ORB knows about, e.g. NameService

- `Object resolve_initial_references( in ObjectId identifier ) raises (InvalidName)`
  - Returns object reference for the requested ORB service
  - `ObjectId` is a String, e.g. "NameService"

5 ORB Interface – org.omg.CORBA.ORB

- Java mapping to abstract class `org.omg.CORBA.ORB`

  ```java
  package org.omg.CORBA;

  public abstract class ORB {
    public abstract String[] list_initial_services();
    public abstract org.omg.CORBA.Object resolve_initial_references( String object_name )
      throws org.omg.CORBA.ORBPackage.InvalidName;

    public abstract String object_to_string( org.omg.CORBA.Object obj );
    public abstract org.omg.CORBA.Object string_to_object( String str );

    ...
  }
  ```
6 ORB Initialisation

- First step in every CORBA application
- Returns a reference to a `CORBA::ORB` object
- PIDL spec:

```plaintext
module CORBA { // PIDL
    typedef string ORBid;
    typedef sequence<string> arg_list;
    ORB ORB_init( inout arg_list argv,
                   in ORBid orb_identifier);
};
```

- Selection of various ORBs (if there is more that one) via ORBid
- ORB parameters in command line arguments
  - e.g. `-ORB<suffix> <value>`

7 ORB Initialisation – `org.omg.CORBA.ORB`

- In Java ORB initialisation via static methods of `org.omg.CORBA.ORB`

```java
public abstract class ORB {
    ...
    public static ORB init(Strings[] args,
                            Properties props);
    public static ORB init(Applet app, Properties props);
    public static ORB init();
    ...
}
```

- Special `init` method for ORB inside an Applet
- `init()` without parameters only returns a Singleton ORB
  - Can only create special structures like Typecodes
  - Not suitable for remote method invocations!

- Java properties to select ORB features, e.g.
  - `org.omg.CORBA.ORBClass` Class that is returned by `init`
    (implements `org.omg.CORBA.ORB`),
    e.g. `com.ooc.CORBA.ORB`
8 Hello World Client

IDL-Interface

```
// Hello.idl

module Example {
    interface Hello {
        string say( in string msg );
    };
};
```

```
// client/HelloClient.java
package client;

import generated.Example.*;
import org.omg.CORBA.*;

public class HelloClient {
    public static void main( String[] args ) {
        try {
            // Initialise ORB
            ORB orb = ORB.init( args, null );
            // Read object reference from file Hello.ior
            String s = ...
            // Create a stub object
            org.omg.CORBA.Object o = orb.string_to_object(s);
            // Narrow to the Hello interface
            Hello hello = HelloHelper.narrow( o );
            // Do the call
            System.out.println( hello.say( " world!" ) );
        } catch(Throwable t) {
            t.printStackTrace();
        }
    }
}
```
8 Hello World Client

- Go to the example directory
  
  ```shell
  > cd /proj/4oods/pub/hello_java_client
  ```

- Compile
  
  ```shell
  > /local/ORBacus-4.0.3/bin/jidl --package generated Hello.idl
  > /local/java-1.3/bin/javac -classpath /local/ORBacus-4.0.3/lib/OB.jar:. client/HelloClient.java
  ```

- Run
  
  ```shell
  ```
8 Hello World Client – Generated Classes

interface HelloOperations
  +say(msg: String): String

org.omg.CORBA.Object
org.omg.CORBA.portable.IDLEntity

interface Hello

org.omg.CORBA.portable.Streamable

HelloHolder

- value: Hello
- HelloHolder()
- HelloHolder(initial: Hello)
- _read(in: org.omg.CORBA.portable.InputStream)
- _write(out: org.omg.CORBA.portable.OutputStream)
- _type(): org.omg.CORBA.TypeCode

HelloHelper

- typeCode_: org.omg.CORBA.TypeCode
- insert(any: org.omg.CORBA.Any, val: Hello)
- extract(any: org.omg.CORBA.Any, ret: Hello)
- _type(): org.omg.CORBA.TypeCode
- _id(): String
- _read(in: org.omg.CORBA.portable.InputStream)
- write(out: org.omg.CORBA.portable.OutputStream)
- narrow(val: org.omg.CORBA.Object): Hello

HelloStub

- _ob_ids_ String[] = {"IDL:Example:HelloOps"}
- _ob_opsClass: java.lang.Class = HelloOps
- _ob_a0: String[]
- _ob_a0: String
- _ob_a0: String

- ob_ids_ String[] = {"IDL:Example:Hello"}
+ ob_opsClass: java.lang.Class = HelloOps
+ _ids(): String[]
+ say(_ob_a0: String): String