Preface

Whenever I refer to the Java language mapping description I will refer to version 1.2 formal/02-08-05 reviewed in August 2002 as provided by Mr. von Loewis. I will call it *the specification*.

Problem 1

Use the IDL-to-Java compiler to generate stubs/proxies according to the given IDL file time.idl. Determine which of all these files are needed for the server-side implementation and which for the client-side implementation.

The given IDL file describes the interface of an imaginary time service:

```
//Vereinfachte Version des TimeService,
//aus formal/98-10-47.idl und formal/98-10-45.idl

#pragma prefix "hpi.uni-potsdam.de"

module TimeBase {

    typedef unsigned long long  TimeT;

    typedef TimeT  InaccuracyT;
    struct UtcT {
        TimeT           time;      // 8 octets
        unsigned long   inacclo;   // 4 octets
        unsigned short  inacchi;   // 2 octets
    };

    struct IntervalT {
        TimeT  lower_bound;
        TimeT  upper_bound;
    };
};

module CosTime {

    enum TimeComparison {
        TCEqualTo,
        TCLessThan,
        TCGreaterThan,
        TCIndeterminate
    };

    enum ComparisonType{
        IntervalC,
        MidC
    };

    enum OverlapType {
        OTContainer,
        OTContained,
        OTOverlap,
        OTNoOverlap
    };

    exception TimeUnavailable {};

    interface UTO {
        readonly attribute TimeBase::TimeT        time;
        readonly attribute TimeBase::InaccuracyT  inaccuracy;
        readonly attribute TimeBase::UtcT         utc_time;
        UTO absolute_time();
        TimeComparison compare_time(
            in  ComparisonType  comparison_type,
            in  CosTime::UTO    uto
        );
    };

    interface TimeService {
        UTO universal_time()
            raises(TimeUnavailable);
```

```
        UTO secure_universal_time()
            raises(TimeUnavailable);
        UTO new_universal_time(
            in TimeBase::TimeT        time,
            in TimeBase::InaccuracyT  inaccuracy);
        UTO uto_from_utc(
            in TimeBase::UtcT         utc);
    };
};
```

I installed the latest Java SDK on my computer (version 1.4.1_02-b06). It comes with an IDL-to-Java compiler called IDLJ (version 3.1). The command line switch `-f` proved to be very useful to explicitely create client- or server-side bindings:

```
idlj.exe -fclient -fd client time.idl
idlj.exe -fserver -fd server time.idl
```

After running the compiler I got two subdirectories both for client and server: CosTime and TimeBase. They represent the modules `CosTime` and `TimeBase` defined by time.idl. First I will give an overview of CosTime and lateron discuss TimeBase: (**bold** files are either necessary for the server *or* the client)

File	Client	Server
ComparisonType.java	☒	☒
ComparisonTypeHelper.java	☒	☒
ComparisonTypeHolder.java	☒	☒
OverlapType.java	☒	☒
OverlapTypeHelper.java	☒	☒
OverlapTypeHolder.java	☒	☒
TimeComparison.java	☒	☒
TimeComparisonHelper.java	☒	☒
TimeComparisonHolder.java	☒	☒
TimeService.java	☒	☒
TimeServiceHelper.java	☒	☐
TimeServiceHolder.java	☒	☐
TimeServicePOA.java	☐	☒
TimeServiceOperations.java	☒	☒
_TimeServiceStub.java	☒	☐
TimeUnavailable.java	☒	☒
TimeUnavailableHelper.java	☒	☒
TimeUnavailableHolder.java	☒	☒
UTO.java	☒	☒
UTOHelper.java	☒	☐
UTOHolder.java	☒	☐
UTOOperations.java	☒	☒
UTOPOA.java	☐	☒
_UTOStub.java	☒	☐

File	Client	Server
InaccuracyTHelper.java	☒	☒
IntervalT.java	☒	☒
IntervalTHelper.java	☒	☒
IntervalTHolder.java	☒	☒
TimeTHelper.java	☒	☒
UtcT.java	☒	☒
UtcTHelper.java	☒	☒
UtcTHolder.java	☒	☒

All files that were found both in the client's and the server's directories are bitwise identical, e.g. UTO.java.

Problem 2

Specify for each file whether its classes and interfaces belong to the Java language mapping or represent a part of the CORBA runtime.

Most files generated by the IDL-to-Java compiler exist in order to provide the language mapping. The few exceptions to that rule either derive directly from `org.omg.CORBA.portable.ObjectImpl` (the stubs) or `org.omg.PortableServer.Servant` (the skeletons).

Classes/interface that ensure the language mapping:

class / interface	signature
ComparisonType	`public class ComparisonType implements org.omg.CORBA.portable.IDLEntity`
ComparisonTypeHelper	`abstract public class ComparisonTypeHelper`
ComparisonTypeHolder	`public final class ComparisonTypeHolder implements org.omg.CORBA.portable.Streamable`
InaccuracyTHelper	`abstract public class InaccuracyTHelper`
IntervalT	`public final class IntervalT implements org.omg.CORBA.portable.IDLEntity`
IntervalTHelper	`abstract public class IntervalTHelper`
IntervalTHolder	`public final class IntervalTHolder implements org.omg.CORBA.portable.Streamable`
OverlapType	`public class OverlapType implements org.omg.CORBA.portable.IDLEntity`
OverlapTypeHelper	`abstract public class OverlapTypeHelper`
OverlapTypeHolder	`public final class OverlapTypeHolder implements org.omg.CORBA.portable.Streamable`
TimeComparison	`public class TimeComparison implements org.omg.CORBA.portable.IDLEntity`
TimeComparisonHelper	`abstract public class TimeComparisonHelper`
TimeComparisonHolder	`public final class TimeComparisonHolder implements org.omg.CORBA.portable.Streamable`
TimeService	`public interface TimeService extends TimeServiceOperations, org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity`
TimeServiceHelper	`abstract public class TimeServiceHelper`
TimeServiceHolder	`public final class TimeServiceHolder implements org.omg.CORBA.portable.Streamable`
TimeServiceOperations	`public interface TimeServiceOperations`
TimeTHelper	`abstract public class TimeTHelper`
TimeUnavailable	`public final class TimeUnavailable extends org.omg.CORBA.UserException`
TimeUnavailableHelper	`abstract public class TimeUnavailableHelper`
TimeUnavailableHolder	`public final class TimeUnavailableHolder implements org.omg.CORBA.portable.Streamable`
UtcT	`public final class UtcT implements org.omg.CORBA.portable.IDLEntity`
UtcTHelper	`abstract public class UtcTHelper`

(next page ...)

class / interface	signature
UtcTHolder	`public final class UtcTHolder implements org.omg.CORBA.portable.Streamable`
UTO	`public interface UTO extends UTOOperations, org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity`
UTOHelper	`abstract public class UTOHelper`
UTOHolder	`public final class UTOHolder implements org.omg.CORBA.portable.Streamable`
UTOOperations	`public interface UTOOperations`

The Holder and Helper classes are defined in chapter 1-4 and 1-5 of the specification, the operations are part of chapter 1-12.

Only four classes are actually needed to perform some CORBA runtime functionality. These are:

class / interface	signature
_TimeServiceStub	`public class _TimeServiceStub extends org.omg.CORBA.portable.ObjectImpl implements CosTime.TimeService`
TimeServicePOA	`public abstract class TimeServicePOA extends org.omg.PortableServer.Servant implements CosTime.TimeServiceOperations, org.omg.CORBA.portable.InvokeHandler`
_UTOStub	`public class _UTOStub extends org.omg.CORBA.portable.ObjectImpl implements CosTime.UTO`
UTOPOA	`public abstract class UTOPOA extends org.omg.PortableServer.Servant implements CosTime.UTOOperations, org.omg.CORBA.portable.InvokeHandler`

I not sure whether these four files are actually part of the CORBA runtime engine because they are part of the Java language mapping specification, too.

Problem 3

Many classes or interfaces implement the interface CosTime::TimeService. Examine them and indicate which definitions come off the language mapping or the implementation strategy. Are there any constructs that are either wrong or missing due to flaws of the IDL compilers ?

language mapping implementation strategy not compliant to the standard

TimeService.java:

```
package CosTime;


/**
* CosTime/TimeService.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

public interface TimeService extends TimeServiceOperations, org.omg.CORBA.Object,
org.omg.CORBA.portable.IDLEntity
{
} // interface TimeService
```

TimeServiceHelper.java:

`_id` and `__typeCode` are not defined by the specification.

```
package CosTime;


/**
* CosTime/TimeServiceHelper.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

abstract public class TimeServiceHelper
{
  private static String  _id = "IDL:hpi.uni-potsdam.de/CosTime/TimeService:1.0";

  public static void insert (org.omg.CORBA.Any a, CosTime.TimeService that)
  {
    org.omg.CORBA.portable.OutputStream out = a.create_output_stream ();
    a.type (type ());
    write (out, that);
    a.read_value (out.create_input_stream (), type ());
  }

  public static CosTime.TimeService extract (org.omg.CORBA.Any a)
  {
    return read (a.create_input_stream ());
  }

  private static org.omg.CORBA.TypeCode __typeCode = null;
  synchronized public static org.omg.CORBA.TypeCode type ()
  {
    if (__typeCode == null)
    {
      __typeCode = org.omg.CORBA.ORB.init ().create_interface_tc (CosTime.TimeServiceHelper.id
(), "TimeService");
    }
```

```
    return __typeCode;
  }

  public static String id ()
  {
    return _id;
  }

  public static CosTime.TimeService read (org.omg.CORBA.portable.InputStream istream)
  {
    return narrow (istream.read_Object (_TimeServiceStub.class));
  }

  public static void write (org.omg.CORBA.portable.OutputStream ostream, CosTime.TimeService
value)
  {
    ostream.write_Object ((org.omg.CORBA.Object) value);
  }

  public static CosTime.TimeService narrow (org.omg.CORBA.Object obj)
  {
    if (obj == null)
      return null;
    else if (obj instanceof CosTime.TimeService)
      return (CosTime.TimeService)obj;
    else if (!obj._is_a (id ()))
      throw new org.omg.CORBA.BAD_PARAM ();
    else
    {
      org.omg.CORBA.portable.Delegate delegate =
((org.omg.CORBA.portable.ObjectImpl)obj)._get_delegate ();
      CosTime._TimeServiceStub stub = new CosTime._TimeServiceStub ();
      stub._set_delegate(delegate);
      return stub;
    }
  }

}
```

TimeServiceHolder.java:

```
package CosTime;

/**
* CosTime/TimeServiceHolder.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

public final class TimeServiceHolder implements org.omg.CORBA.portable.Streamable
{
  public CosTime.TimeService value = null;

  public TimeServiceHolder ()
  {
  }

  public TimeServiceHolder (CosTime.TimeService initialValue)
  {
    value = initialValue;
  }

  public void _read (org.omg.CORBA.portable.InputStream i)
  {
    value = CosTime.TimeServiceHelper.read (i);
  }

  public void _write (org.omg.CORBA.portable.OutputStream o)
  {
```

```
    CosTime.TimeServiceHelper.write (o, value);
  }

  public org.omg.CORBA.TypeCode _type ()
  {
    return CosTime.TimeServiceHelper.type ();
  }

}
```

TimeServiceOperations.java:

```
package CosTime;


/**
* CosTime/TimeServiceOperations.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

public interface TimeServiceOperations
{
  CosTime.UTO universal_time () throws CosTime.TimeUnavailable;
  CosTime.UTO secure_universal_time () throws CosTime.TimeUnavailable;
  CosTime.UTO new_universal_time (long time, long inaccuracy);
  CosTime.UTO uto_from_utc (TimeBase.UtcT utc);
} // interface TimeServiceOperations
```

TimeServicePOA.java:

```
package CosTime;


/**
* CosTime/TimeServicePOA.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

public abstract class TimeServicePOA extends org.omg.PortableServer.Servant
 implements CosTime.TimeServiceOperations, org.omg.CORBA.portable.InvokeHandler
{

  // Constructors

  private static java.util.Hashtable _methods = new java.util.Hashtable ();
  static
  {
    _methods.put ("universal_time", new java.lang.Integer (0));
    _methods.put ("secure_universal_time", new java.lang.Integer (1));
    _methods.put ("new_universal_time", new java.lang.Integer (2));
    _methods.put ("uto_from_utc", new java.lang.Integer (3));
  }

  public org.omg.CORBA.portable.OutputStream _invoke (String $method,
                                org.omg.CORBA.portable.InputStream in,
                                org.omg.CORBA.portable.ResponseHandler $rh)
  {
    org.omg.CORBA.portable.OutputStream out = null;
    java.lang.Integer __method = (java.lang.Integer)_methods.get ($method);
    if (__method == null)
```

```
      throw new org.omg.CORBA.BAD_OPERATION (0,
org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);

    switch (__method.intValue ())
    {
       case 0:  // CosTime/TimeService/universal_time
       {
         try {
           CosTime.UTO $result = null;
           $result = this.universal_time ();
           out = $rh.createReply();
           CosTime.UTOHelper.write (out, $result);
         } catch (CosTime.TimeUnavailable $ex) {
           out = $rh.createExceptionReply ();
           CosTime.TimeUnavailableHelper.write (out, $ex);
         }
         break;
       }

       case 1:  // CosTime/TimeService/secure_universal_time
       {
         try {
           CosTime.UTO $result = null;
           $result = this.secure_universal_time ();
           out = $rh.createReply();
           CosTime.UTOHelper.write (out, $result);
         } catch (CosTime.TimeUnavailable $ex) {
           out = $rh.createExceptionReply ();
           CosTime.TimeUnavailableHelper.write (out, $ex);
         }
         break;
       }

       case 2:  // CosTime/TimeService/new_universal_time
       {
         long time = TimeBase.TimeTHelper.read (in);
         long inaccuracy = TimeBase.InaccuracyTHelper.read (in);
         CosTime.UTO $result = null;
         $result = this.new_universal_time (time, inaccuracy);
         out = $rh.createReply();
         CosTime.UTOHelper.write (out, $result);
         break;
       }

       case 3:  // CosTime/TimeService/uto_from_utc
       {
         TimeBase.UtcT utc = TimeBase.UtcTHelper.read (in);
         CosTime.UTO $result = null;
         $result = this.uto_from_utc (utc);
         out = $rh.createReply();
         CosTime.UTOHelper.write (out, $result);
         break;
       }

       default:
         throw new org.omg.CORBA.BAD_OPERATION (0,
org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);
    }

    return out;
  } // _invoke

  // Type-specific CORBA::Object operations
  private static String[] __ids = {
    "IDL:hpi.uni-potsdam.de/CosTime/TimeService:1.0"};

  public String[] _all_interfaces (org.omg.PortableServer.POA poa, byte[] objectId)
  {
    return (String[])__ids.clone ();
  }

  public TimeService _this()
  {
    return TimeServiceHelper.narrow(
    super._this_object());
```

```
  }

  public TimeService _this(org.omg.CORBA.ORB orb)
  {
    return TimeServiceHelper.narrow(
    super._this_object(orb));
  }


} // class TimeServicePOA
```

_TimeServiceStub.java:

```
package CosTime;


/**
* CosTime/_TimeServiceStub.java .
* Generated by the IDL-to-Java compiler (portable), version "3.1"
* from time.idl
* Freitag, 18. April 2003 11.06 Uhr CEST
*/

public class _TimeServiceStub extends org.omg.CORBA.portable.ObjectImpl implements
CosTime.TimeService
{

  public CosTime.UTO universal_time () throws CosTime.TimeUnavailable
  {
            org.omg.CORBA.portable.InputStream $in = null;
            try {
                org.omg.CORBA.portable.OutputStream $out = _request ("universal_time", true);
                $in = _invoke ($out);
                CosTime.UTO $result = CosTime.UTOHelper.read ($in);
                return $result;
            } catch (org.omg.CORBA.portable.ApplicationException $ex) {
                $in = $ex.getInputStream ();
                String _id = $ex.getId ();
                if (_id.equals ("IDL:hpi.uni-potsdam.de/CosTime/TimeUnavailable:1.0"))
                    throw CosTime.TimeUnavailableHelper.read ($in);
                else
                    throw new org.omg.CORBA.MARSHAL (_id);
            } catch (org.omg.CORBA.portable.RemarshalException $rm) {
                return universal_time (        );
            } finally {
                _releaseReply ($in);
            }
  } // universal_time

  public CosTime.UTO secure_universal_time () throws CosTime.TimeUnavailable
  {
            org.omg.CORBA.portable.InputStream $in = null;
            try {
                org.omg.CORBA.portable.OutputStream $out = _request ("secure_universal_time",
true);
                $in = _invoke ($out);
                CosTime.UTO $result = CosTime.UTOHelper.read ($in);
                return $result;
            } catch (org.omg.CORBA.portable.ApplicationException $ex) {
                $in = $ex.getInputStream ();
                String _id = $ex.getId ();
                if (_id.equals ("IDL:hpi.uni-potsdam.de/CosTime/TimeUnavailable:1.0"))
                    throw CosTime.TimeUnavailableHelper.read ($in);
                else
                    throw new org.omg.CORBA.MARSHAL (_id);
            } catch (org.omg.CORBA.portable.RemarshalException $rm) {
                return secure_universal_time (        );
            } finally {
                _releaseReply ($in);
            }
```

```
  } // secure_universal_time

  public CosTime.UTO new_universal_time (long time, long inaccuracy)
  {
            org.omg.CORBA.portable.InputStream $in = null;
            try {
                org.omg.CORBA.portable.OutputStream $out = _request ("new_universal_time",
true);
                TimeBase.TimeTHelper.write ($out, time);
                TimeBase.InaccuracyTHelper.write ($out, inaccuracy);
                $in = _invoke ($out);
                CosTime.UTO $result = CosTime.UTOHelper.read ($in);
                return $result;
            } catch (org.omg.CORBA.portable.ApplicationException $ex) {
                $in = $ex.getInputStream ();
                String _id = $ex.getId ();
                throw new org.omg.CORBA.MARSHAL (_id);
            } catch (org.omg.CORBA.portable.RemarshalException $rm) {
                return new_universal_time (time, inaccuracy        );
            } finally {
                _releaseReply ($in);
            }
  } // new_universal_time

  public CosTime.UTO uto_from_utc (TimeBase.UtcT utc)
  {
            org.omg.CORBA.portable.InputStream $in = null;
            try {
                org.omg.CORBA.portable.OutputStream $out = _request ("uto_from_utc", true);
                TimeBase.UtcTHelper.write ($out, utc);
                $in = _invoke ($out);
                CosTime.UTO $result = CosTime.UTOHelper.read ($in);
                return $result;
            } catch (org.omg.CORBA.portable.ApplicationException $ex) {
                $in = $ex.getInputStream ();
                String _id = $ex.getId ();
                throw new org.omg.CORBA.MARSHAL (_id);
            } catch (org.omg.CORBA.portable.RemarshalException $rm) {
                return uto_from_utc (utc        );
            } finally {
                _releaseReply ($in);
            }
  } // uto_from_utc

  // Type-specific CORBA::Object operations
  private static String[] __ids = {
    "IDL:hpi.uni-potsdam.de/CosTime/TimeService:1.0"};

  public String[] _ids ()
  {
    return (String[])__ids.clone ();
  }

  private void readObject (java.io.ObjectInputStream s) throws java.io.IOException
  {
     String str = s.readUTF ();
     String[] args = null;
     java.util.Properties props = null;
     org.omg.CORBA.Object obj = org.omg.CORBA.ORB.init (args, props).string_to_object (str);
     org.omg.CORBA.portable.Delegate delegate = ((org.omg.CORBA.portable.ObjectImpl)
obj)._get_delegate ();
     _set_delegate (delegate);
  }

  private void writeObject (java.io.ObjectOutputStream s) throws java.io.IOException
  {
     String[] args = null;
     java.util.Properties props = null;
     String str = org.omg.CORBA.ORB.init (args, props).object_to_string (this);
     s.writeUTF (str);
  }
} // class _TimeServiceStub
```

Java does not support unsigned types even though they are allowed by the CORBA IDL. For example, the data type `TimeT` is mapped from IDL's `unsigned long long` to Java's `long` without explicitly verifying any overflow. Very large values (typically $>2^{31}$) may be mis-interpreted as negative numbers causing runtime errors usually hard to detect. By default, Java supports Unicode. Unlike CORBA all characters in Java are treated as wide chars – a behavior potentially raising exceptions when mapping IDL's char to Java's char. A good workaround is to avoid IDL's `char` and switch to `wchar` all the time (at the cost of increased ORB traffic). The same applies to `string`/`wstring`.

The IDL data type `long double` is not supported by the language mapping yet.