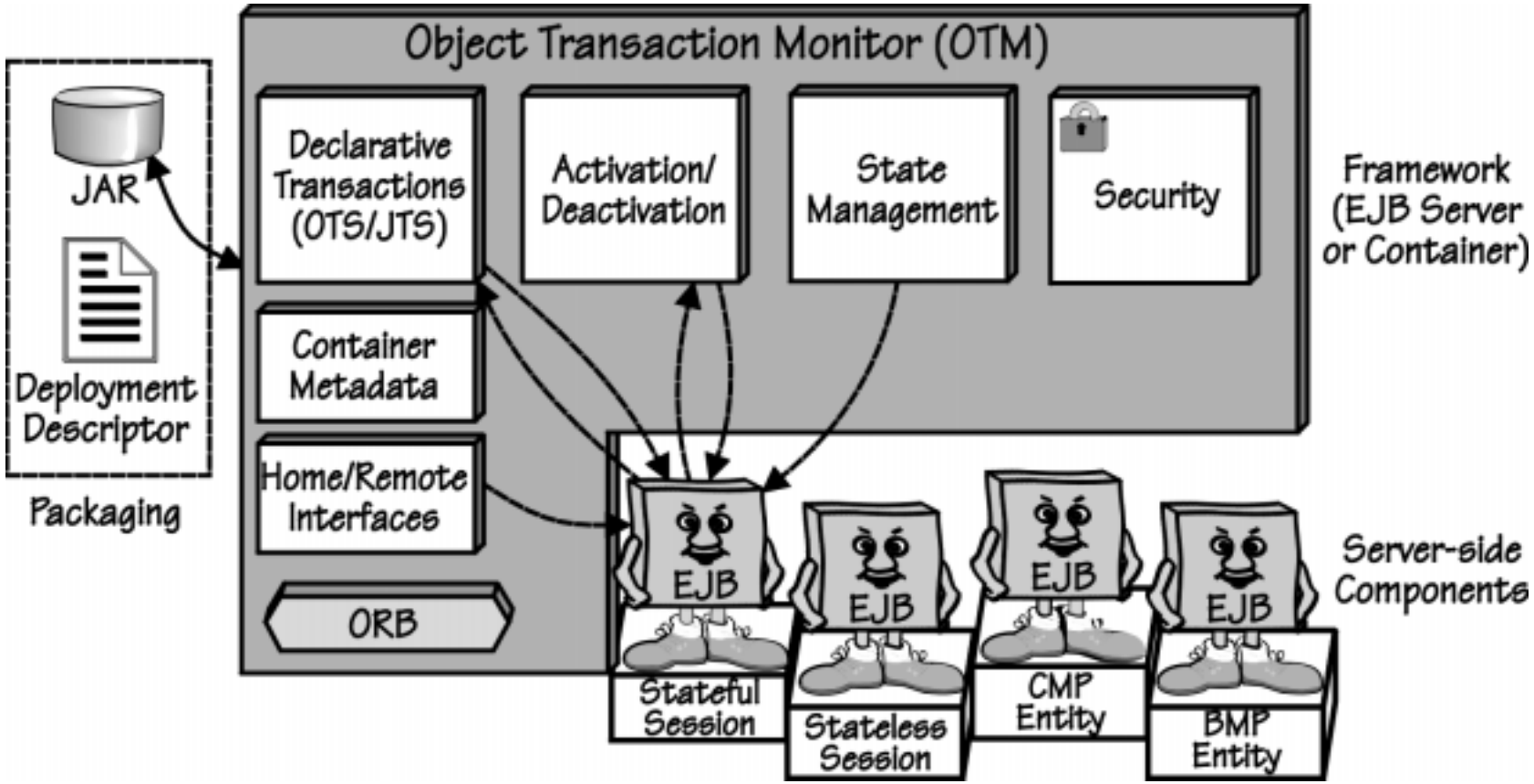
Enterprise JavaBeans

Dan Harkey
Director
Client/Server and Distributed Objects Program
San Jose State University
dharkey@email.sjsu.edu
www.corbajava.engr.sjsu.edu

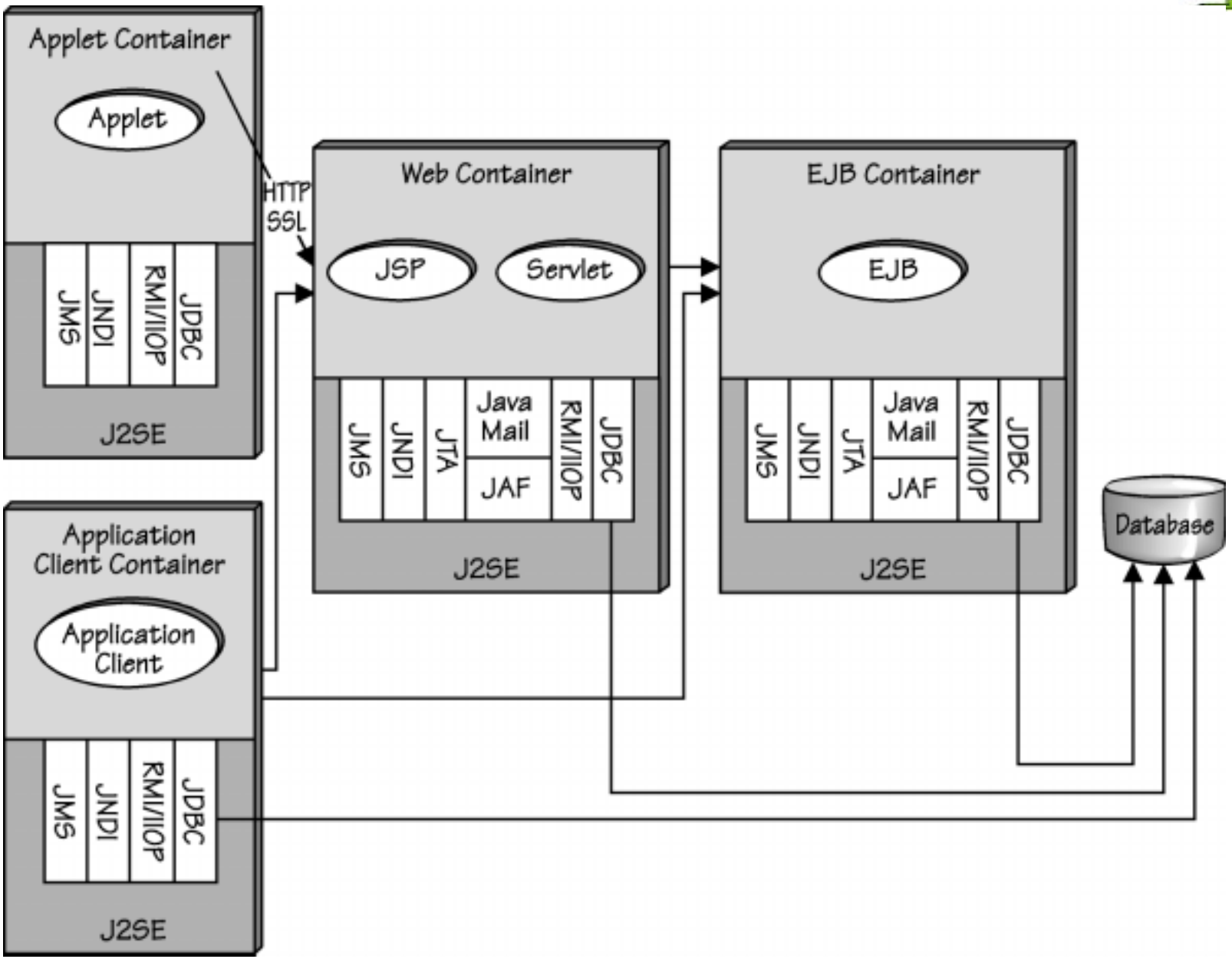
Agenda

- ✓ Enterprise JavaBeans (EJB) Overview
- ✓ Your first EJB code examples
- ✓ EJB Vendors: Meet the players

Enterprise JavaBeans (EJB) Overview



EJB and Java 2 Enterprise Edition



EJB Related Technologies: JNDI

- ✓ Java Naming and Directory Service (JNDI)
  - Java API layer built on top of existing Naming or Directory Service
    - COS Naming
    - LDAP
    - etc
  - Provides a look-up mechanism for EJB clients
  - Used to provide deployment information to EJBs
    - Home Interface look-up
    - Environment properties
    - Resource locations

EJB Related Technologies: JTA/JTS

✓ Java Transaction API (JTA)

- Provides transaction interfaces for applications, application servers, and resource managers
- Consists of three parts
  - High-level application interface that provides begin, commit, rollback semantics
  - Java Mapping of X/Open XA protocol
  - Transaction Manager interface for an application server to control transactions

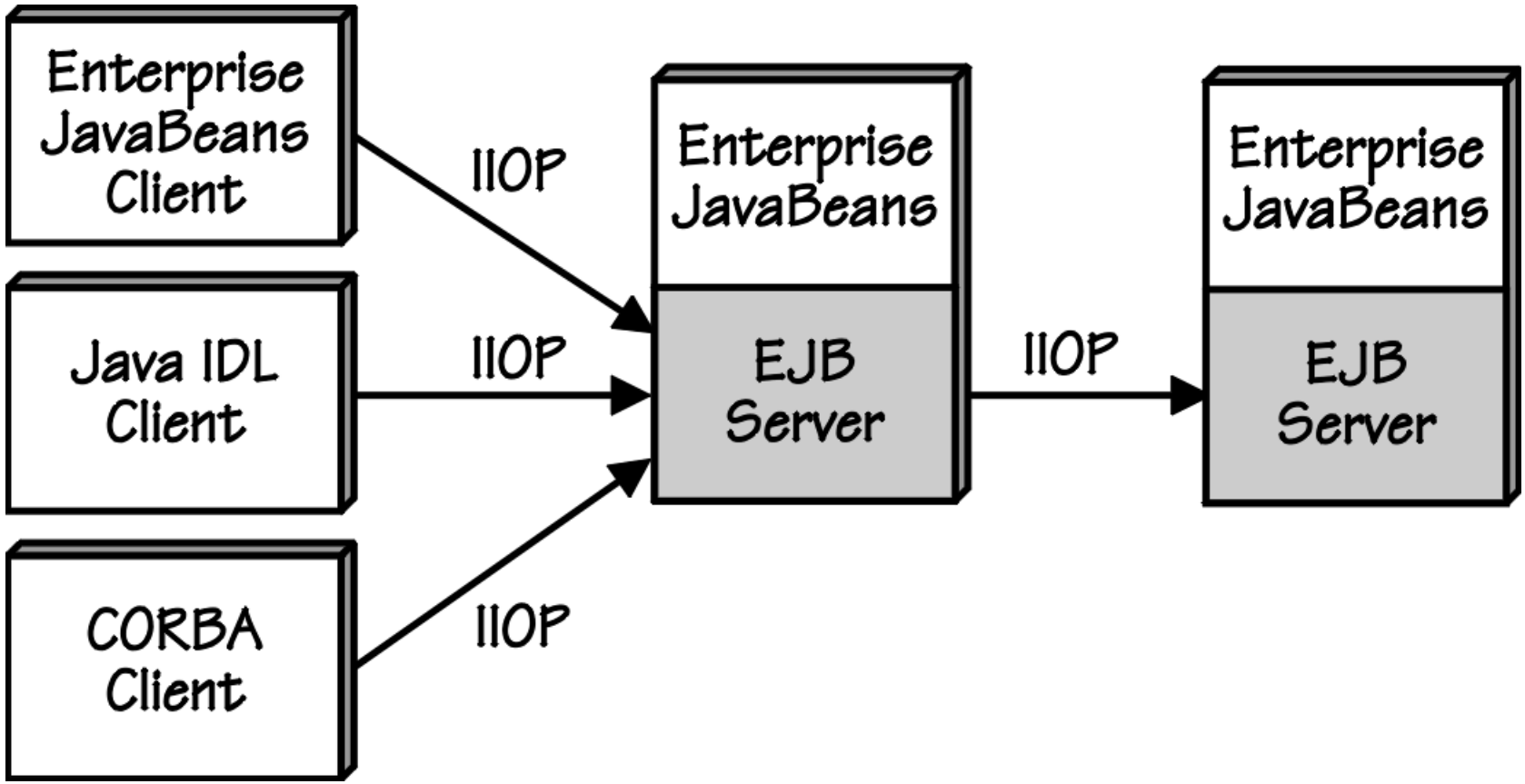
EJB Related Technologies: JTA/JTS (cont.)

- ✓ Java Transaction Service (JTS)
  - Specifies the implementation of a transaction manager using the JTA interfaces
  - Uses the Java mapping of the CORBA Object Transaction Service (OTS) version 1.1 for the implementation

EJB Related Technologies: RMI/IIOP

- ✓ Remote Method Invocation (RMI)/Internet Inter-ORB Protocol (IIOP)
  - EJBs and their clients use RMI programming APIs
  - IIOP is used as the transport protocol for interoperability and to provide transaction and security contexts

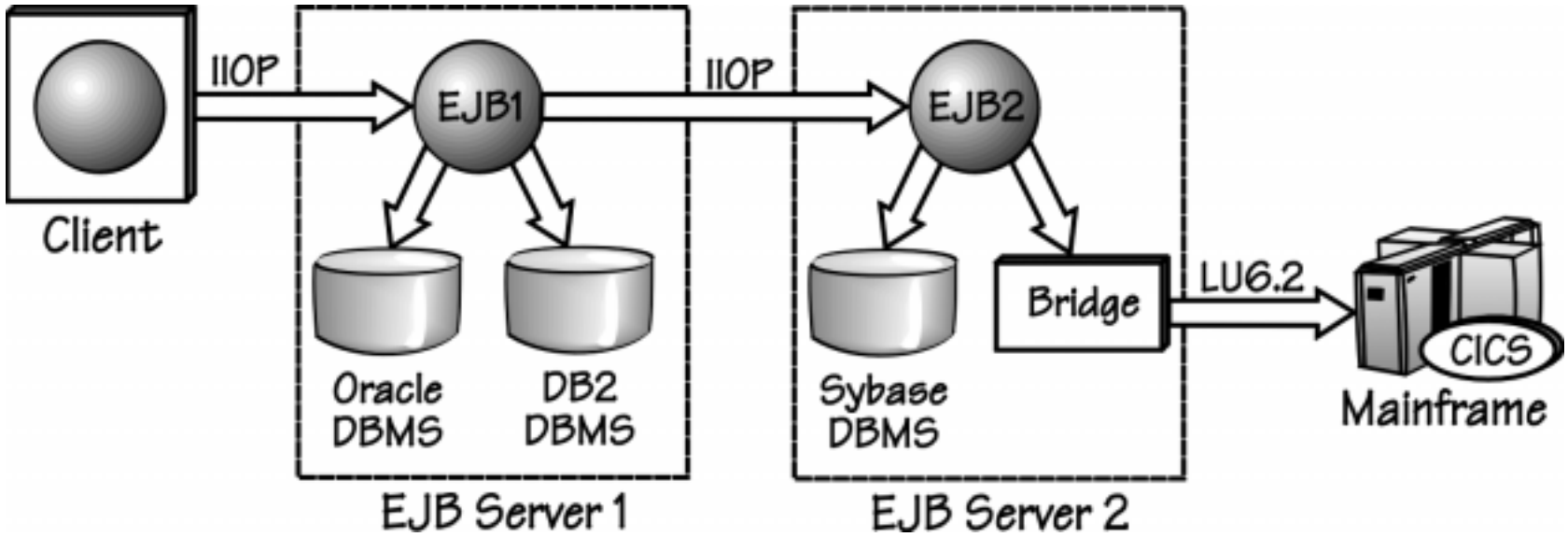
EJB Interoperability



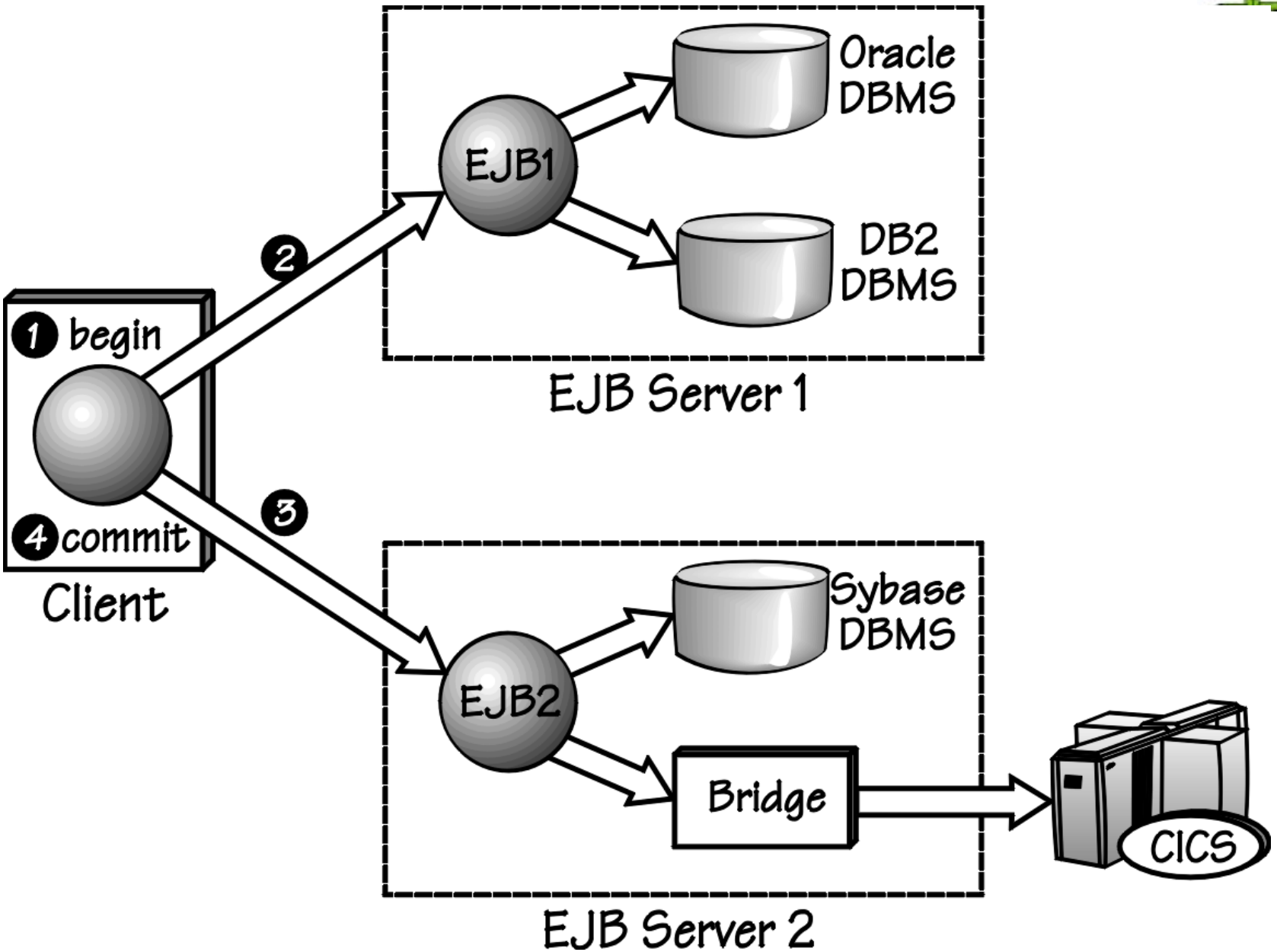
EJB Transactions

- ✓ Container-managed transactions
  - Six types of declarative transactions: NotSupported, Required, Supports, RequiresNew, Mandatory, Never
  - Transaction types declared in deployment descriptor for bean methods
- ✓ Session beans can manage transactions using begin, commit, and rollback methods of the UserTransaction interface
- ✓ Clients can also use the UserTransaction interface to manage transactions

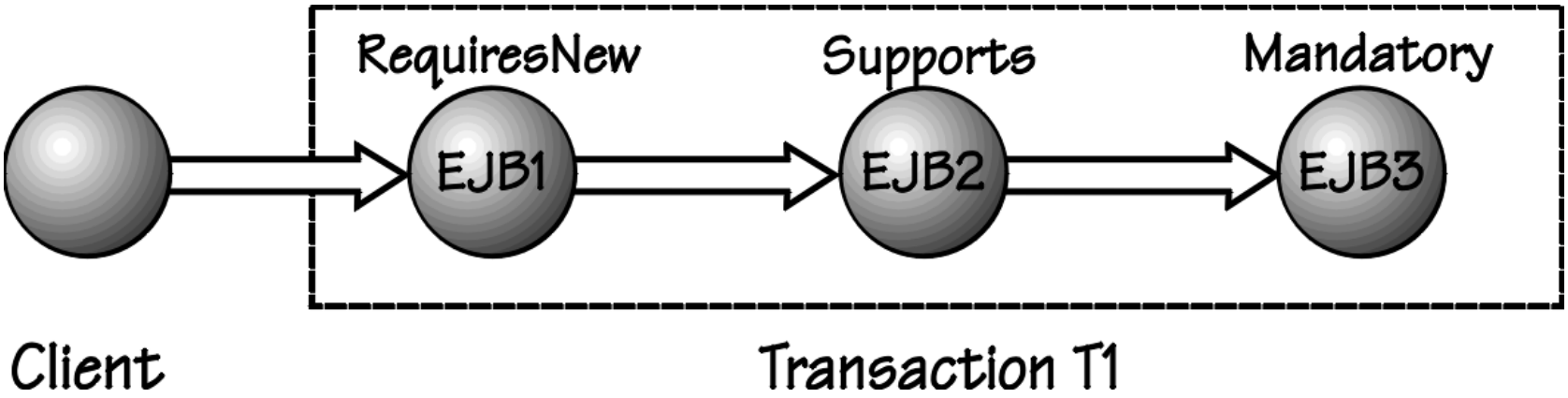
EJB Transactions: Transaction Context Propagation



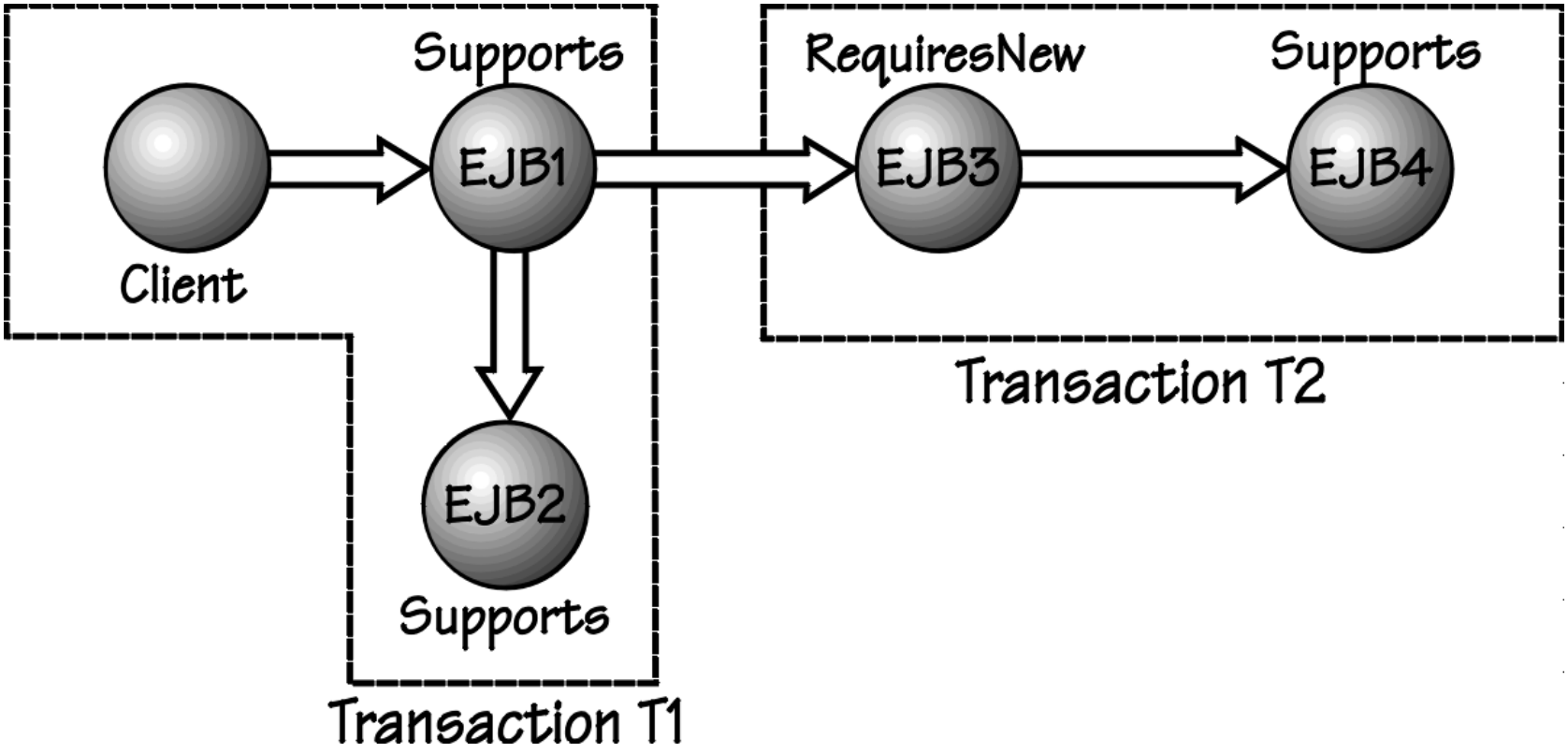
EJB Transactions: Client Demarcated Transactions



EJB Declarative Transactions



EJB Declarative Transactions (cont.)



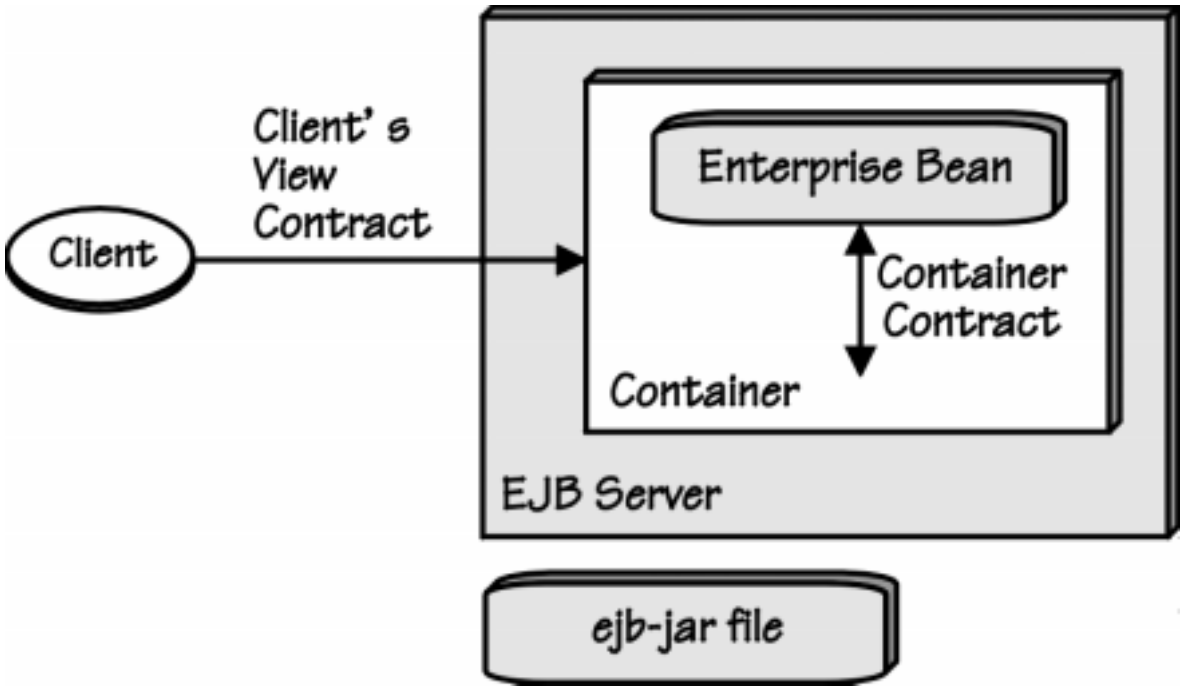
EJB Security

- ✓ Roles and method permissions defined in deployment descriptor
- ✓ APIs defined for EJBs to get Principal and test if caller is defined in a role
- ✓ Authentication, encryption, auditing, etc. is outside of the EJB spec definition and is left to the EJB Server vendor

EJB Packaging

- ✓ EJBs are stored in JAR files
- ✓ Multiple EJBs used to assemble an application
- ✓ Application is deployed to an EJB server
- ✓ XML used to describe bean, application, and deployment information

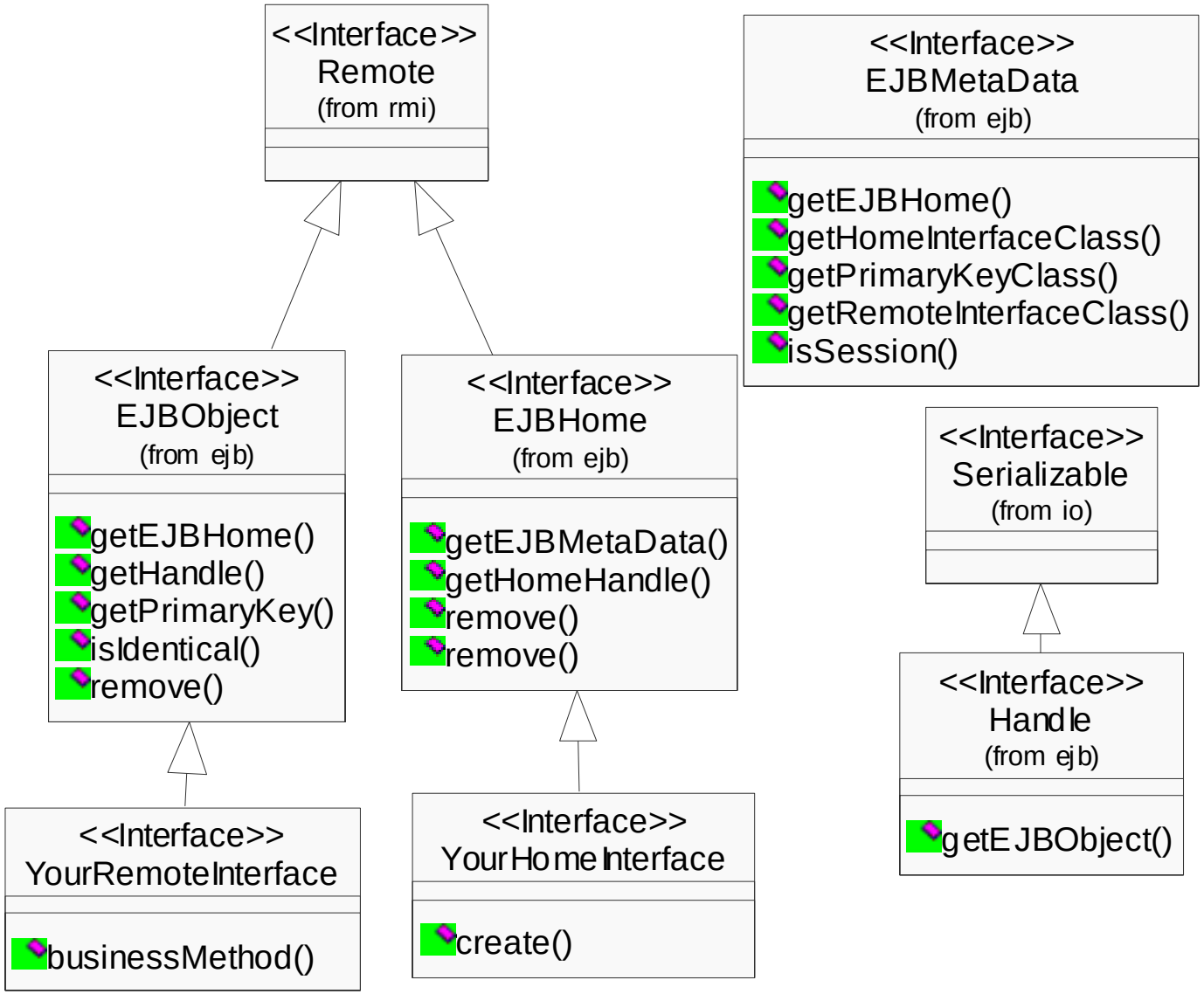
EJB Contracts



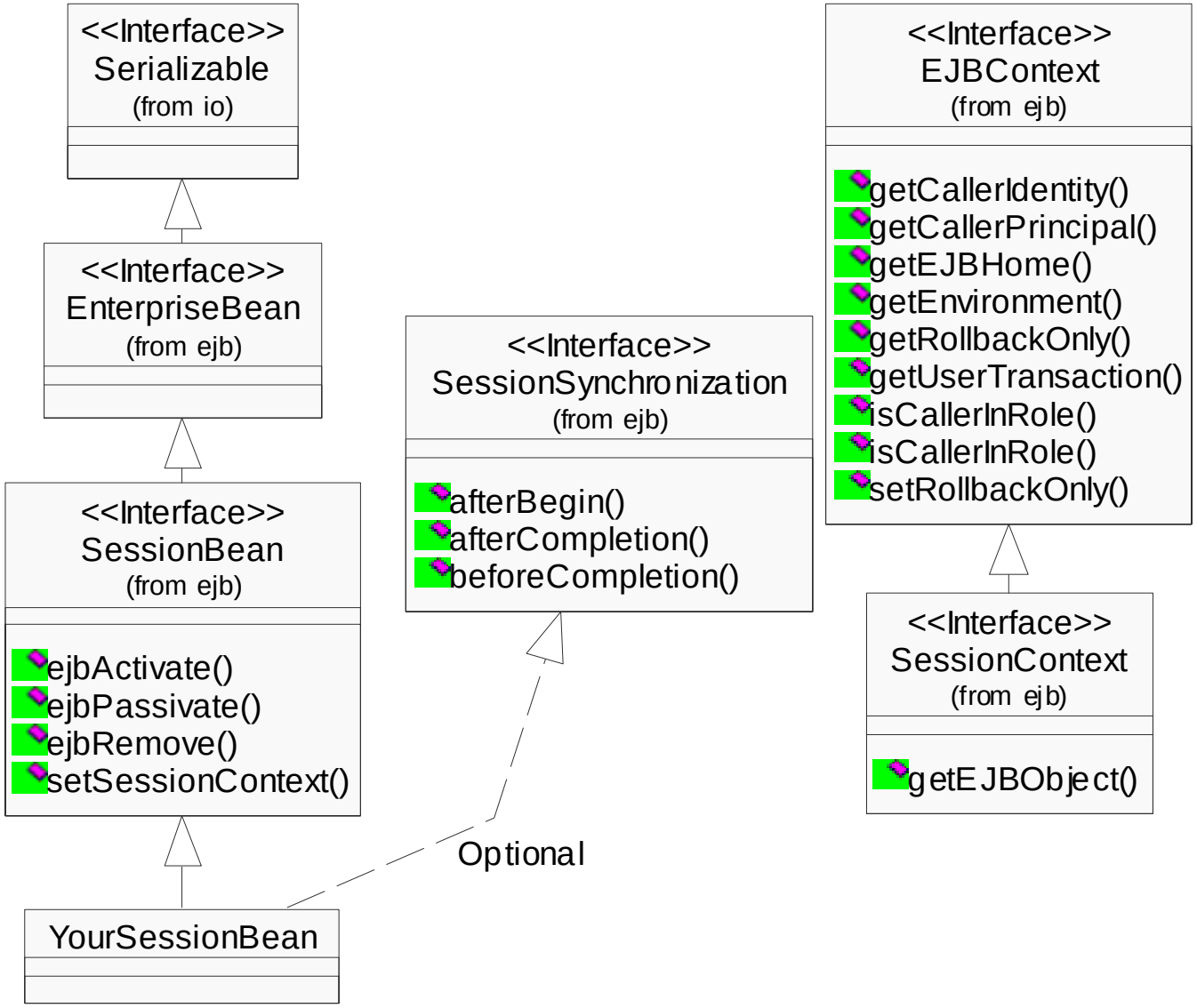
EJB Session Beans

- ✓ Implement business logic that runs on an EJB server
- ✓ Are not shared between clients--logical extension of a single client
- ✓ Do not maintain persistent state
- ✓ Two types
  - Stateless--no client specific state is maintained between client method calls
  - Stateful--can maintain conversational state

Client View of Session Bean



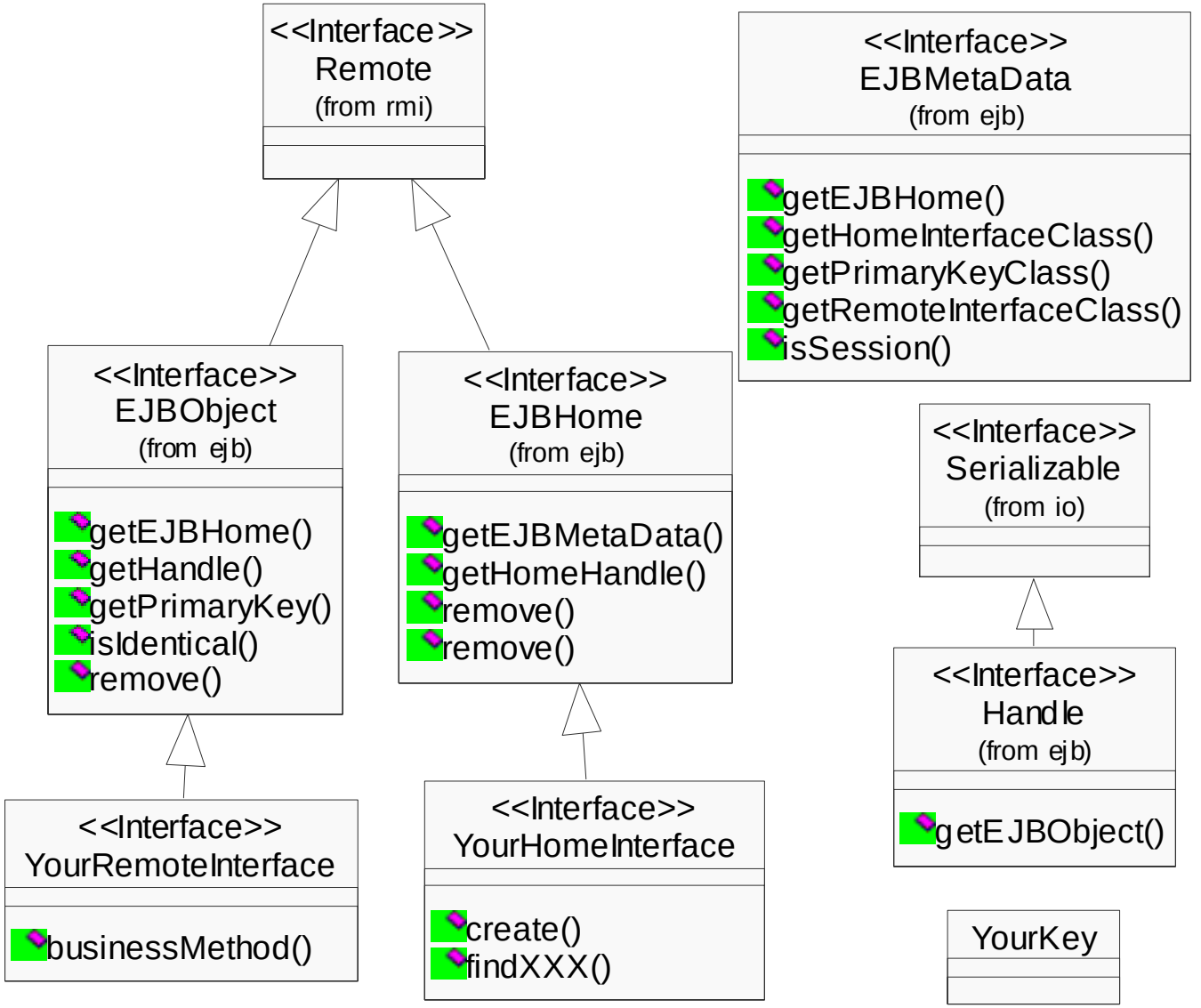
Session Bean Classes



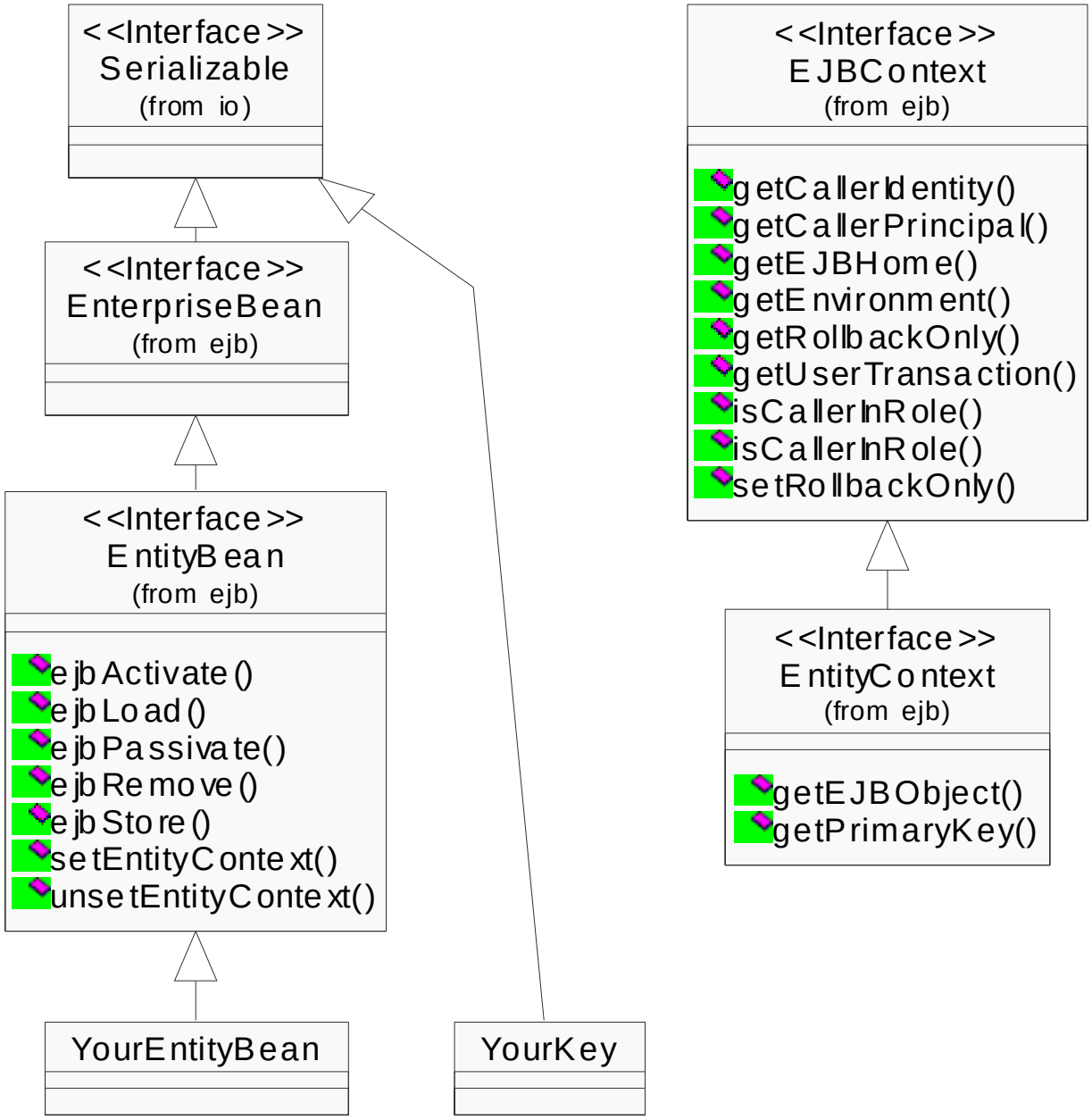
EJB Entity Beans

- ✓ An object-oriented view of an "entity" in persistent storage
- ✓ Can be accessed by multiple clients concurrently
- ✓ Container manages state synchronization between bean and persistent storage
- ✓ Entity bean identity established using a key value

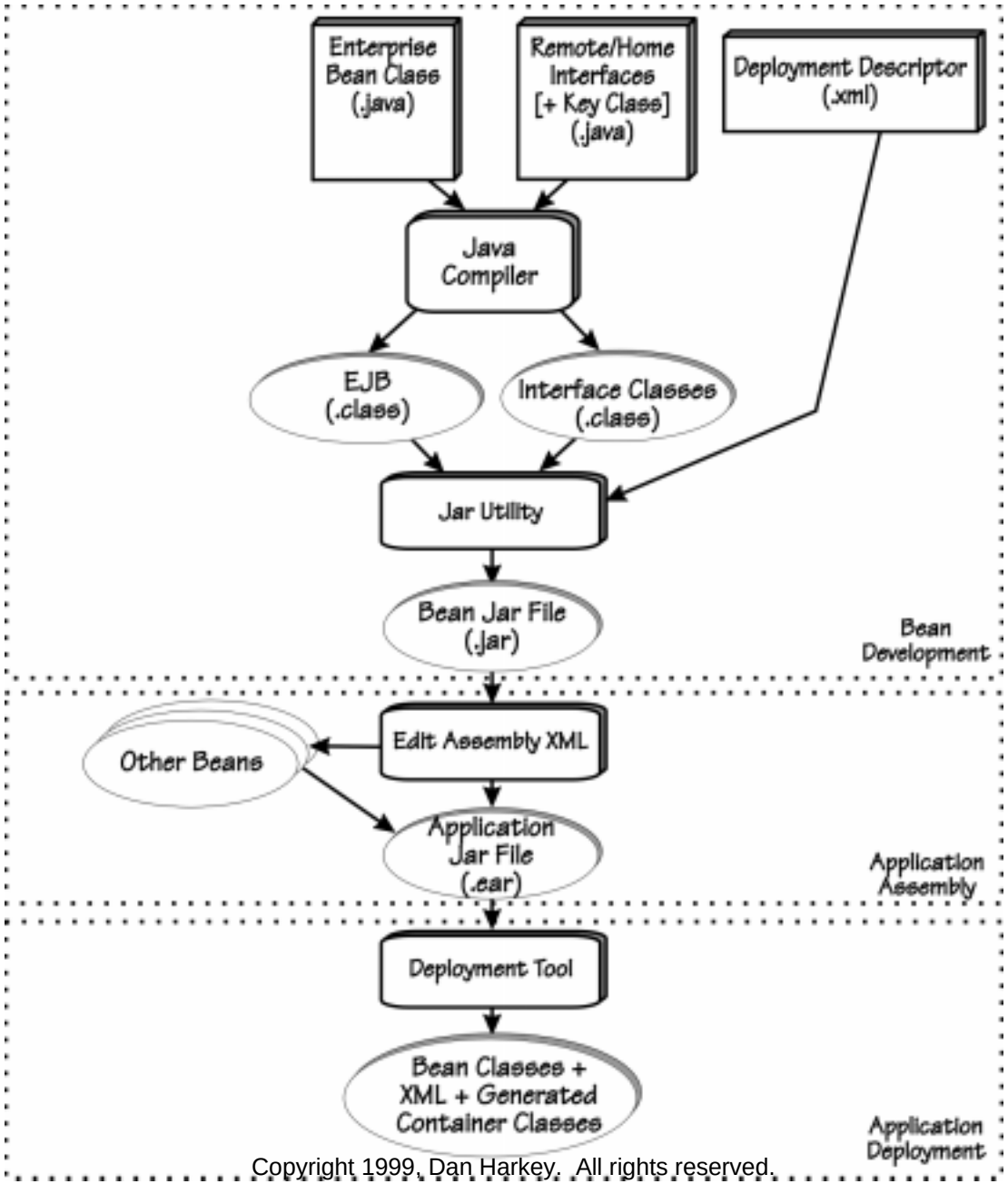
Client View of an Entity Bean



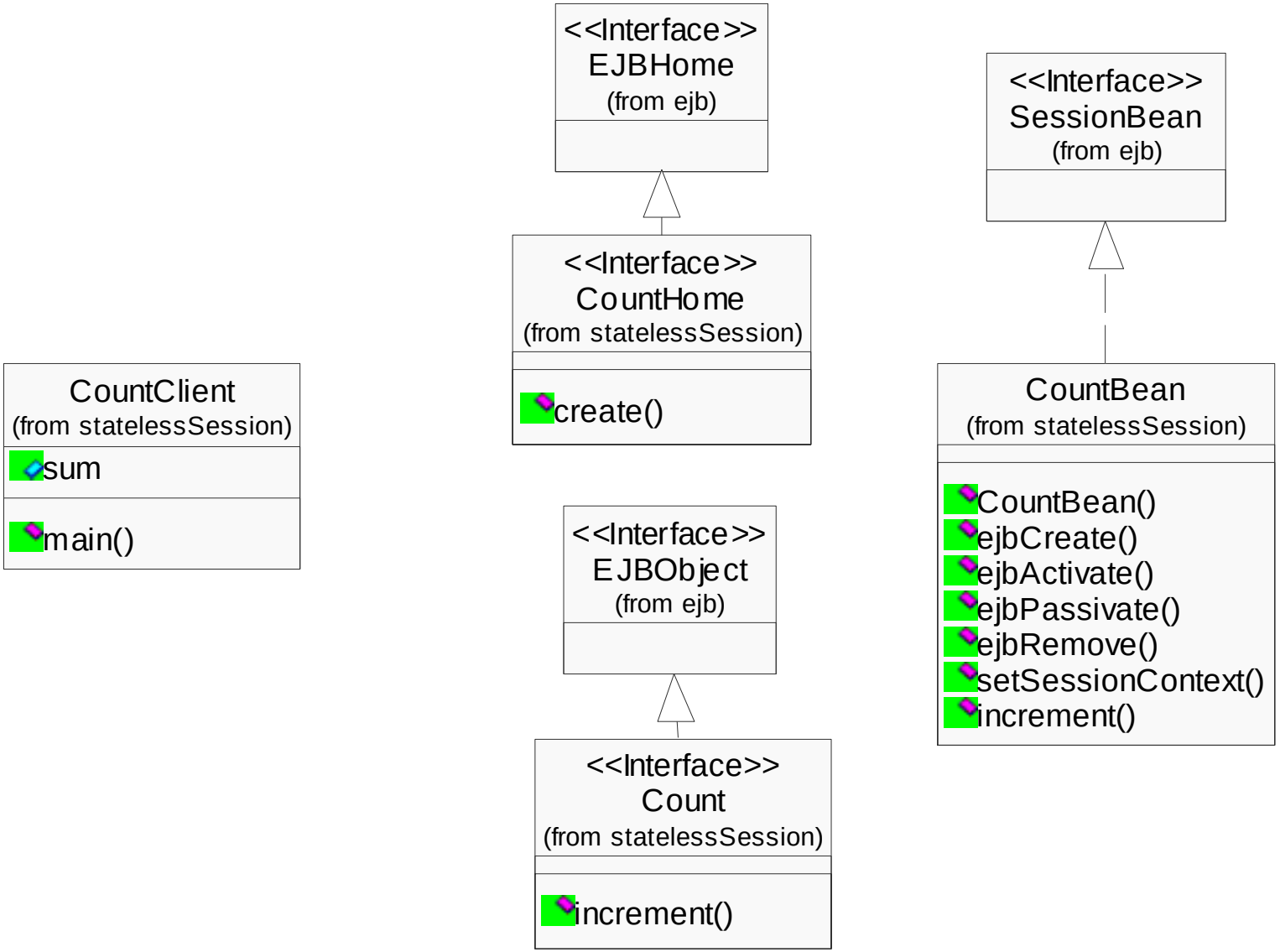
Entity Bean Classes



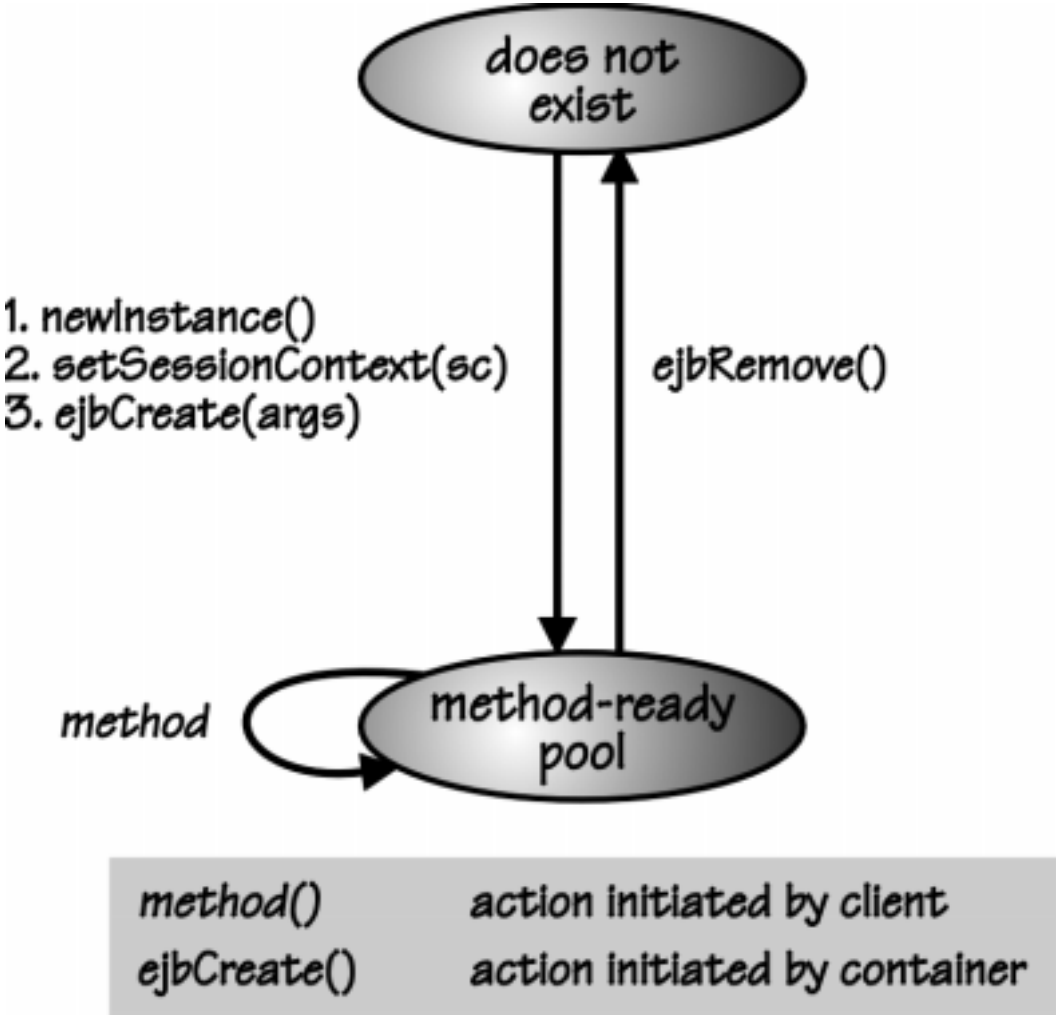
EJB Development Process



Count Stateless Session Bean



Stateless Session Bean State Diagram



Count Stateless Session Bean: Home Interface

```
// CountHome.java
package count.statelessSession;

import javax.ejb.EJBHome;
import java.rmi.RemoteException;
import javax.ejb.CreateException;

public interface CountHome extends EJBHome
{
  public Count create() throws RemoteException, CreateException;
}
```

Count Stateless Session Bean: Remote Interface

```
// Count.java
package count.statelessSession;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Count extends EJBObject
{
  public int increment(int sum) throws RemoteException;
}
```

Count Stateless Session Bean: EJB Class

```
// CountBean.java
package count.statelessSession;

import javax.ejb.*;
import java.rmi.RemoteException;

public class CountBean implements SessionBean
{
  // no arg contructor
  public CountBean() {}

  public void ejbCreate() {}
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void ejbRemove() {}
  public void setSessionContext(SessionContext ctx ) {}

  public int increment(int sum)
  {
    return ++sum;
  }
}
```

Count Stateless Session Bean: Deployment Descriptor (1 of 2)

```
<?xml version="1.0" encoding="Cp1252"?>

<ejb-jar>
  <description>no description</description>
  <display-name>Ejb1</display-name>
  <enterprise-beans>
    <session>
      <description>no description</description>
      <display-name>statelessSession</display-name>
      <ejb-name>statelessSession</ejb-name>
      <home>count.statelessSession.CountHome</home>
      <remote>count.statelessSession.Count</remote>
      <ejb-class>count.statelessSession.CountBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
```

Count Stateless Session Bean: Deployment Descriptor (2 of 2)

```
  <assembly-descriptor>
    <container-transactions>
      <container-transaction>
        <method>
          <ejb-name>statelessSession</ejb-name>
          <method-intf>count.statelessSession.Count</method-intf>
          <method-name>increment</method-name>
          <method-param>int</method-param>
        </method>
        <trans-attribute>NotSupported</trans-attribute>
      </container-transaction>
    </container-transactions>
  </assembly-descriptor>
</ejb-jar>
```

Count Stateless Session Bean: Client (1 of 3)

```
// CountClient.java
package count.statelessSession;

import count.statelessSession.Count;      // Count EJB Remote
import count.statelessSession.CountHome;  // Count EJB Home

import java.rmi.*;
import java.util.*;
import java.io.*;

import javax.rmi.*;
import javax.naming.*;
import javax.ejb.*;

public class CountClient
{
  public static void main ( String args[] )
  {
    int sum; // current sum
    CountHome countHome;
```

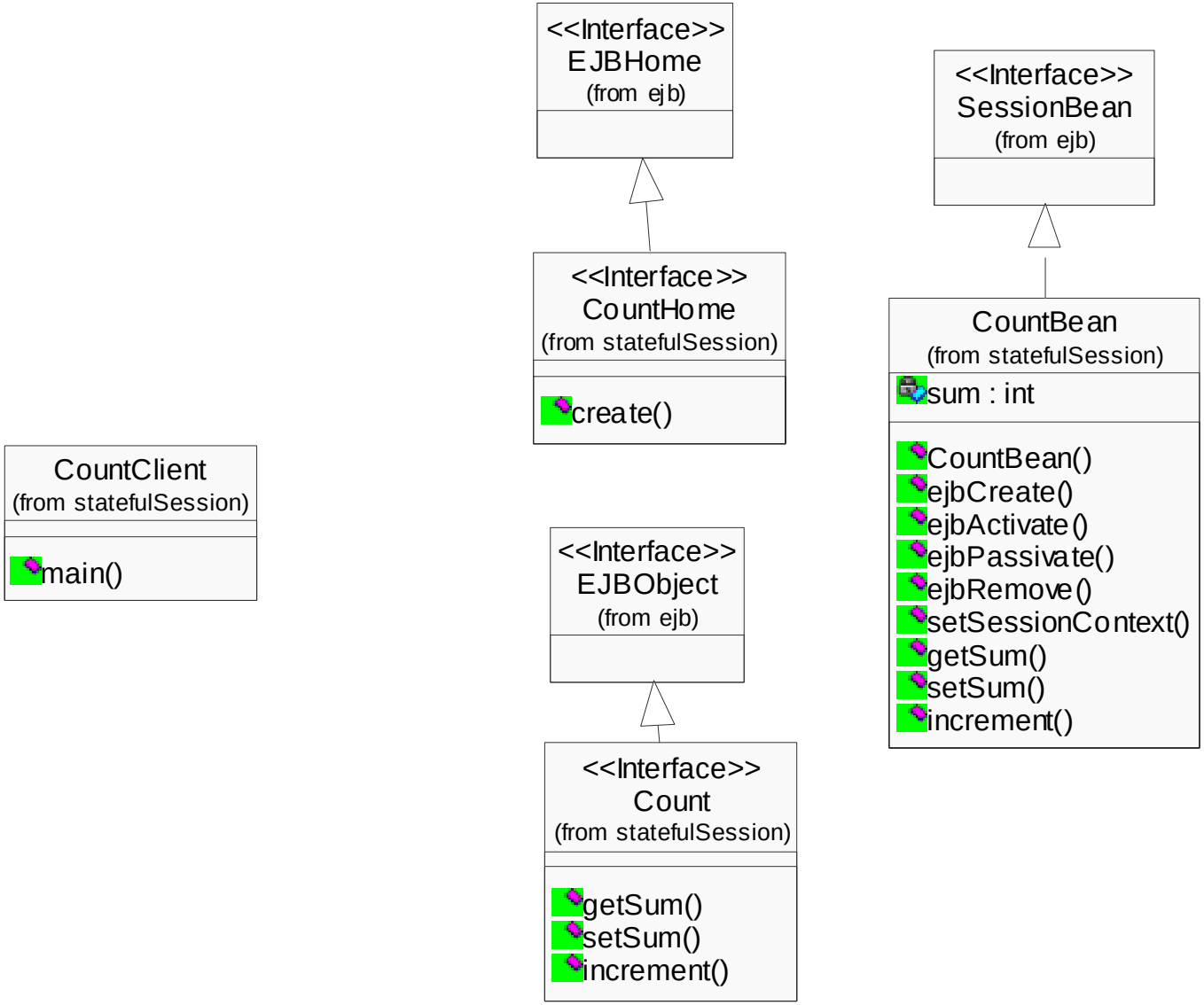
Count Stateless Session Bean: Client (2 of 3)

```
try
{
  // Get the initial JNDI context
  System.out.println("Getting initial context");
  InitialContext context = new InitialContext();

  // Find the home interface and narrow to CountHome
  Object object = context.lookup("statelessSessionCountHome");
  countHome =
    (CountHome)PortableRemoteObject.narrow(object, CountHome.class);

  // Create EJB
  System.out.println("Creating EJB");
  Count count = countHome.create();

  // Set sum to initial value of 0
  System.out.println("Setting sum to 0");
  sum = 0;
```

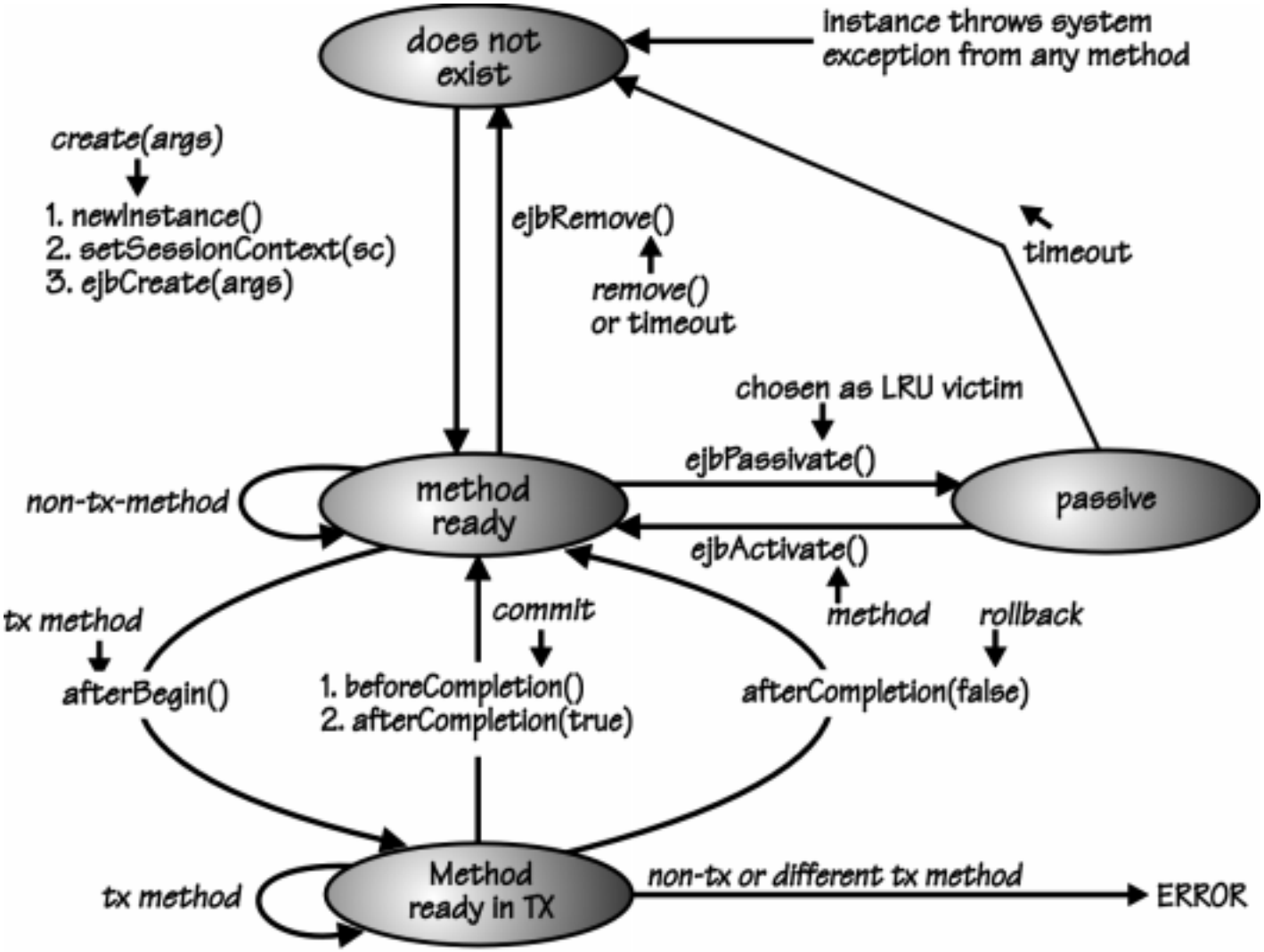
Count Stateless Session Bean: Client (3 of 3)

```
    // Calculate Start time
    long startTime = System.currentTimeMillis();

    // Increment 1000 times
    System.out.println("Incrementing");
    for (int i = 0 ; i < 1000 ; i++ )
    { sum = count.increment(sum);
    }

    // Calculate stop time; print out statistics
    long stopTime = System.currentTimeMillis();
    System.out.println("Avg Ping = "
                    + ((stopTime - startTime)/1000f) + " msecs");
    System.out.println("Sum = " + sum);
  } catch(Exception e)
  { System.err.println("Exception");
    System.err.println(e);
  }
}
```

Count Stateful Session Bean



Stateful Session Bean Lifecycle Diagram



Count Stateful Session Bean: Home Interface

```
// CountHome.java
package count.statefulSession;

import javax.ejb.EJBHome;
import java.rmi.RemoteException;
import javax.ejb.CreateException;

public interface CountHome extends EJBHome
{
  public Count create(int initialSum)
              throws RemoteException, CreateException;
}
```

Count Stateful Session Bean: Remote Interface

```
// Count.java
package count.statefulSession;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Count extends EJBObject
{
  public int  getSum() throws RemoteException;
  public void setSum(int val) throws RemoteException;
  public int  increment() throws RemoteException;
}
```

Count Stateful Session Bean: EJB Class (1 of 2)

```
// CountBean.java
package count.statefulSession;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.ejb.EJBException;
import java.rmi.RemoteException;

public class CountBean extends Object implements SessionBean
{
  private int sum;  // state variable

  // no arg contructor
  public CountBean() {}

  public void ejbCreate (int initialSum )
          throws  javax.ejb.CreateException
  { sum = initialSum;
  }
```

Count Stateful Session Bean: EJB Class (2 of 2)

```
  public void ejbActivate() {}
  public void ejbPassivate() {}
  public void ejbRemove() {}
  public void setSessionContext(SessionContext ctx) {}

  // get sum
  public int getSum()
  { return sum;
  }

  // set sum
  public void setSum(int val)
  { sum = val;
  }

  public int increment ( )
  { sum++;
    return sum;
  }
}
```

Count Stateful Session Bean: Deployment Descriptor (1 of 2)

```
<?xml version="1.0" encoding="Cp1252"?>

<ejb-jar>
  <description>no description</description>
  <display-name>Ejb2</display-name>
  <enterprise-beans>
    <session>
      <description>no description</description>
      <display-name>Count</display-name>
      <ejb-name>Count</ejb-name>
      <home>count.statefulSession.CountHome</home>
      <remote>count.statefulSession.Count</remote>
      <ejb-class>count.statefulSession.CountBean</ejb-class>
      <session-type>Stateful</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
```

Count Stateful Session Bean: Deployment Descriptor (2 of 2)

```
  <assembly-descriptor>
    <container-transactions>
      <container-transaction>
        <method>
          <ejb-name>Count</ejb-name>
          <method-intf>count.statefulSession.Count</method-intf>
          <method-name>increment</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
      </container-transaction>

              ...

    </container-transactions>
  </assembly-descriptor>
</ejb-jar>
```

Count Stateful Session Bean: Client (1 of 3)

```
// CountClient.java
package count.statefulSession;

import count.statefulSession.Count;      // Count EJB
import count.statefulSession.CountHome;  // Count EJB Home

import java.rmi.*;
import java.io.*;
import java.util.*;

import javax.rmi.*;
import javax.naming.*;
import javax.ejb.*;

public class CountClient
{
  public static void main(String args[])
  {
    CountHome countHome;
```

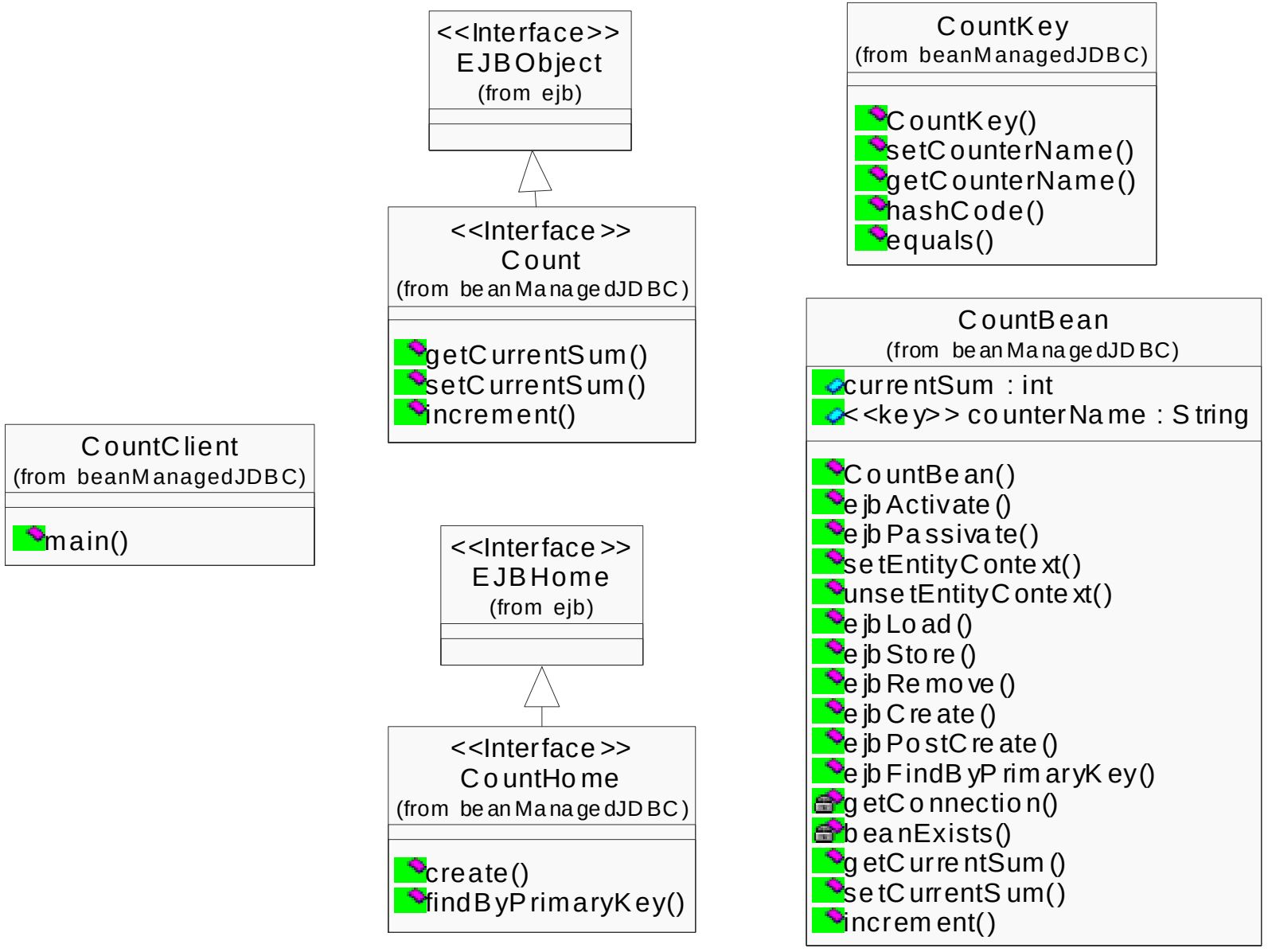
Count Stateful Session Bean: Client (2 of 3)

```
try
{
  // Get the initial JNDI context
  System.out.println("Getting initial context");
  InitialContext context = new InitialContext();

  // Find the home interface and narrow to CountHome
  java.lang.String jndiName =
         new String("statefulSessionCountHome");
  System.out.println("Finding EJB home using " + jndiName);
  Object object = context.lookup(jndiName);
  countHome =
   (CountHome)PortableRemoteObject.narrow(object, CountHome.class);

  // Create EJB with intial sum of 0
  System.out.println("Creating EJB");
  Count count = countHome.create(0);
```

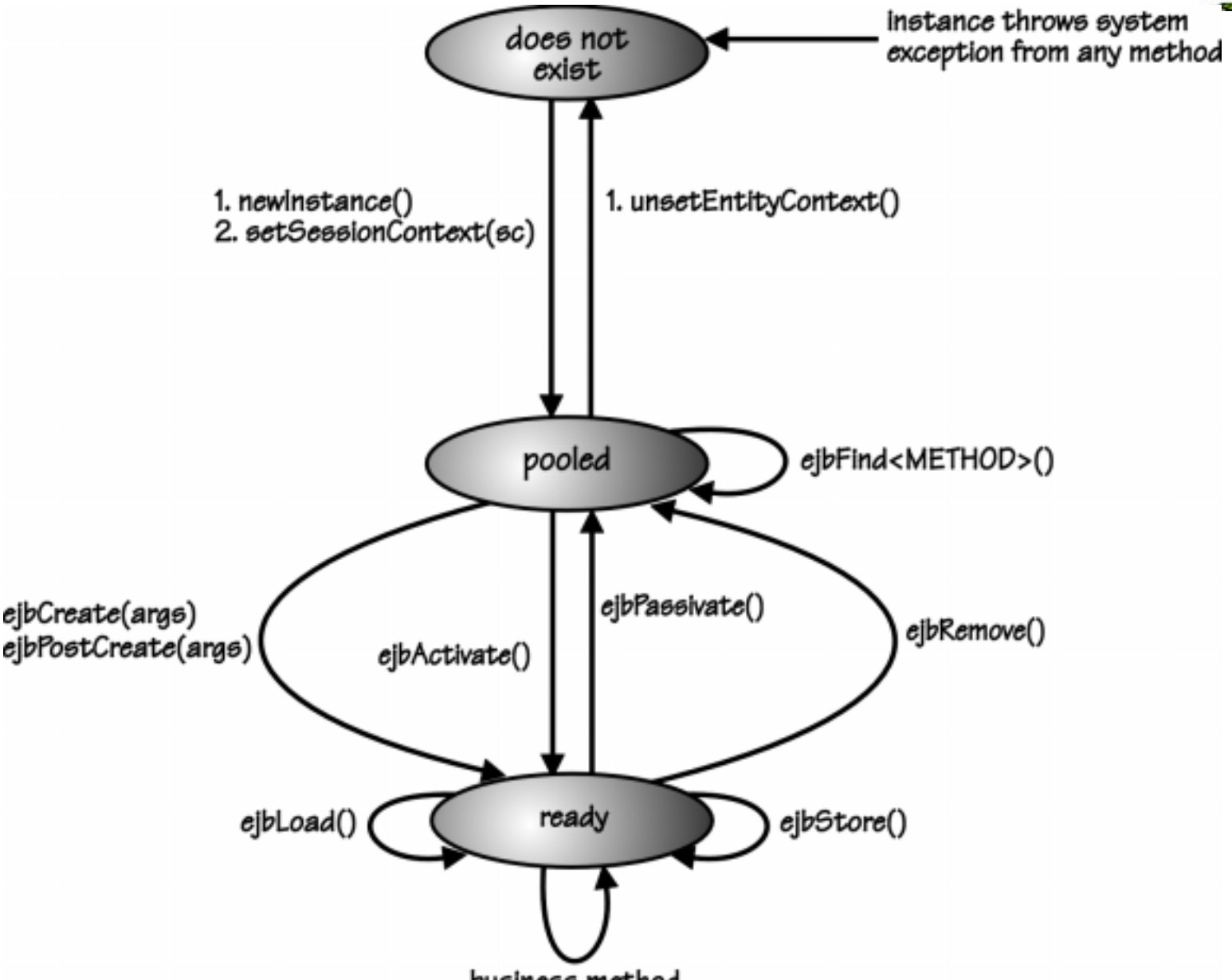
Count Stateful Session Bean: Client (3 of 3)

```
      // Calculate Start time
      long startTime = System.currentTimeMillis();

      // Increment 1000 times
      System.out.println("Incrementing");
      for (int i = 0 ; i < 1000 ; i++ )
      { count.increment();
      }

      // Calculate stop time; print out statistics
      long stopTime = System.currentTimeMillis();
      System.out.println("Avg Ping = "
                         + ((stopTime - startTime)/1000f) + " msecs");
      System.out.println("Sum = " + count.getSum());
    } catch(Exception e)
    { System.err.println("Exception");
      System.err.println(e);
    }
  }
}
```

Count Bean-Managed Entity Bean



Entity Bean Lifecycle Diagram



Count Bean-Managed Entity: Home Interface

```
// CountHome.java
package count.beanManagedJDBC;
import count.beanManagedJDBC.Count;

import java.rmi.*;
import javax.ejb.*;

public interface CountHome extends EJBHome
{
  public Count create(String name, int sum)
         throws RemoteException, CreateException;

  public Count findByPrimaryKey(CountKey primaryKey)
         throws RemoteException, FinderException;
}
```

Count Bean-Managed Entity: Remote Interface

```
// Count.java
package count.beanManagedJDBC;

import javax.ejb.*;
import java.rmi.*;

public interface Count extends EJBObject
{
  public int  getCurrentSum() throws RemoteException;
  public void setCurrentSum(int val) throws RemoteException;
  public int  increment() throws RemoteException;
}
```

Count Bean-Managed Entity: Key Class (1 of 2)

```
// CountKey.java
package count.beanManagedJDBC;

import count.beanManagedJDBC.CountBean;

public class CountKey implements java.io.Serializable
{
  public String counterName;

  public CountKey()
  { counterName = "";
  }

  public CountKey(CountBean bean)
  { counterName = bean.counterName;
  }

  public CountKey(String name )
  { counterName = name;
  }
```

Count Bean-Managed Entity: Key Class (2 of 2)

```
  public void setCounterName(String name )
  { counterName = name;
  }

  public String getCounterName( )
  {return counterName;
  }

   public int hashCode ( )
  { return counterName.hashCode();
  }

  // Required by IBM implementation
  public boolean equals ( Object obj )
  { return ((obj instanceof count.beanManagedJDBC.CountKey) &&
           ((((CountKey)obj).counterName).equals(counterName)));
  }
}
```

Count Bean-Managed Entity: EJB Class (1 of 11)

```
// CountBean.java

package count.beanManagedJDBC;

import count.beanManagedJDBC.CountKey;

import java.util.Properties;
import java.io.*;
import java.sql.*;

import javax.naming.*;
import javax.ejb.*;
import javax.sql.*;

public class CountBean implements EntityBean
{
  public String counterName; /* key */
  public int currentSum;

  private EntityContext context;
  private Connection connection;
```

Count Bean-Managed Entity: EJB Class (2 of 11)

```
// no arg contructor
public CountBean()
{ counterName = null;
  currentSum = 0;
}

public void ejbActivate()
{
  CountKey key = (CountKey)context.getPrimaryKey();
  counterName = key.getCounterName();
}

public void ejbPassivate() {}
public void setEntityContext(EntityContext ctx) {context = ctx;}
public void unsetEntityContext() {context = null;}
```

Count Bean-Managed Entity: EJB Class (3 of 11)

```
public void ejbLoad()
{ try
  {
    connection = getConnection();
    PreparedStatement ps = connection.prepareStatement(
         "SELECT currentSum FROM CountBean WHERE counterName = ?");
    ps.setString(1, counterName);
    ResultSet rs = ps.executeQuery();
    rs.next();
    currentSum = rs.getInt(1);
    ps.close();
  } catch (Exception e)
  { throw new EJBException("Error loading state for "
                           + counterName + ", " + e.getMessage());
  } finally
  { try
    { connection.close();
    } catch (Exception e)
    { throw new EJBException(e);
    }
  }
}
```

Count Bean-Managed Entity: EJB Class (4 of 11)

```
public void ejbStore()
{
  try
  {
    connection = getConnection();
    PreparedStatement ps = connection.prepareStatement(
       "UPDATE CountBean SET currentSum = ? WHERE counterName = ?");
    ps.setInt(1, currentSum);
    ps.setString(2, counterName);
    ps.executeUpdate();
    ps.close();
  } catch (Exception e)
  { throw new EJBException("Error storing state for "
                           + counterName + ", " + e.getMessage());
  } finally
  { try
    { connection.close();
    } catch (Exception e)
    { throw new EJBException(e);
    }
  }
}
```

Count Bean-Managed Entity: EJB Class (5 of 11)

```
public void ejbRemove ( ) throws RemoveException
{
  try
  {
    connection = getConnection();
    PreparedStatement ps = connection.prepareStatement(
         "DELETE FROM CountBean WHERE counterName = ?");
    ps.setString(1, counterName);
    int resultCount = ps.executeUpdate();
    ps.close();
  } catch (Exception e)
  {
    throw new EJBException("Error removing bean "
                           + counterName + ", " + e.getMessage());
  } finally
  { try
    { connection.close();
    } catch (Exception e)
    { throw new EJBException(e);
    }
  }
}
```

Count Bean-Managed Entity: EJB Class (6 of 11)

```
public CountKey ejbCreate( String name, int initialSum)
       throws CreateException
{ // Set the initial instance data
  counterName = name;
  currentSum = initialSum;

  try
  { connection = getConnection();

    if (beanExists(counterName))
    { throw new DuplicateKeyException(
                      "Bean " + counterName + " already exists.");
    }
```

Count Bean-Managed Entity: EJB Class (7 of 11)

```
      PreparedStatement ps = connection.prepareStatement(
      "INSERT INTO CountBean (counterName, currentSum) values (?, ?)");
      ps.setString(1, counterName);
      ps.setInt(2, currentSum);
      ps.executeUpdate();
      ps.close();
    } catch (Exception e)
    { throw new EJBException("Error creating "
                             + counterName + ", " + e.getMessage());
    } finally
    { try
      { connection.close();
      } catch (Exception e)
      { throw new EJBException(e);
      }
    }
    return new CountKey(counterName);
}

public void ejbPostCreate(String name, int initialSum)
       throws CreateException
{
}
```

Count Bean-Managed Entity: EJB Class (8 of 11)

```
public CountKey ejbFindByPrimaryKey(CountKey primaryKey )
       throws FinderException
{ counterName = primaryKey.getCounterName();

  try
  { connection = getConnection();
    if (beanExists(counterName))
    {  return primaryKey;
    }
    throw new FinderException("Error finding " + counterName);
  } catch (Exception e)
  {
    throw new EJBException("Error finding "
                           + counterName + ", " + e.getMessage());
  } finally
  { try
    { connection.close();
    } catch (Exception e)
    { throw new EJBException(e);
    }
  }
}
```

Count Bean-Managed Entity: EJB Class (9 of 11)

```
// private methods
private Connection getConnection () throws SQLException
{
  Connection connection;

  Properties environmentProperties = context.getEnvironment();

  // get the JNDI name of the data source
  String dataSourceName =
      environmentProperties.getProperty("DataSourceName");

  /* as specified in the XML/props */
  if (dataSourceName == null || dataSourceName.equals(""))
  { throw new EJBException(
          "Environment property DataSourceName not found.");
  }

// look up the data source
  try
  { InitialContext context = new InitialContext();
    DataSource dataSource =
         (DataSource) context.lookup(dataSourceName);
```

Count Bean-Managed Entity: EJB Class (10 of 11)

```
      connection = dataSource.getConnection();
    } catch (NamingException ne)
    { throw new EJBException("DataSourceName not found");
    }
    return connection;
  }

private boolean beanExists(String key) throws SQLException
  { PreparedStatement ps = connection.prepareStatement(
          "SELECT currentSum FROM CountBean WHERE counterName = ?");
    ps.setString(1, counterName);
    ResultSet rs = ps.executeQuery();
    if ( !rs.next() )
    { ps.close();
      return false;
    } else
    { ps.close();
      return true;
    }
  }
```

Count Bean-Managed Entity: EJB Class (11 of 11)

```
  // Business Methods

  // get sum
  public int getCurrentSum()
  {
    return currentSum;
  }

  // set sum
  public void setCurrentSum(int val)
  {
    currentSum = val;
  }

  public int increment()
  {
    currentSum++;
    return currentSum;
  }
}
```

Count Container-Managed Entity: EJB Class (1 of 3)

```
// CountBean.java
package count.containerManagedJDBC;

import count.beanManagedJDBC.CountKey;
import javax.ejb.*;

public class CountBean implements EntityBean
{ public String counterName; /* key */
  public int currentSum;
  private EntityContext context;

  // no arg contructor
  public CountBean()
  { counterName = null;
    currentSum = 0;
  }
```

Count Container-Managed Entity: EJB Class (2 of 3)

```
public void ejbActivate() {}
public void ejbPassivate() {}
public void setEntityContext(EntityContext ctx) {context = ctx;}
public void unsetEntityContext() {context = null;}
public void ejbLoad() {}
public void ejbStore() {}
public void ejbRemove () {}

public CountKey ejbCreate( String name, int initialSum)
       throws CreateException
{ // Set the initial instance data
  counterName = name;
  currentSum = initialSum;
  return new CountKey(counterName);
}

public void ejbPostCreate(String name, int initialSum)
       throws CreateException  { }

public CountKey ejbFindByPrimaryKey(CountKey primaryKey )
       throws FinderException
{ counterName = primaryKey.getCounterName();
}
```

Count Container-Managed Entity: EJB Class (3 of 3)

```
  // get sum
  public int getCurrentSum()
  {
    return currentSum;
  }

  // set sum
  public void setCurrentSum(int val)
  {
    currentSum = val;
  }

  public int increment()
  {
    currentSum++;
    return currentSum;
  }
}
```

Count Bean-Managed Entity: Deployment Descriptor (1 of 3)

```
<?xml version="1.0" encoding="Cp1252"?>

<!DOCTYPE ejb-jar>

<ejb-jar>
  <description>no description</description>
  <display-name>Ejb1</display-name>
  <enterprise-beans>
    <entity>
      <description>no description</description>
      <display-name>Count</display-name>
      <ejb-name>Count</ejb-name>
      <home>count.beanManagedJDBC.CountHome</home>
      <remote>count.beanManagedJDBC.Count</remote>
      <ejb-class>count.beanManagedJDBC.CountBean</ejb-class>
      <persistence-type>Bean</persistence-type>
      <primkey-class>count.beanManagedJDBC.CountKey</primkey-class>
      <reentrant>false</reentrant>
```

Count Bean-Managed Entity: Deployment Descriptor (2 of 3)

```
    <env-props>
      <env-prop>
        <env-prop-name>DataSourceName</env-prop-name>
        <env-prop-value>jdbc/odbc</env-prop-value>
      </env-prop>
    </env-props>
    <resource-refs>
      <resource-ref>
        <res-ref-name>DataSourceName</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
      </resource-ref>
    </resource-refs>
  </entity>
</enterprise-beans>
```

Count Bean-Managed Entity: Deployment Descriptor (1 of 3)

```
  <assembly-descriptor>
    <container-transactions>
      <container-transaction>
        <method>
          <ejb-name>Count</ejb-name>
          <method-intf>int</method-intf>
          <method-name>increment</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
      </container-transaction>

       ....

  </assembly-descriptor>
</ejb-jar>
```

Count Bean-Managed Entity: Client (1 of 4)

```
// CountClient.java
package count.beanManagedJDBC;

import count.beanManagedJDBC.Count;       // Count EJB
import count.beanManagedJDBC.CountKey;    // Count EJB Key
import count.beanManagedJDBC.CountHome;   // Count EJB Home

import java.rmi.*;
import java.io.*;
import java.util.*;
import javax.naming.*;
import javax.ejb.*;
import javax.rmi.*;

public class CountClient
{
  public static void main ( String args[] )
  {
    CountHome countHome;
    Count count = null;
```

Count Bean-Managed Entity: Client (2 of 4)

```
try
{
  if (args.length != 2)
  { System.out.println("Parameter error");
    System.out.println("Usage: CountClient cmd counter-name");
    System.out.println("       Where cmd = create, find, or remove");
    System.exit(1);
  }

  // Get the initial context
  System.out.println("Getting initial context");
  InitialContext context = new InitialContext();

  // Lookup the home interface and narrow to CountHome
  System.out.println("Finding EJB home");
  Object object = context.lookup("beanManagedJDBCCountHome");
  System.out.println("Narrowing EJB home");
  countHome =
    (CountHome)PortableRemoteObject.narrow(object, CountHome.class);
```

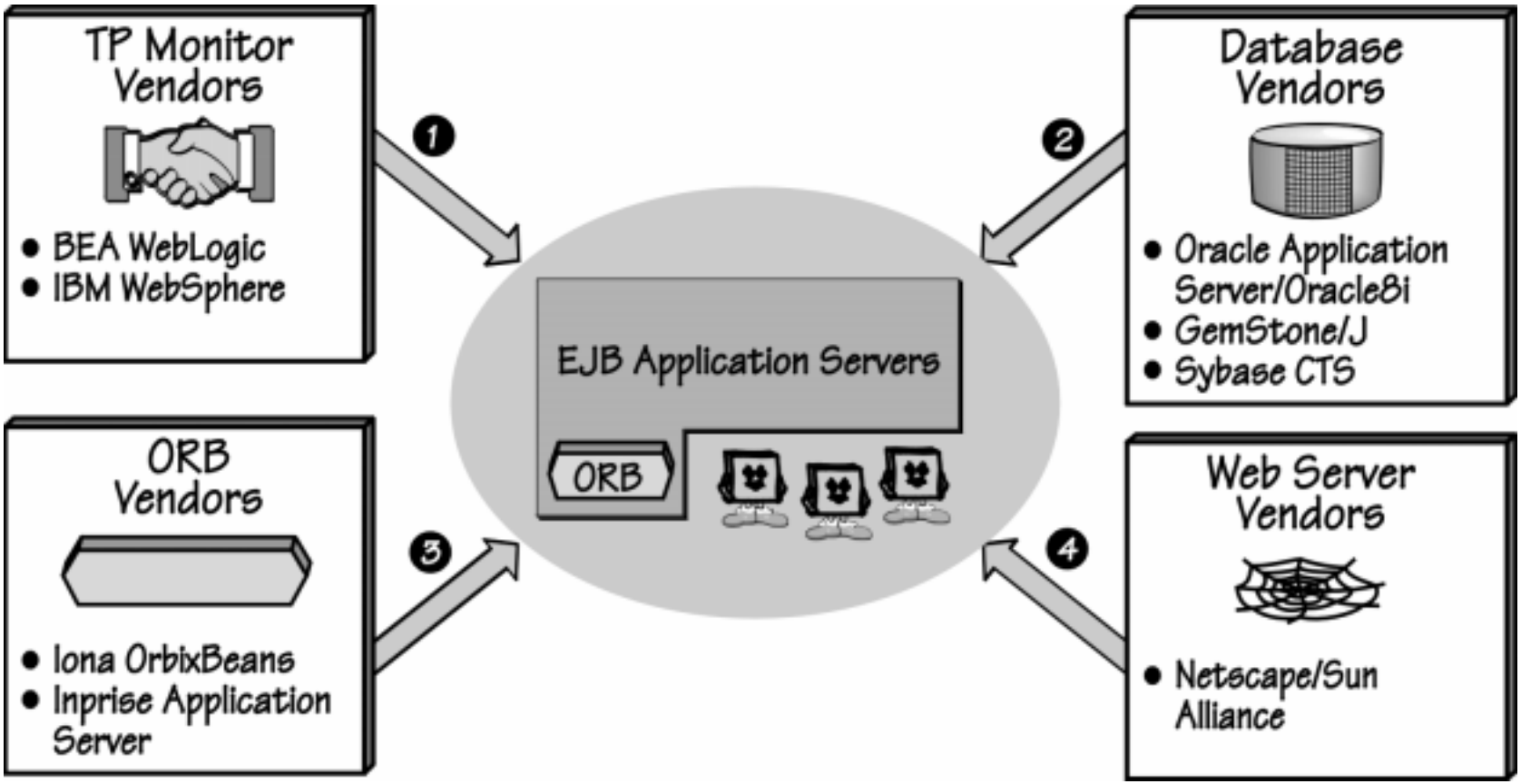
Count Bean-Managed Entity: Client (3 of 4)

```
// Remove an existing EJB
if (args[0].equals("remove"))
{ System.out.println("Removing " + args[1]);
  countHome.remove(new CountKey(args[1]));
  System.exit(0);  // no counting, just exit
}

// Create a new EJB
if (args[0].equals("create"))
{ System.out.println("Creating " + args[1]);
  count = countHome.create(args[1], 0);
} else

// Find an existing EJB
if (args[0].equals("find"))
{ System.out.println("Finding " + args[1]);
  CountKey key = new CountKey(args[1]);
  count = countHome.findByPrimaryKey(key);
} else
{ System.out.println("Invalid parameters");
  System.exit(1);
}
```

Count Bean-Managed Entity: Client (4 of 4)

```
      // Calculate Start time
      long startTime = System.currentTimeMillis();

      // Increment 1000 times
      System.out.println("Incrementing");
      for (int i = 0 ; i < 1000 ; i++ )
      { count.increment();
      }

      // Calculate stop time; print out statistics
      long stopTime = System.currentTimeMillis();
      System.out.println("Avg Ping = "
                         + ((stopTime - startTime)/1000f) + " msecs");
      System.out.println("Sum = " + count.getCurrentSum());
    } catch(Exception e)
    { System.err.println("Exception");
      e.printStackTrace();
    }
  }
}
```

Enterprise JavaBeans: Meet the Players



Enterprise JavaBeans: Meet the Players (cont.)

- ✓ See List of Players at java.sun.com/ejb
  - ~35 server vendors have announced products
  - ~30 tool vendors
  - Third party component market developing

For More Information...

- ✓ Sun EJB Home Page -- java.sun.com/ejb
  - EJB Specifications and Documentation
  - List of Vendors
  - White Papers
- ✓ San Jose State University program -- www.corbajava.engr.sjsu.edu
- ✓ Client/Server Survival Guide, Third Edition, Orfali, Harkey, and Edwards (Wiley, 1999)
- ✓ Client/Server Programming with Java and CORBA, Second Edition, Orfali andHarkey (Wiley, 1998)