CIS 386 Course

Advanced Enterprise Java Programming

Enterprise JavaBeans: BMP and CMP Entity Beans

René Doursat

Guest Lecturer

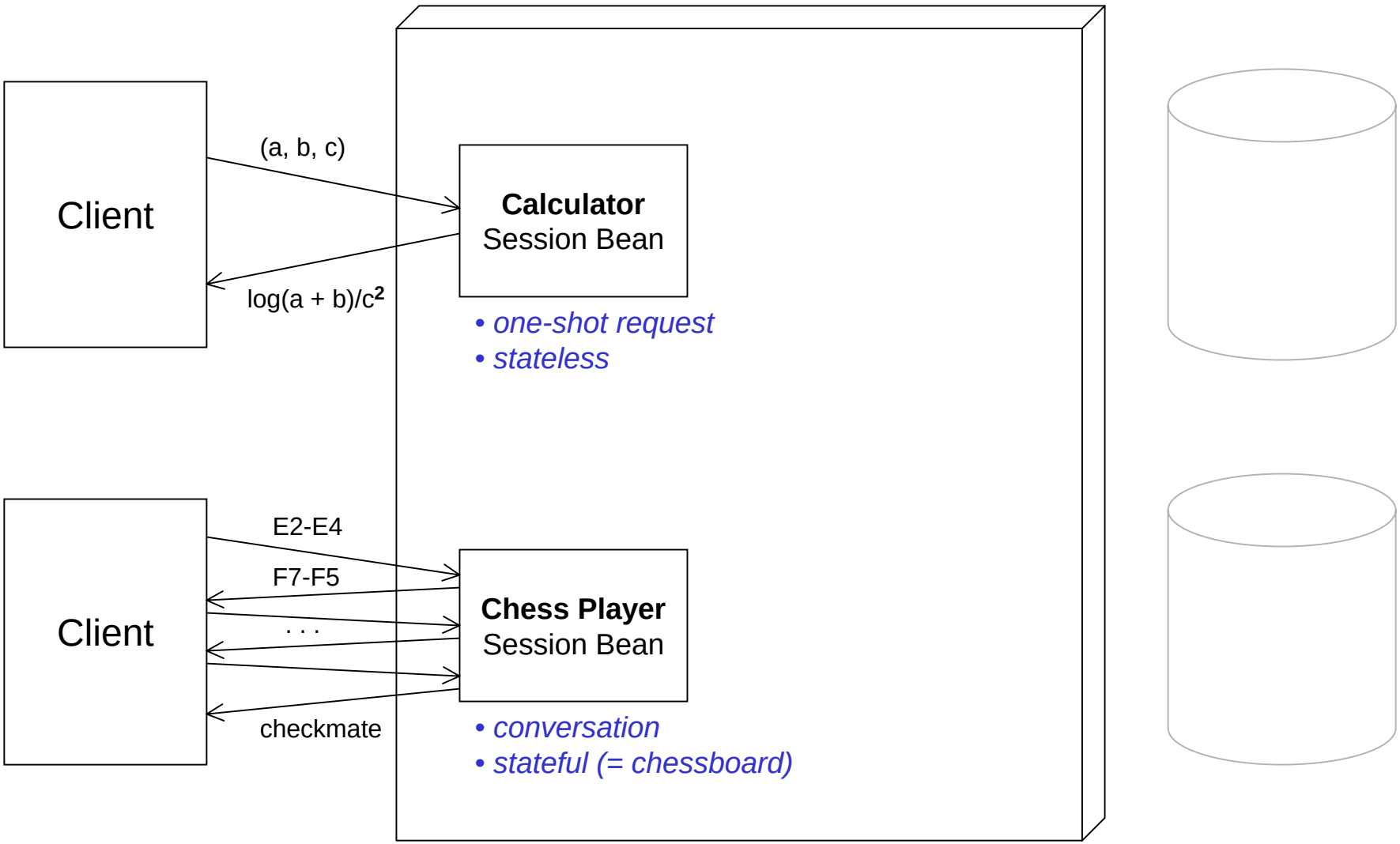
Golden Gate University, San Francisco

February 2003

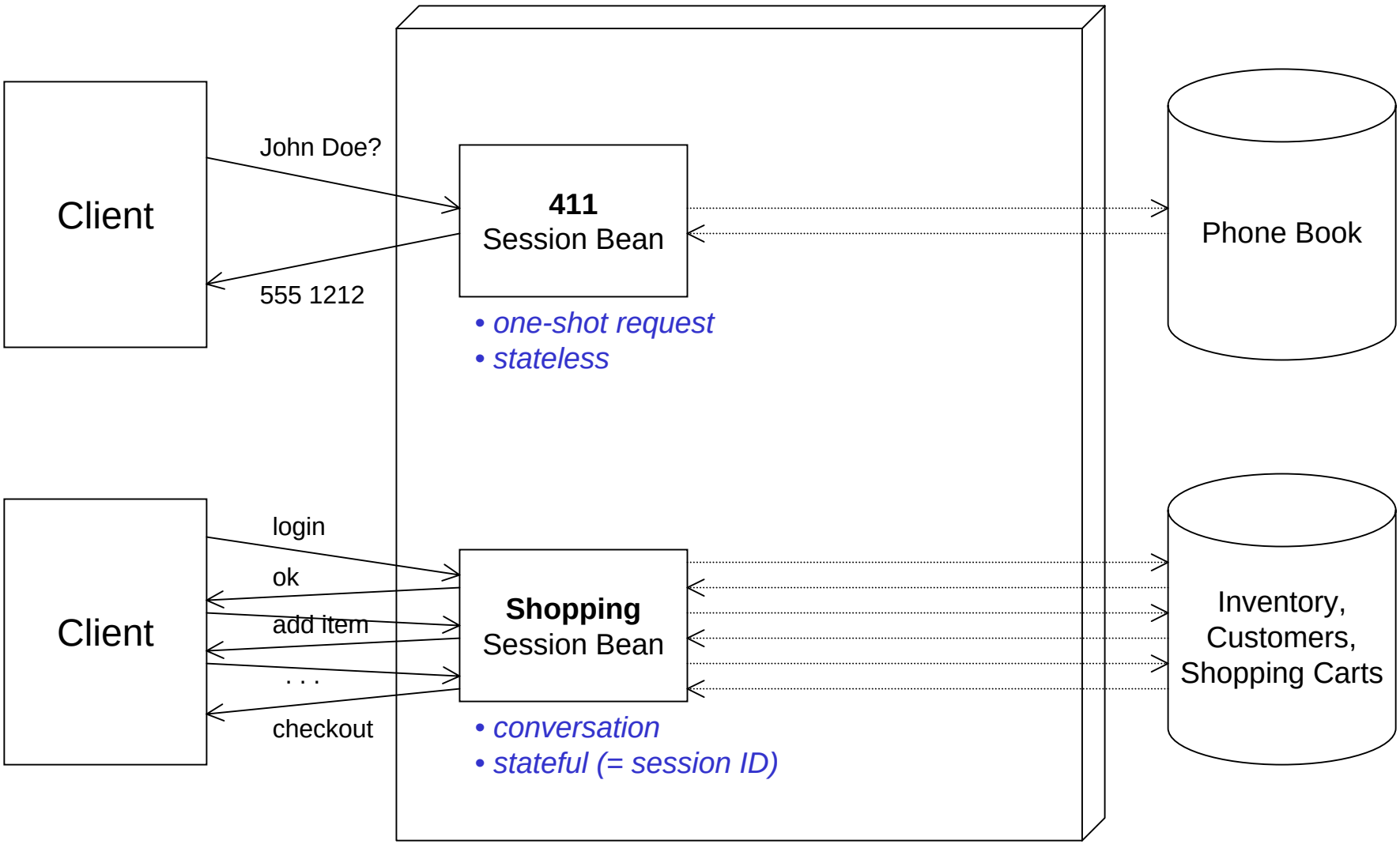
EJB Trail

- Session Beans
  - Stateless
  - Stateful
- Entity Beans
  - Bean-Managed Persistence (BMP)
  - Container-Managed Persistence (CMP)
- Message-Driven Beans

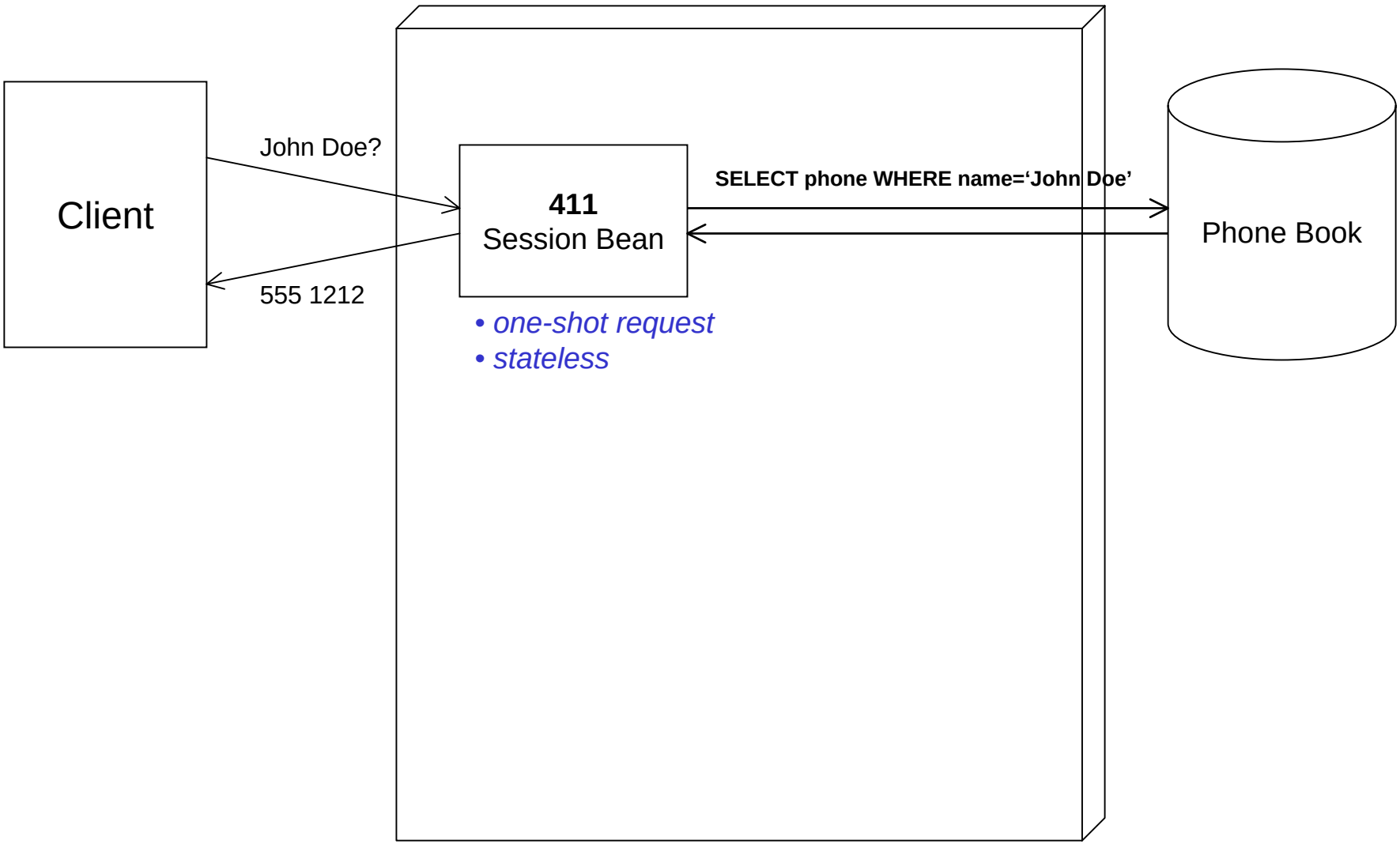
Algorithmic Computation: No External Data



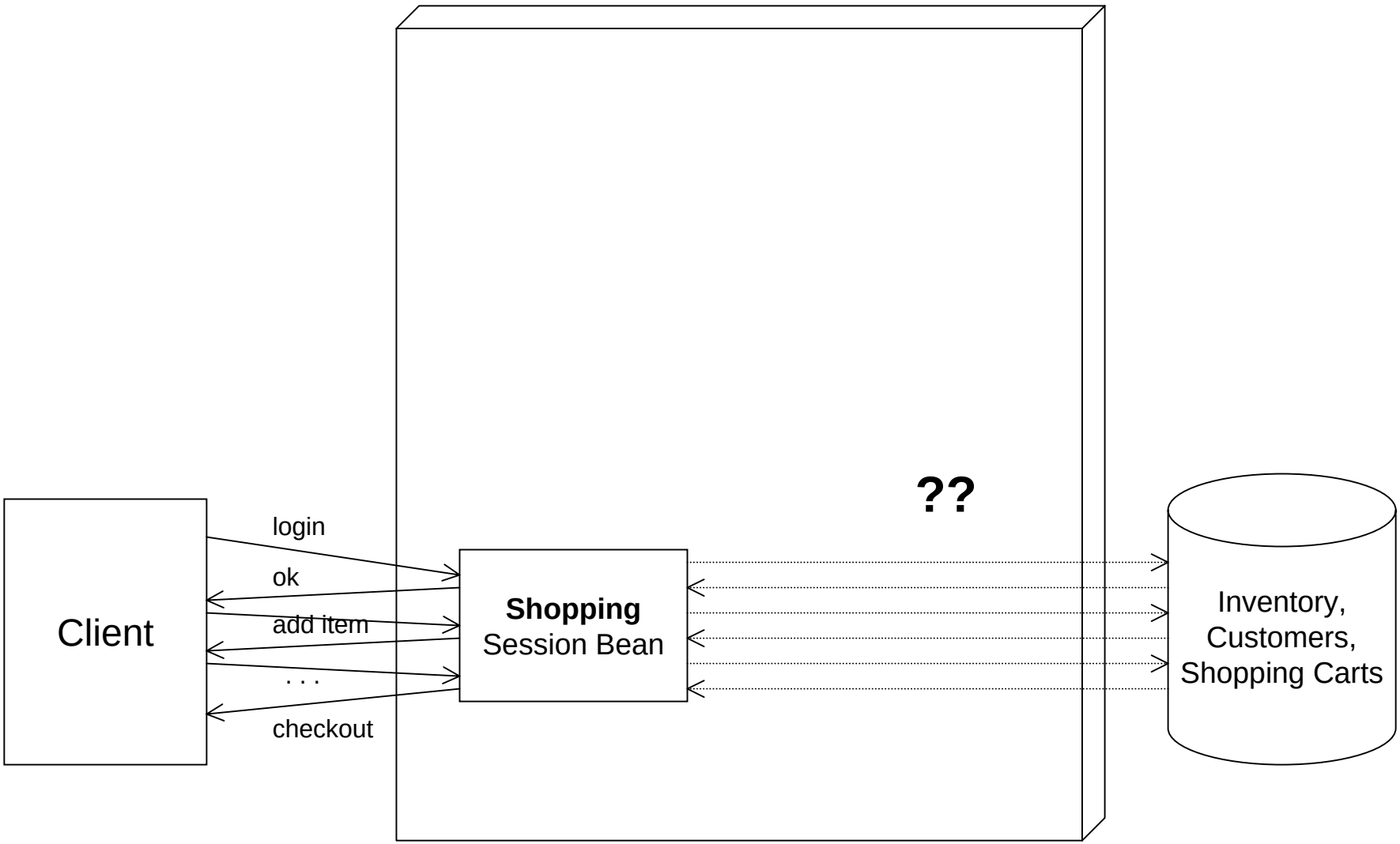
Business Computation: Manages External Data



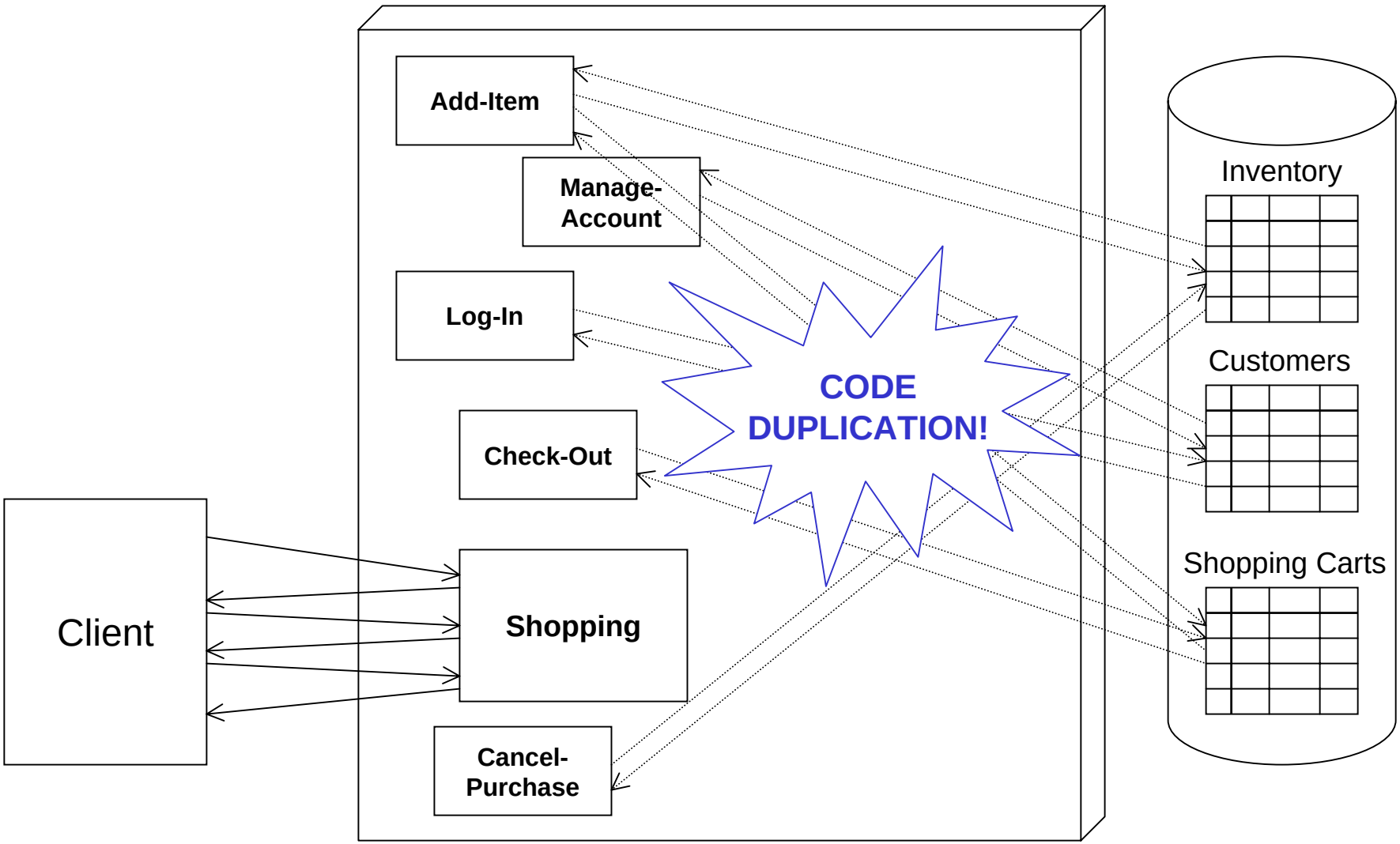
Simple Session: Direct Data Access Possible



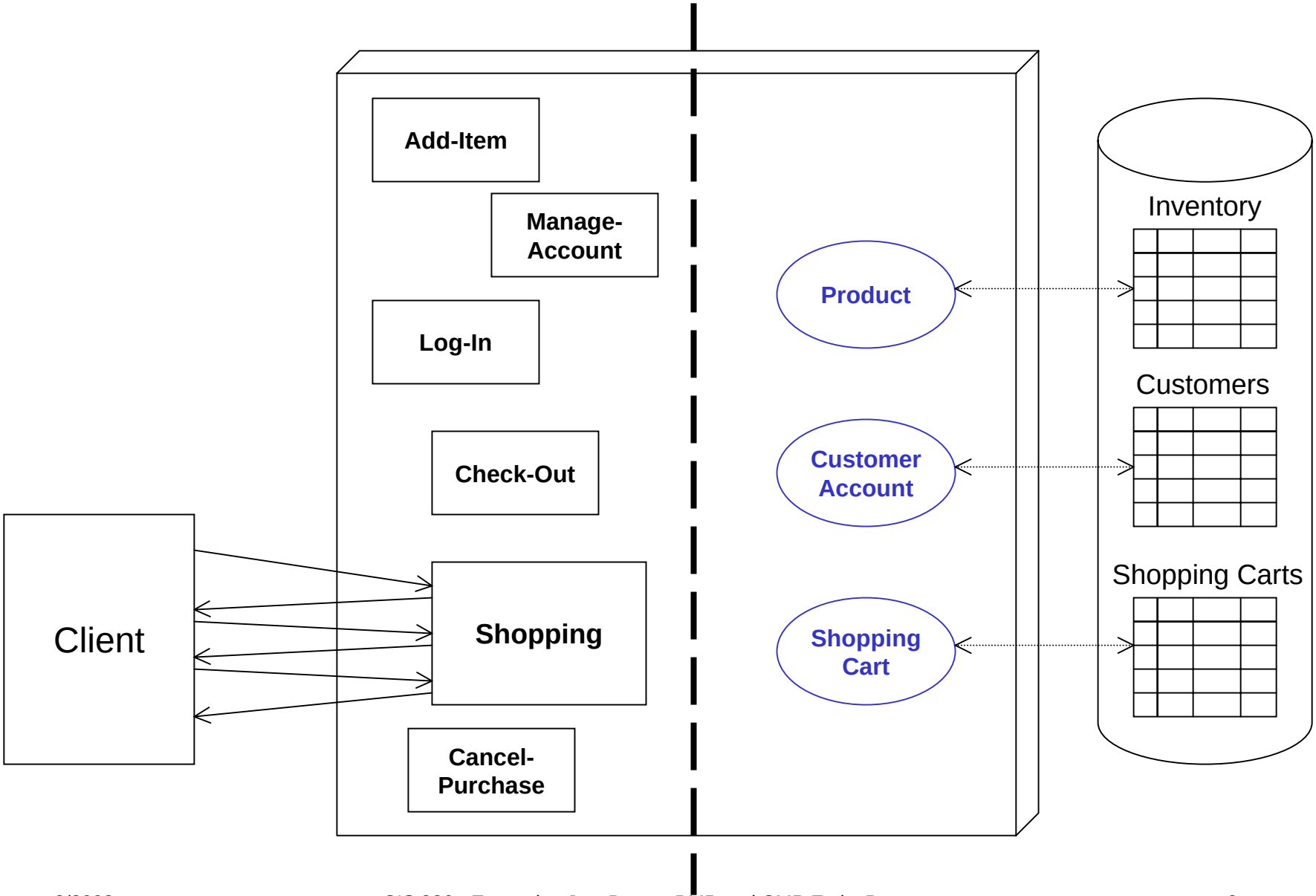
Complex Session: How To Access Data?



Complex Session: Direct Data Access Not Advised

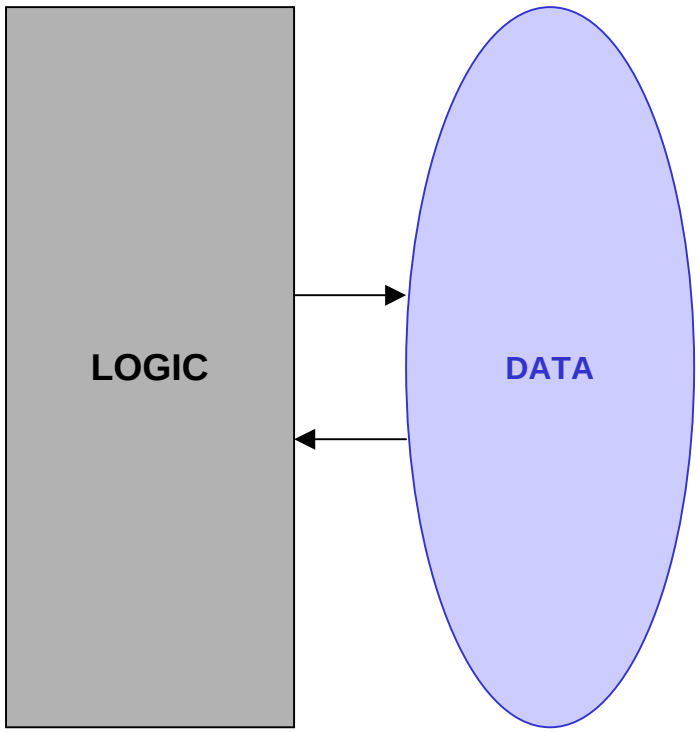


Complex Session: The Case For An Object Layer

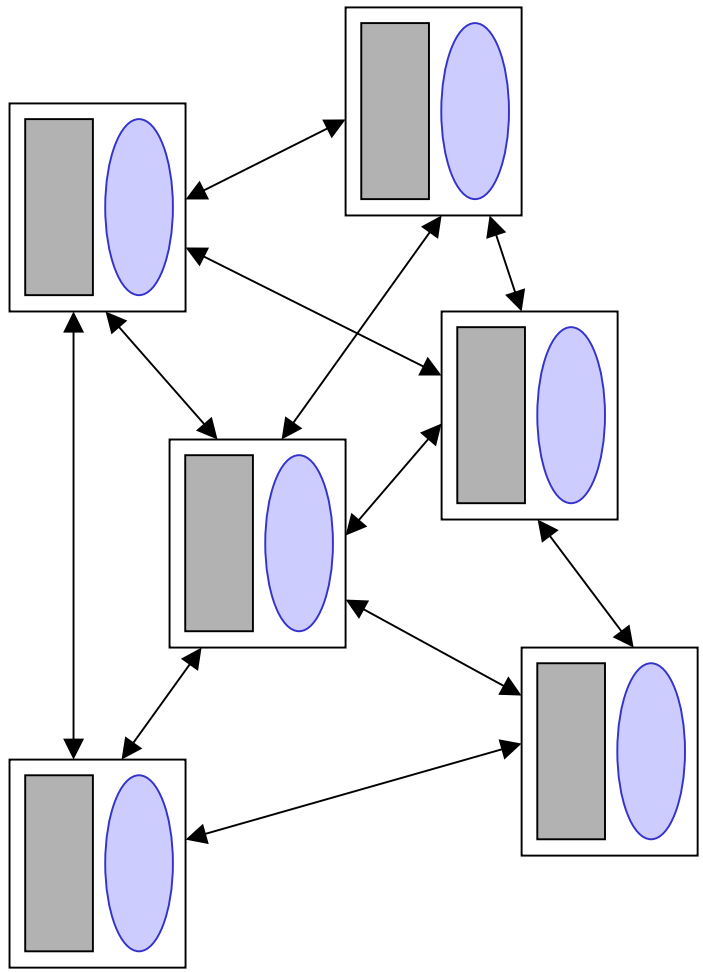


Logic vs. Data: Toward Encapsulation

procedural architecture

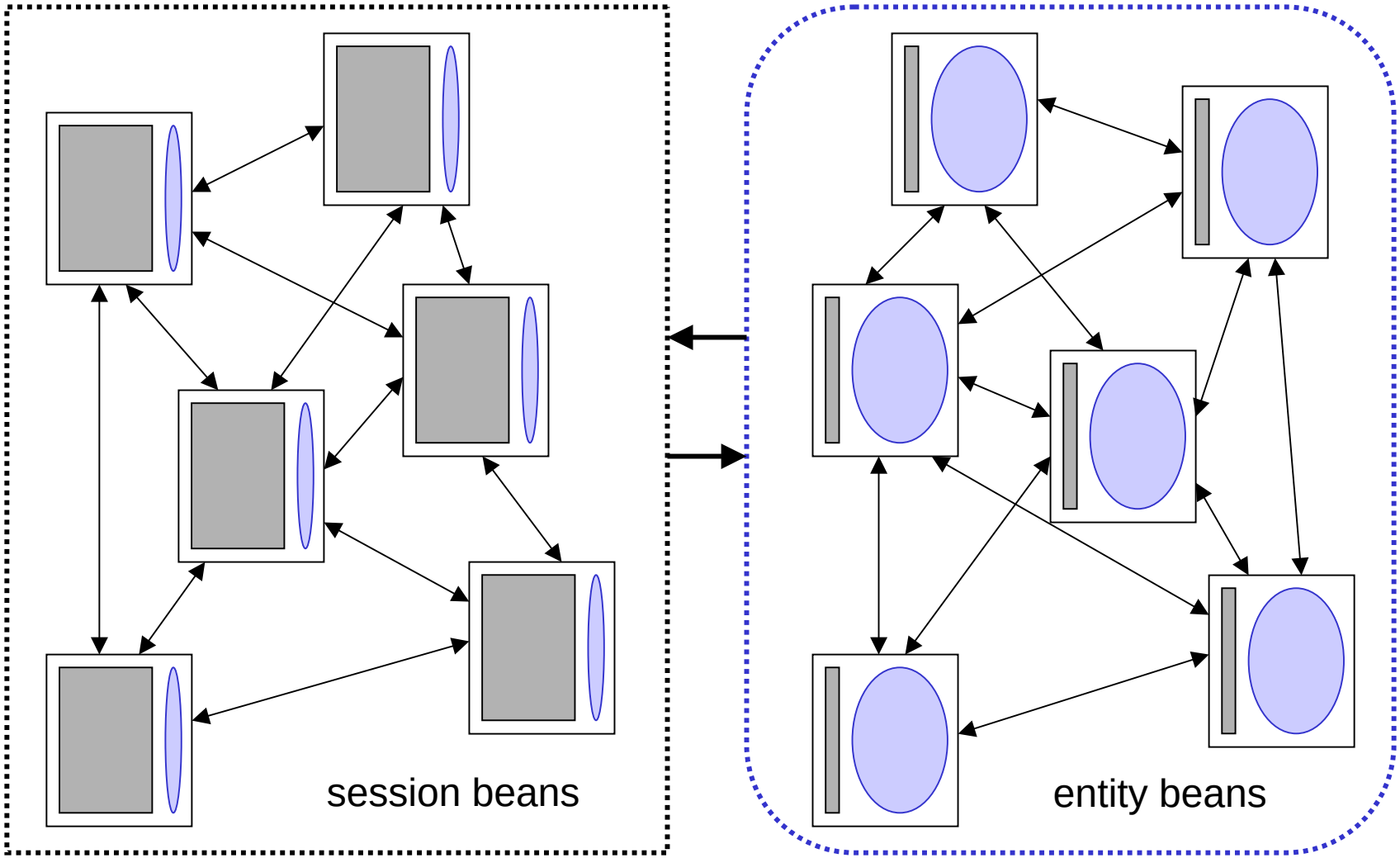


object-oriented architecture

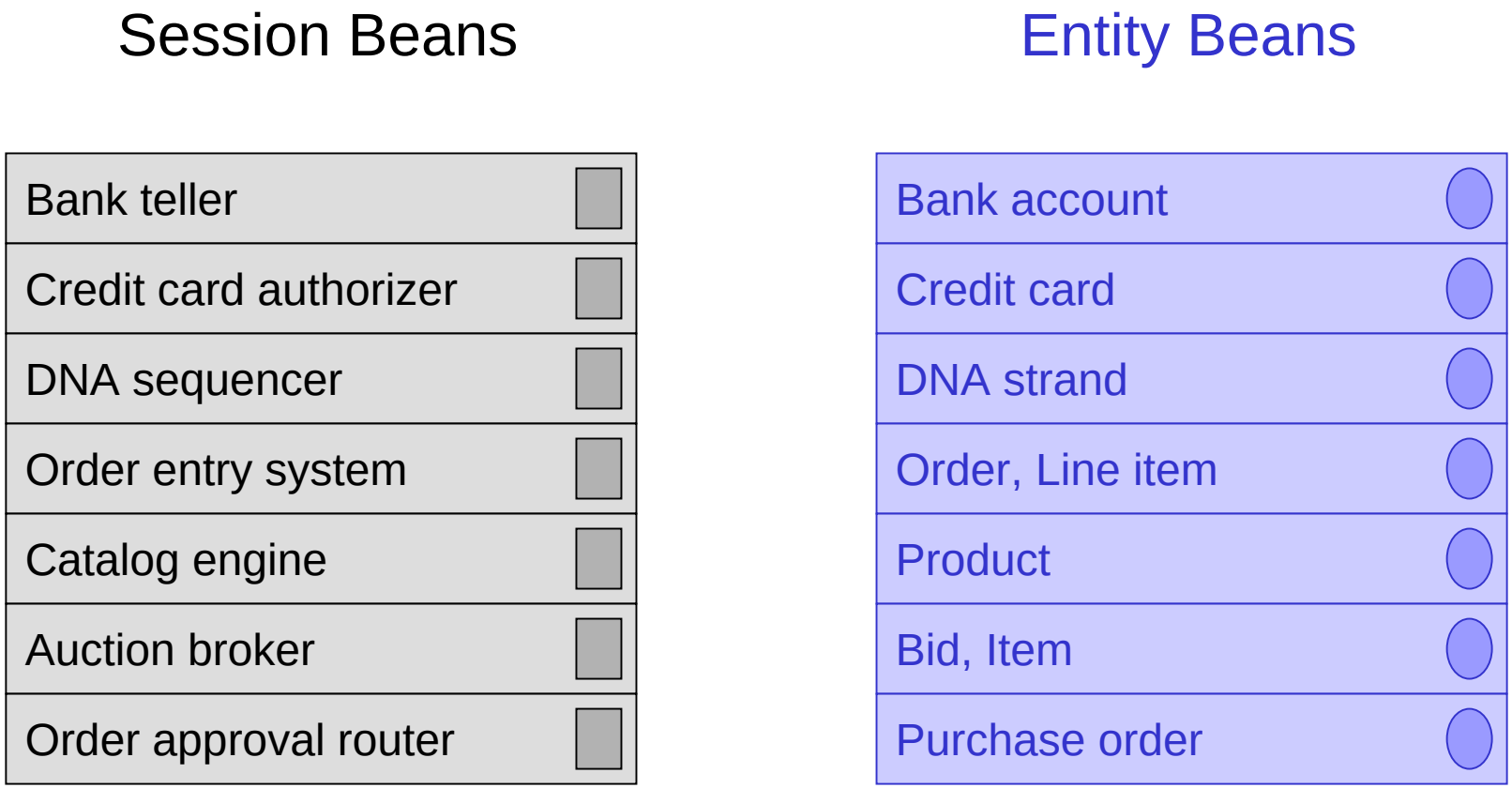


Logic vs. Data: Reintroducing Separation (at a higher level)

EJB architecture



Logic vs. Data: Examples

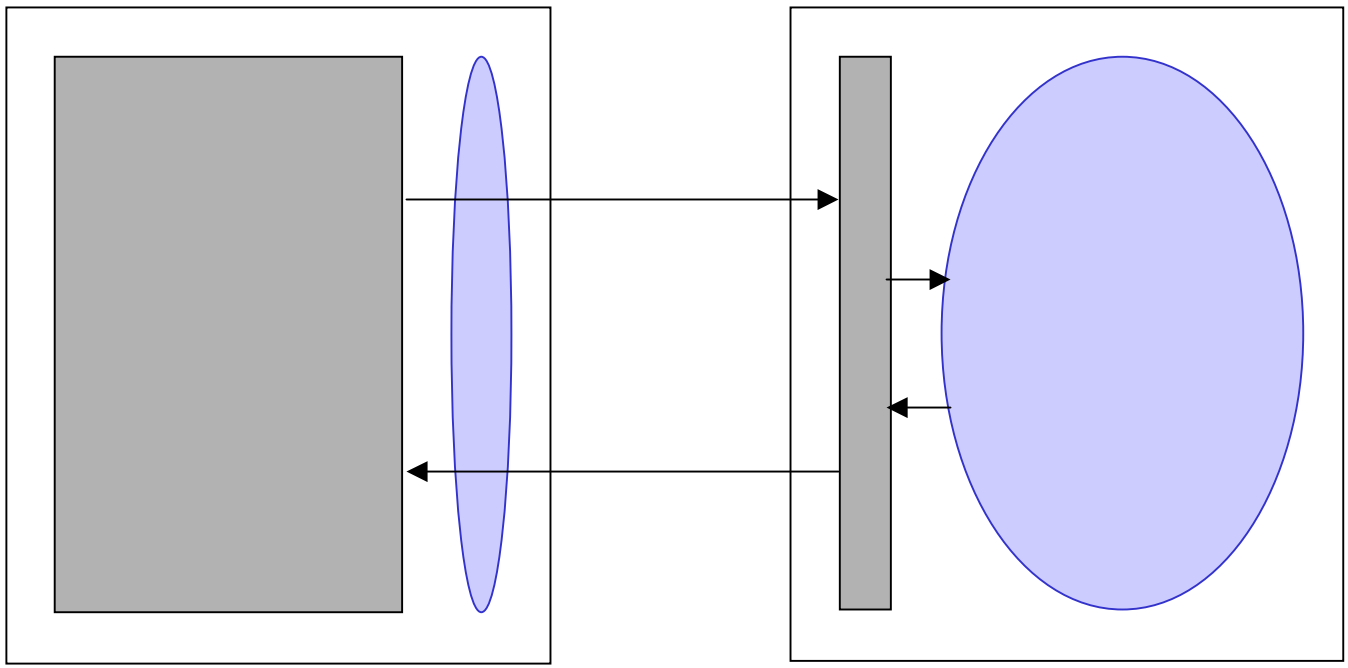


(after *Mastering EJBs*, Ed Roman, p231)

Logic vs. Data: Thick vs. Thin

Bank teller session bean

Bank account entity bean



complex bank operations

simple conversational state (customer ID)

elementary account ops (deposit, withdraw)

extensive account data

Definitions

*Given that **Enterprise Beans** are server-side components deployable in a distributed multi-tier environment:*

- **Session Beans** model the business's processes: they perform work for clients calling them and are short-lived.
- **Entity Beans** model the business's fundamental underlying data: they are persistent objects stored in permanent storage.
- **Message-Driven Beans** are a type of asynchronous Session Beans.

Feature Comparison

Session Beans

- execute on behalf of a single client
- can be transaction-aware
- do not represent directly shared data in the database (but may access and update such data)
- relatively short-lived
- removed when container crashes

Entity Beans

- allow shared access from multiple users
- is associated to a primary key (defining data *uniqueness*)
- provide an object view of data in the database
- can be long-lived
- survive container crash (automatic state reset)

Concepts of Persistence

- **Serialization**
  - bit-blobs written to storage media
  - not practical for selective search
- **Object-to-Relational Mapping**
  - data stored to and loaded from RDBMS
  - generally: 1 object class = 1 table, 1 object instance = 1 row, but not always
  - mapping scheme can be complex
- **ODBMS**
  - ideal for transparent object persistence, however not a wide-spread product

Bean Class Methods

SessionBean

- setSessionContext(ctx)
- ejbCreate . . . (. . .)
- ejbActivate()
- ejbPassivate()

- ejbRemove()

EntityBean

- setEntityContext(ctx)

- ejbFind . . . (. . .)

- ejbHome . . . (. . .)

- ejbCreate(. . .)

- ejbPostCreate(. . .)
- ejbActivate()
- ejbLoad()
- ejbStore()
- ejbPassivate()

- ejbRemove()
- unsetEntityContext()

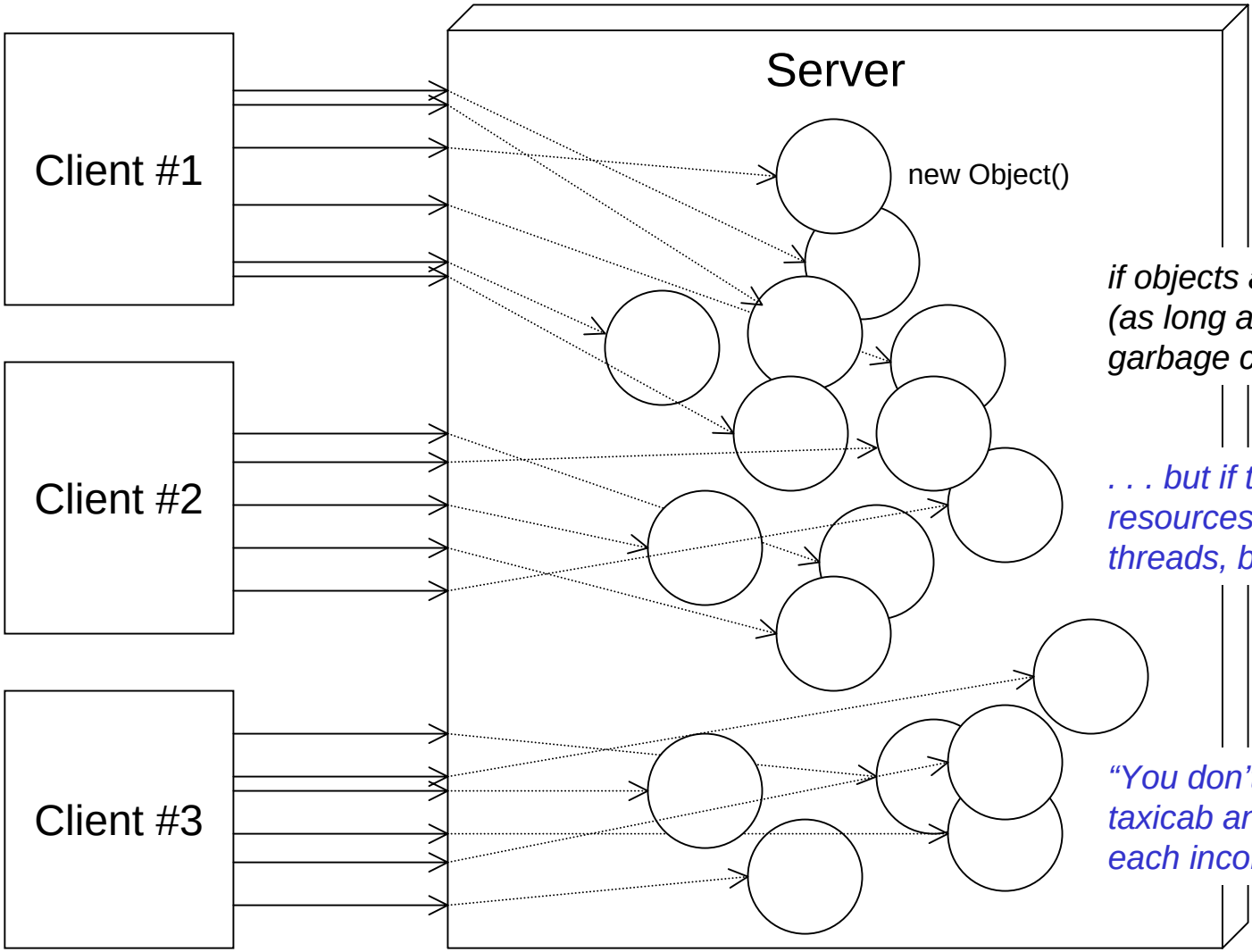
SessionBean & EntityBean

- *businessMethod*(. . .)
- get . . . (. . .)
- set . . . (. . .)

EJB Trail

- Session Beans
  - Stateless
  - Stateful
- Entity Beans
  - Bean-Managed Persistence (BMP)
  - Container-Managed Persistence (CMP)
- Message-Driven Beans

Without Object Pooling: Multiply And Waste

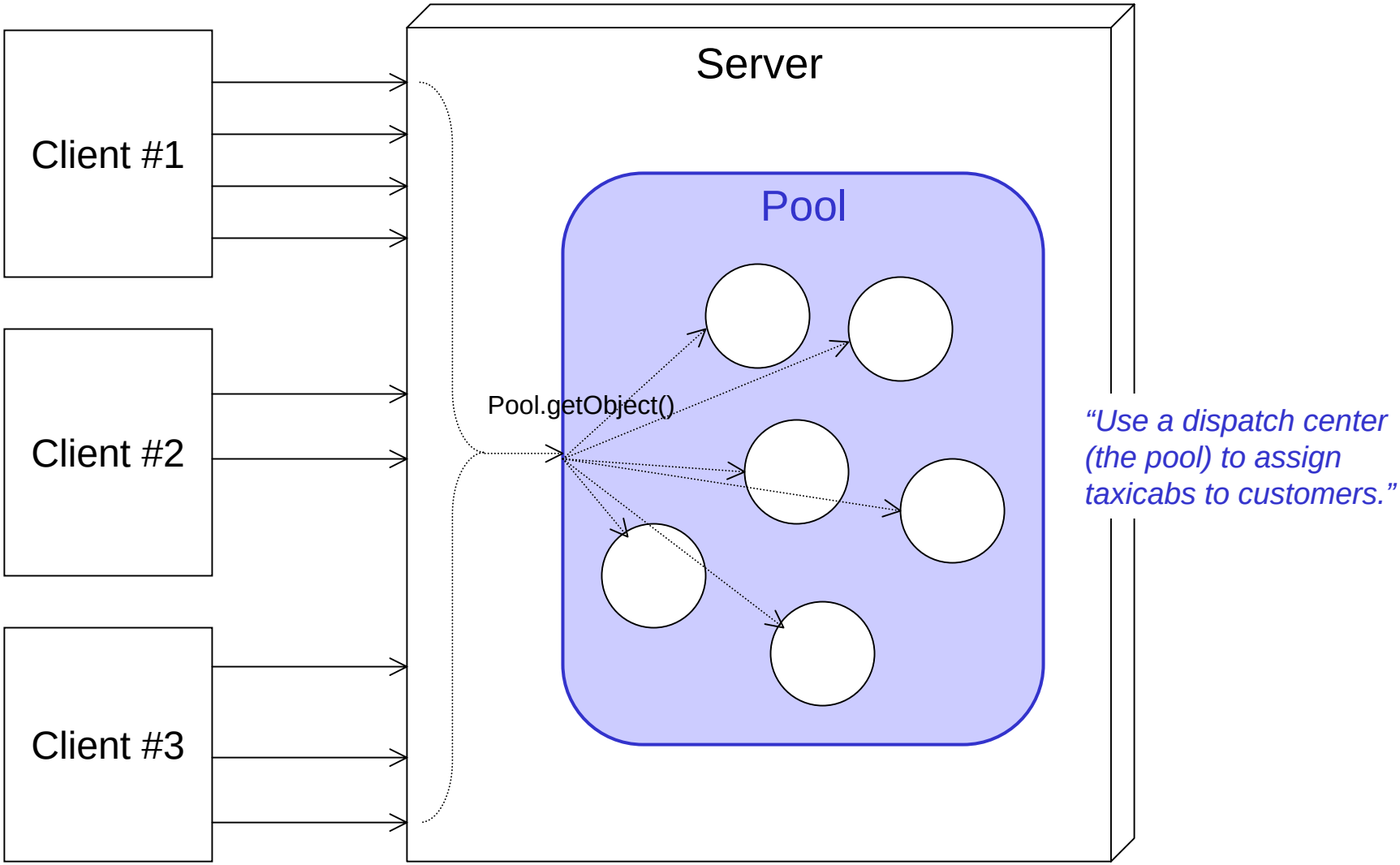


if objects are "lightweight": OK (as long as they are freed or garbage collected)

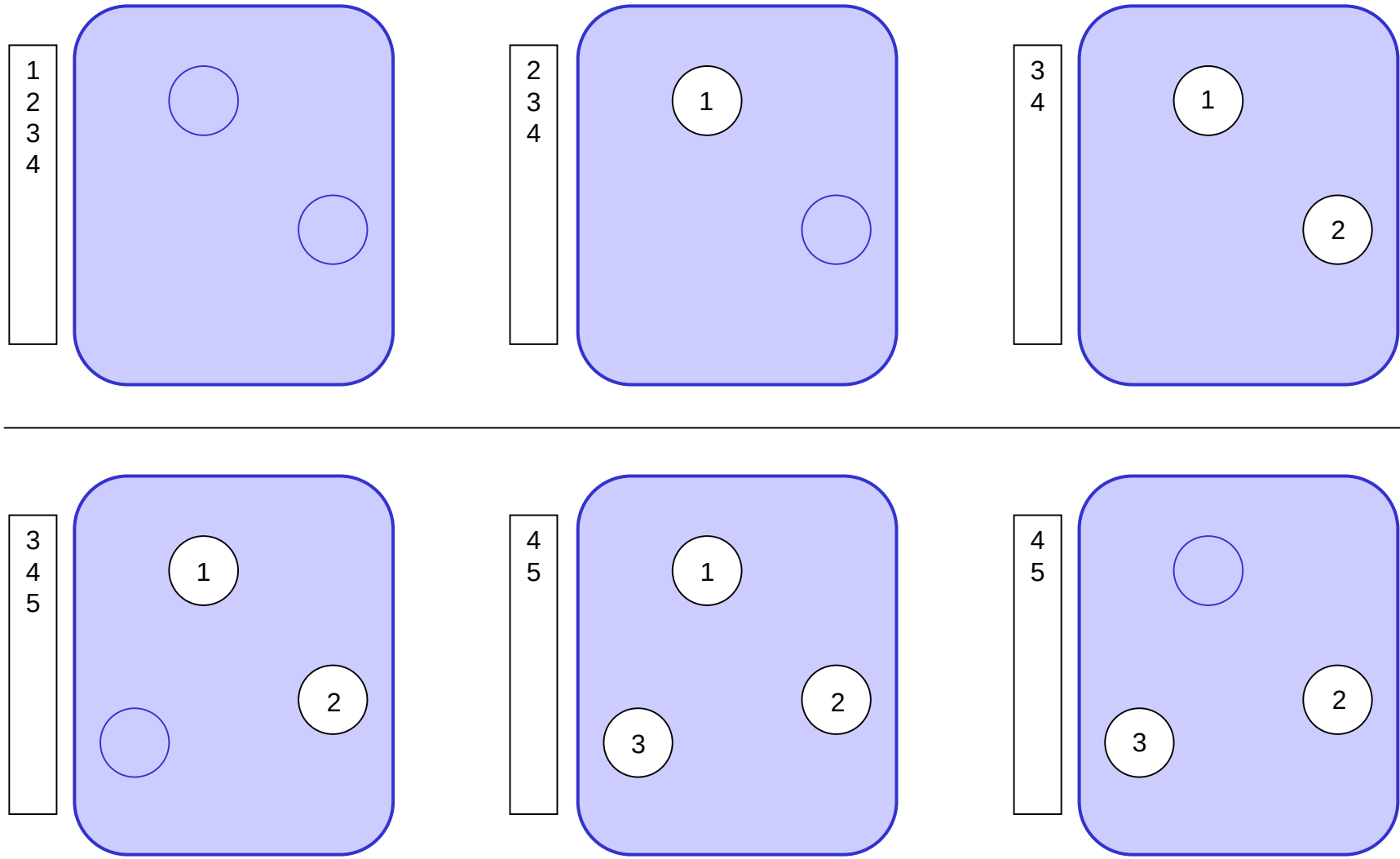
. . . but if they are expensive resources (connections, threads, beans, etc.): BAD!

"You don't want to build a new taxicab and hire a new driver for each incoming customer call."

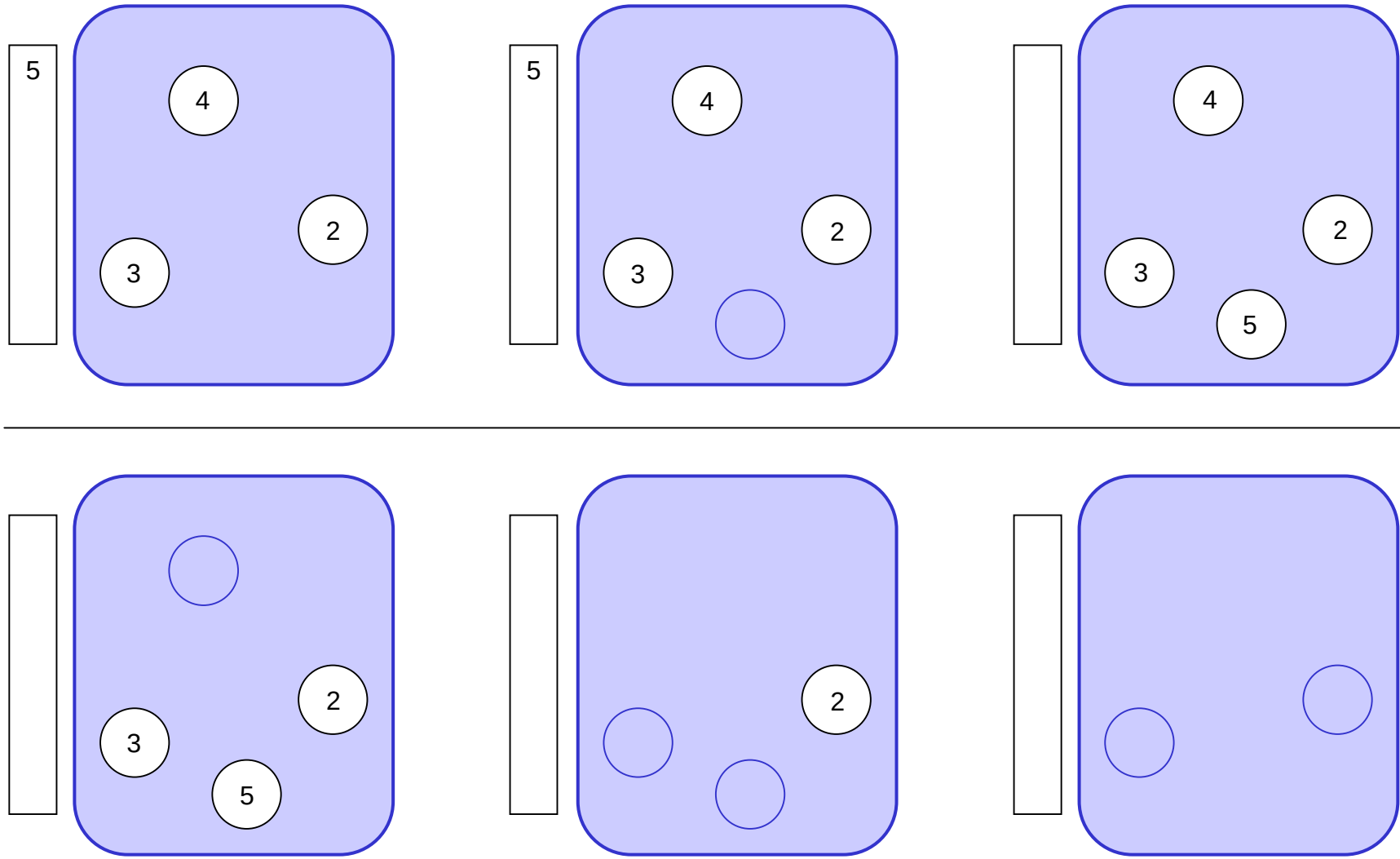
Object Pooling: Conserve And Reuse



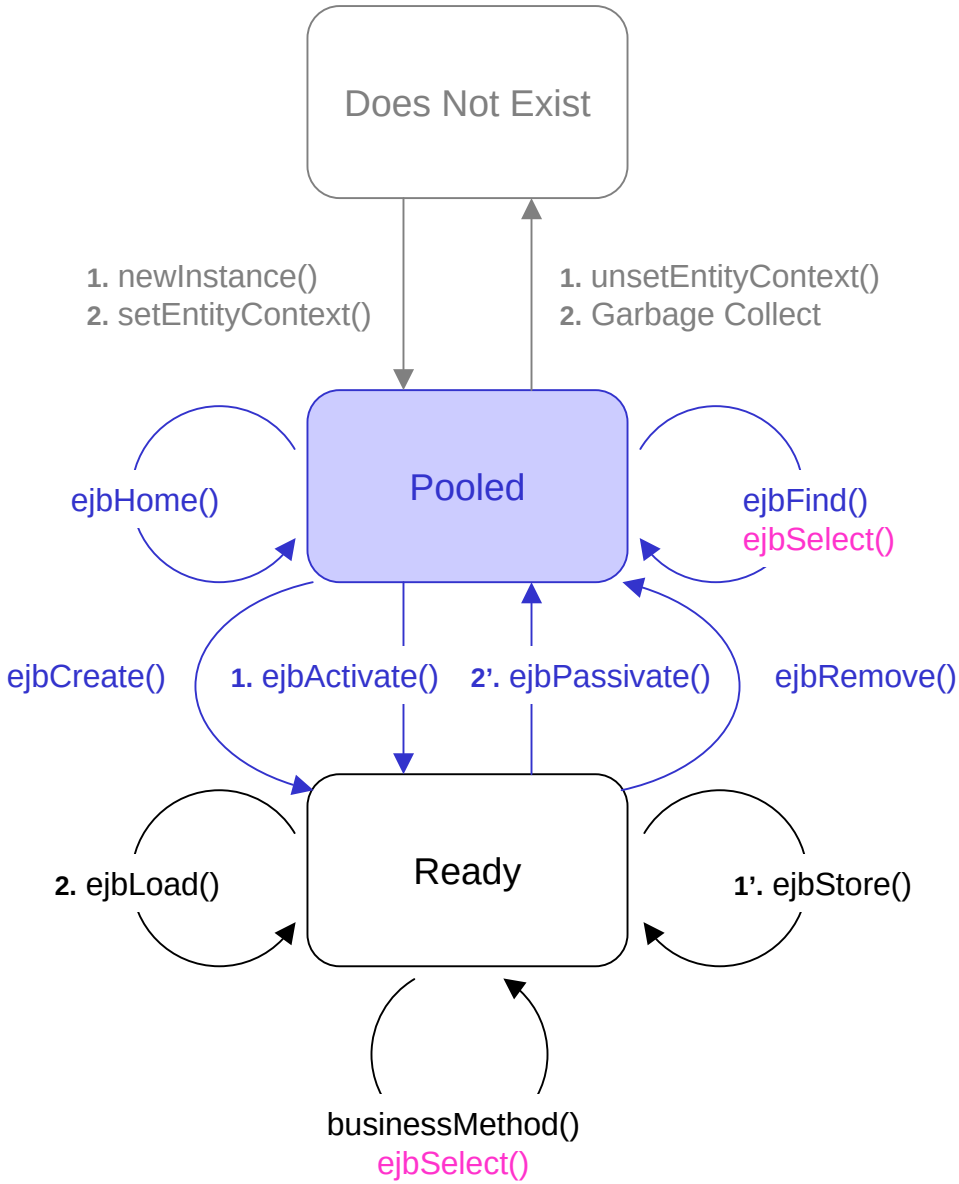
Object Pooling: Cycle



Object Pooling: Cycle



Entity Bean's Lifecyle: The "Easter Eggs" Lab



Entity Beans: BMP & CMP

BMP Entity Beans

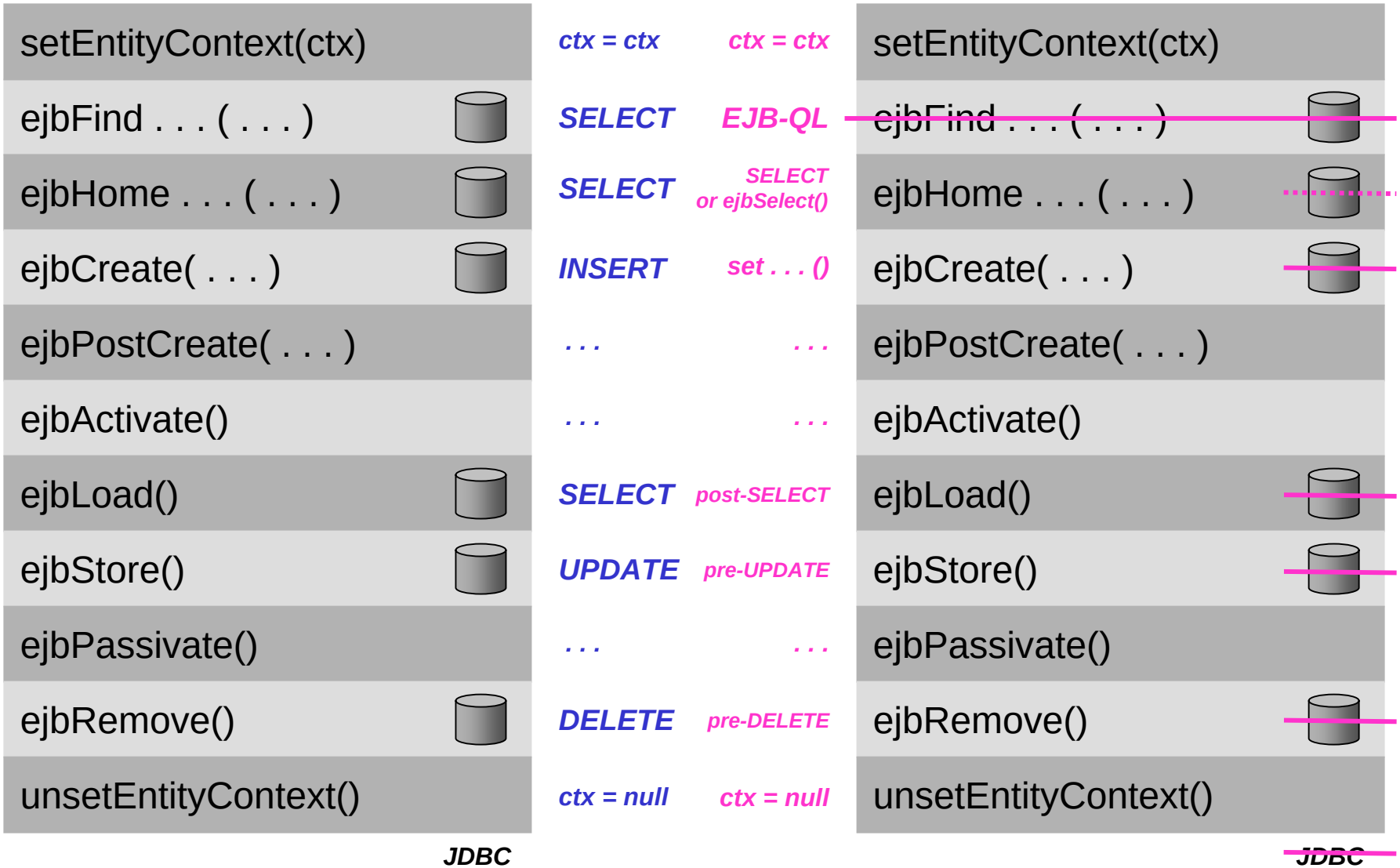
- persistence logic is ***programmed*** in the bean class
- big Java code for bean class
  - persistent fields & get/set methods
  - edit & search queries with JDBC
- small XML deployment descriptors

CMP Entity Beans

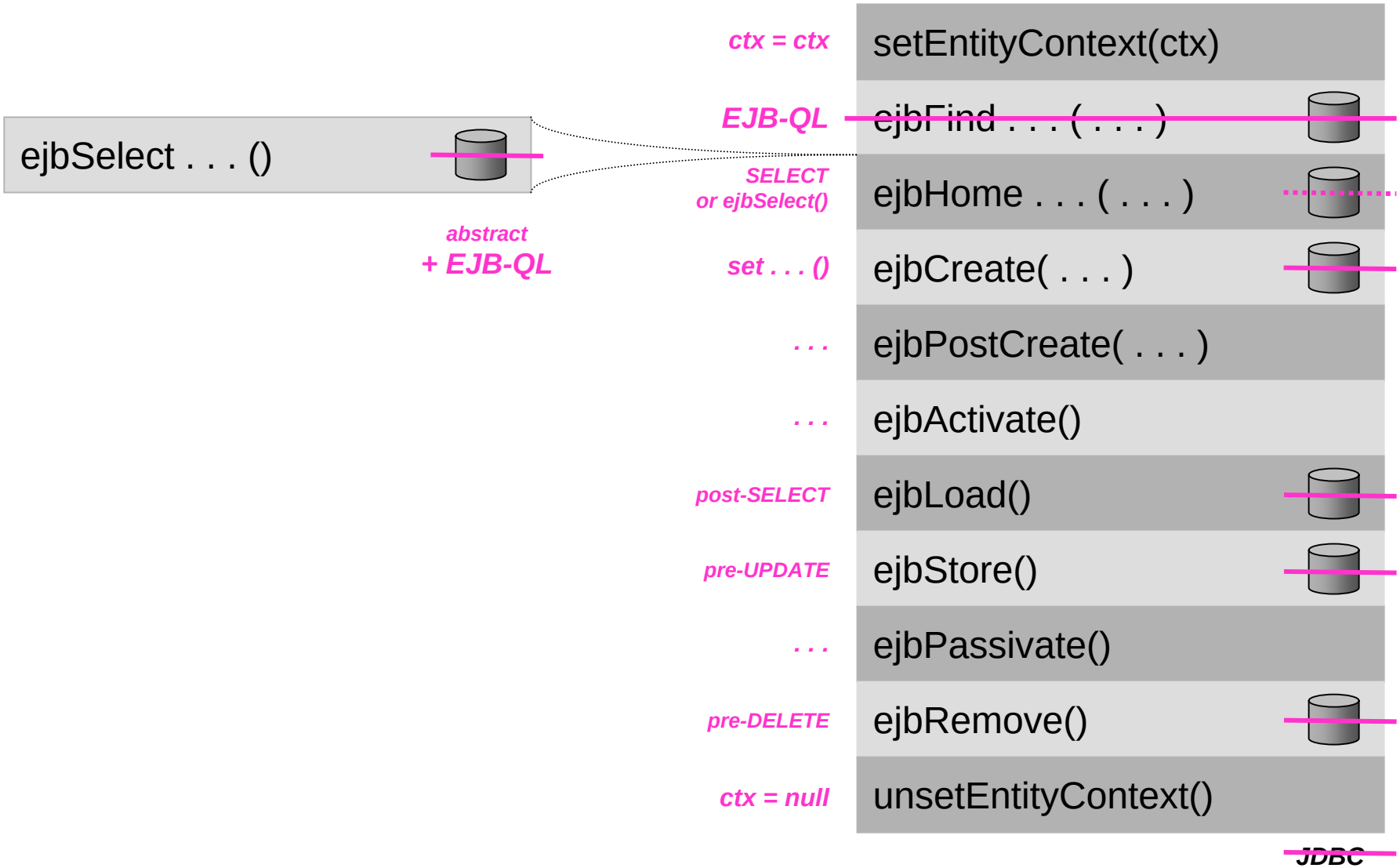
- persistence logic is ***declared*** in the deployment descriptors
- smaller Java code for bean class
  - ***no*** persistent fields, abstract get/set
  - almost ***no*** JDBC
- bigger XML deployment descriptors
  - persistent fields
  - EJB-QL search queries
  - container-specific DB mapping

Entity Bean's Business Logic: BMP & CMP

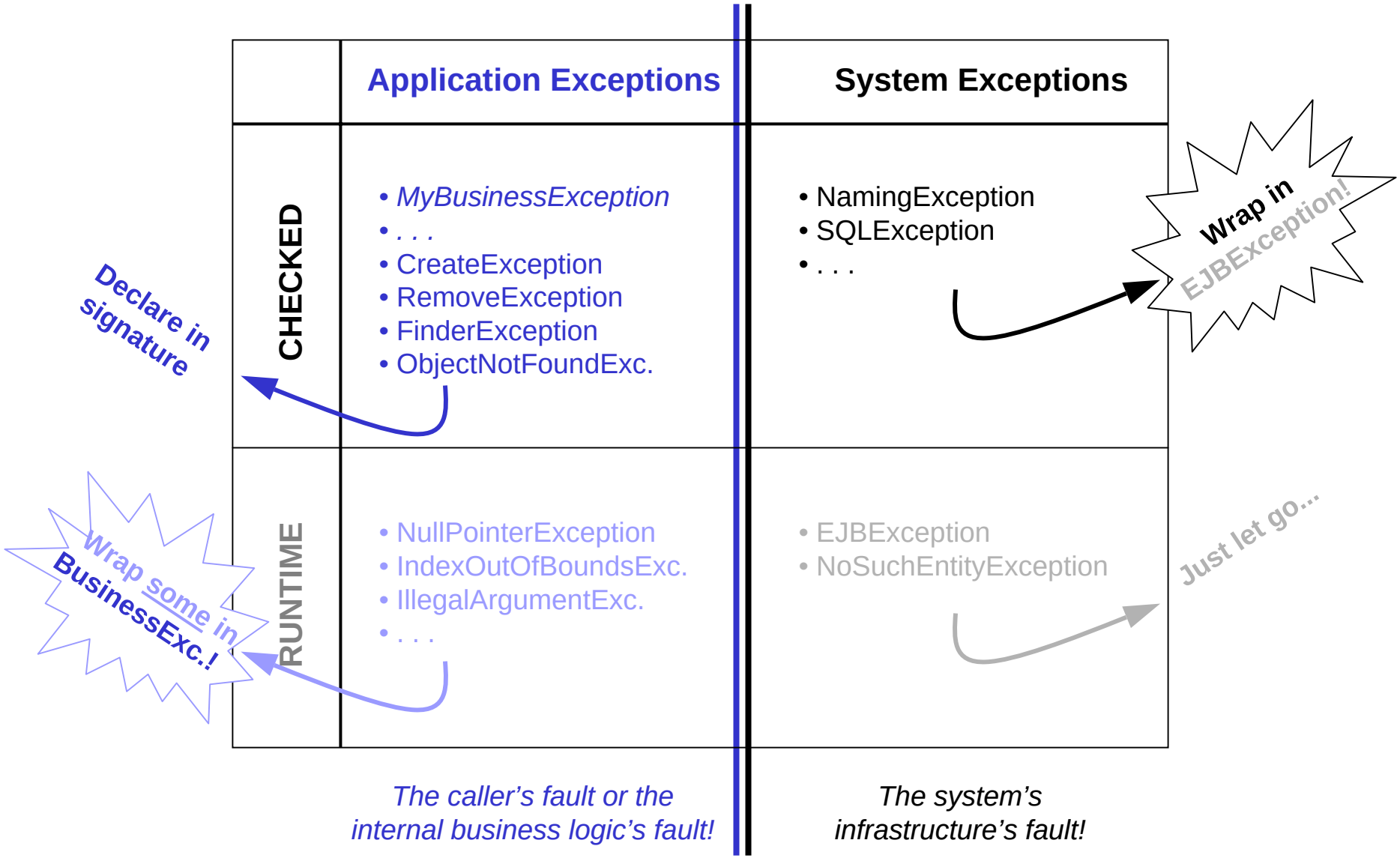
Entity Bean's Container Methods: BMP & CMP



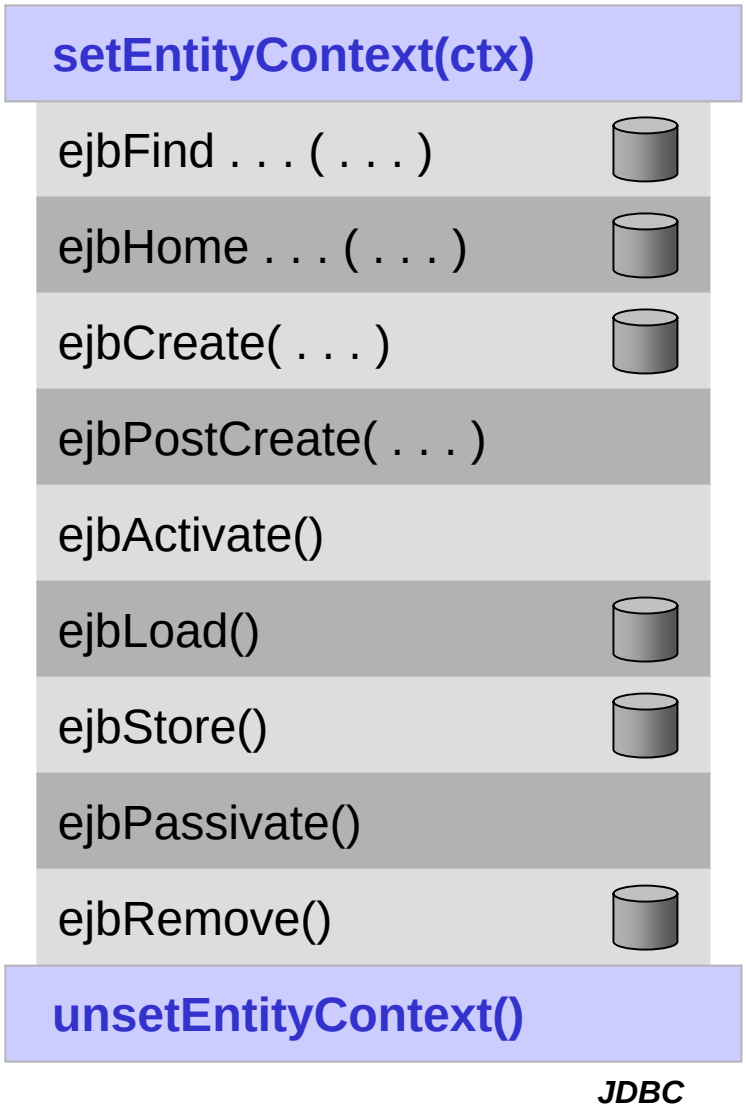
Entity Bean's Container Methods: CMP



Exception Handling



Entity Bean's Container Methods: BMP

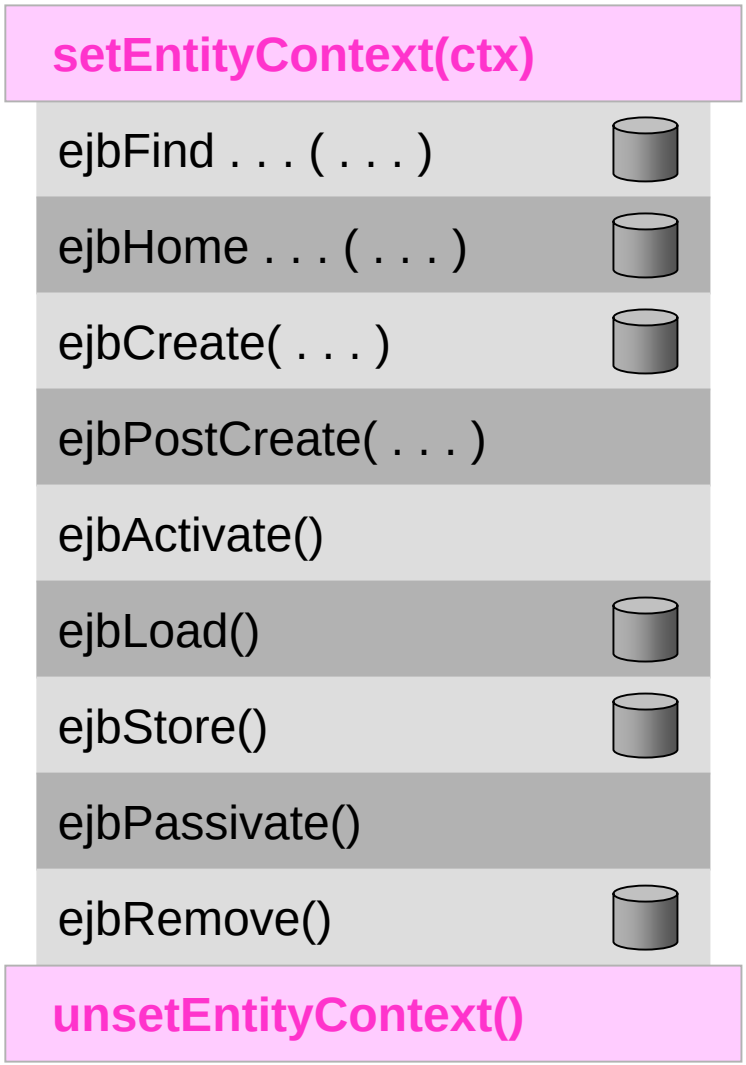



```
public void setEntityContext(EntityContext ctx)

{
    this.ctx = ctx;
}
```

```
public void unsetEntityContext()

{
     this.ctx = null;
}
```

Entity Bean's Container Methods: CMP

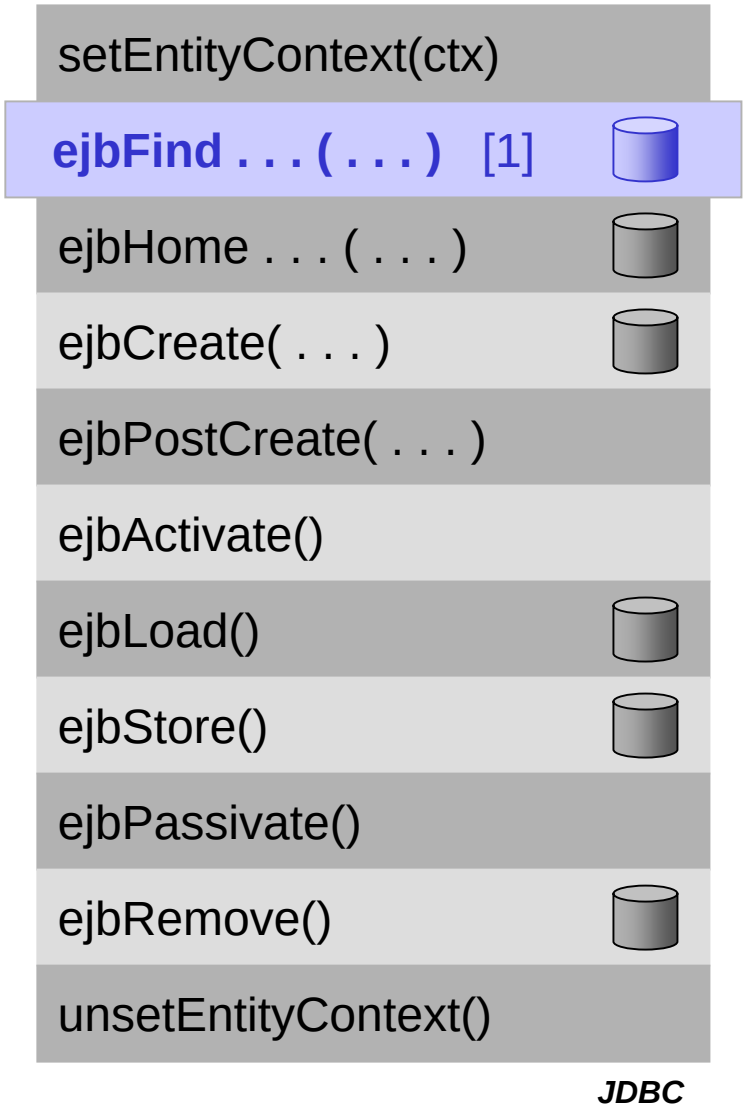
→ *SAME*




```
public void setEntityContext(EntityContext ctx)

{
    this.ctx = ctx;
}
```

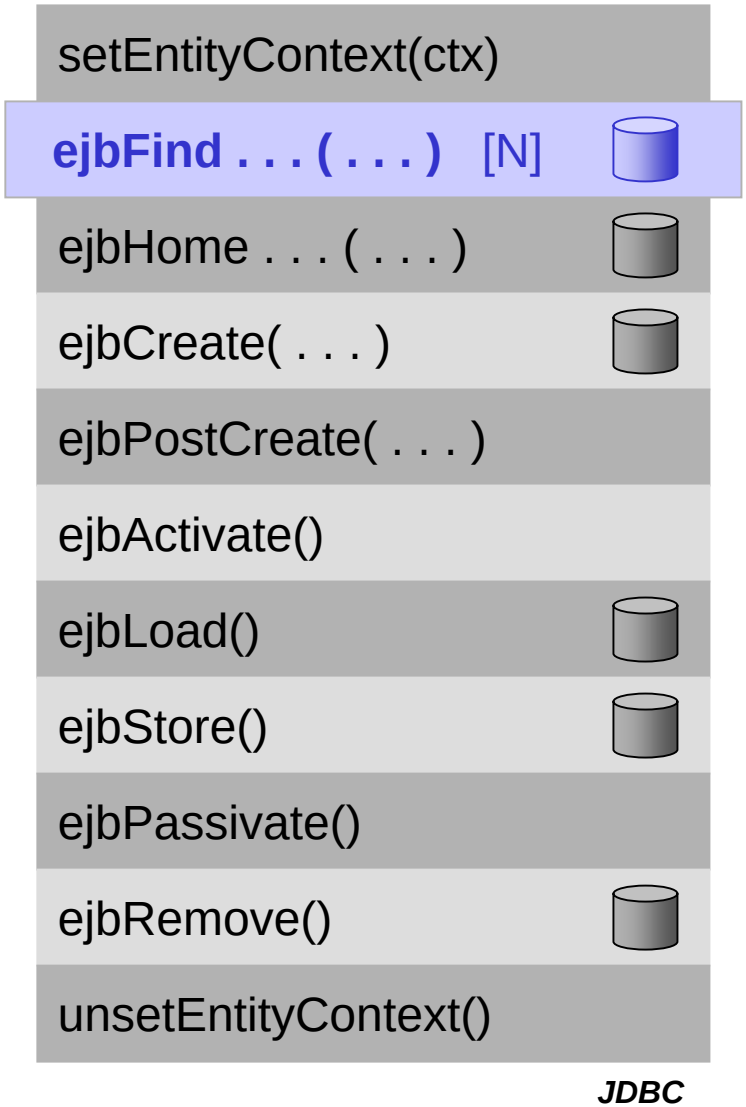
```
public void unsetEntityContext()

{
     this.ctx = null;
}
```

Entity Bean's Container Methods: BMP




```
public Key ejbFind . . . ( args )
    throws FinderException
{

    try {
        ...get DB connection

        ...prepare SELECT statement
        ...set args in statement

        ...execute statement query
        if (empty result) {
            ...throw FinderException
        }
        ...return key from result
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

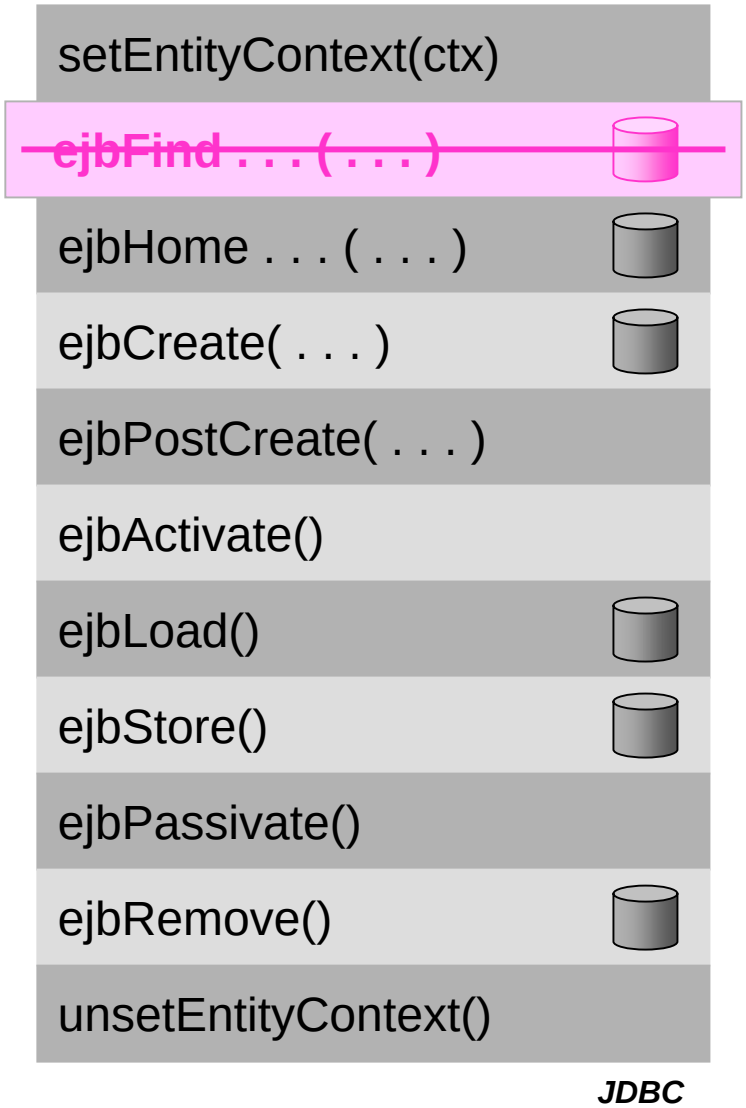
Entity Bean's Container Methods: BMP




```
public Collection ejbFind . . . ( args )
    throws FinderException
{

    try {
        ...get DB connection

        ...prepare SELECT statement
        ...set args in statement

        ...execute statement query
        while (not empty result) {
            ...fill coll. with keys from result
        }
        ...return coll.
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

Entity Bean's Container Methods: CMP

→ *IMPLEMENT IN EJB-QL*




```
public Key/Collection ejbFind . . . ( args )
    throws FinderException
```

ejb-jar.xml

```
<entity>
  . . .
  <persistence-type>Container</persistence-type>
  . . .
  <query>
    <query-method>
      <method-name>find . . . </method-name>
      <method-params>args</method-params>
    </query-method>
    <ejb-ql>
      SELECT OBJECT(q) FROM . . .
    </ejb-ql>
  </query>
  . . .
</entity>
```

Entity Bean's Container Methods: CMP

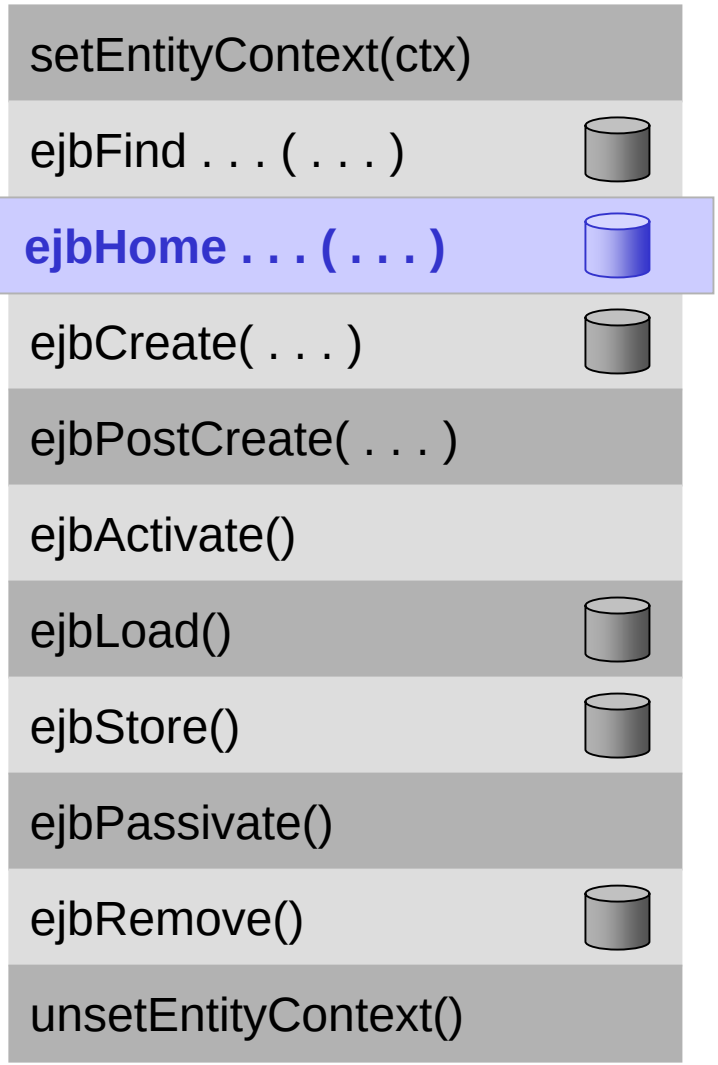
→ *ABSTRACT METHOD AND EJB-QL*


```
public abstract Collection ejbSelect . . . ()
    throws FinderException;
```

ejb-jar.xml

```
<entity>
  . . .
  <persistence-type>Container</persistence-type>
  . . .
  <query>
    <query-method>
      <method-name>ejbSelect . . . </method-name>
      <method-params></method-params>
    </query-method>
    <ejb-ql>
      SELECT q.xxx FROM . . .
    </ejb-ql>
  </query>
  . . .
</entity>
```

Entity Bean's Container Methods: BMP




```
public Value ejbHome . . . ( args )
    throws BusinessException
{

    try {
        ...get DB connection

        ...prepare SELECT statement
        ...set args in statement

        ...execute statement query
        while (not empty result) {
            ...aggregate value from result
        }
        ...return value
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

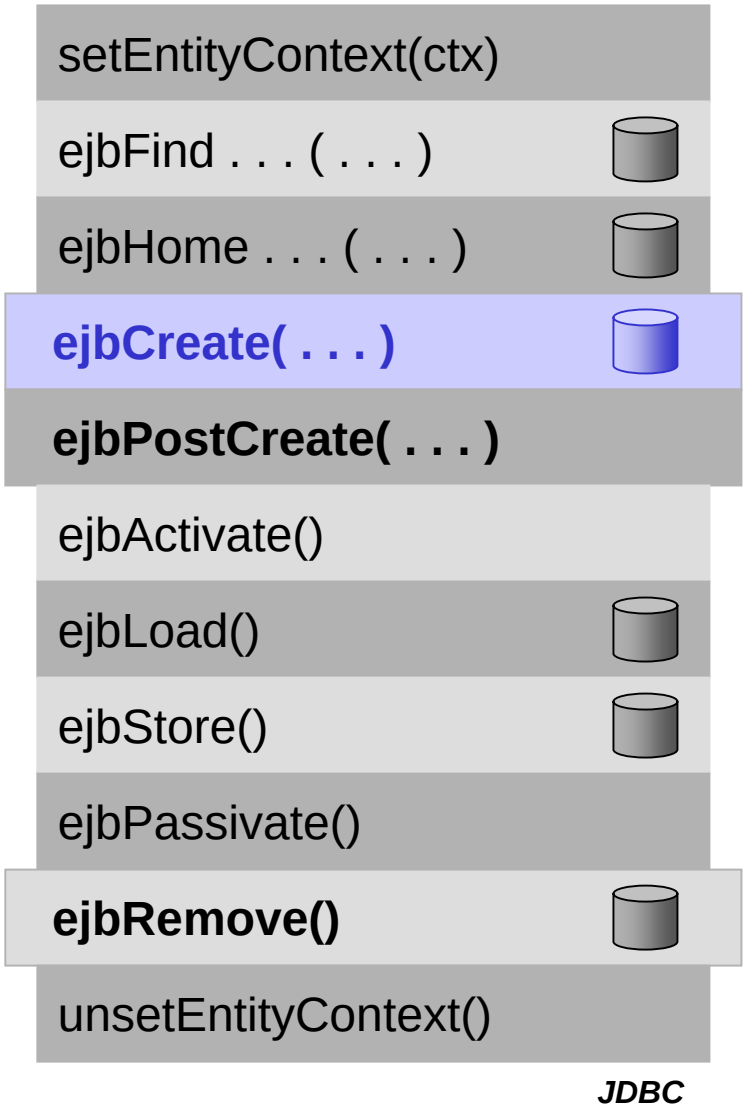
Entity Bean's Container Methods: CMP

→ *SAME OR USE ejbSelect()*


```
public Value ejbHome . . . ( args )
    throws BusinessException
{

    try {
        ...get DB connection

        ...prepare SELECT statement
        ...set args in statement

        ...execute statement query
        while (not empty result) {
            ...aggregate value from result
        }
        ...return value
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

Collection c = ejbSelect()

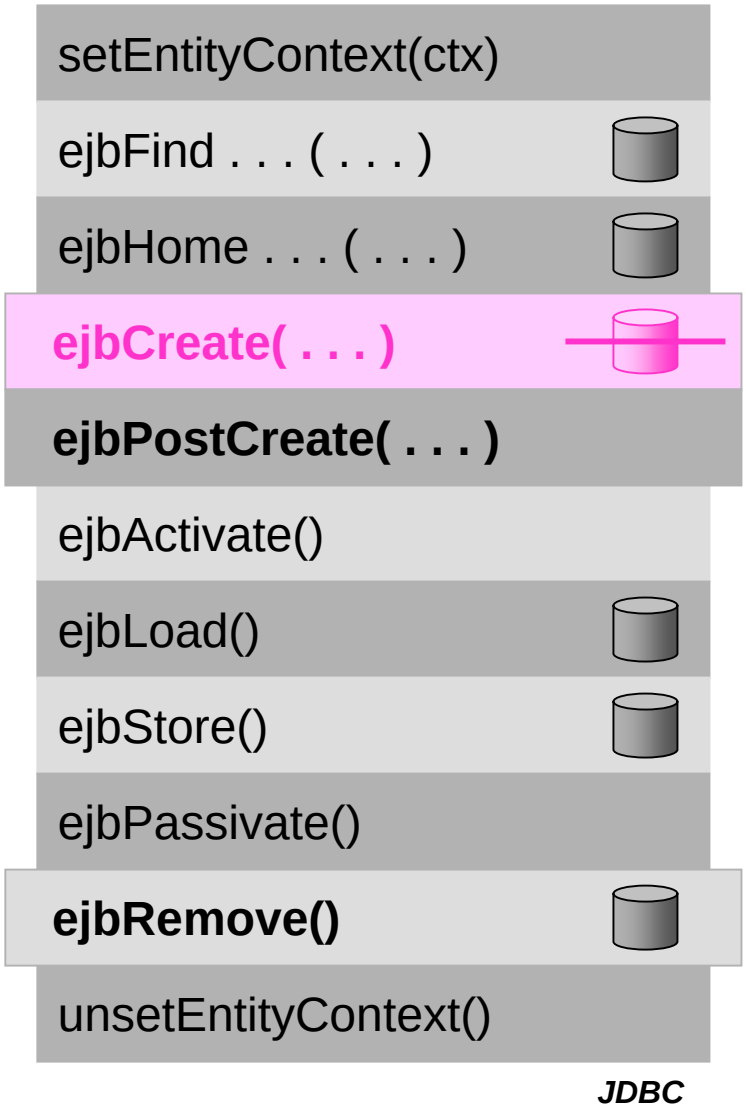
Entity Bean's Container Methods: BMP




```
public Key ejbCreate( args )
    throws CreateException
{
    ...initialize fields with args
    try {
        ...get DB connection

        ...prepare INSERT statement
        ...set key & fields in statement

        ...execute statement update
        if (zero count) {
            ...throw CreateException
        }
        ...return key
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

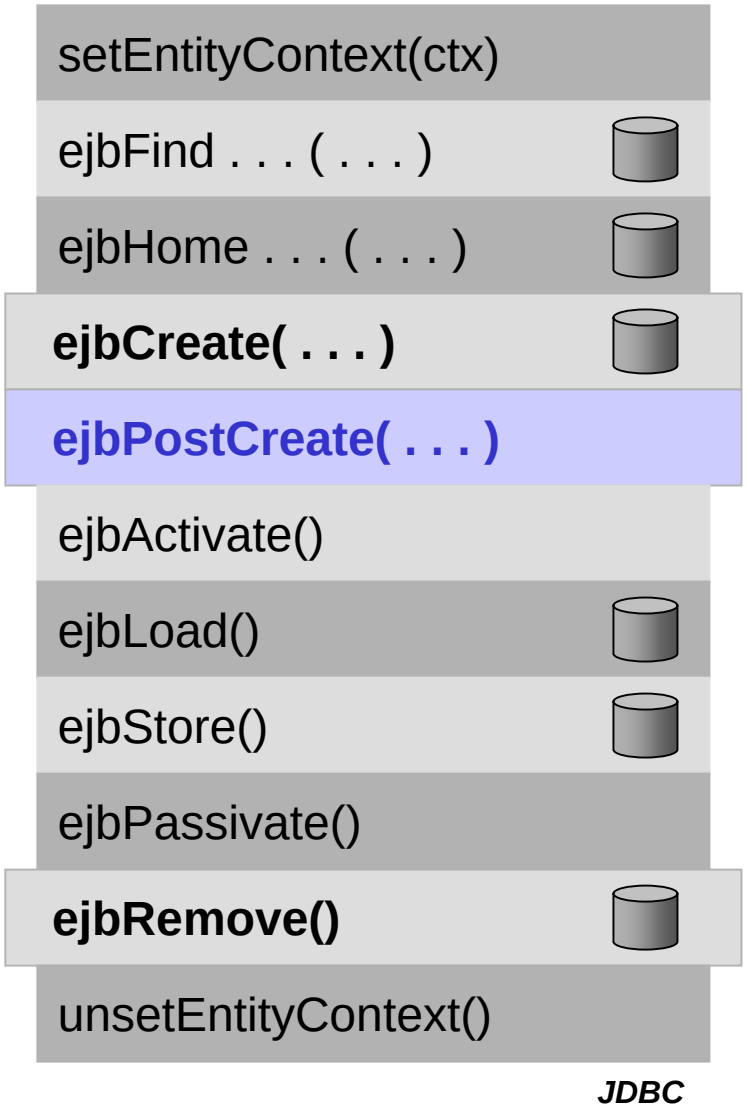
Entity Bean's Container Methods: CMP

→ *NO JDBC: set . . . () ONLY*




```
public Key ejbCreate( args )
    throws CreateException
{
    ...initialize fields with args, using setXxx()

    ...return key

}
```

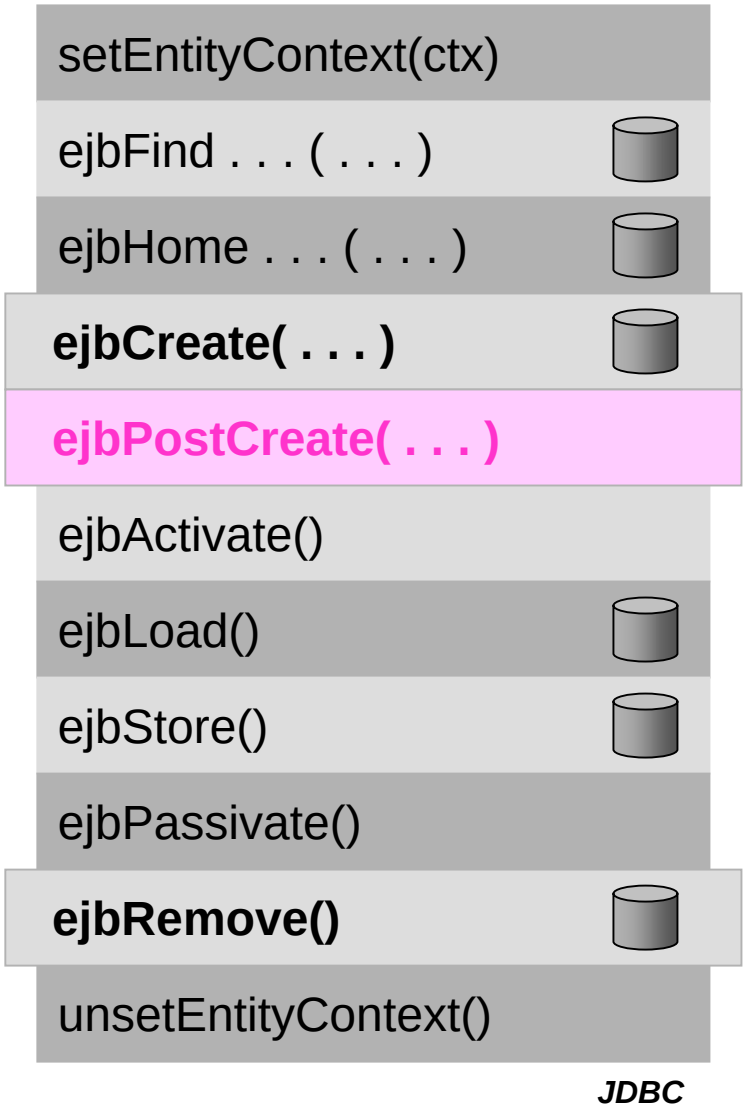
Entity Bean's Container Methods: BMP




```
public void ejbPostCreate( args )

{
    ...use EJBObject, if needed
}
```

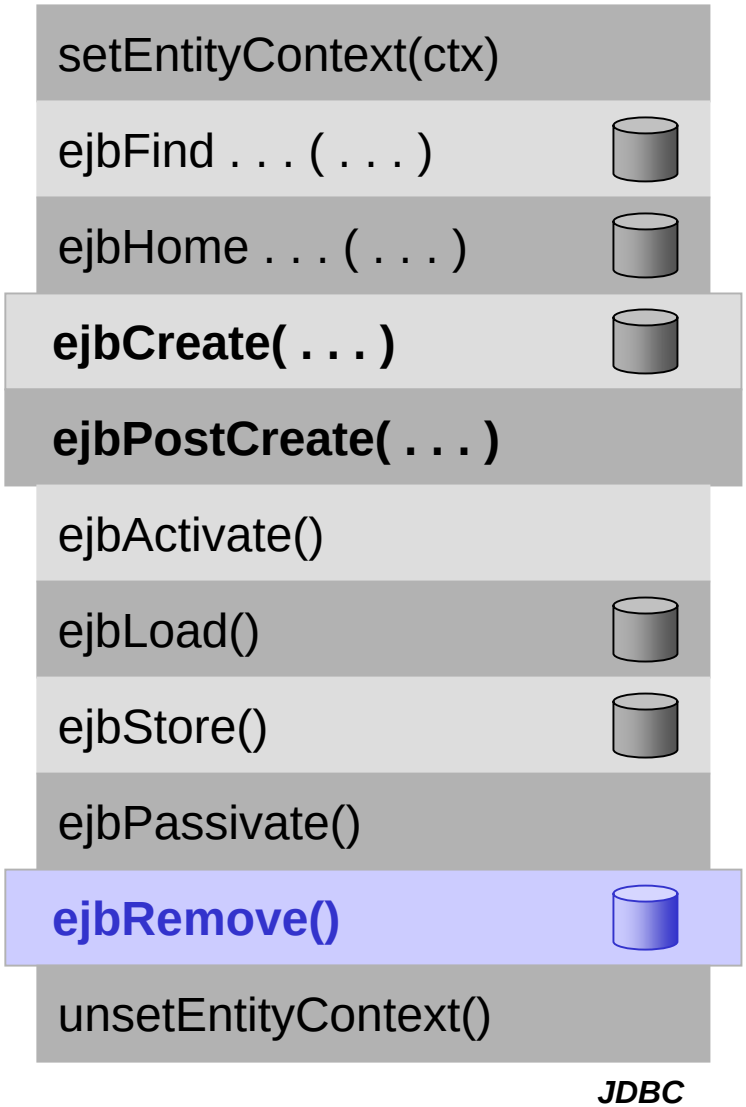
Entity Bean's Container Methods: CMP

→ *SAME*




```
public void ejbPostCreate( args )

{
    ...use EJBObject, if needed
}
```

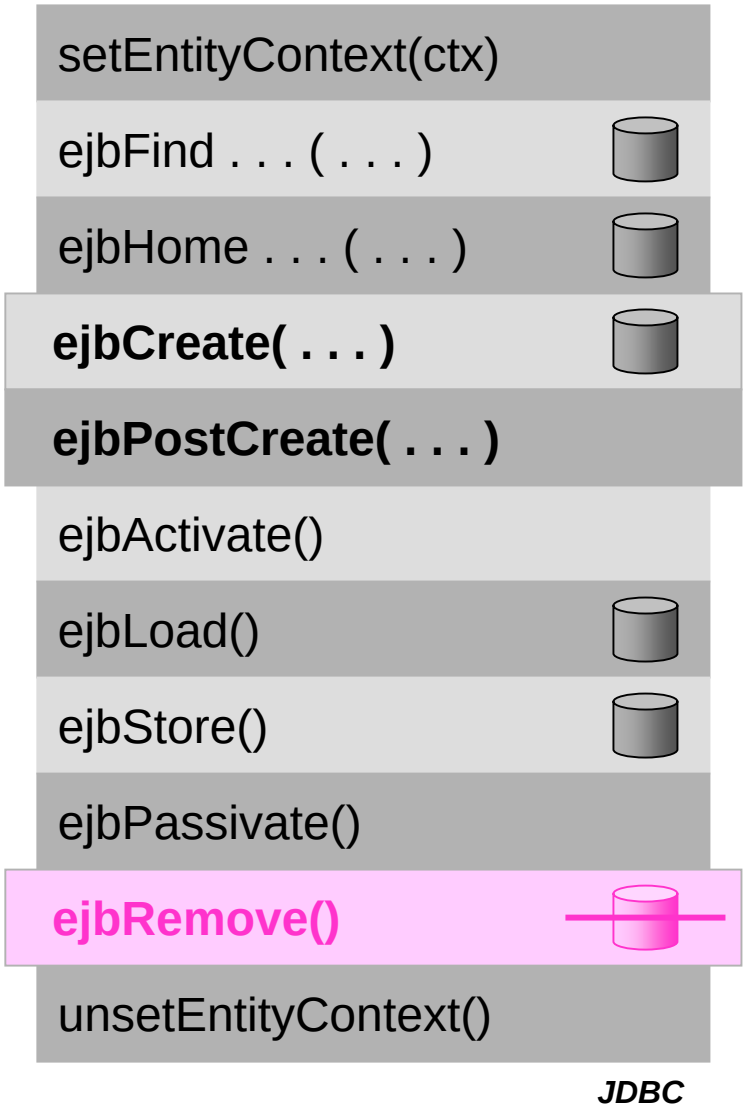
Entity Bean's Container Methods: BMP




```
public void ejbRemove()
    throws RemoveException
{

    try {
        ...get DB connection

        ...prepare DELETE statement
        ...set key in statement

        ...execute statement update
        if (zero count) {
            ...throw RemoveException
        }

    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

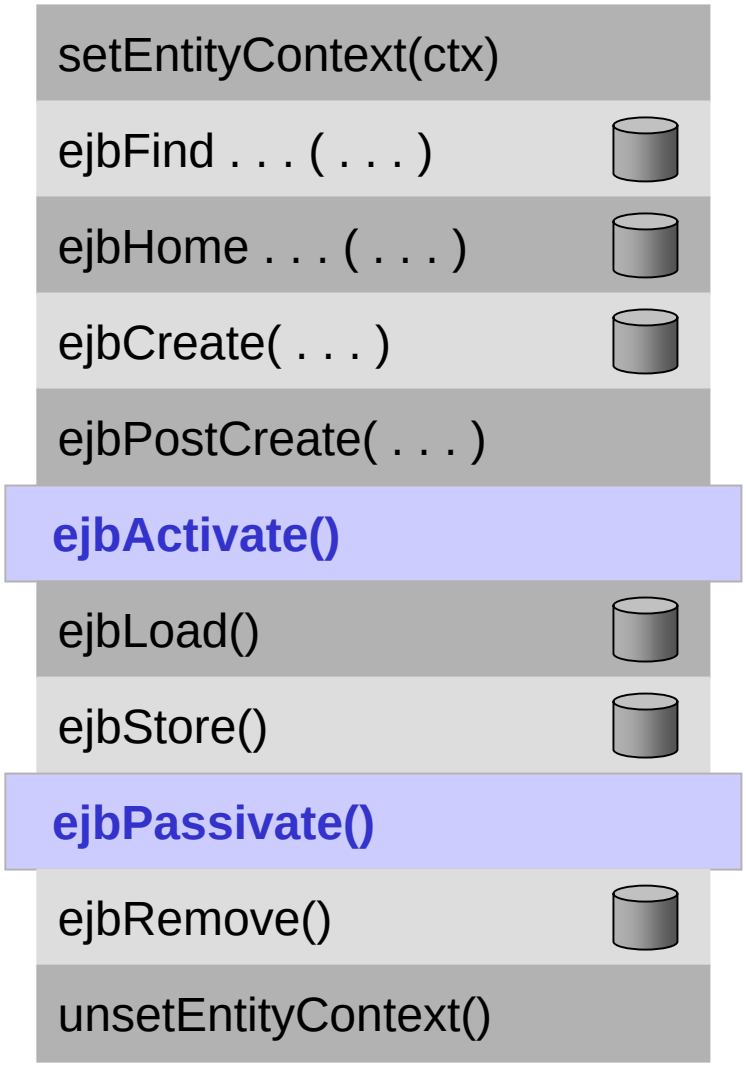
Entity Bean's Container Methods: CMP

→ *NO JDBC: pre-DELETE ONLY*




```
public void ejbRemove()
    throws RemoveException
{
    ...possible pre-DELETE processing

}
```

Entity Bean's Container Methods: BMP

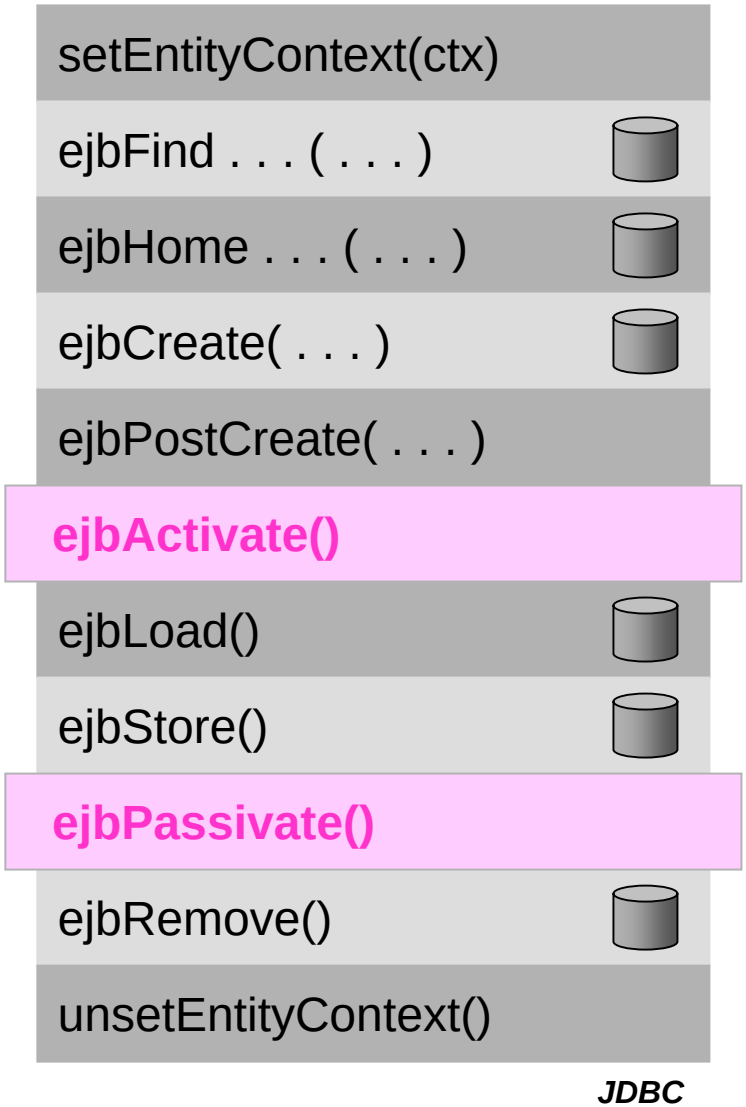



```
public void ejbActivate()

{
    ...acquire resources, if needed
}
```

```
public void ejbPassivate()

{
    ...release held resources, if any
}
```

Entity Bean's Container Methods: CMP

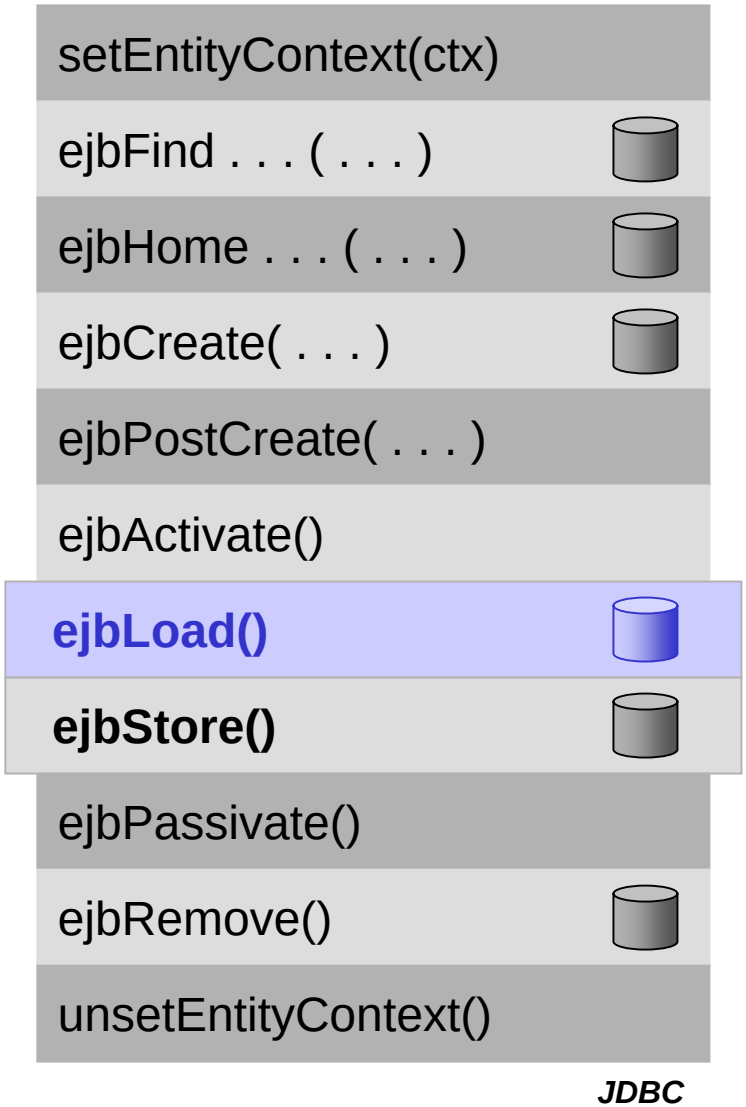
→ *SAME*




```
public void ejbActivate()

{
    ...acquire resources, if needed
}
```

```
public void ejbPassivate()

{
    ...release held resources, if any
}
```

Entity Bean's Container Methods: BMP



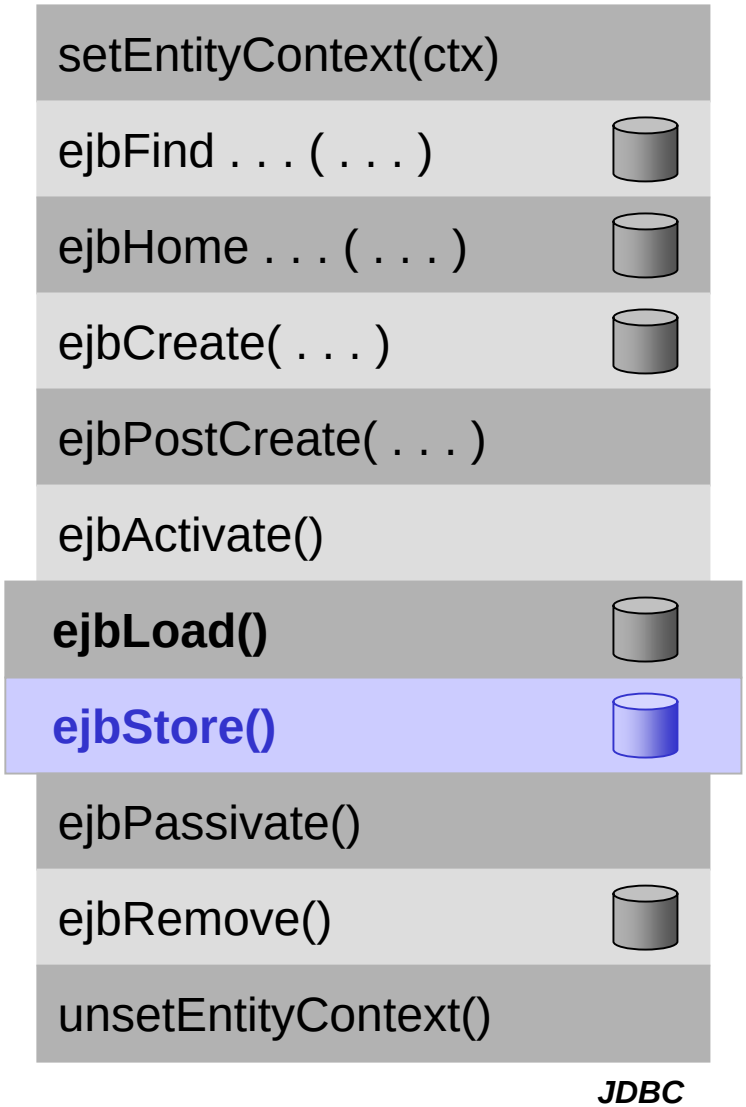

```
public void ejbLoad()

{

    try {
        ...get DB connection

        ...prepare SELECT statement
        ...set key in statement

        ...execute statement query
        if (empty result) {
            ...throw NoSuchEntityException
        }
        ...get fields from result
    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

Entity Bean's Container Methods: CMP

→ *NO JDBC: post-SELECT ONLY*


```
public void ejbLoad()

{

    ...possible post-SELECT processing
}
```

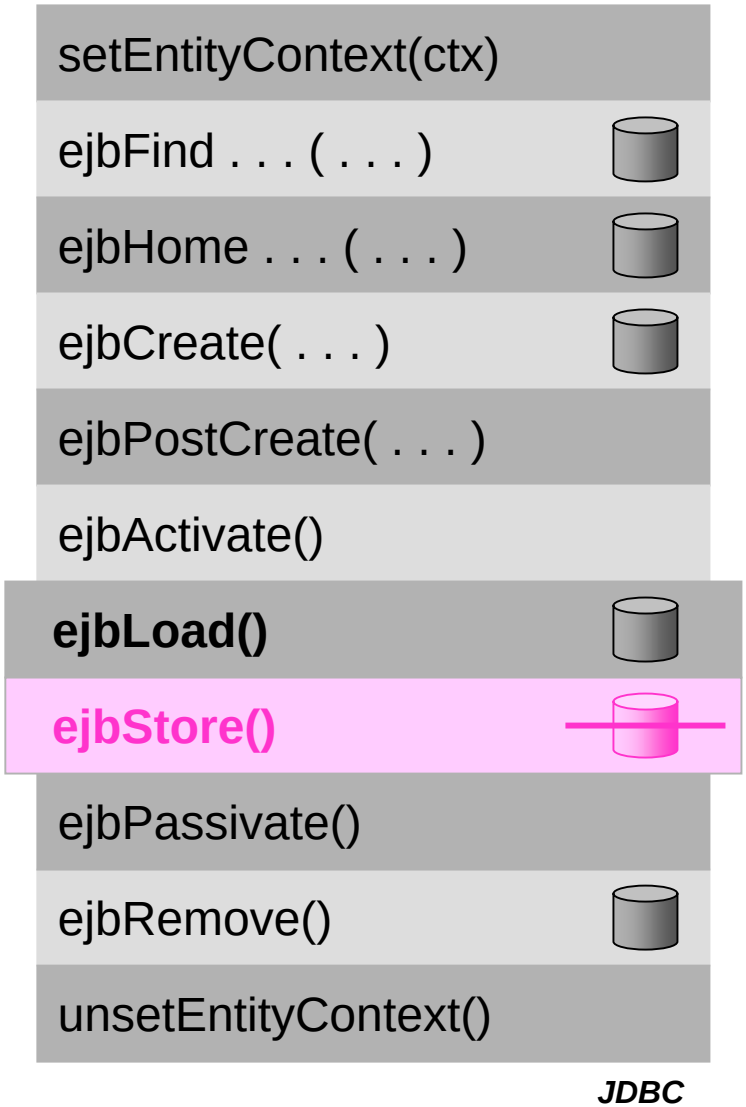
Entity Bean's Container Methods: BMP




```
public void ejbStore()

{

    try {
        ...get DB connection

        ...prepare UPDATE statement
        ...set key & fields in statement

        ...execute statement update
        if (zero count) {
            ...throw EJBException
        }

    }
    catch (SQLException, NamingException) {
        ...throw EJBException
    }
    finally {
        ...close statement
        ...close connection
    }
}
```

Entity Bean's Container Methods: CMP

→ *NO JDBC: pre-UPDATE ONLY*

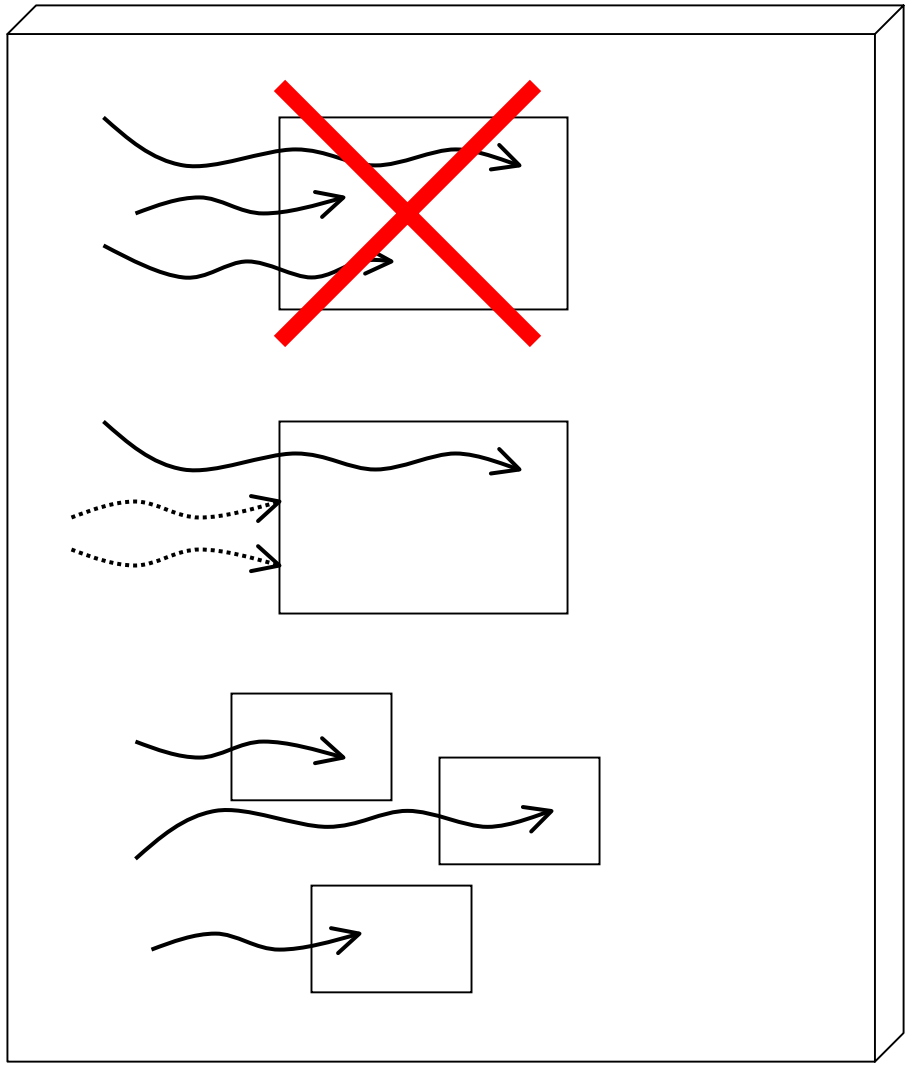



```
public void ejbStore()

{
    ...possible pre-UPDATE processing

}
```

EJBs Are Single-Threaded

Container Strategy 1:
serialize access to a single instance

Container Strategy 2:
allow parallel access to multiple instances



Entity Beans Are Single-Threaded

Container Strategy 1:
serialize access to a single instance

Container Strategy 2:
allow parallel access to multiple instances
and keep data synchronized in DB