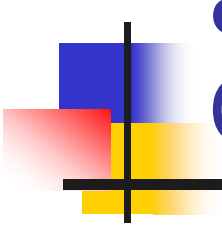
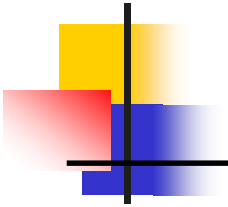


Mapping Java Objects to XML and Relational Databases
O-R and O-X Mapping

Donald Smith

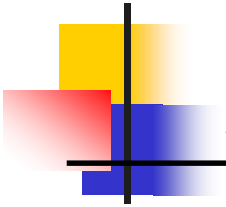
Technology Director, Oracle Corporation

blog: jroller.com/page/dsmith



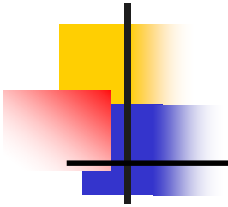
Speaker's Qualifications

- Decade of experience in OO Persistence
- Presented at Java One, Oracle World, OOPSLA, JAOO, Sun Tech Days, TheServerSide Symposium, *etc.*
- Author of numerous articles on persistence challenges



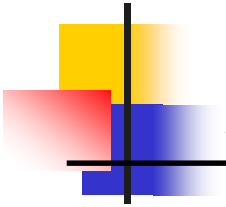
About the Audience…

- Who considers themselves first and foremost to be a DBA or "Database expert"?
- Who considers themselves first and foremost to be a Java and/or Web Services developer?
- Who considers themselves first and foremost to be an Architect?
- Who considers themselves first and foremost to be a manager, and will you admit it?



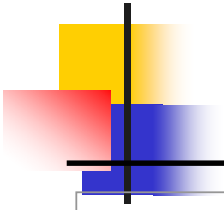
Goal

Discuss and demonstrate the challenges of mapping Java Objects to Relational Databases and XML documents



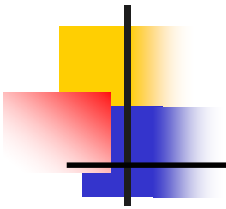
Agenda

- Quick Review of Java Mapping
  - O-R and O-X Overview
  - Terminology
- O-R Mapping
  - Direct, 1-1, Aggregate, 1-M, M-M, Inheritance
- O-X Mapping
  - Direct, Composite, Transformation, Inheritance



Agenda

- Quick Review of Java Mapping
  - O-R and O-X Overview
  - Terminology
- O-R Mapping
- O-X Mapping

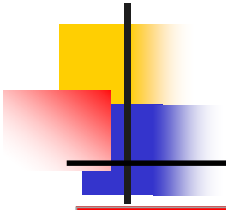


Impedance Mismatch

- Difference in relational, XML and object technology known as "impedance mismatch"
- Challenging problem
  - Requires relational, XML and object expertise

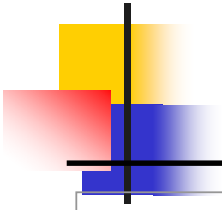
O-R Impedance Mismatch

	Factor	J2EE	Relational Database
Technical	Logical Data Format	Objects, methods, inheritance	Tables, SQL, stored procedures
	Scale	Hundreds of megs	Gigabytes, terabytes
	Relationship	Memory references	Foreign keys
	Uniqueness	Internal object identity	Primary keys
Business	Key Skills	Java development, object modeling	SQL, Stored Procedures, data management, Data modeling
	Tools	IDE, Source code management, Object Modeler	Schema designer, query manager, performance profilers, database config
Political	Corporate Org. Structure	"Newer technology" often with weak organizational ties to database mgmt	Often mature infrastructure with significant legacy considerations



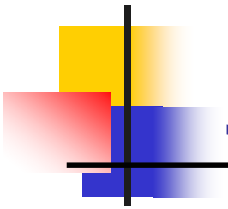
O-X Impedance Mismatch

	Factor	J2EE	XML
Technical	Logical Data Format	Objects, methods, inheritance	XSD, XPATH
	Scale	Hundreds of megs	Depends
	Relationship	Memory references	Generally use Aggregation
	Uniqueness	Internal object identity	Unique Identifier
Business	Key Skills	Java development, object modeling	XML, XSD, XPATH, XQUERY
	Tools	IDE, Source code management, Object Modeler	XML Design Tools and Viewers
Political	Corporate Org. Structure	Often integrating or using legacy application code	High expectations of data portability



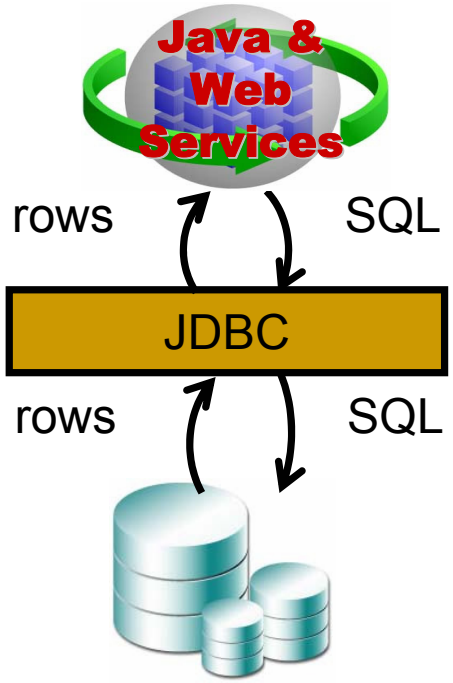
Agenda

- Quick Review of Java Mapping
  - O-R and O-X Overview
  - Terminology
- O-R Mapping
- O-X Mapping



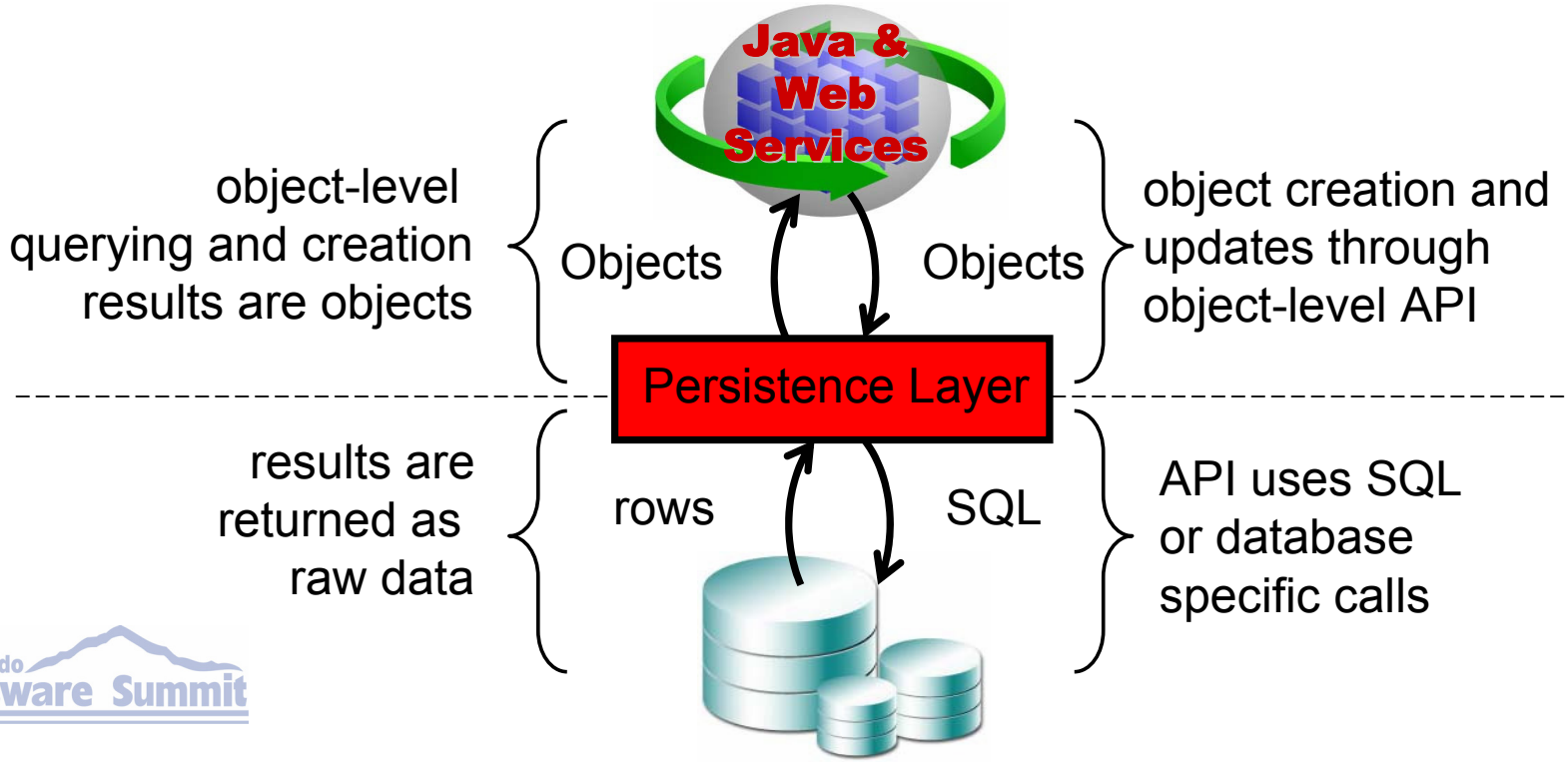
JDBC

- Java standard for accessing data sources
- Simply – data source connection
- OK with:
  - "Window on data" applications
  - Business logic entrenched on database
  - Java nothing more than GUI tool



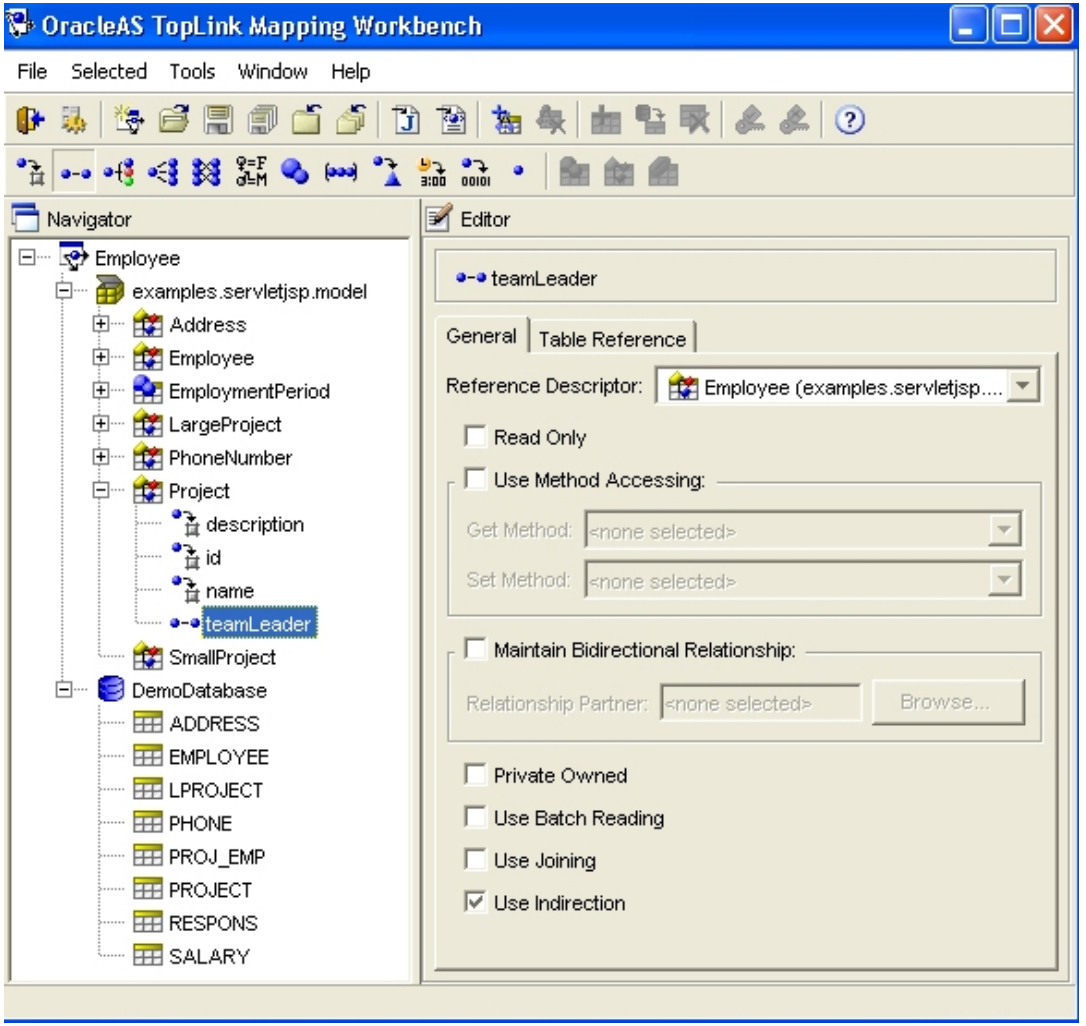
Persistence Layer

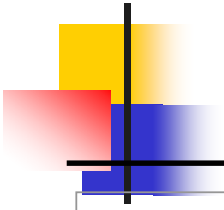
- Abstract persistence details from application layer
- Requires Object to Data source mapping…



O-R Mapping Tools

- O-R Mapping Tools help you manage the mapping's meta data

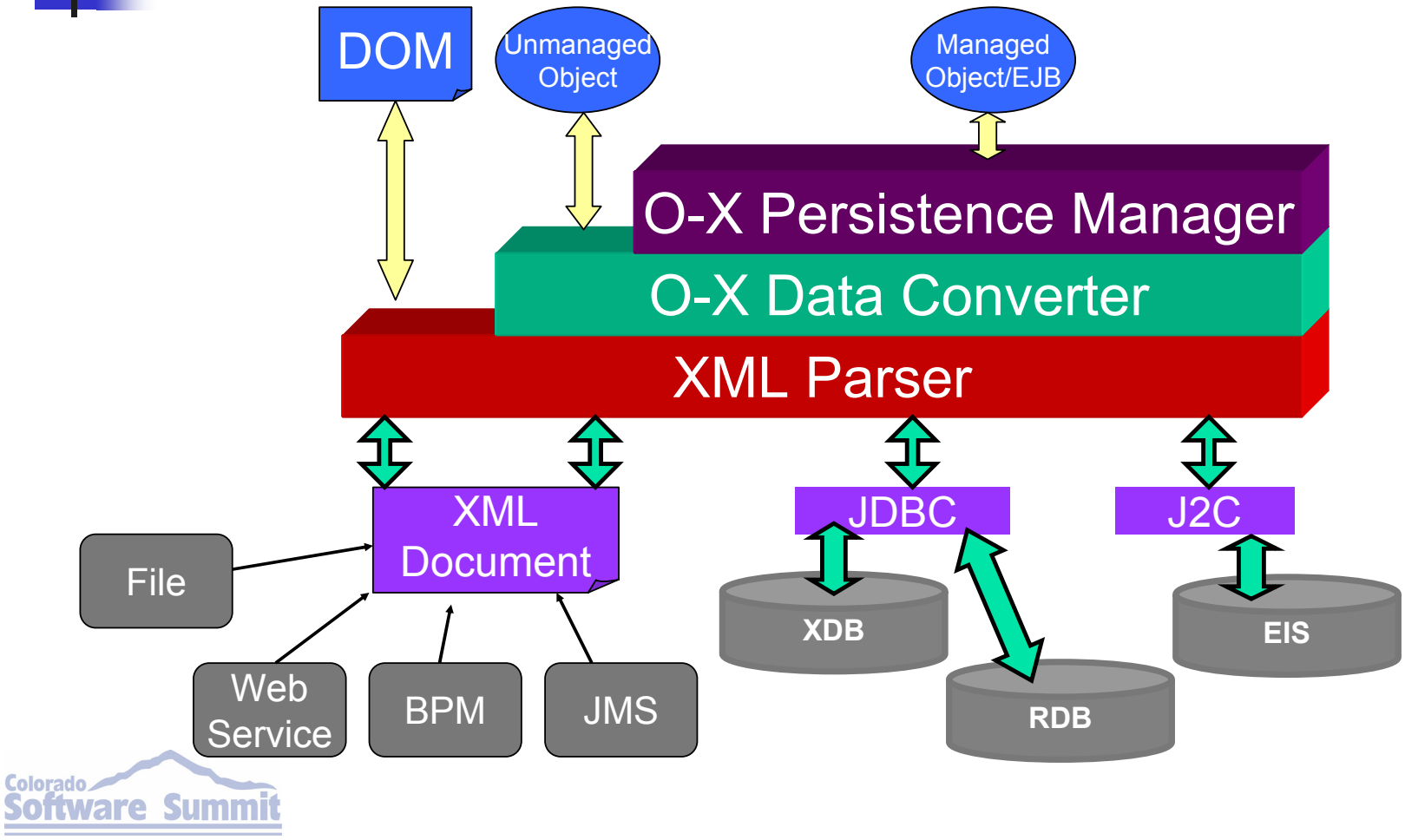


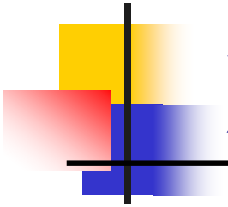


Agenda

- Quick Review of Java Mapping
  - O-R and O-X Overview
  - Terminology
- O-R Mapping
- O-X Mapping

3 Levels of XML Representation

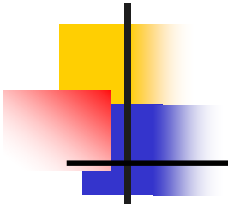




XML Parser

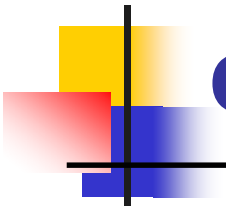
- JAXP – Java API for XML Parsing
  - DOM
  - SAX

- Very low level
- Akin to straight using straight JDBC for database interactions
- Useful for simple and raw GUI based apps where a business model is overkill



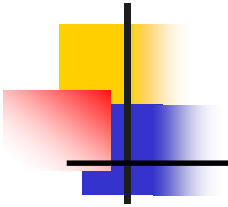
O-X Data Converter

- Converts XML data to Java objects and vice-versa
- Accessed by applications through Marshal/Unmarshal interface
- Usually for non-transactional data sources
- JAXB implementations



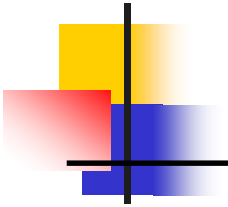
O-X Data Converter Limitations

- Beware of your JAXB implementation…
  - Most generate Java classes from XML Schema – static, inflexible
  - Most offer no control over the mappings
  - Can't use your own Java classes
  - Application code is tightly coupled to a specific XSD
- Usually no GUI tools to do mappings
- Conversion only, no run-time manager available for transactional data sources
- Homogeneous data support
  - Specific interfaces and generation for XML



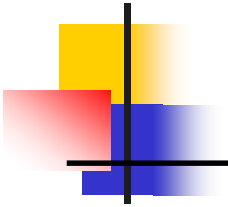
O-X Persistence Manager

- Flexible mapping, developers control how objects are mapped to XML – "meet in the middle"
  - Can use developer-defined Java classes
  - Independence between object model and XML schema
  - Business logic can be safely added into Java model
  - Classes can be mapped to multiple schemas – *vice versa*
  - Can be JAXB compliant



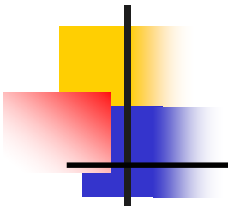
O-X Persistence Manager

- Support complex XML mappings
  - Positional, path information
  - Examples coming…
- May provide visual mapping interface
- May support other data sources – relational and EIS



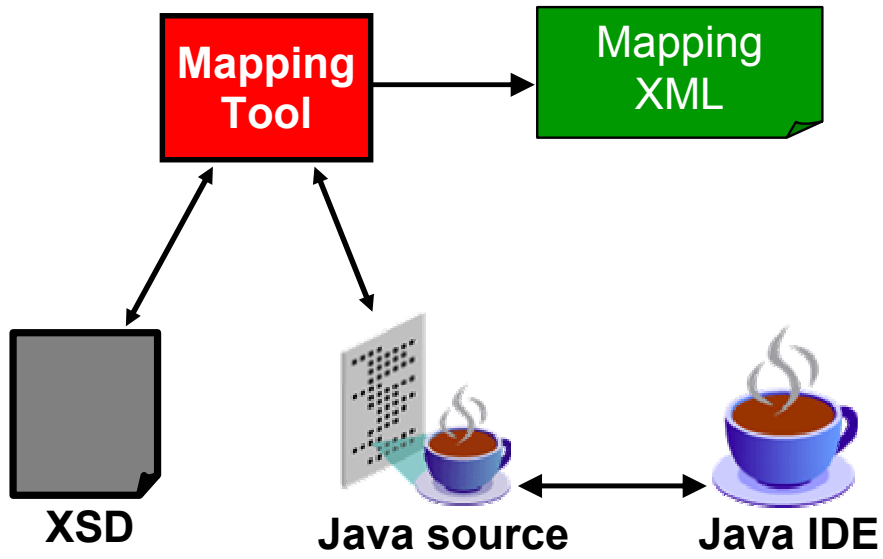
O-X Persistence Manager

- Persistence manager functionality may be required for transactional XML data sources such as EIS systems, XML databases.
- Provides additional capabilities on top of data conversion such as:
  - Caching
  - Querying
  - Transactions
  - Concurrency



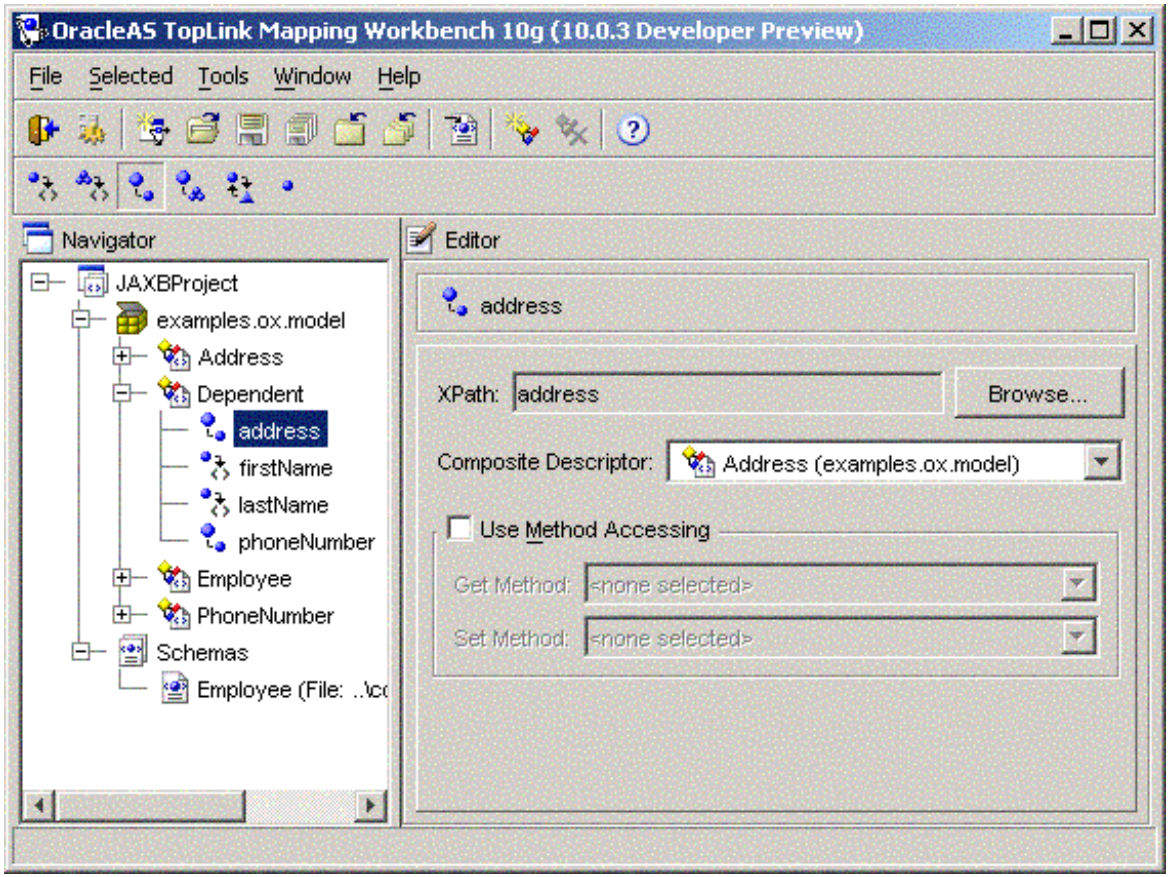
O-X Mapping

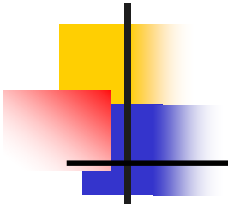
- Map Object Model to XSD
- Either code gen XSD from Object Model, *vice versa*, or "meet in the middle"



O-X Mapping Tools

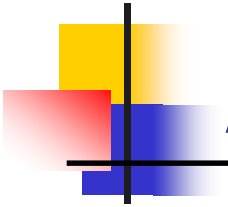
- O-X Mapping Tools help you manage the mapping's meta data





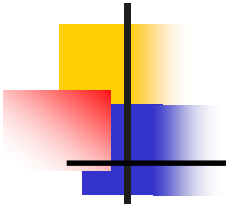
O-R *versus* O-X

- O-X
  - Much more aggregation (*aka* "denormalized")
  - Relationships represented in structure of document
  - Related data usually contained in single "bite"
    - "Less Chatty"
- O-R
  - Relationships represented by data in table (FK)
  - Frequent round-tripping
  - Data storage and practices more standardized



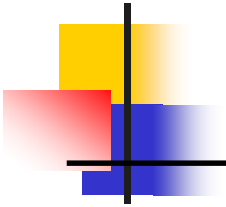
Agenda

- Quick Review of Java Mapping
  - O-R and O-X Overview
  - Terminology
- O-R Mapping
- O-X Mapping



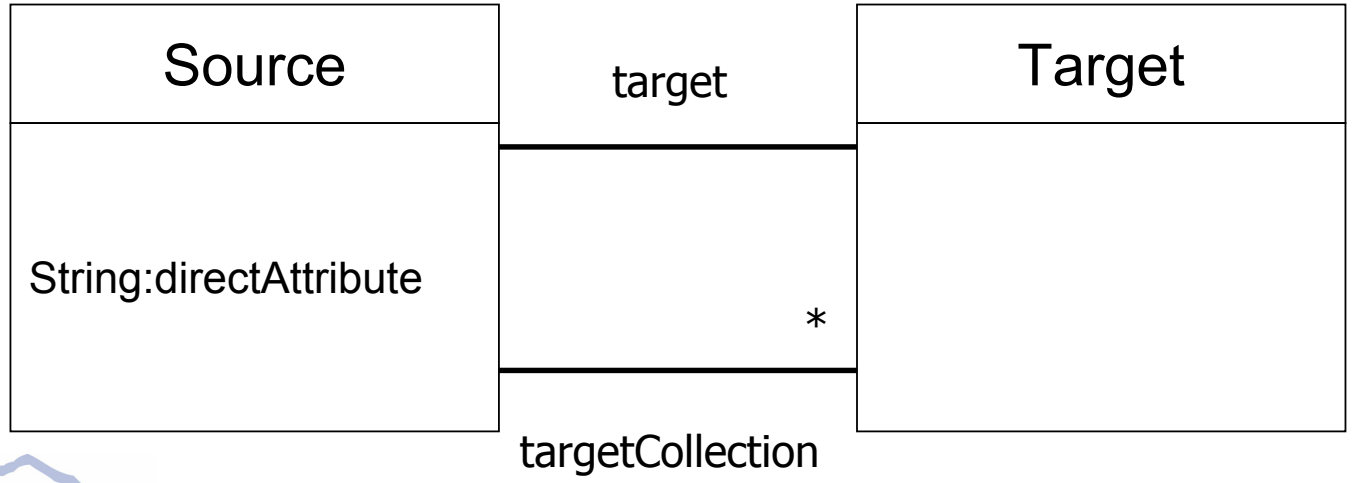
Mapping Terminology

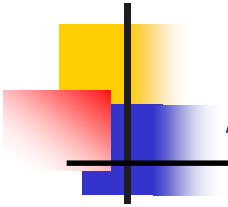
- Direct Mappings
  - Field in Object model maps to field in datasource
    - May require conversion, aggregation or serialization
- Relationship Mappings
  - Thought of always as uni-directional
    - "Bi-directional" relationships are two distinct mappings
  - Source is the originator of the relationship
  - Target is referenced object or set of objects from the source



Mapping Terminology

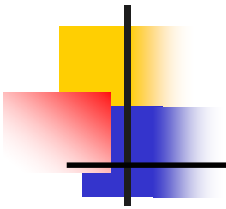
```
public class Source {
  private String directAttribute;
  private Target target;
  private Collection targetCollection;

}
```



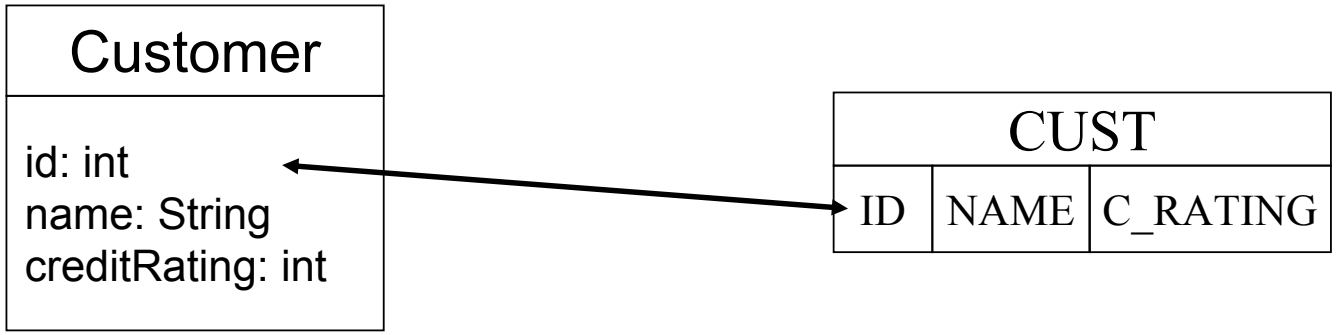


Agenda

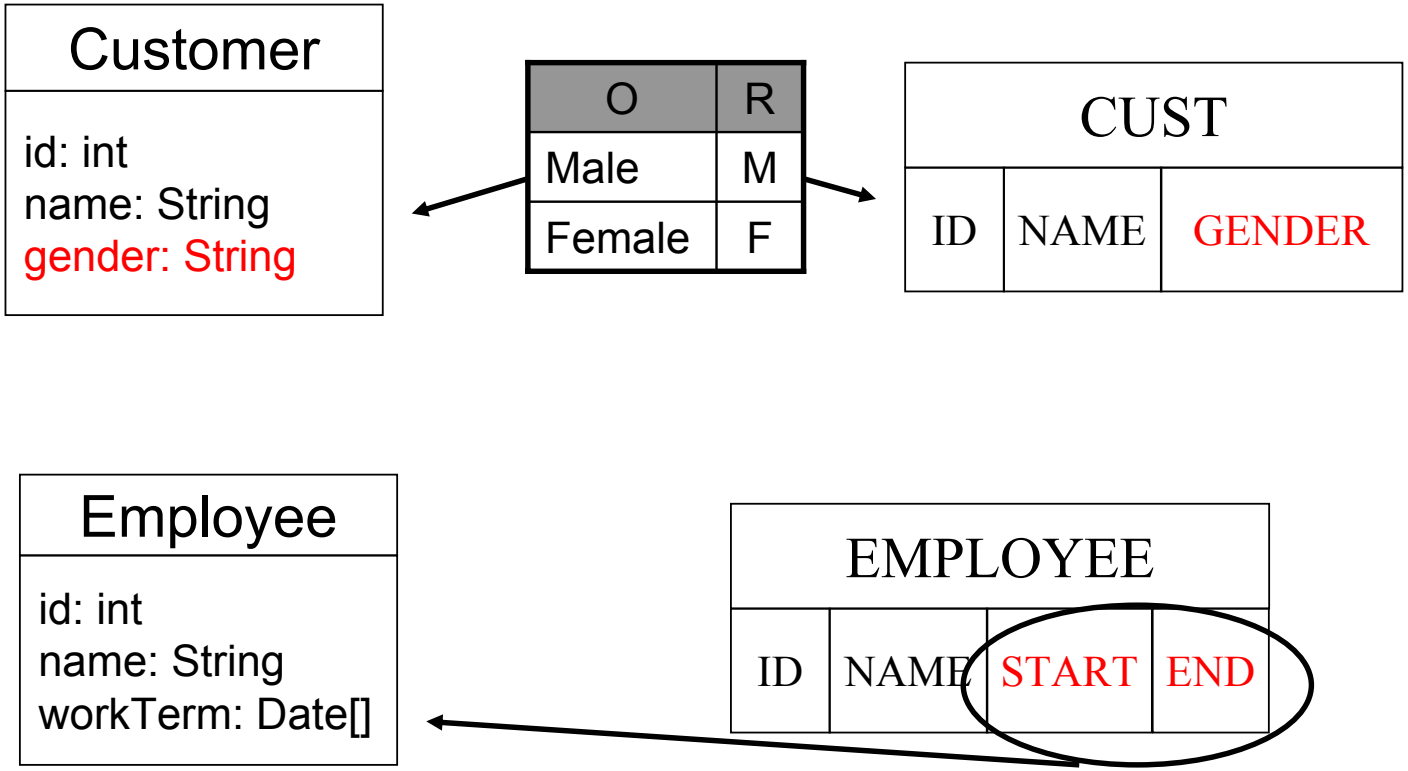
- Quick Review of Java Mapping
- O-R Mapping
  - Direct, 1-1, Aggregate, 1-M, M-M, Inheritance
- O-X Mapping

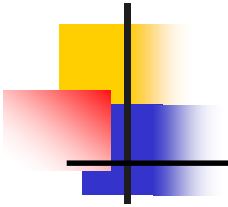


Simple Direct Mapping



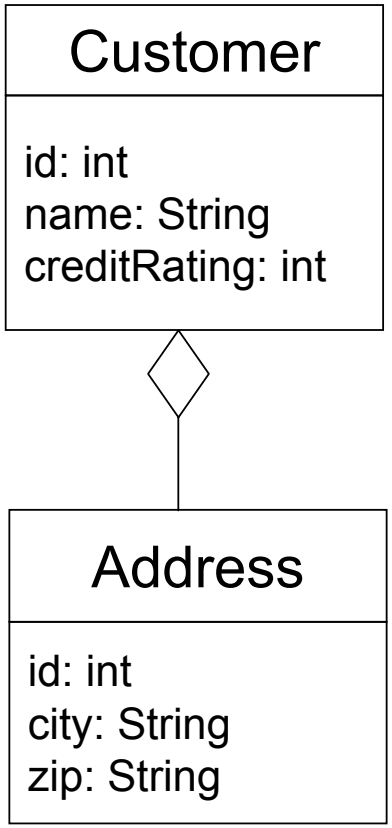
Direct Conversion Mappings



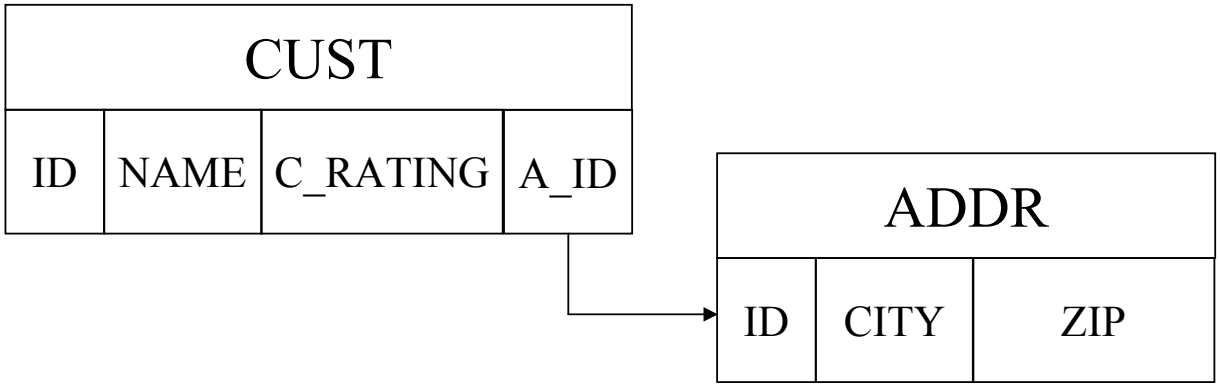


1-1 Object Model

1:1 Relationship



Typical 1-1 Relationship



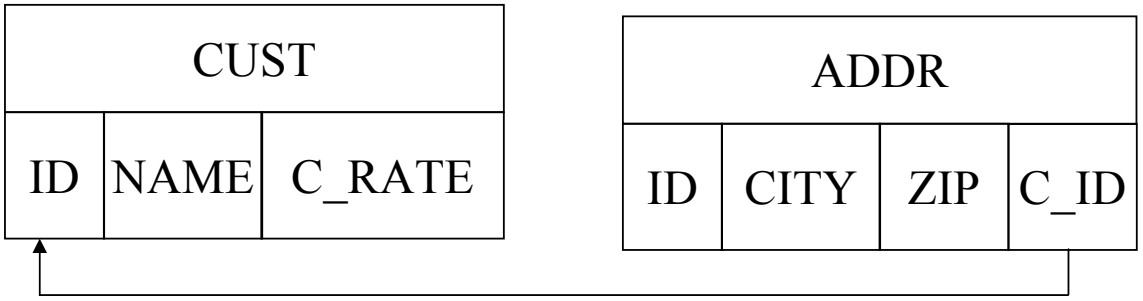
CUST

ID	NAME	A_ID
7	Don	3
13	Trisha	17

ADDR

ID	CITY	ZIP
3	Ottawa	K2J4X7
17	Toronto	K1P2T2

1-1 FK in Target Table



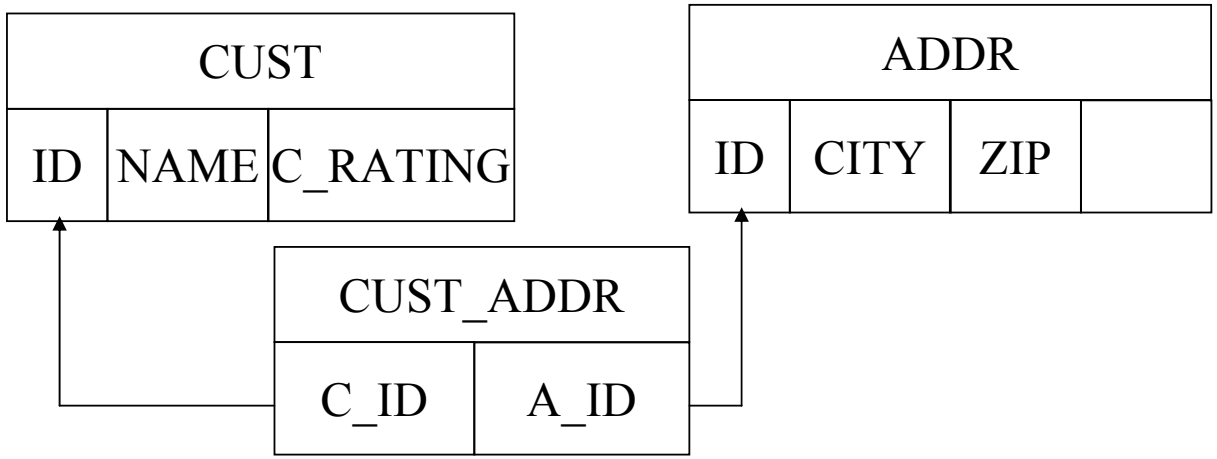
CUST

ID	NAME
7	Don
13	Trisha

ADDR

ID	CITY	ZIP	C_ID
3	Ottawa	K2J4X7	7
17	Toronto	K1P2T2	13

Association Table



CUST

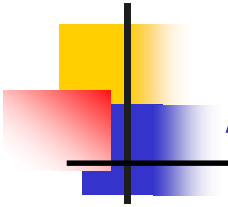
ID	NAME
7	Don
13	Trisha

ADDR

ID	CITY	ZIP
3	Ottawa	K2J4X7
17	Toronto	K1P2T2

CUST_ADDR

C_ID	A_ID
7	3
13	17



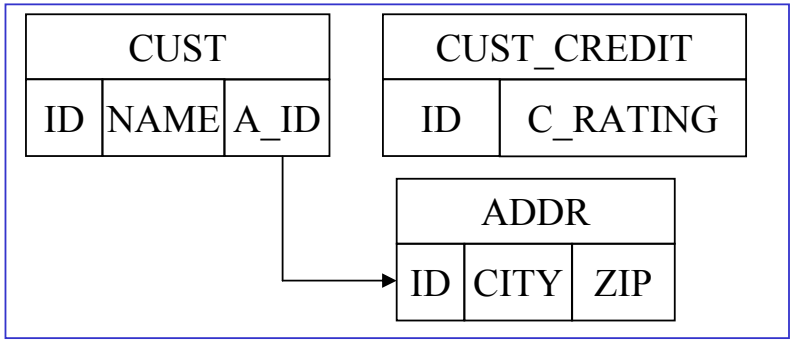
Aggregation

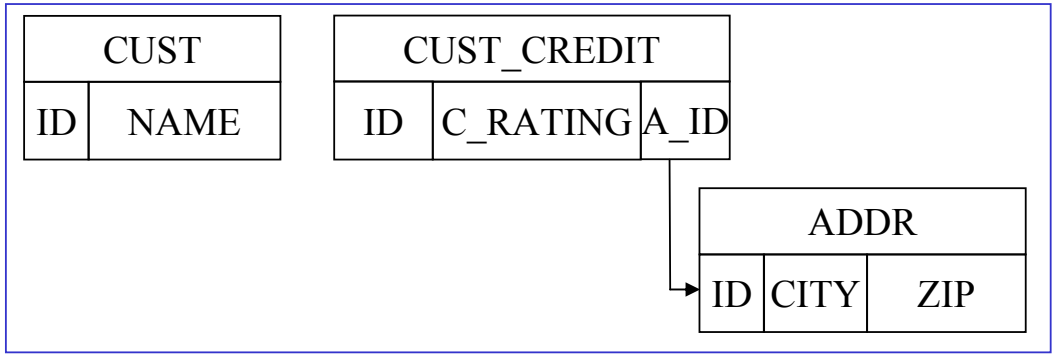
CUST				
ID	NAME	C_RATING	CITY	ZIP

CUST

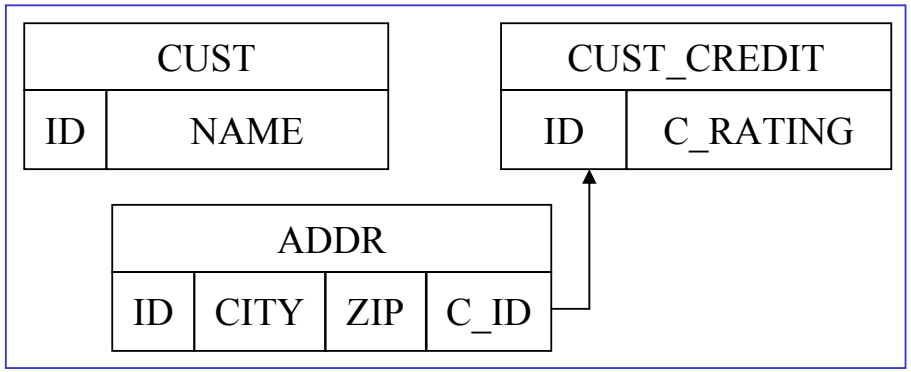
ID	NAME	CITY	ZIP
7	Don	Ottawa	K2J4X7
13	Trisha	Toronto	K1P2T2

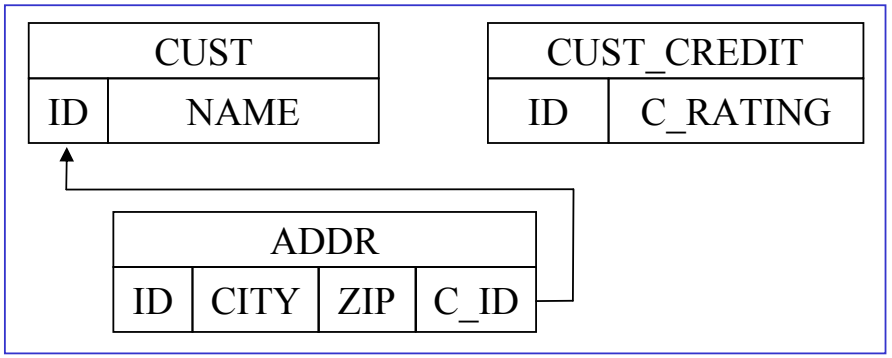
Multiple Table Mappings

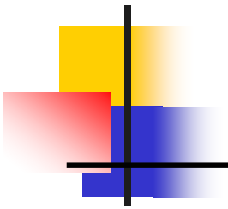




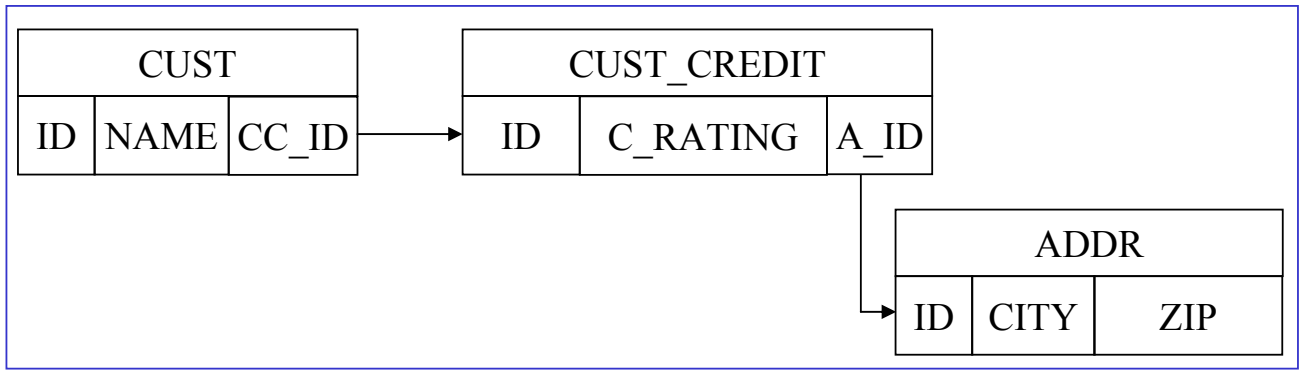
Multiple Table Mappings

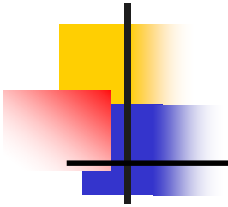






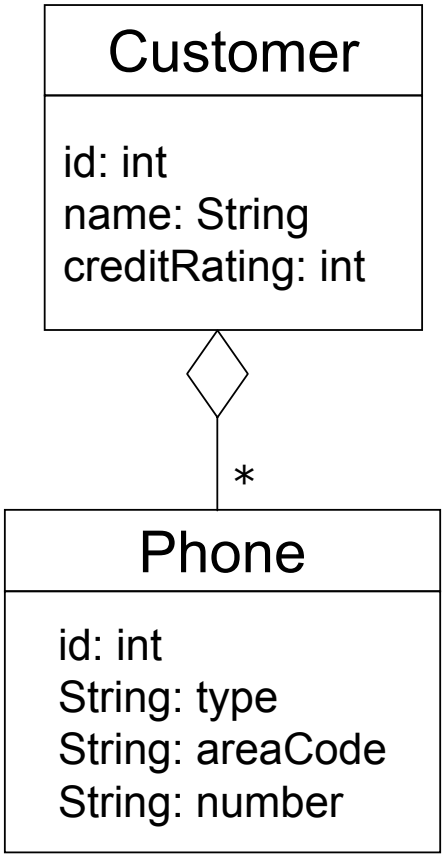
Multiple Table Mappings

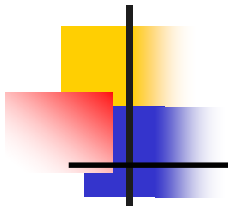




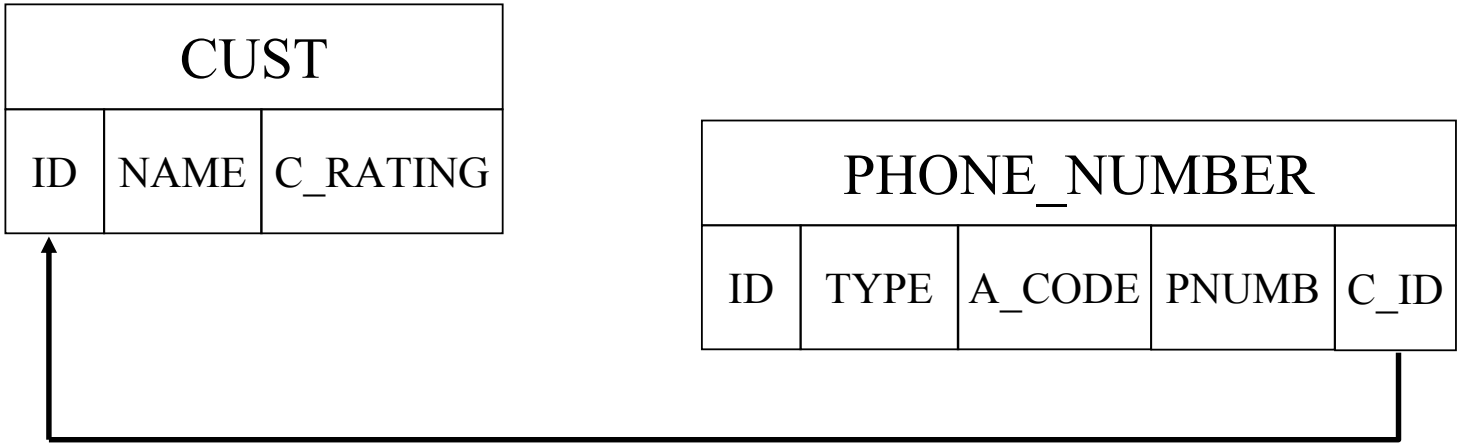
1-M Object Model

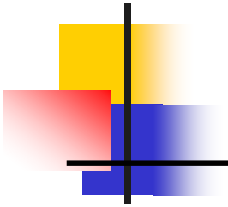
1:M Relationship





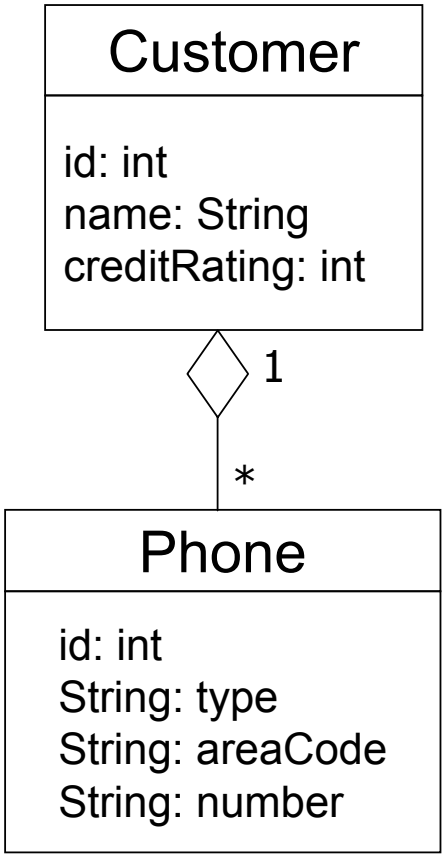
1-M Relationship





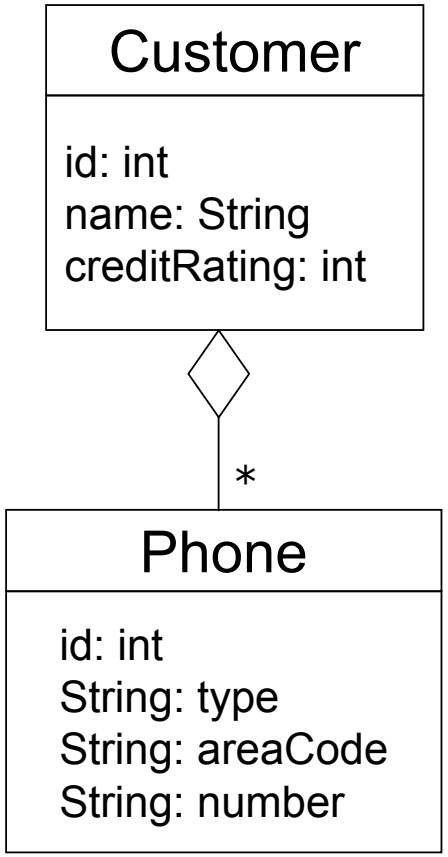
1-M Relationship Trick

- Mapping a 1-1 "back reference" from the target of the 1-M to the source
- Allows persistence layer to more easily handle the foreign key



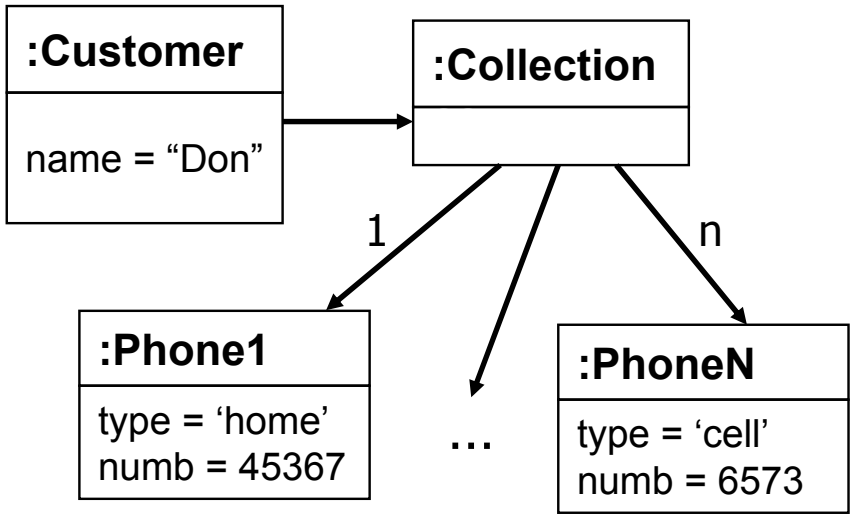
1-M Relationship Trap!

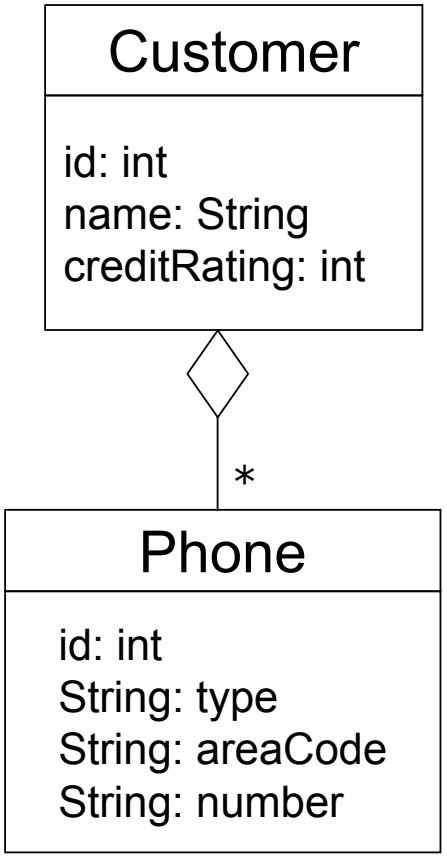
- What if order is important?
- Specify field to order by

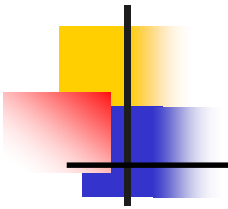


1-M Relationship Trap!

- What if *collection index* is the order criteria?



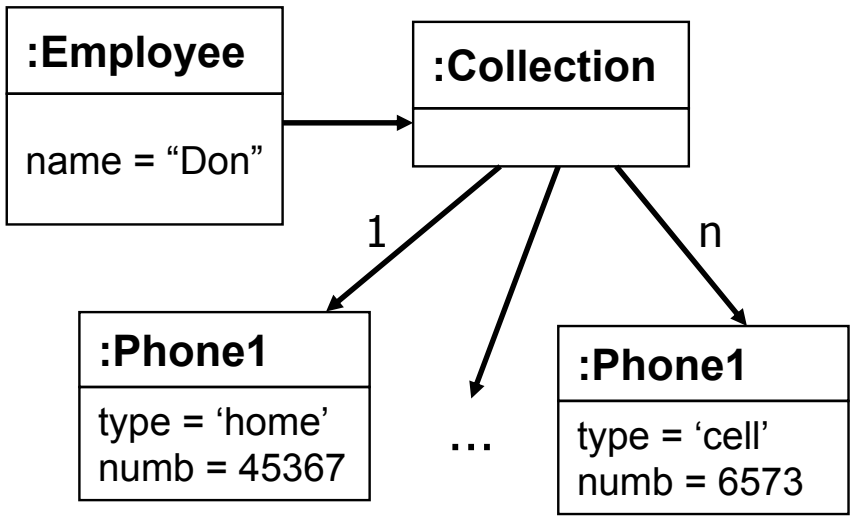




1-M Relationship Trap!

- Challenging problem because "state" from Java utility class needs to be persisted
- Best solution is to find or make a persistent attribute of the target the order by field
- Regardless, database requires SOMETHING to order by

1-M Relationship Trap!

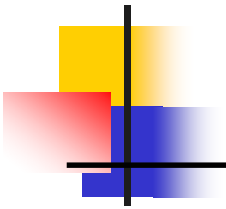


EMP

ID	NAME
7	Don

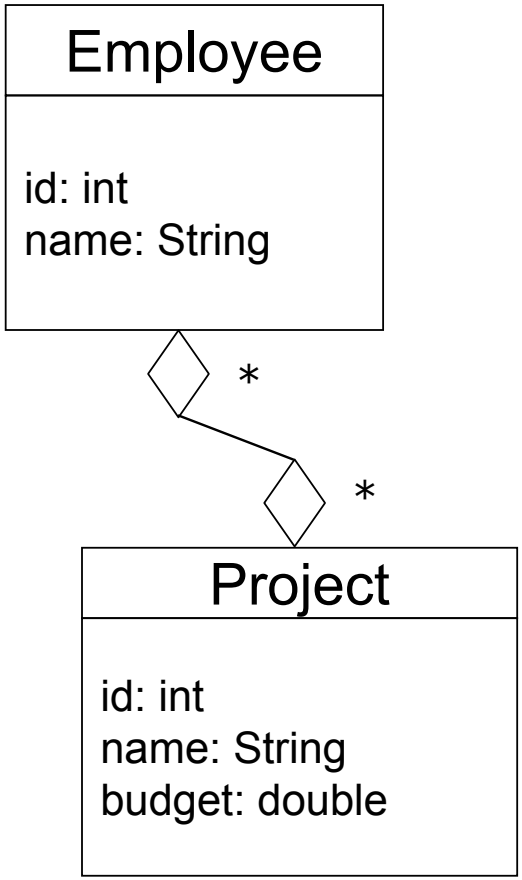
PHONE

ID	TYPE	NUMBER	ORDER
3	home	45367	1
17	cell	6573	3
15	work	99882	2

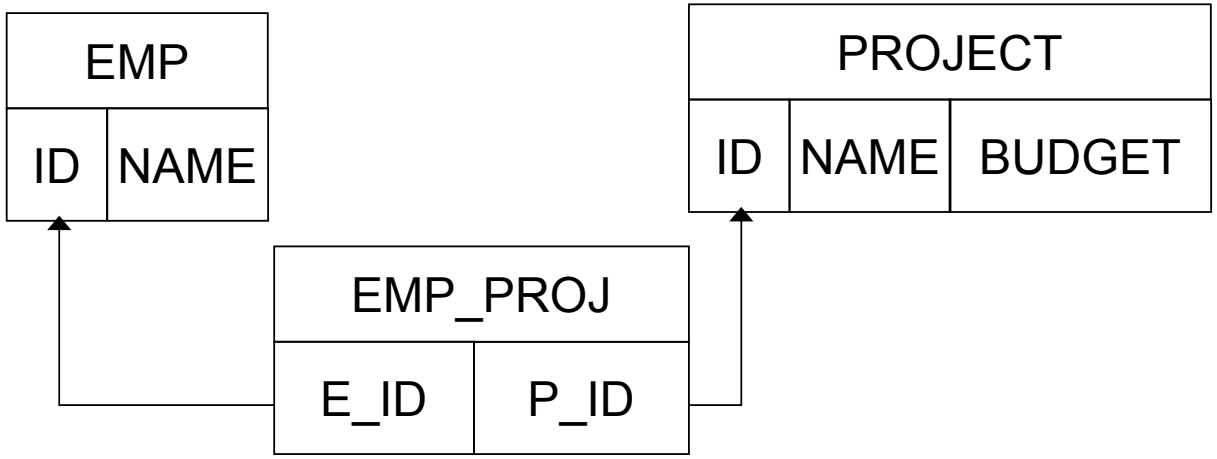


M-M Object Model

M:M Relationship



M-M Relationship



EMP

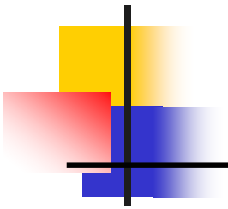
ID	NAME
7	Don
13	Mike

PROJECT

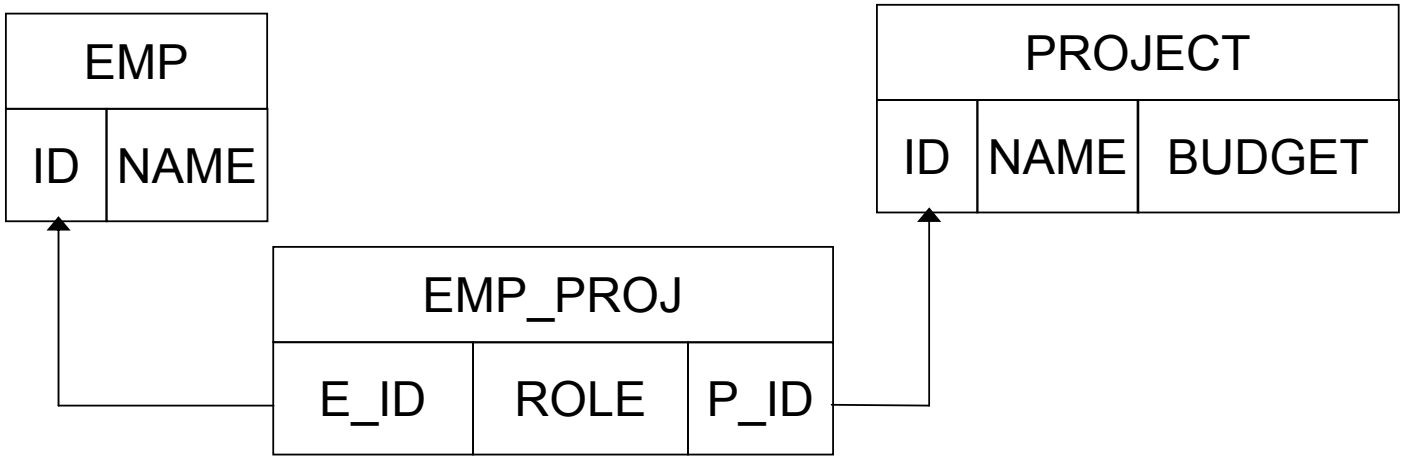
ID	NAME	BUDGET
3	Tiger	5,000,000
17	Dragon	1,000,000

EMP_PROJ

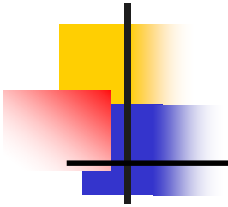
E_ID	P_ID
7	3
7	17



M-M Relationship Trap!

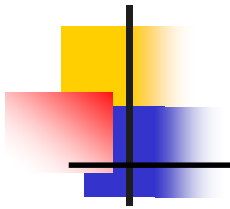


Common challenge – RDBs often intertwine data and relationships. Where would this go in object model?

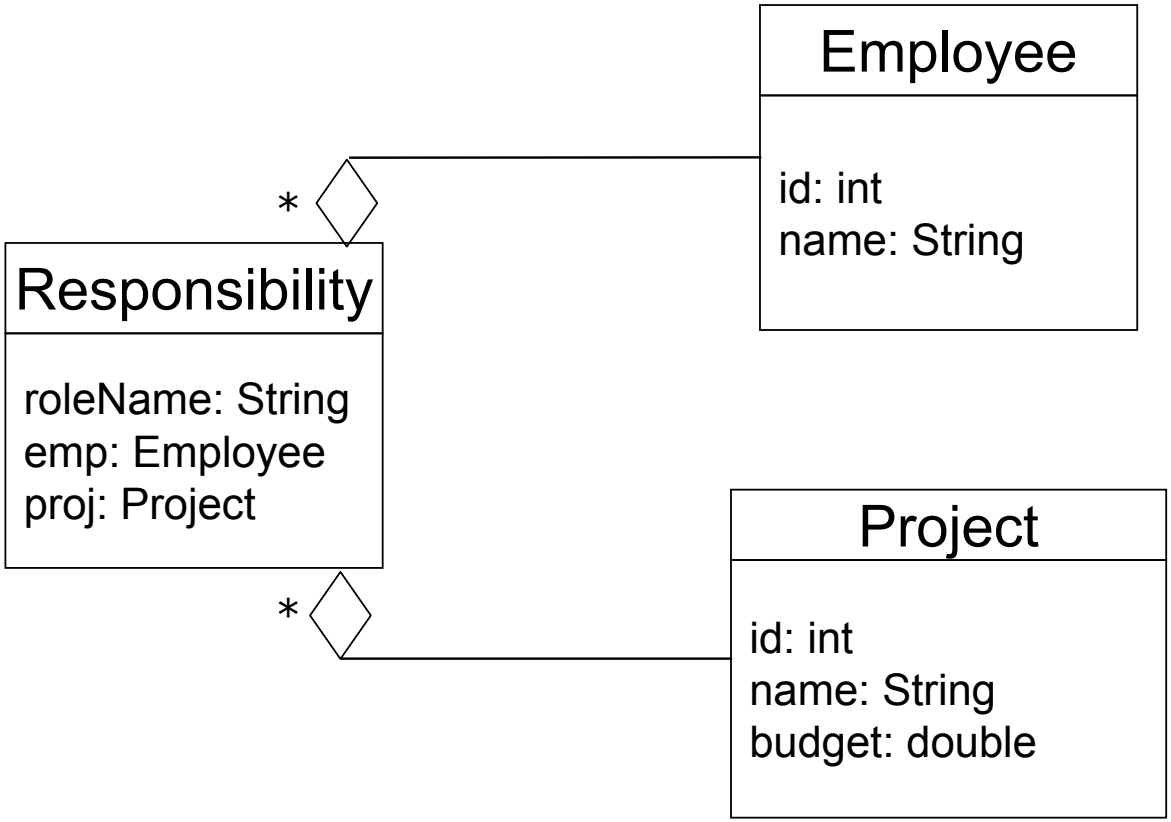


M-M Relationship Trap!

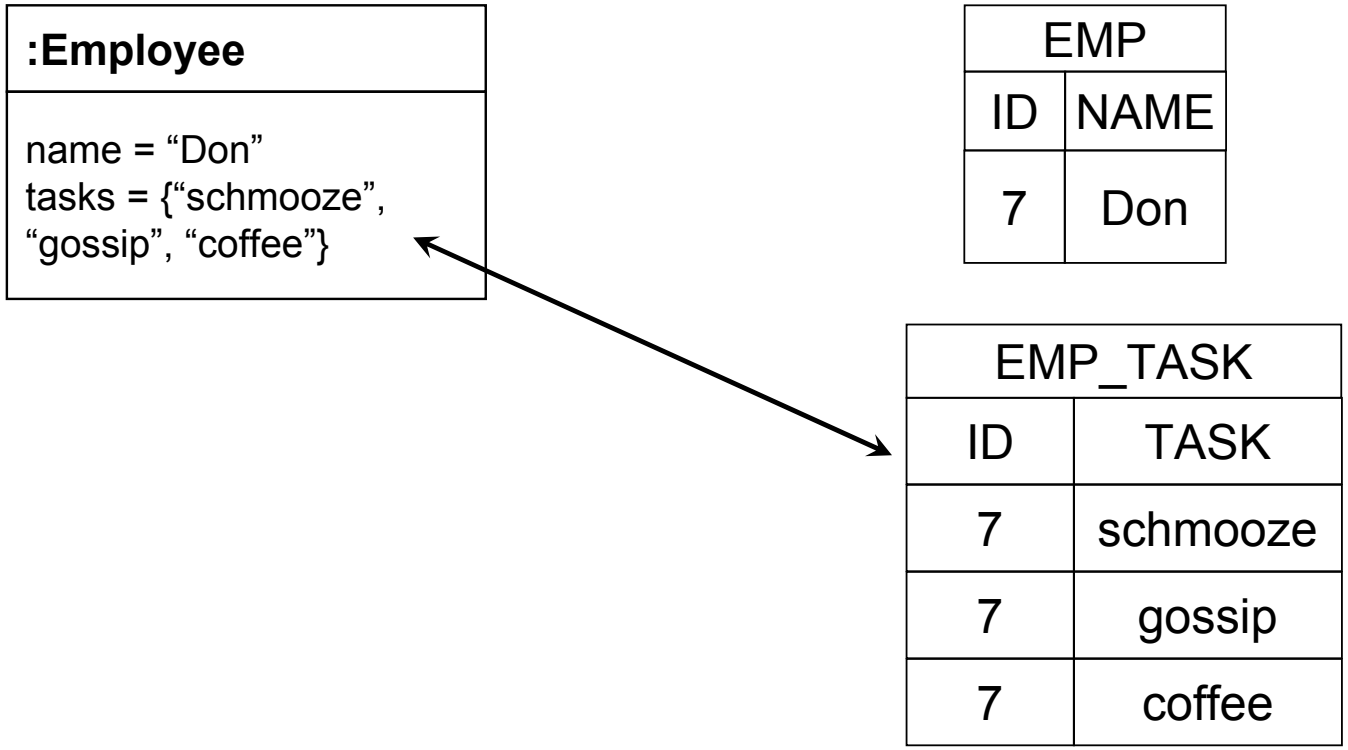
- In relational world, putting "data" on a relationship is not uncommon
- Not possible in Java
- Must store data in an object – requires some re-thinking of Object model

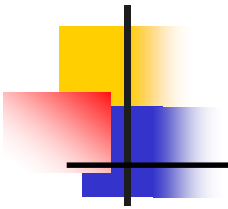


M-M Trap Solution



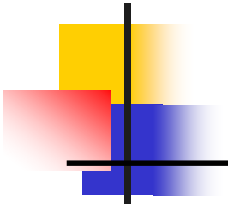
Direct Collection Mapping





Direct Collection Issues

- Inefficient for large direct collections
  - Changes require lots of "change detection" analysis, or massive purge and insert on database
- Only useful for PRIMITIVES, more than 1 piece of date requires 1-M mapping
- Consider Maps of Primitives…



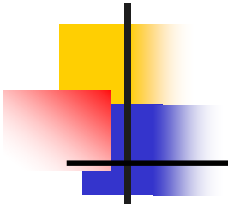
Direct Map Mapping

:Employee

name = “Don”
contact = {
“email” -> “d@m.com”,
“home” -> “555-1212”,
“cell” -> “555-2323”
}

EMP	
ID	NAME
7	Don

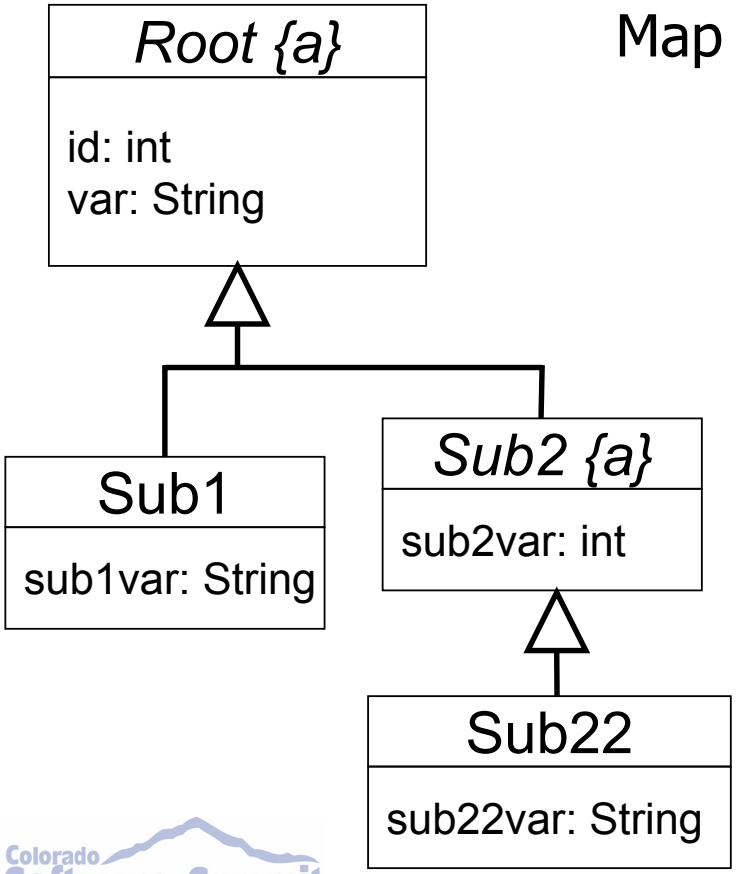
CONTACT		
ID	KEY	VALUE
7	email	d@m.com
7	home	555-1212
7	cell	555-2323



Inheritance

- Very straight forward at first, but can become challenging because of the number of possible solutions
- Compounded with relationships
- Can be source of contention with DBA – generally leads to lots of joining on queries!

"Leaf Table Mapping"



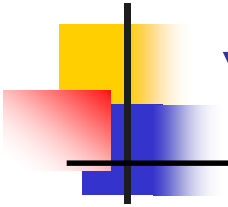
Map only concrete classes

SUB1

ID	VAR	SUB1VAR
7	Foo	Pish

SUB22

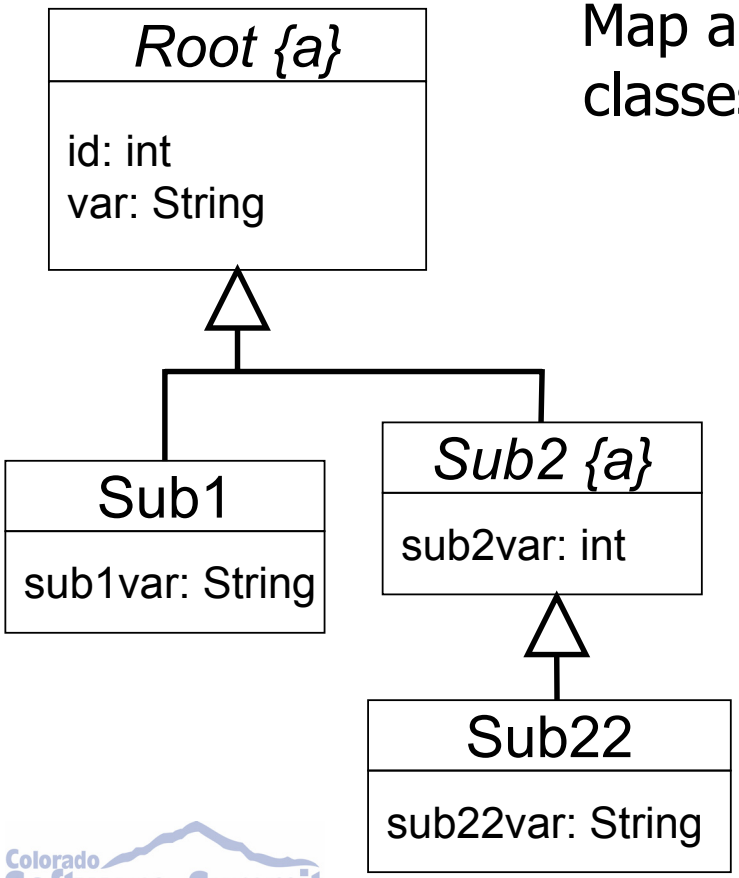
ID	VAR	SUB2VAR	SUB22VAR
13	Tim	677	Flim



"Leaf Table" Issues

- How do you query for the Root class?
- Data is not normalized

"All Table Mapping"



Map all classes

ROOT

ID	VAR
7	Foo
13	Bar

SUB1

ID	SUB1VAR
7	Pish

Problem! How to tell type?!

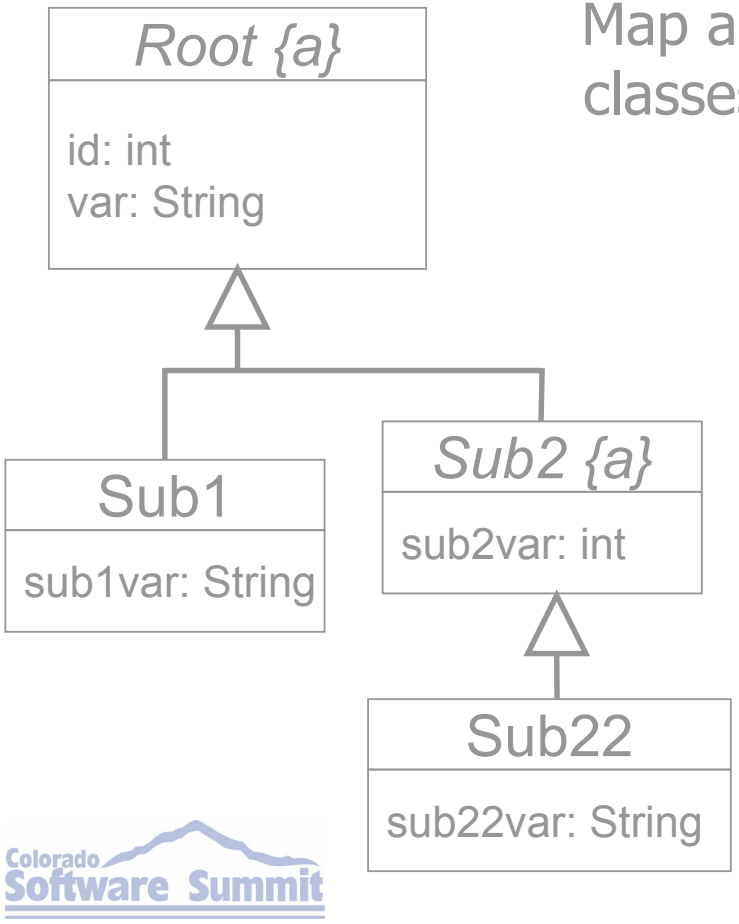
SUB2

ID	SUB2VAR
13	677

SUB22

ID	SUB22VAR
13	Flim

Identifying Type



Map all classes

ROOT

ID	VAR	TYPE
7	Foo	S1
13	Bar	S22

Need Type Identifier!

SUB1

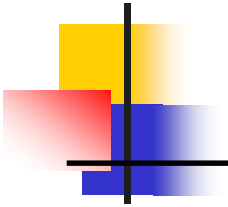
ID	SUB1VAR
7	Pish

SUB2

ID	SUB2VAR
13	677

SUB22

ID	SUB22VAR
13	Flim



Identifying Type

ROOT

ID	VAR	TYPE
7	Foo	S1
13	Bar	S22

Lots of strategies exist…

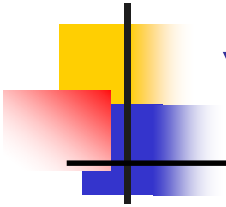
ROOT

ID	VAR	TYPE
7	Foo	com.foo.Sub1
13	Bar	com.foo.Sub22

ROOT

ID	VAR
7	Foo
13	Bar

```
if id < 10
    type = Sub1
else
    type = Sub22
```

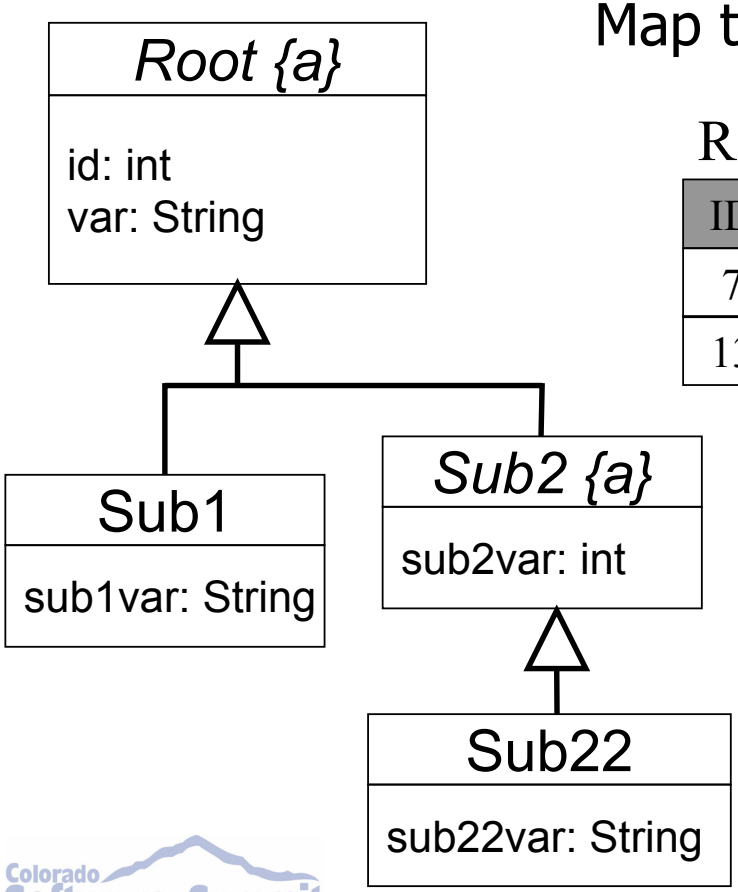


"All Table" Issues

- Lots of joining
- Type identification can be inefficient

"Single Table Mapping"

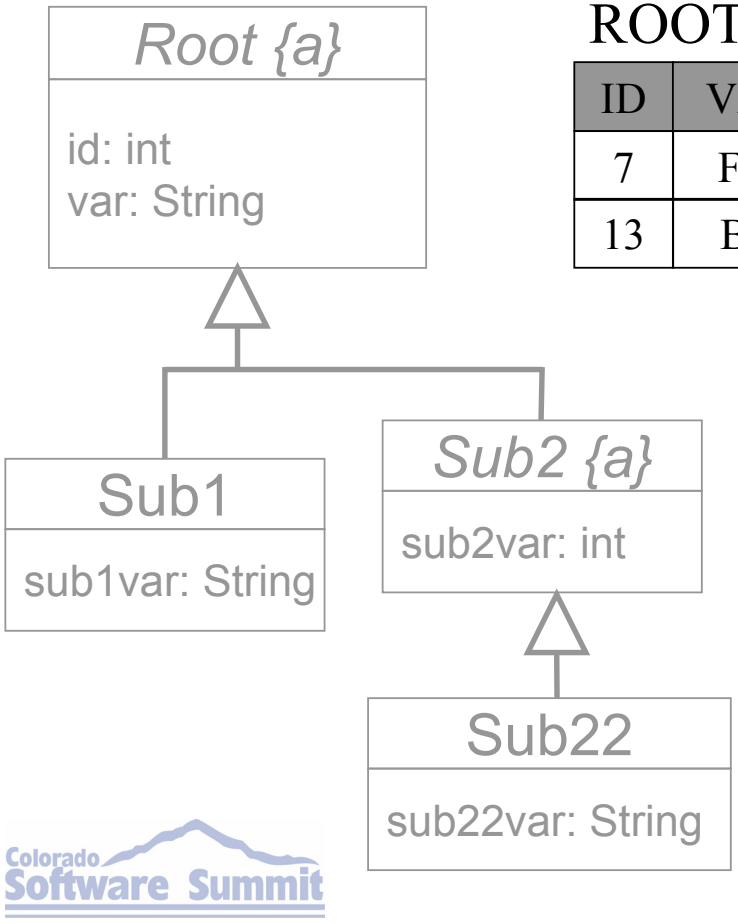
Map to one table



ROOT

ID	VAR	SUB1VAR	SUB2VAR	SUB22VAR
7	Foo	Pish	null	null
13	Bar	null	677	Flim

Identifying Type

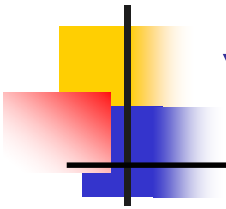


ROOT

ID	VAR	SUB1VAR	SUB2VAR	SUB22VAR	TYPE
7	Foo	Pish	null	null	S1
13	Bar	null	677	Flim	SUB22

Two options for type detection

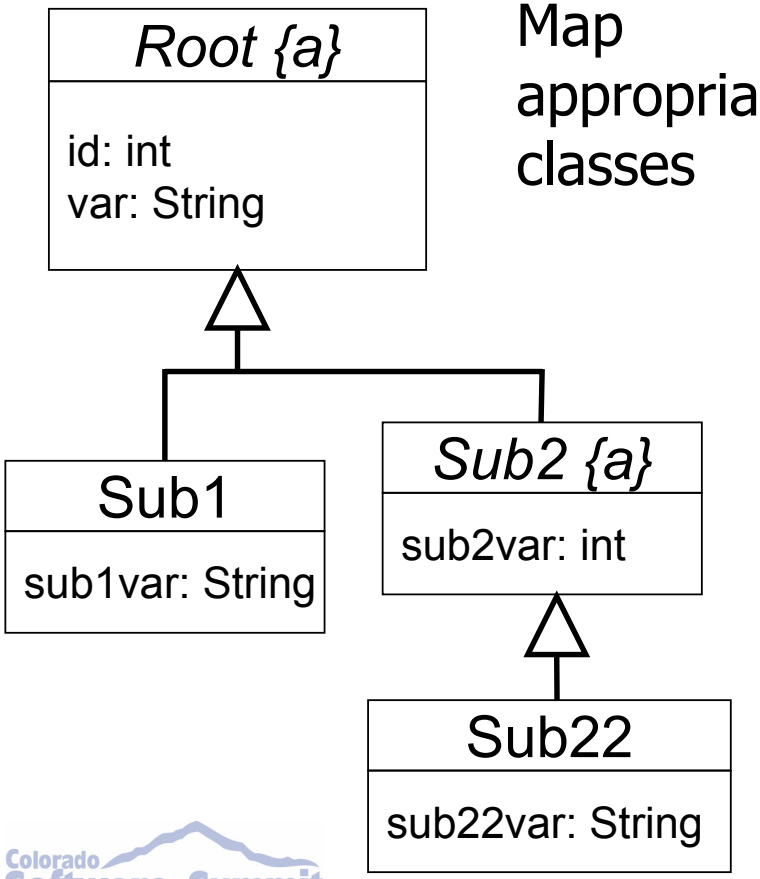
```
if SUB1VAR != null
    type = Sub1
else
    type = Sub22
```



"Single Table" Issues

- DBA will freak out – de-normalized database
- Type identification can be inefficient

"Combination Table Mapping"



Map appropriate classes

ROOT

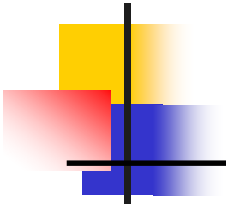
ID	VAR	TYPE
7	Foo	S1
13	Bar	S22

SUB1

ID	SUB1VAR
7	Pish

SUB22

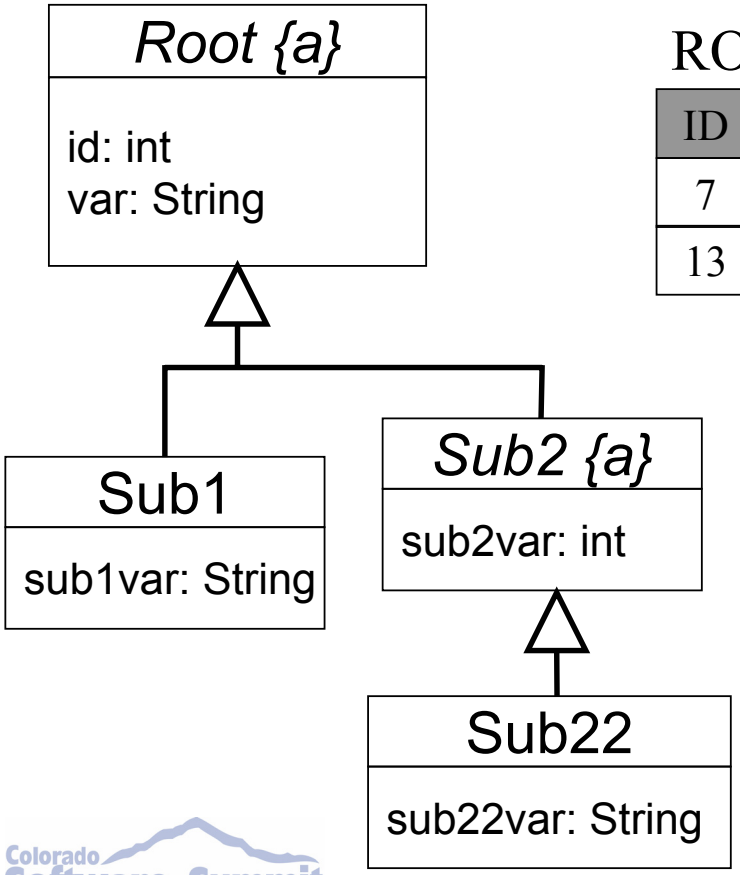
ID	SUB22VAR	SUB2VAR
13	Flim	677



Which Strategy to Use?

- Leaf Table
  - When never querying at abstract class level
- All Table
  - When not doing much querying
    - Too many joins
- Single Table
  - When one subclass is much more prevalent, and your DBA doesn't care about normalization
- Combination
  - Use to optimize above situations

Wouldn't It Be Cool?



Map to view for reads...

ROOT (*View*)

ID	VAR	SUB1VAR	SUB2VAR	SUB22VAR
7	Foo	Pish	null	null
13	Bar	null	677	Flim

Map to tables for writes...

ROOT

ID	VAR
7	Foo
13	Bar

SUB1

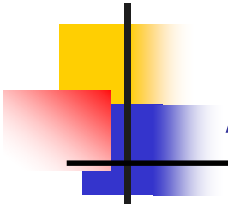
ID	SUB1VAR
7	Pish

SUB2

ID	SUB2VAR
13	677

SUB22

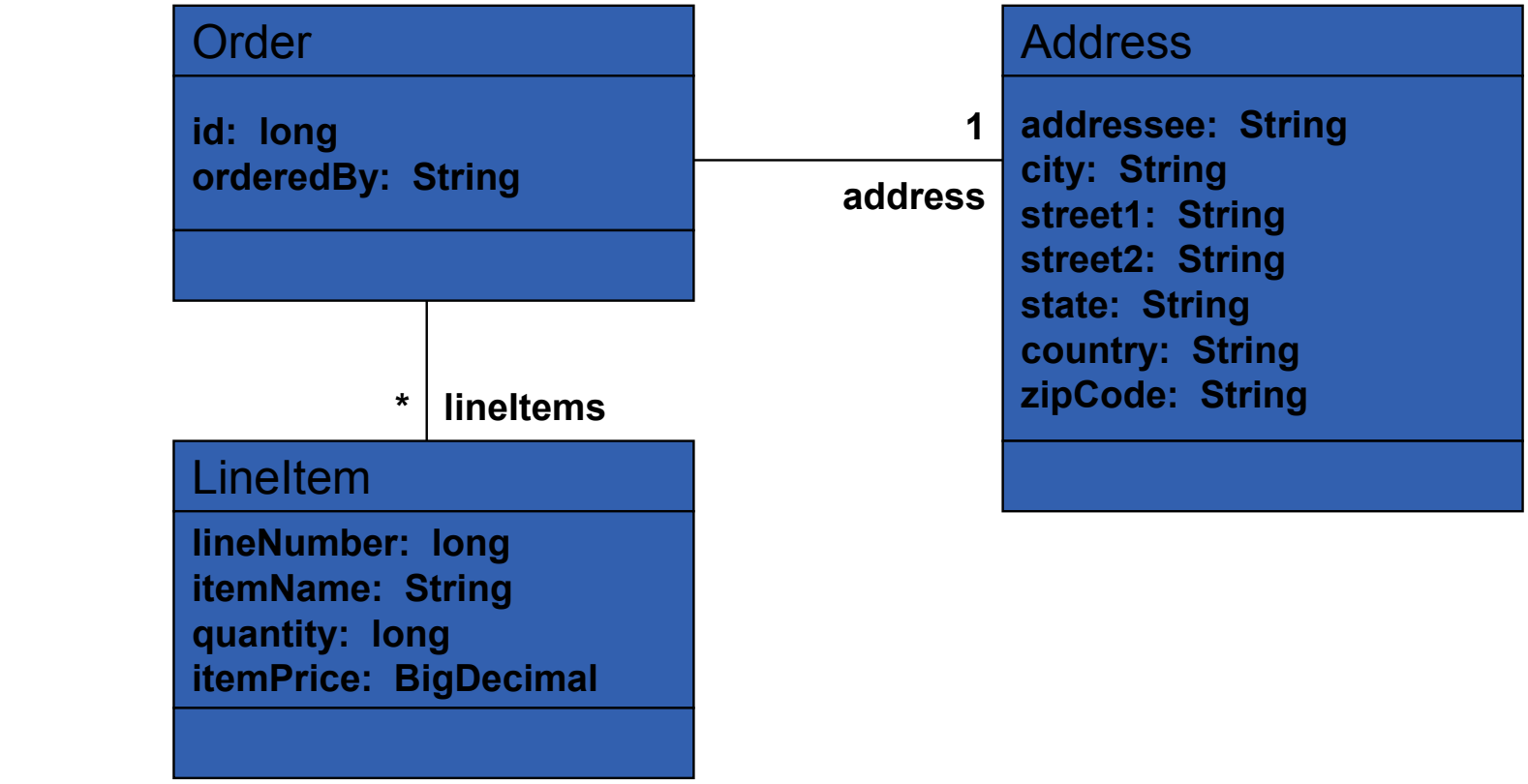
ID	SUB22VAR
13	Flim



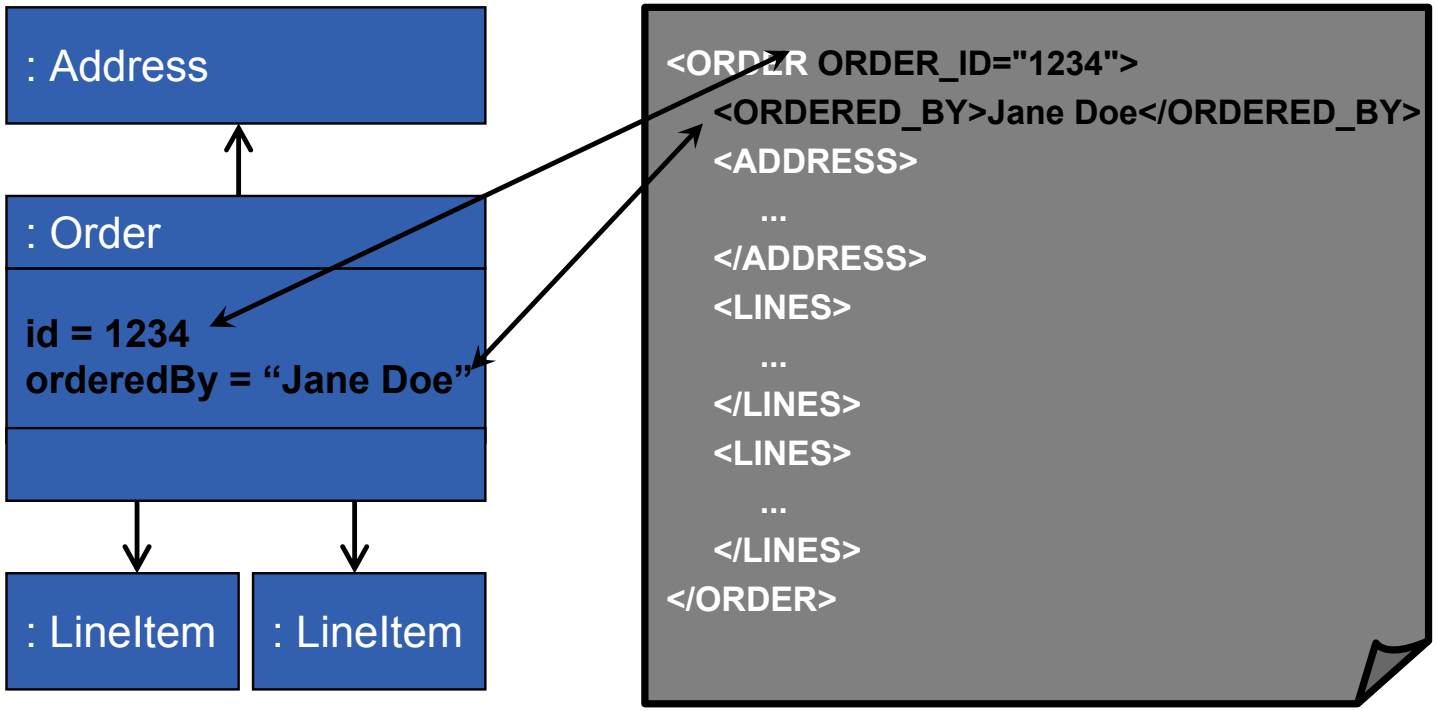
Agenda

- Quick Review of Java Mapping
- O-R Mapping
- O-X Mapping
  - Direct, Composite, Transformation, Inheritance

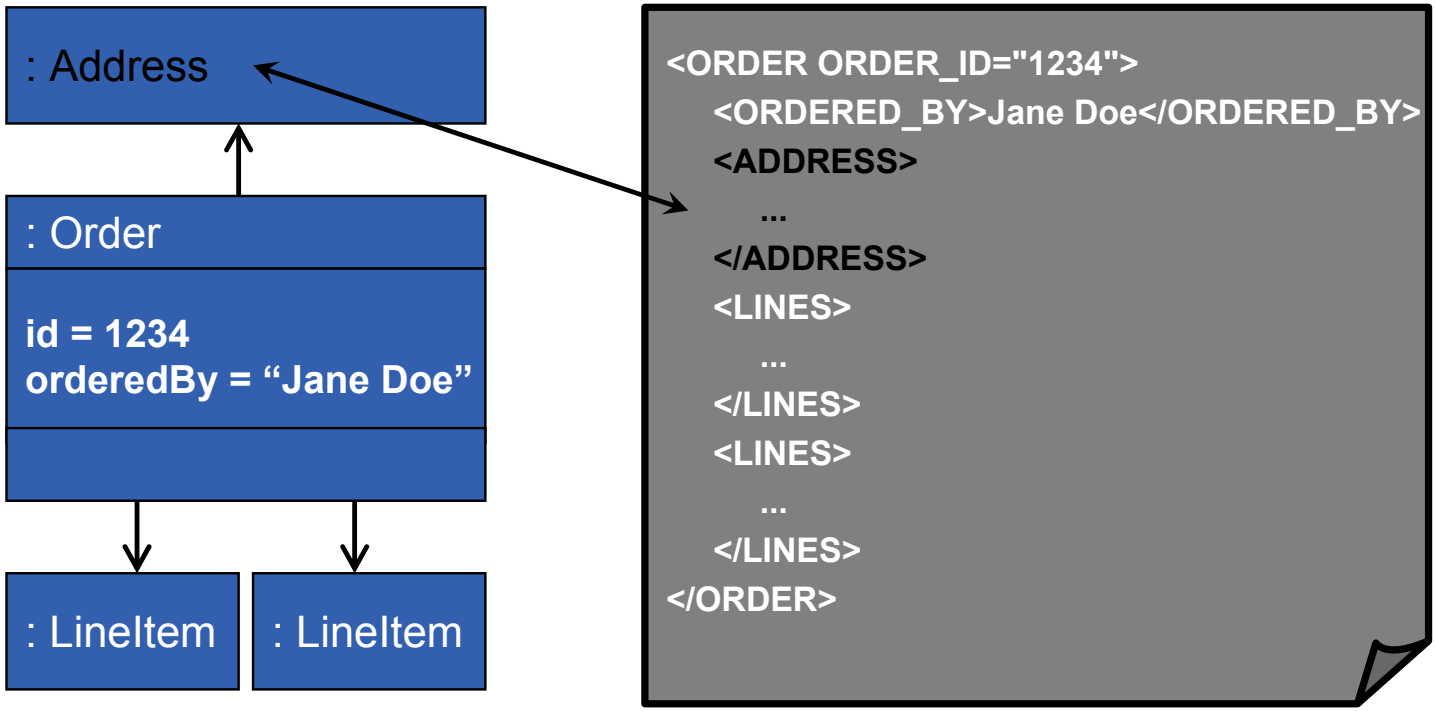
Example Object Model



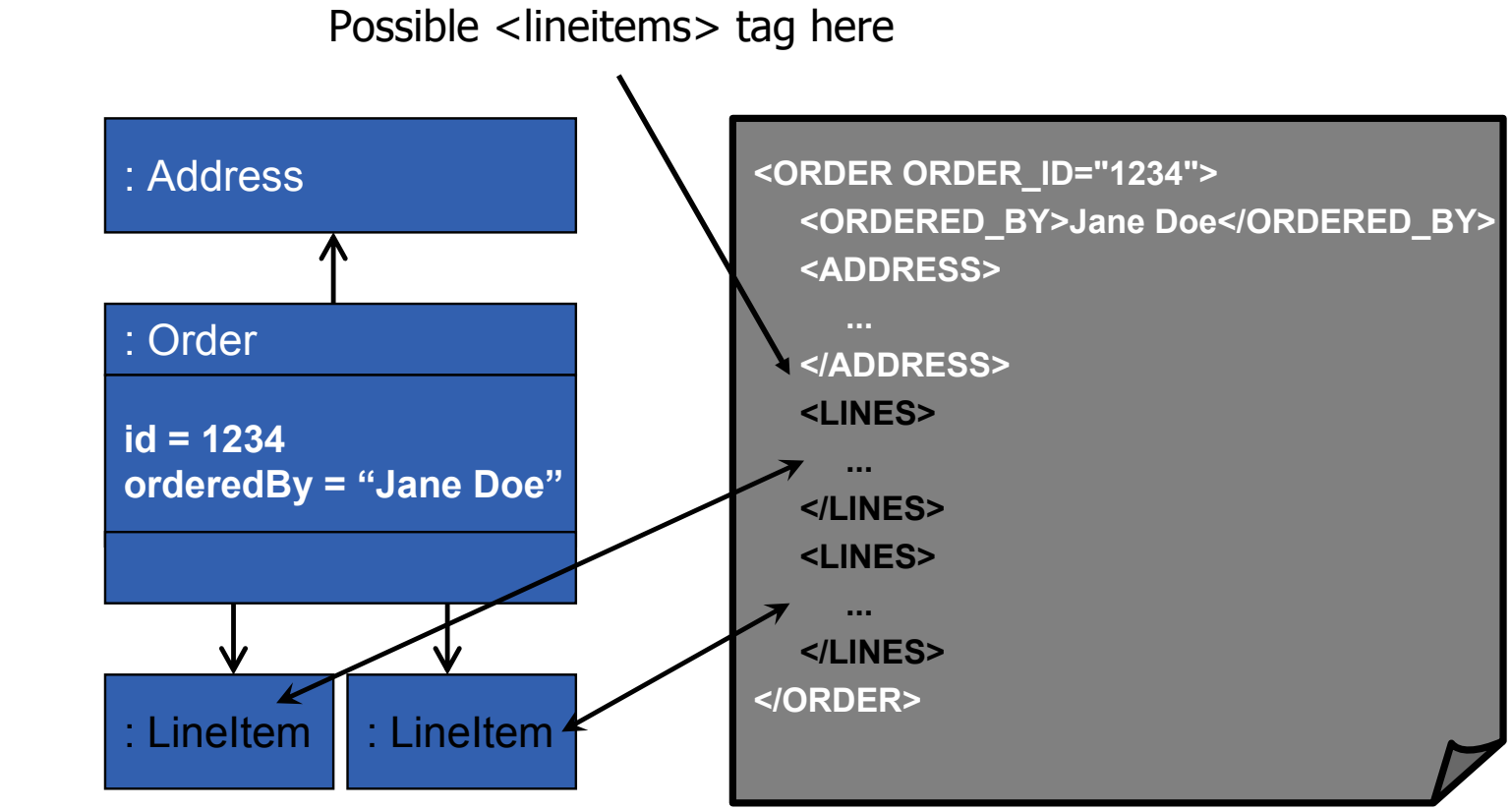
Direct Mapping

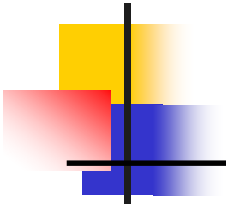


Composite Object Mapping



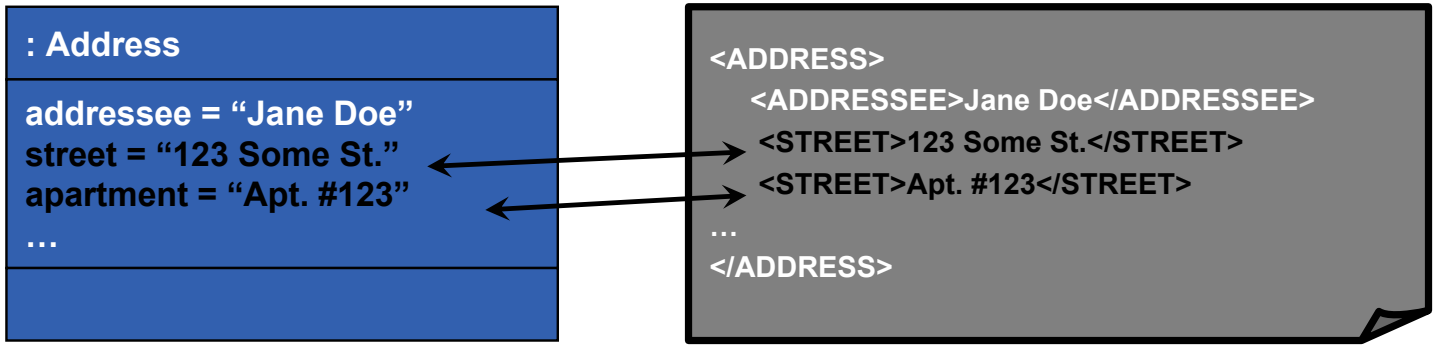
Composite Collection Mapping



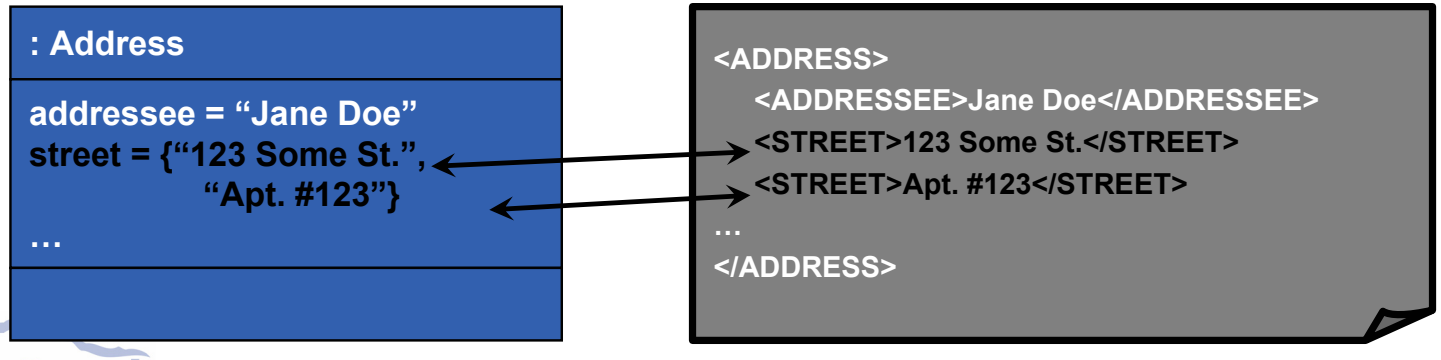


Positional Information

Good O-X Support

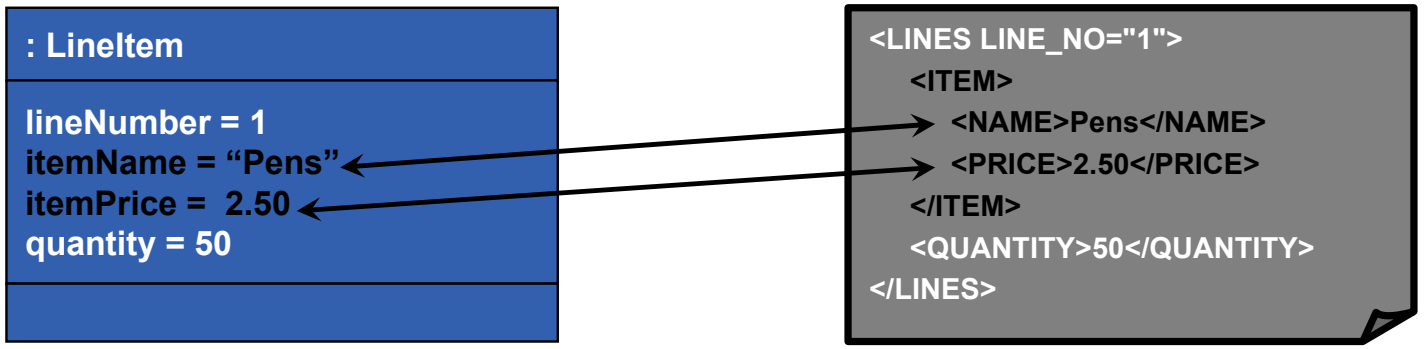


JAXB/Class Generation Today…

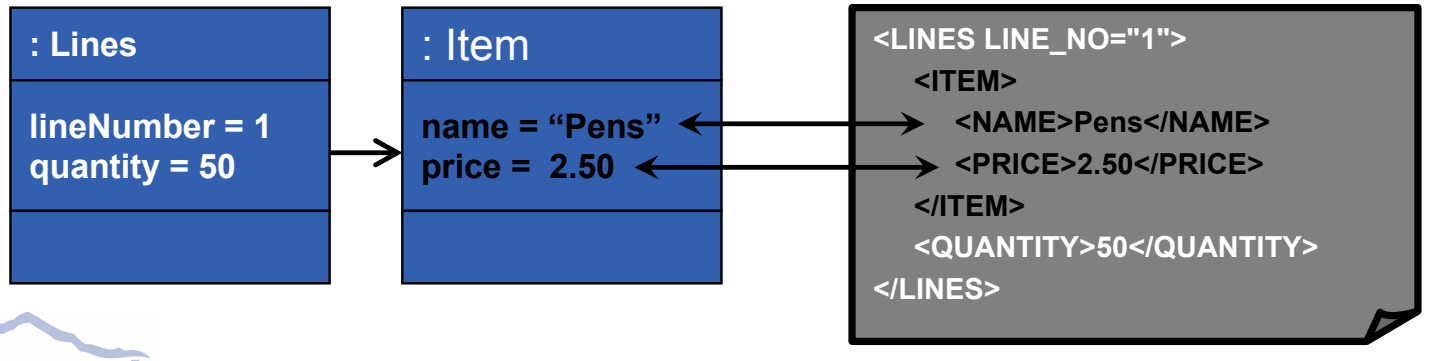


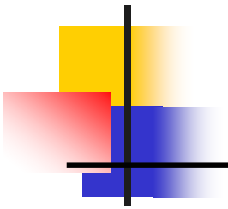
Path Information

Good O-X Mapping



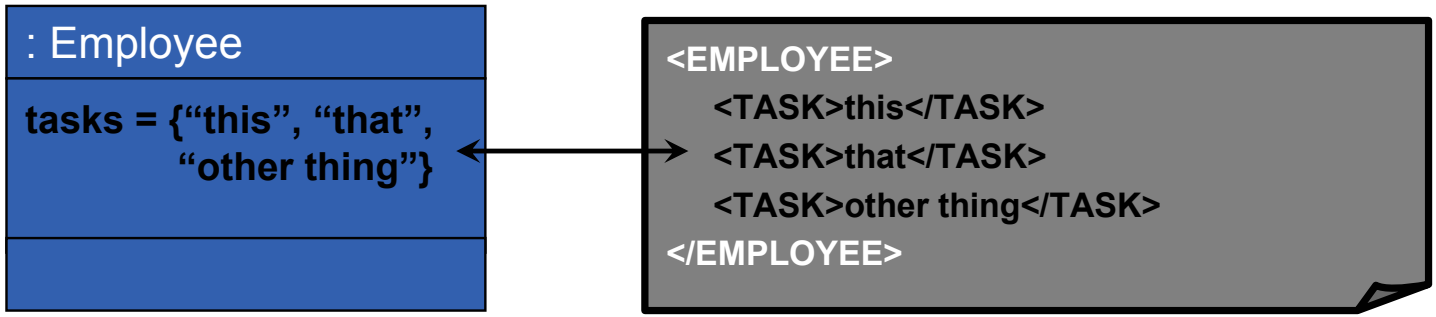
JAXB/Class Generation Today…





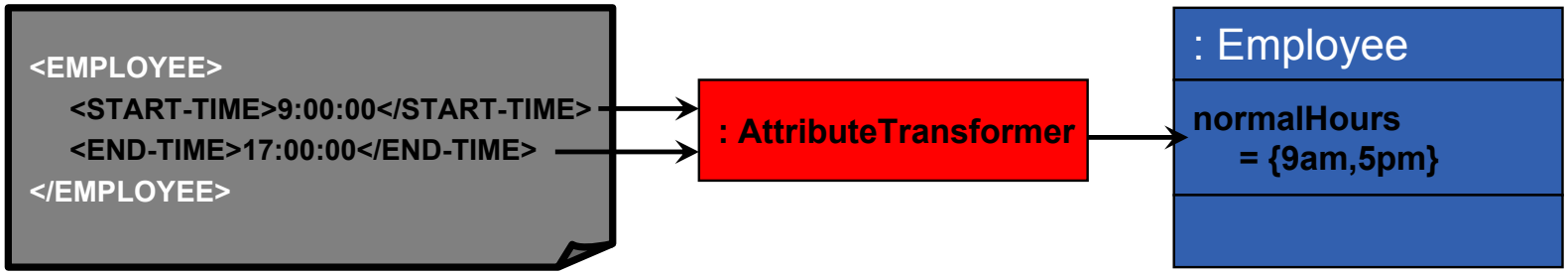
Direct Collection Mapping

- Compare with Positional Mapping

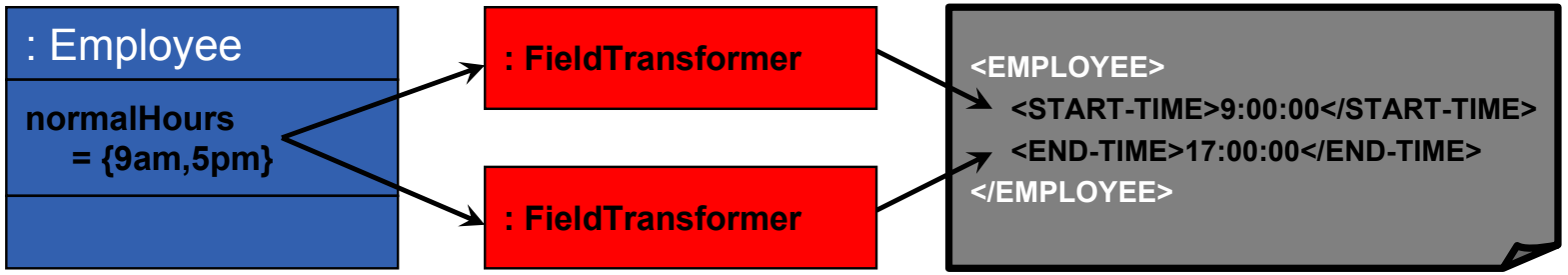


Transformation Mapping

Unmarshal (Read)



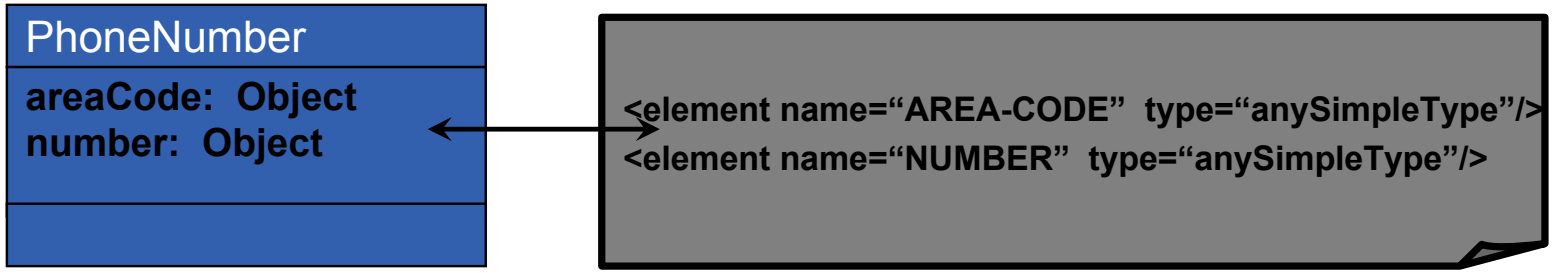
Marshal (Write)



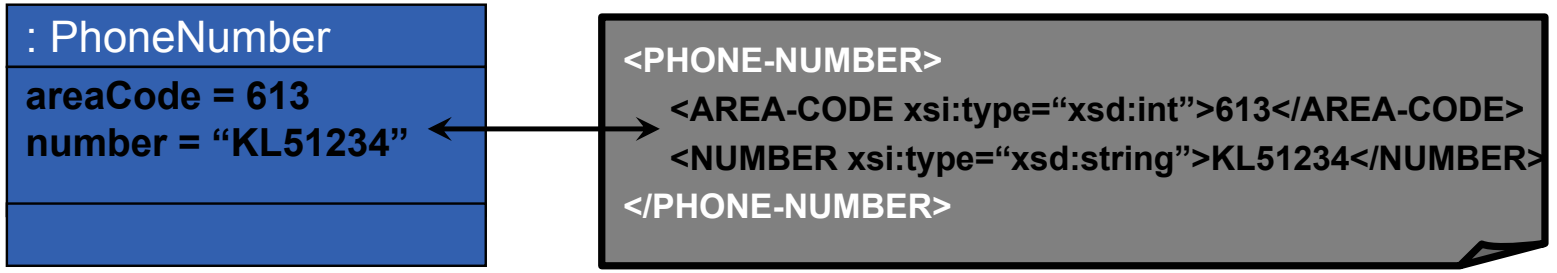
Conversion Mapping

Simple Type Translator

Class Model



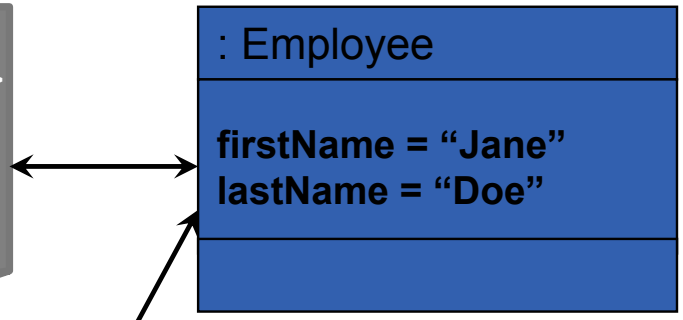
Instance Model



Namespace Support

```
<abc:EMPLOYEE xmlns:abc="http://X" xmlns:def="http://Y">
    <def:FIRST-NAME>Jane</def:FIRST-NAME>
    <def:LAST-NAME>Doe</def:LAST-NAME>
</abc:EMPLOYEE>
```

```
<ns1:EMPLOYEE xmlns:ns1="http://X" xmlns:ns2="http://Y">
    <ns2:FIRST-NAME>Jane</ns2:FIRST-NAME>
    <ns2:LAST-NAME>Doe</ns2:LAST-NAME>
</ns1:EMPLOYEE>
```



Employee Default Root = abc:EMPLOYEE
XPath firstName = def:FIRST-NAME
XPath lastName = def:LAST-NAME

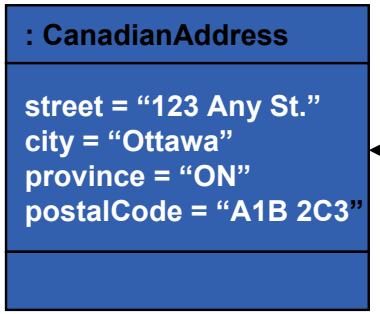
Namespace Resolver:
abc = http://X
def = http://Y

Inheritance

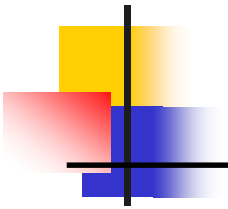
Class Model


```
<element name="ADDRESS" type="ADDRESS-TYPE"/>
<complexType name="ADDRESS-TYPE">
    .....
</complexType>
<complexType name="CANADIAN-ADDRESS-TYPE>
    <complexContent>
        <extension base="ADDRESS-TYPE">
            ...
        </extension>
    </complexContent>
</complexType>
```

Instance Model

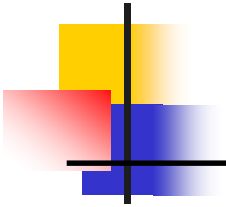



```
<ADDRESS xsi:type="CANADIAN-ADDRESS-TYPE">
    <STREET>123 Any St.</STREET>
    <CITY>Ottawa</CITY>
    <PROVINCE>ON</PROVINCE>
    <POSTAL-CODE>A1B 2C3</POSTAL-CODE>
</EMPLOYEE>
```



Conclusions

- Much more to O-R and O-X Mapping than first meets the eye
- O-R and O-X Share several common challenges, but each have their own unique perspectives



Recommended Sessions…

- Donald Smith
  - Impact of Tiger on Java Persistence
- Mike Keith
  - Standards Based Persistence through Web Services
  - Evolution of the EJB Entity
- Neil Graham
  - Lower-level Data Manipulation with J2SE 5.0
- Matthew Wakeling
  - [Advanced Techniques | Performance] in Object Warehousing