JDO
Java Data Object

Didier DONSEZ

Université Joseph Fourier (Grenoble 1)

IMA – IMAG/LSR/ADELE

Didier.Donsez@imag.fr

Rappel
Java, Persistance et Bases de Données

- Rendre persistant des objets Java
  - 35% du travail du développeur passe dans le mapping Objet/JDBC
- Plusieurs solutions de stockage
  - Sérialisation + Fichier
    - ☹ ne permet pas le partage et la recherche
    - ☹ Le chargement n 'est pas incrémental
  - JDBC
    - API bas-niveau
    - ☹ impedance mismatch
  - SQLJ
    - Pre-Processeur « Embedded SQL in Java » vers JDBC
    - ☹ impedance mismatch
  - JavaBlend
    - ODMG
  - JDO

JavaBlend

- OML Java de l ’ODMG2.0 (www.odmg.org)
  - classes additionnelles
    - `PersistentRoot` et `OID`
    - `Dcollection`, ...
- Transparence au SQL
  - Surcouche à JDBC

Java Data Objects (JDO)

- Permet de rendre persistants des instances de n'importe quelles classes
  - Persistance transparente
    - Accès direct aux membres

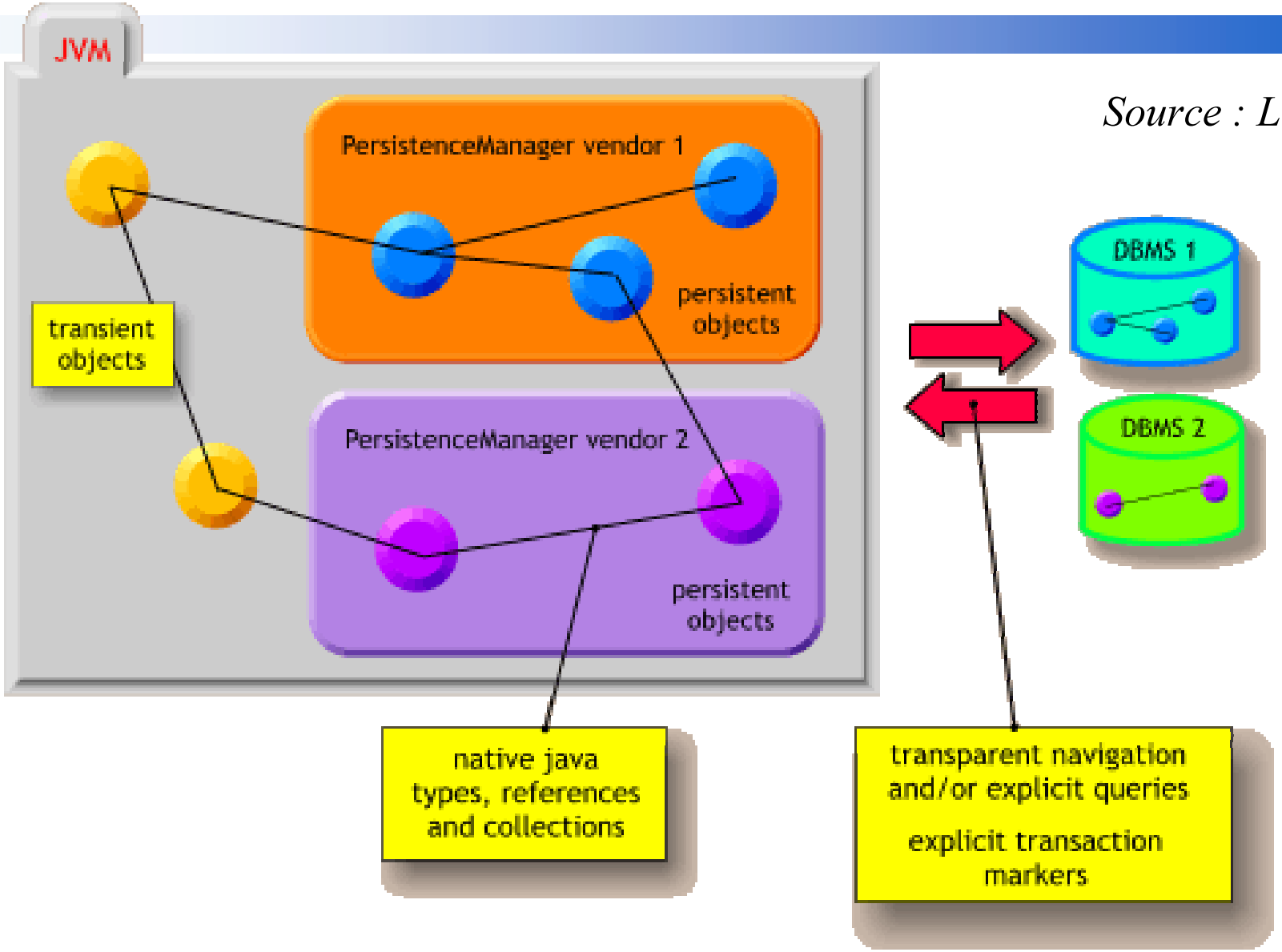
      Les méthodes set/get (accesseur/mutateur) ne sont pas obligatoires (AspectJ, AOP)
    - Déréferenciation par .
  - Instances persistantes / transientes
- Editions
  - J2EE, J2SE et J2ME
- V1.0 : 28/03/2002

JDBC versus JDO

Source : Libelis

JDBC	JDO
SQL-oriented	Object-**Oriented**
Explicit intrusive code	**Fully** transparent
Manual cache management	Advanced **cache management**
Manual mapping	Automatic **Mapping**
RDBMS centric	Universal

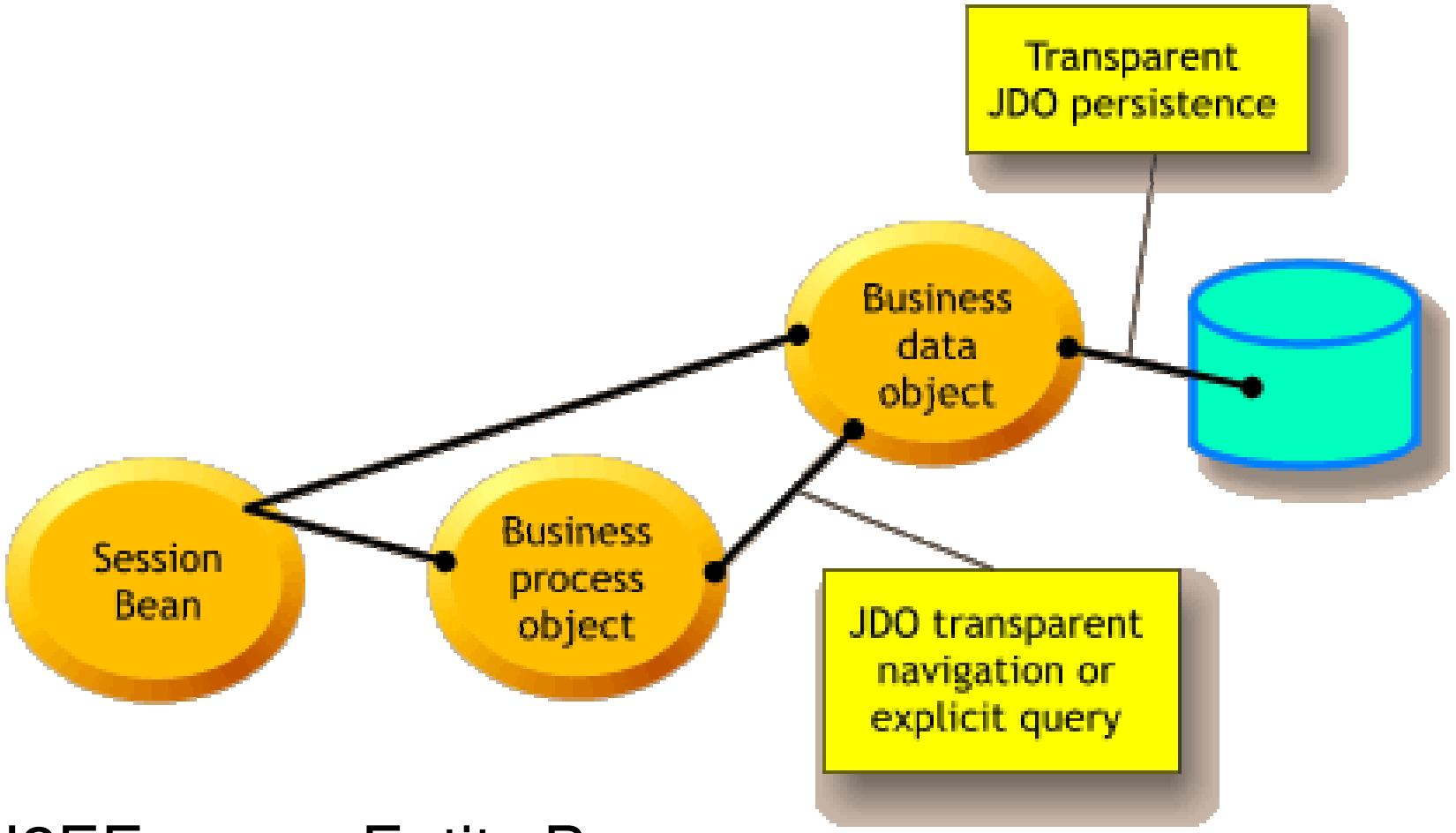
Architecture



Source : Libelis

Usage : Persistance transparente

Source : Libelis

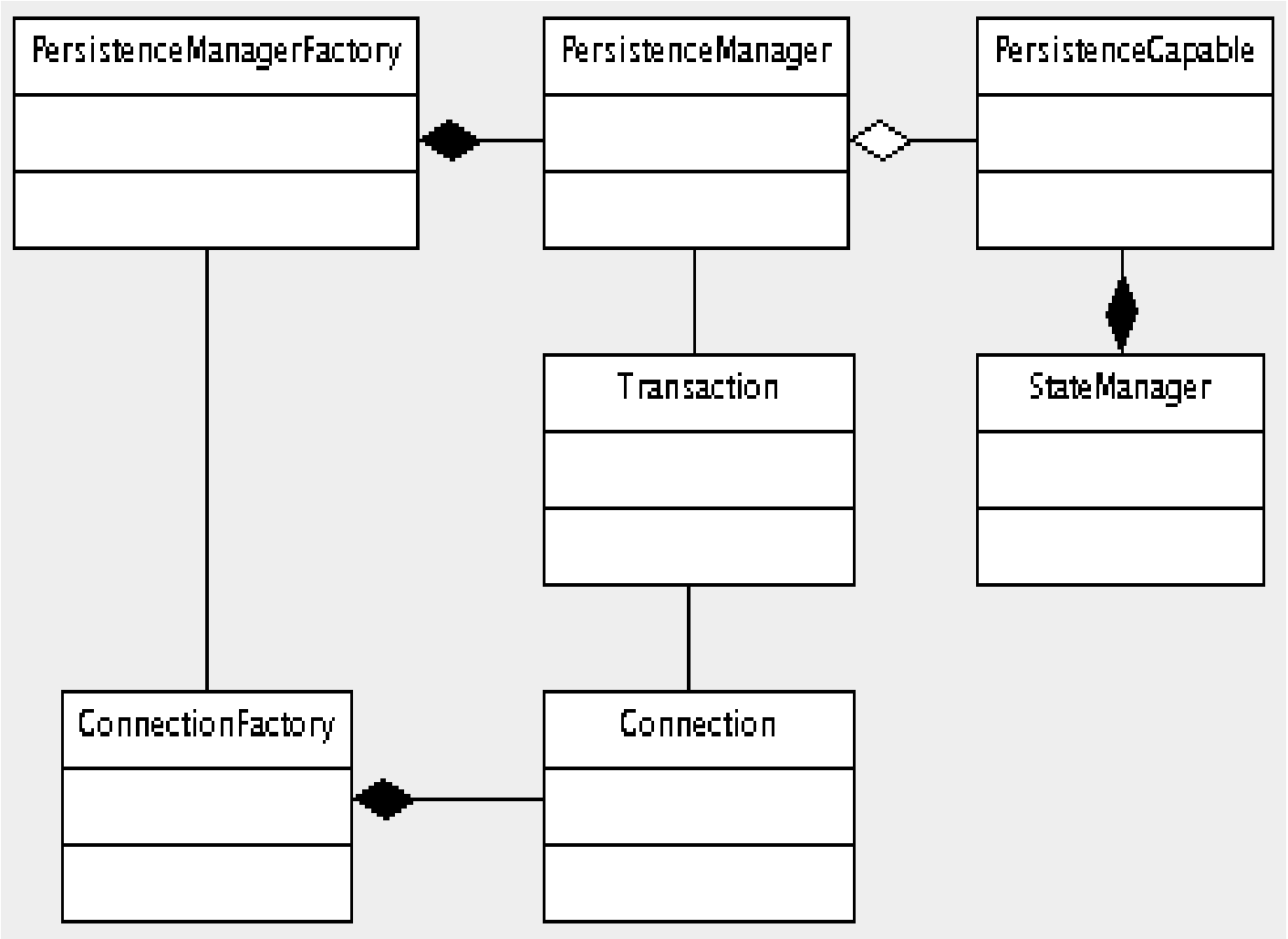


- J2EE apps : Entity Beans

API JDO

- javax.jdo.PersistenceManagerFactory
  - Agit comme un pool de connections
- javax.jdo.PersistenceManager
  - Gère les accès, la sauvegarde, les transactions et les recherches entre les applications et les Data Stores
- javax.jdo.PersistenceCapable
  - Interface que doit implémenter une classe dont des instances peuvent être persistantes
- javax.jdo.InstanceCallback
  - Defines some hooks that allows to do "special things" (like initialisations of transient attributes) during database operations (like before/after read, before/after write, …).
- javax.jdo.Transaction
- javax.jdo.Query
  - Recherche sur critère.

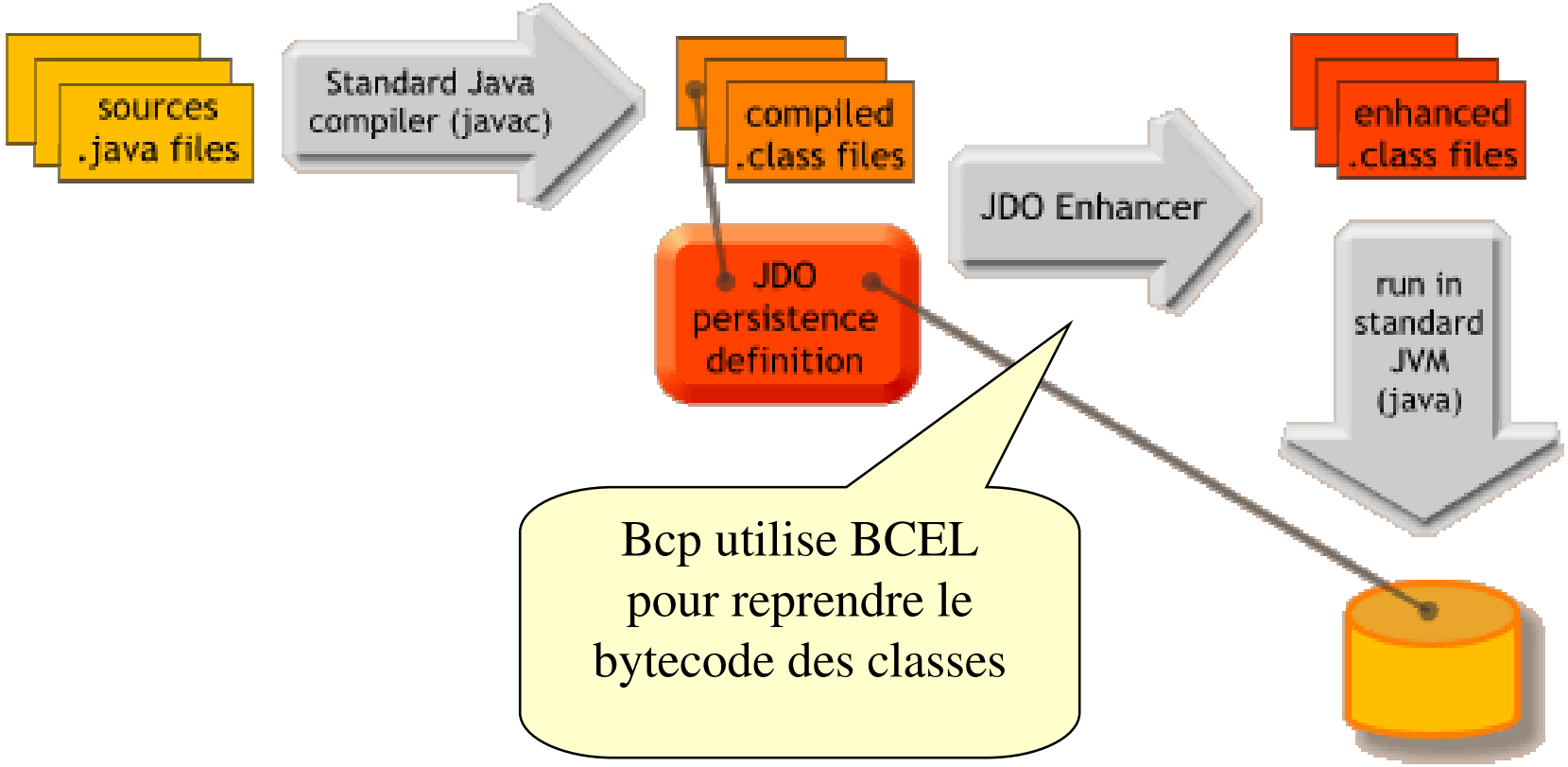
API JDO


PersistenceManagerFactory
PersistenceManager
PersistenceCapable
Transaction
StateManager
ConnectionFactory
Connection

Cycle de développement

- Autorise debugging, profiling (JPDA)
- Intégration : IDE, ANT

Source : Libelis



JDO identity

- How to uniquely identify an object ?

Primary key

- Defined by application, enforced in database
- Used for access to legacy RDBMS

Managed by datastore

- Not tied into any instance values
- New RDBMS or ODBMS

Managed by implementation

- To guarantee uniqueness in the JVM, not datastore
- For property files, ASCII files, XML files, ...

```
pmf = (PersistenceManagerFactory) (
    Class.forName("org.libelis.jdo.versant.PersistenceManagerFactory")
    .newInstance()
);
pmf.setConnectionURL(dbName);
pm = pmf.getPersistenceManager();
tx = pm.currentTransaction();
// tx.setOptimistic(true);
tx.begin();
…
tx.commit();
pm.close();
```

JDO - Cycle de vie des objets

- Création/Persistance

```
Employee e= new Employee("John DOW");
pm.makerPersistent(e); // e now persists (after commit)
Address a = new Address("2 rue de la Paix","Paris"); // a transient
e.address = a ; // a now persists (after commit)
```

- OID

```
Object oid = pm.getObjectId(pc);
oid.writeObject(out); // serialisation
```

- Recherche par OID

```
OID oid = new OID(); oid.readObject(in);
PersistenceCapable pc =(PersistenceCapable)pm1.getObjectById(oid, true);
pm.deletePersistent(pc);
```

- Suppression

```
pm.deletePersistent(e); // delete e
```

JDO - Query

```
Collection extent = pm.getExtent(Employee.class, false);
Query query = pm.newQuery (
                Employee.class, extent, "salary > 50000" );
Collection result = q.execute();

for (iterator i = result.iterator(); i.hasNext(); ) {
    Employee emp = (Employee)i.next();
   out.println(emp.name+":"+emp.salary)
 }
```

JDO - QUery

- **Basic query with ordering.**
  - Class empClass = Employee.class;
  - Extent clnEmployee = pm.getExtent (empClass, false);
  - String filter = “salary > 30000”;
  - Query q = pm.newQuery (empClass, clnEmployee, filter);
  - q.setOrdering (“salary ascending”);
  - Collection emps = q.execute ();

JDO - Query

- **Parameter passing.**
  - Class empClass = Employee.class;
  - Extent clnEmployee = pm.getExtent (empClass, false);
  - String filter = “salary > sal”;
  - Query q = pm.newQuery (empClass, clnEmployee, filter);
  - String param = “Float sal”;
  - q.declareParameters (param);
  - Collection emps = (Collection) q.execute (new Float (30000.));

- **Navigation through single-valued field.**

```
Class empClass = Employee.class;
Extent clnEmployee = pm.getExtent (empClass, false);
String filter = “dept.name == dep”;
Query q = pm.newQuery (empClass, clnEmployee, filter);
q.declareParameters (“String dep”);
Collection emps = (Collection) q.execute (“R&D”);
```

Navigation through multi-valued field.

```
Class depClass = Department.class;
Extent clnDepartment = pm.getExtent (depClass, false);
String filter = "emps.contains (emp) & emp.salary > sal";
String vars = "Employee emp";
String param = "float sal";
Query q = pm.newQuery (depClass, clnDepartment, filter);
q.declareParameters (param);
q.declareVariables (vars);
Collection deps = (Collection) q.execute (new Float (30000.));
```

Produits JDO

- Implémentation de référence de SUN
  - Cible : FOStore (File/Object Store)
- Data Source
  - SGBDs OO
    - ObjectStore (PSE Pro/Java), Orient, Matisse, FastObject, Versant, …
  - SGBDs R
    - JDBC, Oracle, DB2, PostGres, MySql, …
  - Fichiers plats
- Produits
  - Libelis LiDO
  - Castor JDO
  - ObjectWeb JORM / JDO
  - …

API relatives

- JDBC
- JCA
- JTA/JTS

Bibliographie

- JSR-000012 Java™ Data Objects Specification
  - http://www.jcp.org/aboutJava/communityprocess/first/jsr012/
- Robin Roos, "Java Data Objects", Ed Addison-Wesley, ISBN 0-321-12380-8
  - Disponible librement en version PDF non imprimable