ODMG Java

Main Features

- Unified type system between the Java language and the database.
- Java classes can be made persistence-capable.
- Persistence is by reachability.
- Transparent access and update to the database.
- Collections.
- Query facilities.

Class Declaration: example

```
class Person {
        String name;
        Person spouse;
        Set  family;
}
```

Persistent Root: Example

```
Database base = new Database();
base.open ("origin");
Transaction trans = new Transaction();
trans.begin ( );
Person adam = new Person ("adam");
base.bind (adam, "adam");
trans.commit ( );
base.close ( );
```

Retrieving and Creating Objects: Example

```
Database base = new Database;
base.open ("origin");
Transaction trans = new Transaction;
trans.begin ( );
Person adam = base.lookup("adam");
Person eve = new Person ("eve");
Person cain = new Person ("cain");
adam.spouse = eve;
adam.family.insertElement(cain);
trans.commit ( );
base.close ( );
```

Persistency

- Persistency is by reachability from persistent roots. All the persistent objects are directly or transitively attached to the persistent root.
- At commit time if an object is connected to a persistent root then the object is written automatically into the database.
- The class Database allows to define persistent roots and to retrieve them:
  - bind (Object, String) // give a name
  - lookup (String) // retrieve an object by its name

Primitive Java Types

int	integer	Integer class
short		Short class
long		Long class
float		Float class
boolean		Boolean class
char		Char class
byte		
String		

Mapping from ODMG to Java

ODMG	Java Type
Date	java.sql.Date
Struct or tuple	Java Class
Set (t)	interface Set
Bag(t)	interface Bag
List (t)	interface List
Array (t)	array type [ ] or Vector
Iterator	Enumeration

Java OML

- The syntax for manipulating persistent or transient objects is identical and follows Java syntax.
- Object creation is done with the new operator. If an object type intermixes persistent data and transient data only persistent data is loaded in memory, transient data is set to default.
- There is no object deletion since persistency is by reachability.
- Modification of objects is done with standard Java operations. At commit time modifications are written back automatically.

Interface Collection

```
public interface Collection {
        public int size ( );
        public boolean isEmpty ( );
        public removeElement ( );
        .........
        public Collection query ( String predicate);
}
```

Interface Set

```
public interface Set extends Collection {
        public Set union (Set otherSet);
        public Set intersection (Set otherSet);
        .......
        public boolean properSubsetOf (Set otherSet);
        .......
}
```

Interface List

```
public interface List extends Collection {
        public void add (int index, Object obj)
                throws ArrayIndexOutOfBounds;
        public void put (int index, Object obj)
                throws ArrayIndexOutOfBounds;
        ........
        public Object get (int index)
                throws ArrayIndexOutOfBounds;
        public List concat (List other);

}
```

Interface Database

```
public class Database {
        public static Database open ( String name)
                throws ODMGException;
        public void close ( )
                throws ODMGException;
        public void bind (Object obj, String name);
        public Object lookup (String name)
                throws ObjectNaneNotFoundException;
}
```

Transaction

- The interface Transaction provides standard operations as: begin ( ), commit ( ), abort ( ).
- Operations are also defined for using the possibility of maultithreading in Java, such operations are: join, current, etc.

Java OQL

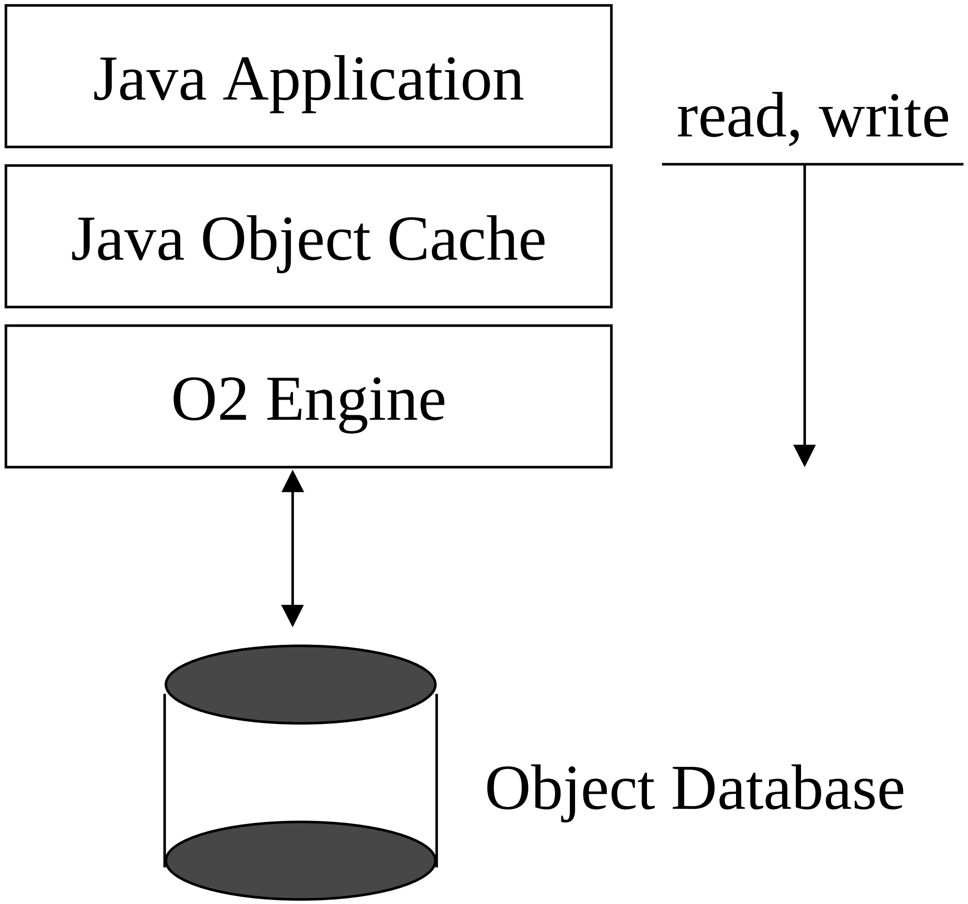
- Filtering a collection with the function query defined in the interface Collection
  - SetOfObject mathematiciens;
  - mathematiciens = Students.query("exists c in this.takes : c.subject = 'Math' ");
- Using the class OQLQuery.

C. Delobel

OQLQuery Class

```
class OQLQuery {
        public OQLQuery (String query);
        public bind (Object parameter);
        public Object execute ( ) throws ODMGException;
}
Set Students, young;
OQLQuery query;
query = new OQLQuery ( "select s from s in $1 where s.age = $2");
query.bind(Students); query.bind(25);
young = (Set) query.execute ( );
```

General Architecture



Conclusion

- Java is well adapted to become a database programming language.
- Some features of the ODMG model are not implemented: relationship and collections are not parametrized by a type or a class.
- The concepts of Java interface and implementation allow the possibility to define very powerful implementations for Set and List with HashSet, ArraySet, ArrayList, LinkedList, etc.