Why Object Relational Mapping (ORM)?

When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects. Consider the following Java Class with proper constructors and associated public function:

## What is ORM?

ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java etc. An ORM system has following advantages over plain JDBC

* Hibernate is ORM solution for Java Applications.
* Hibernate is ORM tool given to the transfer the data between a java(object)application and a database(Relational) in the form of the objects. Hibernate is the open source, light weight tool given by **Gavin King**.
* Hibernate is non-invasive framework, which means it doesn't forces the programmer to extend or implement any class or interface.
* Hibernate can run with-in or with-out server, hence Hibernate is suitable for both stand-alone applications(desktop application) and web-applications.
* Hibernate is purely meant for persistence.

**Persistence :** The process of storing enterprise data in to relational database.

The following is a list of popular ORM (Object Relational Mapping) frameworks that are used in Java / J2EE applications to persist data objects into relational database.

* EJB – Enterprise Java Beans
* Enterprise Objects Framework
* Hibernate
* iBATIS
* Java Data Objects (JDO)
* Java Object Oriented Querying (jOOQ)
* Java Persistence API (JPA)
* JPOX
* Kodo
* MyBatis
* Object Relational Bridge (Apache OJB)
* ObjectDB
* OpenJPA
* ORMLite
* QuickDB ORM
* TopLink
* Torque