HIBERNATE

What is ORM?

- ORM stands for Object-Relational Mapping (ORM)
- It is a programming technique for converting data between RDBMS and OOPL
- Advantages of ORM
  - Business logic code accesses data as object not as DB tables.
  - Hides details of SQL queries
  - Internally uses JDBC
  - Business logic code is independent of database implementation
  - Software entities are based on business concepts rather than DB tables.
  - Automatic key generation and transaction management
  - Fast development and easy maintenance of applications

Why Object Relational Mapping (ORM)?

- In Object-Oriented software/application development
  - Mismatch between
    - Object model (interconnected graph of objects)
    - Relational database model (tabular format of data)

- Two typical scenarios:
  - Change in design of database at later stage of application development
  - Load and store objects (data objects) in a relational database.

What ORM provides?

- API to perform CRUD operations
- Configuration for specifying mapping between object and table
- DBMS independent query language or API (e.g., HQL)

Why Object Relational Mapping (ORM)?

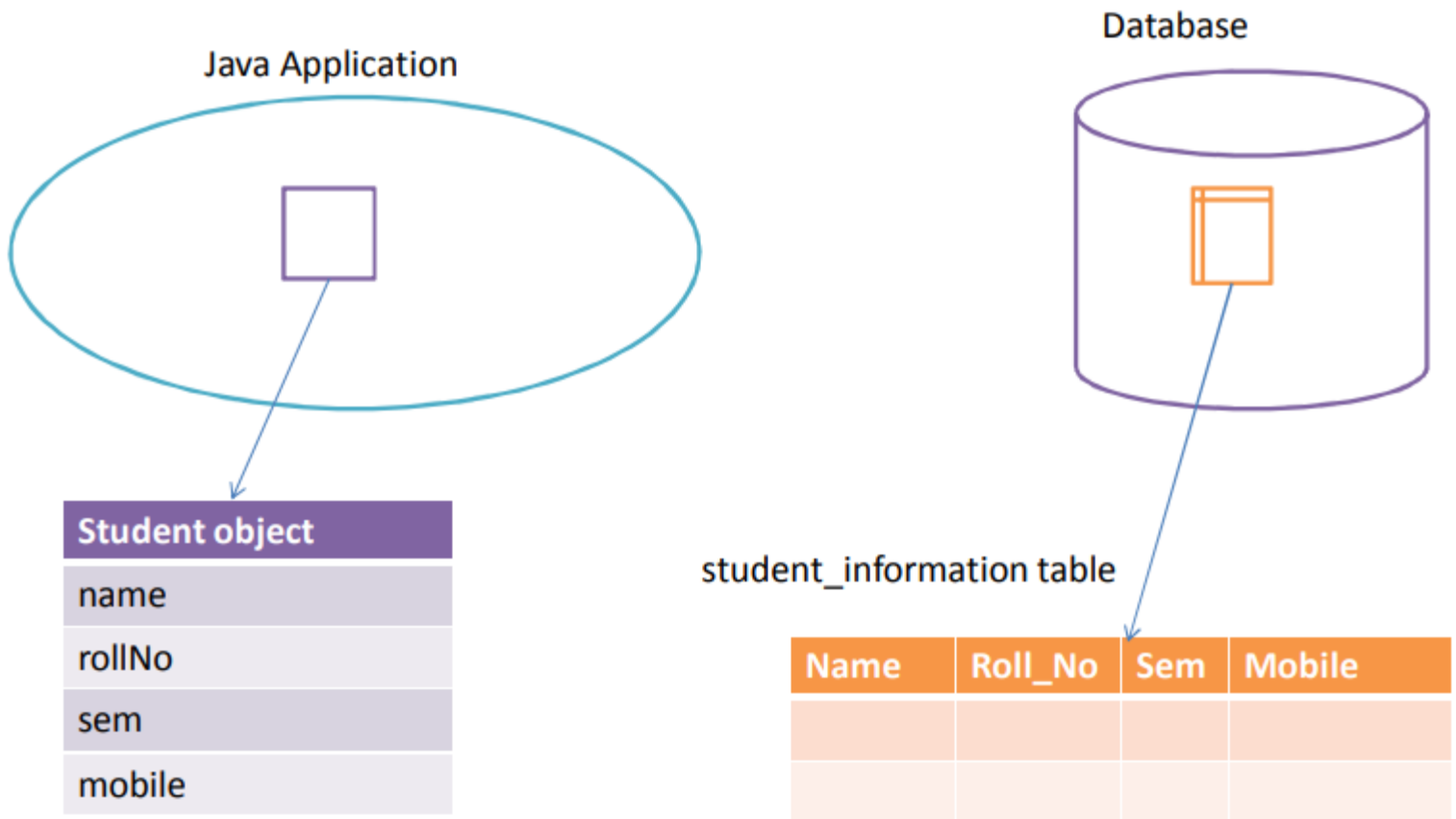
- Mismatch problems

Mismatch	Meaning
Granularity	Two or more objects may represent a single database table
Inheritance	RDBMS systems do not have anything like Inheritance available in OO programming languages
Identity	RDBMS defines exactly one meaning of 'sameness' : primary key. Java has object identity (ob1==ob2) and object equality (ob1.equals(ob2))
Associations	OOPL represents association using object references. RDBMS represents it using foreign key column.
Navigation	Ways of navigation are different in RDBMS and Java

ORM frameworks in Java

- Enterprise Java Beans Entity Beans
- JDO-Java Data Objects
- Castor
- TopLink
- Spring DAO
- Hibernate, and many other

Java Application deals with data



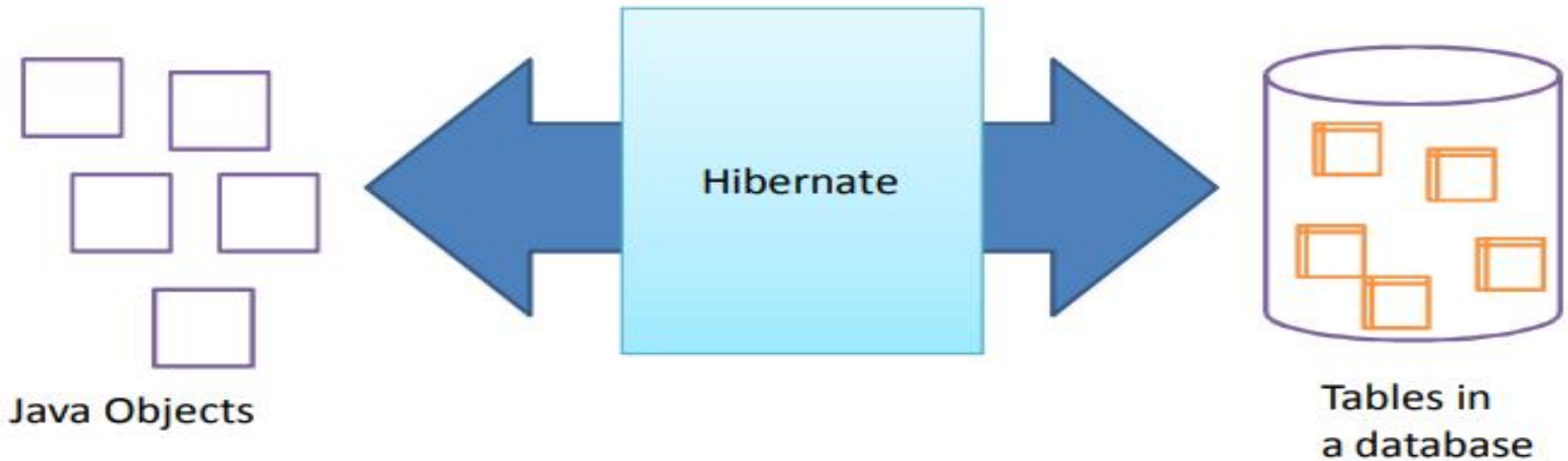
Student s=new Student(“abc”, 1, 6, 111111111);

How Hibernate is useful?

- Programmers need to write minimal code to deal with data
- Database connection is configurable
- Hibernate takes burden of
  - Making database connection
  - Releasing db connection
  - Writing queries
- Hibernate relives the developer from writing more than 95% of common data persistent related code

What is Hibernate?

- Hibernate is an ORM tool that provides Object Relational Mapping
- It is open source persistent framework created by Gavin King in 2001.



How Hibernate is useful?

```
public void insert(Student st){
    Session session=factory.openSession();
    Transaction tx=null;
    try{
        tx=session.beginTransaction();
        session.save(st);
        tx.commit();
    }
    catch(HibernateException ex){
        …
    }
    finally{
        session.close();
    }
}
```

```
Student s=new Student("abc", 1, 6, 111111111);
```

Java Application can use JDBC

```
public void insert(Student st){
    Connection con=null; Statement stmt=null;
    // Register JDBC driver
    Class.forName(…);
    // Open a DB connection
    con=DriverManager.getConnection(…);
    // Create a statement
    stmt=con.createStatement();
    String query="INSERT INTO STUDENT_INFORMATION(Name, Roll_No,
    Sem, Mobile) " + "VALUES("+st.getName()+","+st.getRollNo+","+
    st.getSem()+","+st.getMobile()+")";
    //Execute update query
    stmt.executeUpdate(query);
}
```

```
Student s=new Student("abc", 1, 6, 111111111);
```

How Hibernate is useful for data retrieval?

- JDBC code

```
ResultSet rs;
rs=stmt.executeQuery("SELECT * FROM STUDENT_INFORMATION");
List stList=new LinkedList();
while(rs.next()){
    String name=rs.getString("Name");
    int rollNo=rs.getInt("Roll_No");
    int sem=rs.getInt("Sem");
    String mobile=rs.getString("Mobile");

    Student st=new Student(name, rollNo, sem, mobile);
    stList.add(st);
}
```

- Hibernate code

```
List stList= session.createQuery("FROM STUDENT_INFORMATION").list();
```

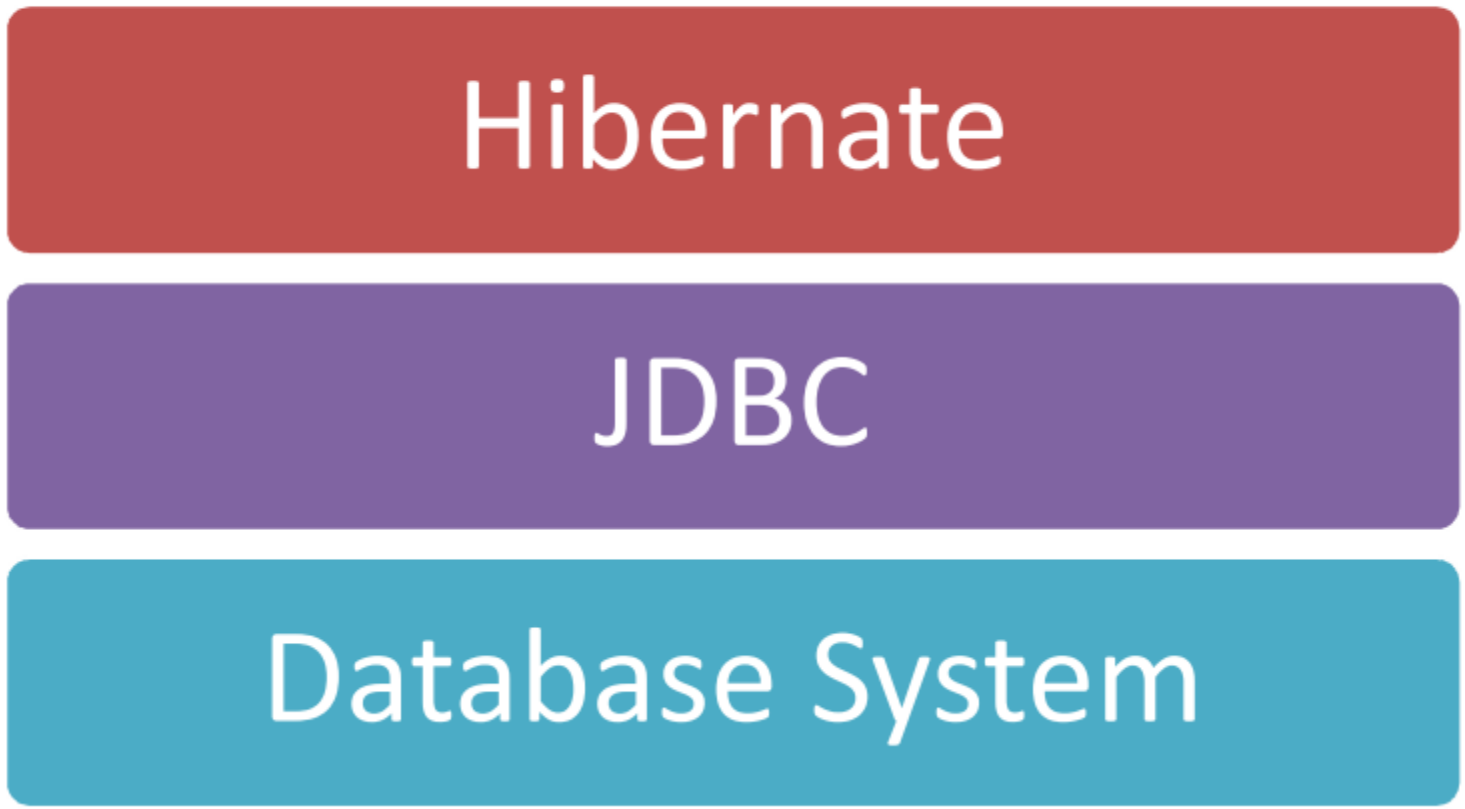
Other features of Hibernate

- Good caching mechanism for faster retrieval of data
  - Avoid database query if object is in cache
- Opening and closing db connection handled by Hibernate
- Application can support almost all relational databases
- A little modification is needed in java application for any change in database.
  - E.g., change in the name of a column in a database table
    - Using direct JDBC code, need to reflect at all places
    - Using Hibernate, just change in the configuration file or in annotation

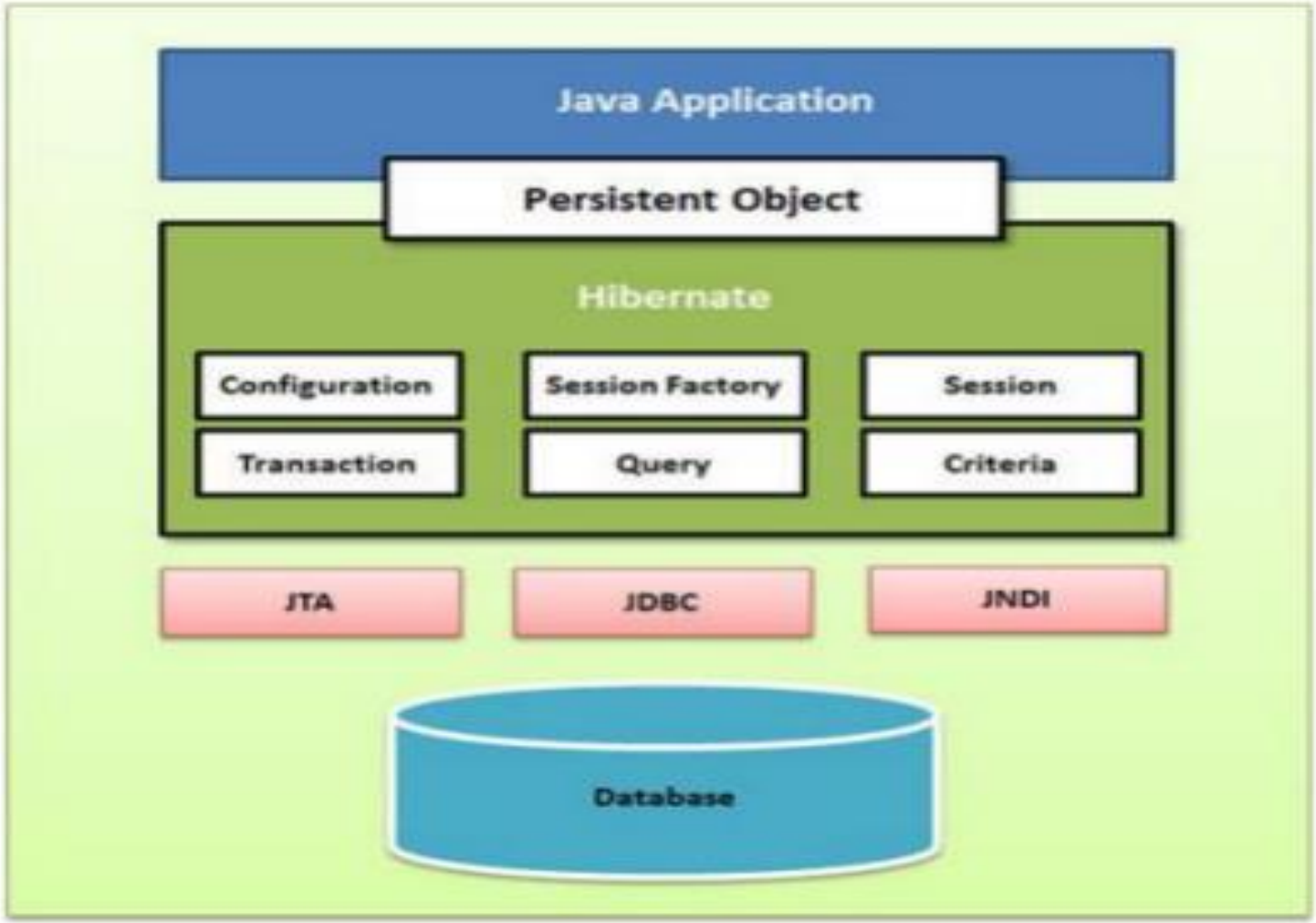
Hibernate advantages

- Hibernate handles mapping Java classes to database tables using XML files
  - We do not need to write any Java code
- Hibernate provides simple API to deal with database tables.
- If we change database, we need to change only XML file, no change in Java code is required.
- Abstracts unfamiliar SQL types and provides Java objects having known data types.
- Minimize frequent database access with intelligent fetching strategies.

Position of Hibernate layer



Hibernate Objects



Supported databases

- Hibernate supports major RDBMS
  - HSQL Database Engine
  - DB2
  - MySQL
  - PostgreSQL
  - FrontBase
  - Oracle
  - Microsoft SQL Server Database
  - Sybase SQL Server
  - Informix Dynamic Server

Hibernate Objects

- Major objects
  - Configuration
  - Session Factory
  - Session
  - Transaction
  - Query
  - Criteria
- Hibernate's interfacing with DBMS
  - Java Transaction API (Used in JEE server)
  - Java Naming and Directory Interface (Used in JEE server)
  - JDBC (Used in traditional/console app)

Configuration object

- Configuration object is created during application initialization.
- It represents configuration or property file needed by the Hibernate.
- Configuration object provides two things:
  - Database connection:
    - It is handled using one or more configuration files. E.g., hibernate.properties and hibernate.cfg.xml
  - Class mapping
    - Establishes connection between Java classes and database tables

Transaction object

- It represents a unit of work (atomic operations) to be performed with the database.
- Transaction concept is provided by RDBMS
- Hibernate handles transactions using underlying transaction manager and transaction

SessionFactory object

- SessionFactory object is created using Configuration object.
- SessionFactory is used to create a Session.
- SessionFactory is thread safe object.
  - Multiple threads can use it
- SessionFactory is heavyweight object. It is created during application startup and is maintained for later use.
- We need one SessionFactory object per DBMS.

Query object

- Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.
- A Query object is used
  - to bind query parameters
  - to limit the number of results returned by the query
  - to execute the query

Session object

- A Session object provides a physical connection with the database.
- Session object is lightweight. It is instantiated each time an application needs interaction with the database.
- Persistent objects are stored and retrieved through a Session object.
- Session objects are not thread safe. Therefore, we should release them once use is over.

Criteria object

- It is used to create and execute object oriented criteria queries to retrieve objects.

Hibernate Configuration

- Hibernate requires the following configuration
  - DBMS related parameters
  - Mapping of Java Classes with Tables
  - Other parameters (e.g., hibernate supported features)
- Configuration information can be kept in hibernate.cfg.xml file.
- This file is stored in the root directory of our application's classpath.

Sample Configuration file for MySQL (in Hibernate 3.X)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <!-- Assume testDB is the database name -->
```

Hibernate Configuration: Properties

Properties and meaning
hibernate.dialect This property enables Hibernate to generate the appropriate SQL for the chosen database.
hibernate.connection.driver_class It indicates the JDBC driver class to use for JDBC connectivity.
hibernate.connection.url The JDBC URL to the database instance
hibernate.connection.username Username for the database.
hibernate.connection.password Password for the database.

Sample Configuration file

```
        <property name="hibernate.connection.url">
           jdbc:mysql://localhost/testDB
        </property>
        <property name="hibernate.connection.username">
           admin
        </property>
        <property name="hibernate.connection.password">
           admin123
        </property>
        <!-- List of XML mapping files -->
        <mapping resource="student.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Hibernate Configuration: Properties

Properties and meaning
hibernate.connection.pool_size Upper limit on the number of connections waiting in the Hibernate database connection pool.
hibernate.connection.autocommit It allows auto-commit mode to be used for the JDBC connection.
hibernate.connection.datasource The JNDI name defined in the application server context.
hibernate.jndi.class The InitialContext class for JNDI.
hibernate.jndi.<JNDIpropertyname> Passes any JNDI related property we want to pass to the JNDI InitialContext.
hibernate.jndi.url It indicates the URL for JNDI.

Configuration specification of Mapping Java classes with Tables

- <mapping> element indicates hibernate mapping file to use for mapping Java classes with Tables

Databases dialect property value for various databases

DBMS	Property value
DB2	org.hibernate.dialect.DB2Dialect
Microsoft SQL Server 2008	org.hibernate.dialect.SQLServer2008Dialect
MySQL	org.hibernate.dialect.MySQLDialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle 11g	org.hibernate.dialect.Oracle10gDialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect

Understanding of Hibernate Mapping (Java class)

- An Object Relational mappings are defined in an XML document.
- Suppose we have a Student java class

```
public class Student {
    private int id;
    private String firstName;
    private String lastName;
    private int rollNo;
    …
```

Hibernate Session

- Persistent objects are saved and retrieved using a Session object.
- The Session object is lightweight object and it is used to get a physical connection with a database.
- When interaction with database is needed, object of Session is created.
- Session objects are not thread safe, so delete them once use is over.
- Session provides methods for the following operations related to instances of mapped entity class
  - Create
  - Read
  - Delete

Understanding of Hibernate Mapping (DB table)

- Let STUDENT_INFO database table

```
create table STUDENT_INFO (
    id INT NOT NULL auto_increment,
    first_name VARCHAR(20) default NULL,
    last_name VARCHAR(20) default NULL,
    roll_no INT  default NULL,
    PRIMARY KEY (id)
);
```

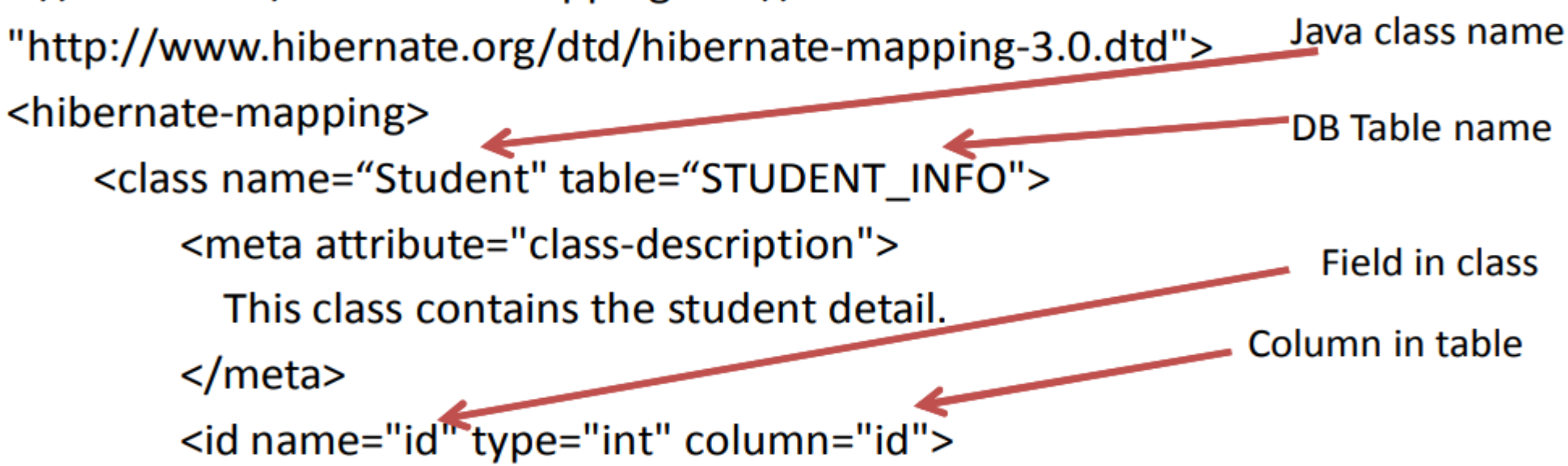
Hibernate Persistent classes

- Hibernate reads Java class attributes and persists them to a database table, and vice versa.
- Hibernate framework uses mapping file
- Persistent classes or entity classes follow rules of POJO (Plain Old Java Object) programming model.
  - Some of rules of POJO (persistent classes) are
    - Default constructor (no argument constructor)
    - ID as a member to identify objects in Hibernate framework and row in database.
    - Declare attributes as private and provide getter/setter methods.

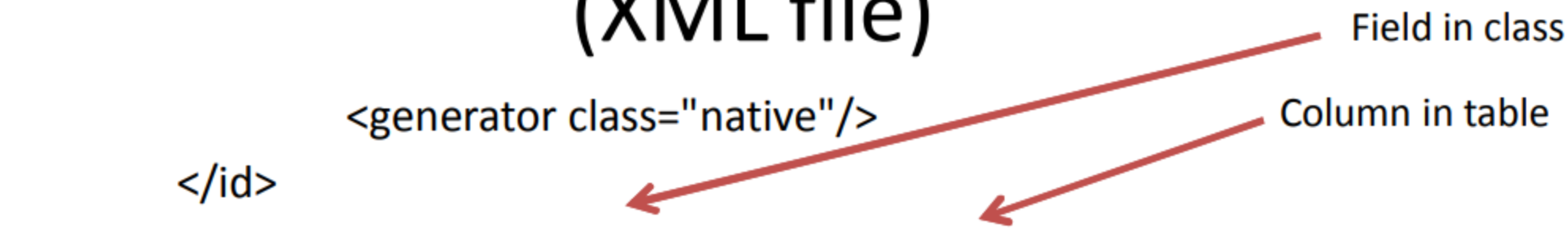
Understanding of Hibernate Mapping (XML file)

- Mapping file to map class (attributes or data members) with database table (fields or columns)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="Student" table="STUDENT_INFO">
        <meta attribute="class-description">
            This class contains the student detail.
        </meta>
        <id name="id" type="int" column="id">
```



Understanding of Hibernate Mapping (XML file)




```
            <generator class="native"/>
        </id>
        <property name="firstName" column="first_name" type="string"/>
        <property name="lastName" column="last_name" type="string"/>
        <property name=“rollNo" column=“roll_no" type="int"/>
    </class>
</hibernate-mapping>
```

- We should save the mapping in a file with the name format <classname>.hbm.xml.
- For Student class, we can store the mapping in Student.hbm.xml file.

Hibernate Mapping data types

- The types specified in mapping files are neither Java data types nor SQL database types.
- The specified types in mapping files are called Hibernate mapping types.

Understanding of Hibernate Mapping file

- The mapping file contains <hibernate-mapping> as the root element
- <class> elements with properties
  - name : The name of Java class
  - table: The name of mapped DB table
- <meta> element : It is optional
- <id> maps unique ID attribute of the class with the primary key in table. It element has the following properties
  - name : The name field in Class
  - type: hibernate mapping type
  - column: the column in database table

Hibernate Mapping data types

Hibernate mapping type	Java data type	SQL data type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT

Understanding of Hibernate Mapping file

- <generator> element within the id element
  - It indicates what primary key generator algorithm to use
  - Its class attribute is set to native. Hibernate uses either identity, sequence, or hilo algorithm to create primary key.
  - Hibernate chooses appropriate one based on the capabilities of the database.
- The <property> element
  - Each property maps a Java class property (data member or field) to a column in the database table.
  - Attributes of the property are name, type, and column (similar to ID)

Hibernate Mapping data types: Date and Time

Hibernate mapping type	Java data type	SQL data type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.TimeStamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Hibernate Mapping data types: Binary and Large Object Types

Hibernate mapping type	Java data type	SQL data type
binary	byte[]	VARBINARY (or BLOB)
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable interface	VARBINARY (or BLOB)
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB