HIBERNATE

IJSE

1. What it is
2. Why do we need that technology
3. Implementation of it
4. Advance Conecpts

What actually software developers do

- Create applications to deal/manage with important data

What if...

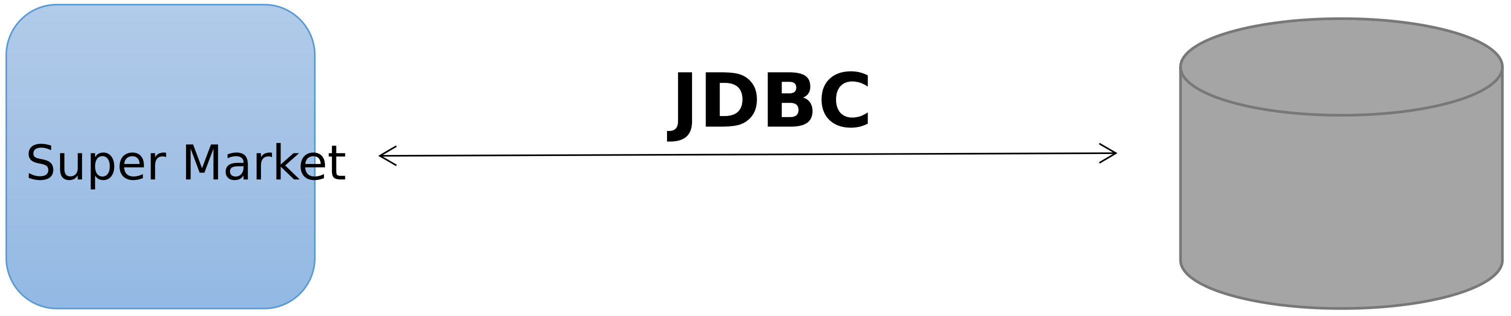
Ø These kind of applications don’t have any database?

v What if you create an application for a super market without using any database...

WHAT WILL HAPPEN ..?

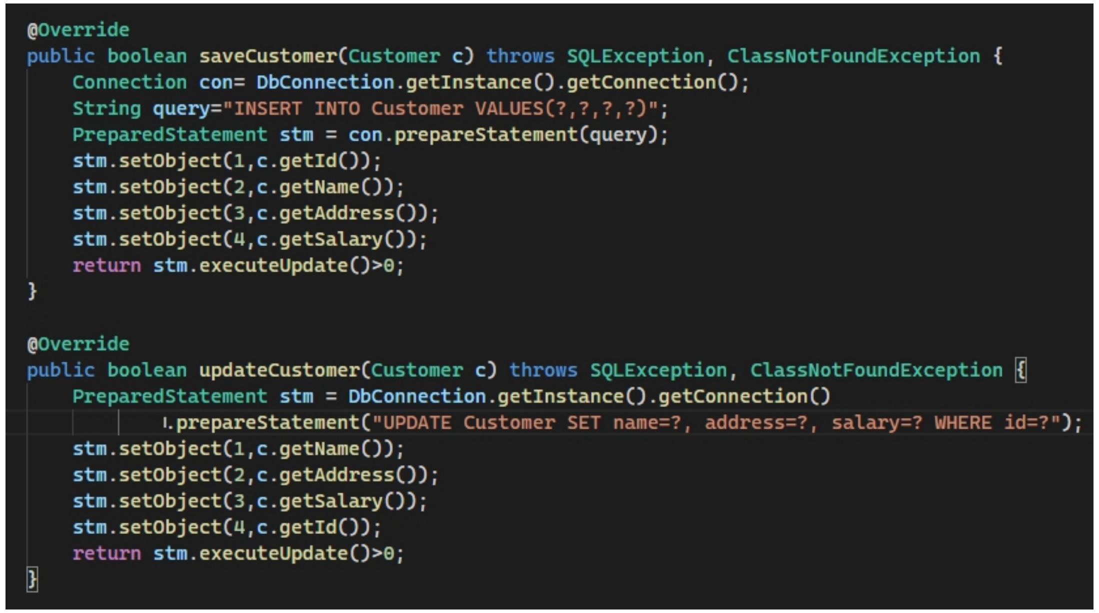
So, What’s the solution?

Persist our DATA



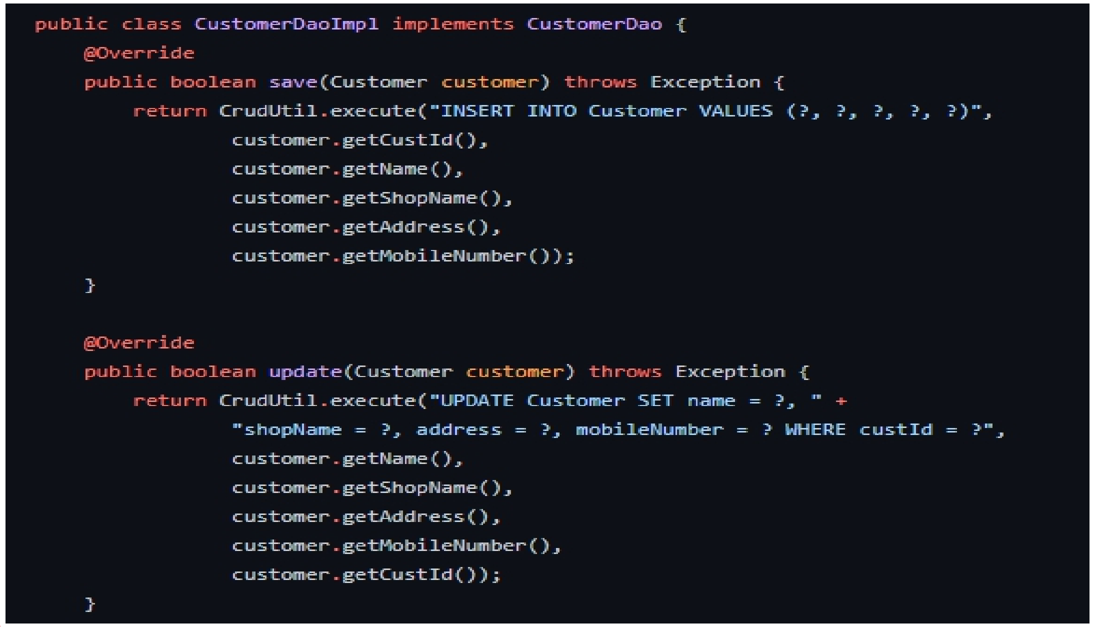
Java application

Mysql DATABSE



```
@Override
public boolean saveCustomer(Customer c) throws SQLException, ClassNotFoundException {
    Connection con= DbConnection.getInstance().getConnection();
    String query="INSERT INTO Customer VALUES(?,?,?,?)";
    PreparedStatement stm = con.prepareStatement(query);
    stm.setObject(1,c.getId());
    stm.setObject(2,c.getName());
    stm.setObject(3,c.getAddress());
    stm.setObject(4,c.getSalary());
    return stm.executeUpdate()>0;
}

@Override
public boolean updateCustomer(Customer c) throws SQLException, ClassNotFoundException {
    PreparedStatement stm = DbConnection.getInstance().getConnection()
            .prepareStatement("UPDATE Customer SET name=?, address=?, salary=? WHERE id=?");
    stm.setObject(1,c.getName());
    stm.setObject(2,c.getAddress());
    stm.setObject(3,c.getSalary());
    stm.setObject(4,c.getId());
    return stm.executeUpdate()>0;
}
```



```
public class CustomerDaoImpl implements CustomerDao {
    @Override
    public boolean save(Customer customer) throws Exception {
        return CrudUtil.execute("INSERT INTO Customer VALUES (?, ?, ?, ?, ?)",
                customer.getCustId(),
                customer.getName(),
                customer.getShopName(),
                customer.getAddress(),
                customer.getMobileNumber());
    }

    @Override
    public boolean update(Customer customer) throws Exception {
        return CrudUtil.execute("UPDATE Customer SET name = ?, " +
                "shopName = ?, address = ?, mobileNumber = ? WHERE custId = ?",
                customer.getName(),
                customer.getShopName(),
                customer.getAddress(),
                customer.getMobileNumber(),
                customer.getCustId());
    }
```

Data Persistence

Ø Persistence means that we make our application’s data outlive the application’s process.

Fun Fact:

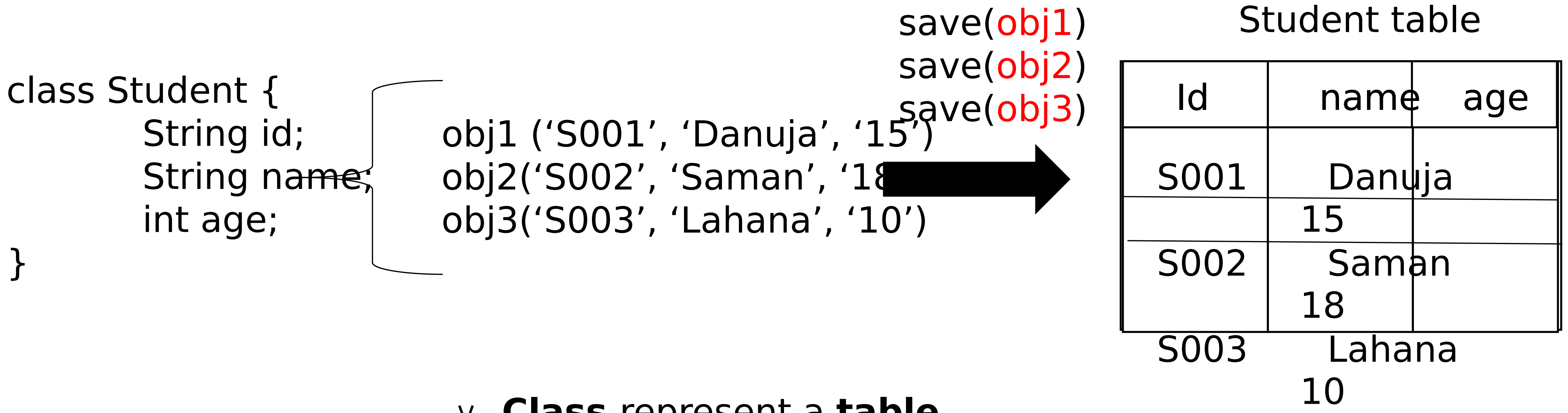
- Actually, programmers are lazy people. They find simple ways to do things.
- That’s why many Java, C, C++, programers are lazy to work with sql queries.

So, What if...

instead of SQL Query, we have ->
save() method to save data in database?

ORM

- Ø ORM is a **concept**. It means **Object-relational mapping.**
- Ø In Java terms, the **state of our objects live beyond the scope of the JVM,** so that the same state is available later.



```
class Student {
        String id;
        String name;
        int age;
}
```

obj1 ('S001', 'Danuja', '15')
obj2('S002', 'Saman', '18
obj3('S003', 'Lahana', '10')

save(obj1)
save(obj2)
save(obj3)

Student table

Id	name	age
S001	Danuja	15
S002	Saman	18
S003	Lahana	10

- v **Class** represent a **table**
- v **Class Property** represent **one column**
- v **Classs Object** represent **one Row**

ORM Tools

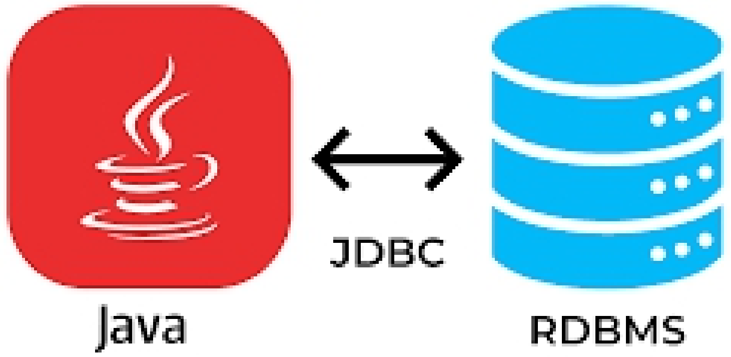
- **Hibernate**
- Sequelize
- SQLAlchemy
- Entity Framework Core
- OpenJPA
- Entity Framework
- Doctrine 2
- MyBatis

Pre-requisite to Hibernate

1. Knowledge about Java core concepts and OOP knowledge

2. SQL query knowledge with RDBMS knowledge

3. JDBC



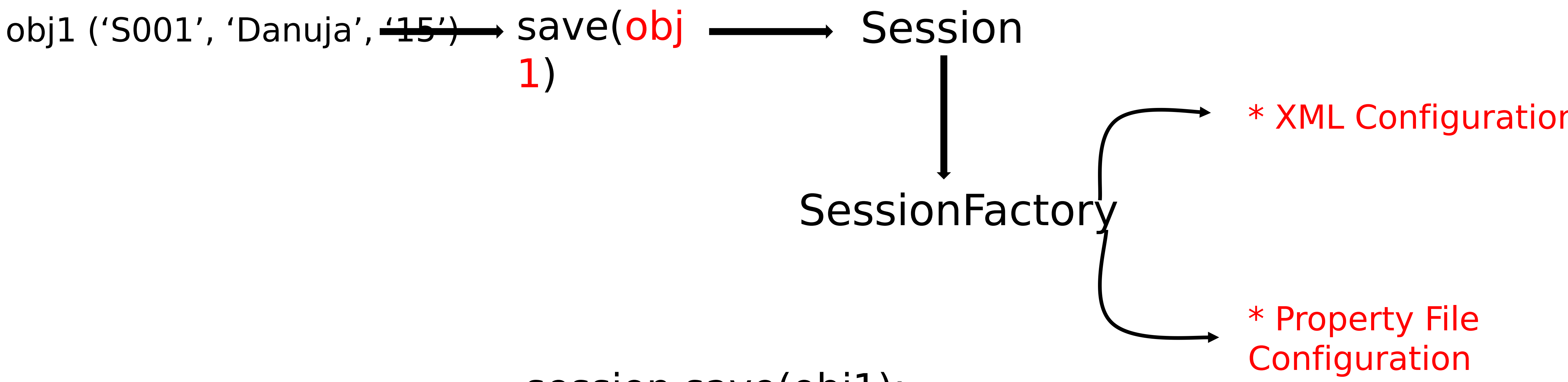
What is Hibernate ?

- Hibernate ORM (Hibernate) is an **object-relational mapping (ORM) tool** for Java programming language.

- Hibernates primary feature is mapping Java classes to database tables, and mapping java data types to SQL data types.

- Hibernate was created in 2001 by **Gavin King** as an alternative tool for **EJB2-style.** He is a software Engineer at **Red Hat.** (Red Hat is the founder of Red Hat Linux)

How Hibernate works?



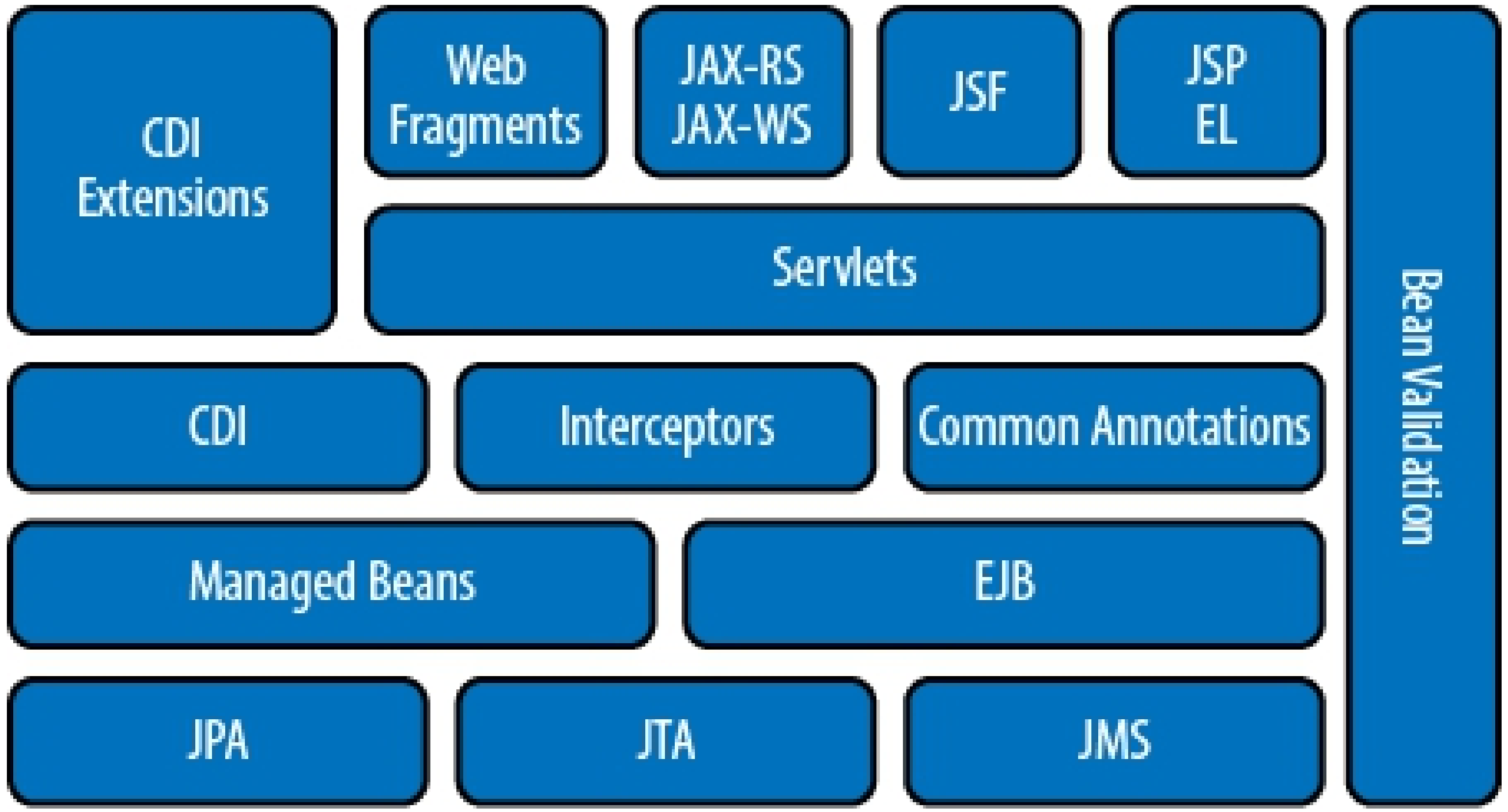
```
session.save(obj1);
session.update(obj1);
session.delete(obj1);
session.get(<className>,
<identifire>)
```

Relation between Hibernate with JPA

- Hibenate is a implementation of **JPA**
- JPA is a specification of **Java EE**
- JPA is describe as Java Persistence API
- It says how you **Persiste** your data with your Java Application

JDK
JRE
JVM

Java SE 8



Java
EE

Mapping Annotations

1. What is Annotation mean?

   Annotation is a meta data for our applications

Ø Few of Annotation is here…

@Entity : The @Entity annotation is used to specify that the currently annotated class repr

@Id : The @Id annotation specifies the entity identifier.

@Column : The @Column annotation is used to specify the mapping between a basic enti
and the database table column.

@Transient : The @Transient annotation is used to specify that a given entity attribute sh

@CreationTimestamp : The @CreationTimestamp annotation is used to specify that the
temporal type must be initialized with the current JVM timestamp

@Embeddable : The @Embeddable annotation is used to specify embeddable types. Like
embeddable types do not have any identity, being managed by their ownin

Hibernate Relations

1. OneToOne
2. OneToMany
3. ManyToMany

Inverse Side/End : Side, that Primary key giving

Owners Side/End : Side, that Primary key recieved

Data Fetching

1. Lazy Fetching
2. Eager Fetching

ü Lazy and Eager are two types of data loading strategies in ORMs such as h

ü These data loading strategies we used when one entity class is having re other Entities like Owner and Pet(Pets in the Owner).

Lazy Loading − Associated data loads only when we explicitly call getter or size method.

- Use Lazy Loading when you are using one-to-many collections.
- Use Lazy Loading when you are

Egare Loading − Data loading happens at the time of their parent is fetched.

- Use Eager Loading when you are sure that you will be using related entities with the main entity

get() vs load()

- In hibernate, **get()** and **load()** are two methods which is used to fetch da
  for the given identifier.

- They both belong to Hibernate session.

- Get() method return null, If no row is available in the session cache or th
  the given identifier

- load() method throws object not found exception

- get() is little bit slower than load()

Further more...

- use **get()** when you want to load an actual object
- use **load()** when you need to obtain a reference to the objct without issueing
  - § for Example, to create a relationship with another object.

Example for **load()** method.

```
          Passport passport = new Passport();
          passport.setPsID("P1");
          passport.setIssueDate("2021-09-25");

          //No any SELECT Query here.
          passport.setPerson(session.load(Person.class,
     "P001"));
```

Advanages and Disadvantages

Advantages

* Hibernate is a open source framework

* Better than JDBC

* Hibernate has its own query language called HQL

* Hibernate has caching mechanism. Using this, number of database hits will be reduced.

Disadvantages

* Hibernate is slow acompared to jDBC because of generating many SQL queries at run time bu this is not considered.

* No sutible for small projects:

* Take lot of time to learn Hibernate

HQL (Hibernate Query Language) vs
Native SQL (Structrued Query Language)

- SQL is a traditional query language that directly interacts with RDBMs.

- whereas HQL is a JAVA-based OOP language that uses the Hibernate interface to convert the OOP code into query statements and then interacts with databases.

- SQL is solely based on RDBMSs but HQL is a combination of OOP with relational databases.

1. Instead of **SQL** work with **Table** and **Column**, **HQL** is work with **Object** and their **properties**.

2. Keywords like SELECT, FROM and WHERE, etc… are not case sensitive, but properties like table and column names are case sensitive in **HQL**

In SQL

SELECT **name** FROM **Customer**;

Column Name Table Name

In HQL

SELECT **name** FROM **Customer**;

Property Name ClassName

HQL few ‘Clauses’

let’s try...

- FROM
- SELECT
- WHERE
- ORDER BY
- GROUP BY

query.list()

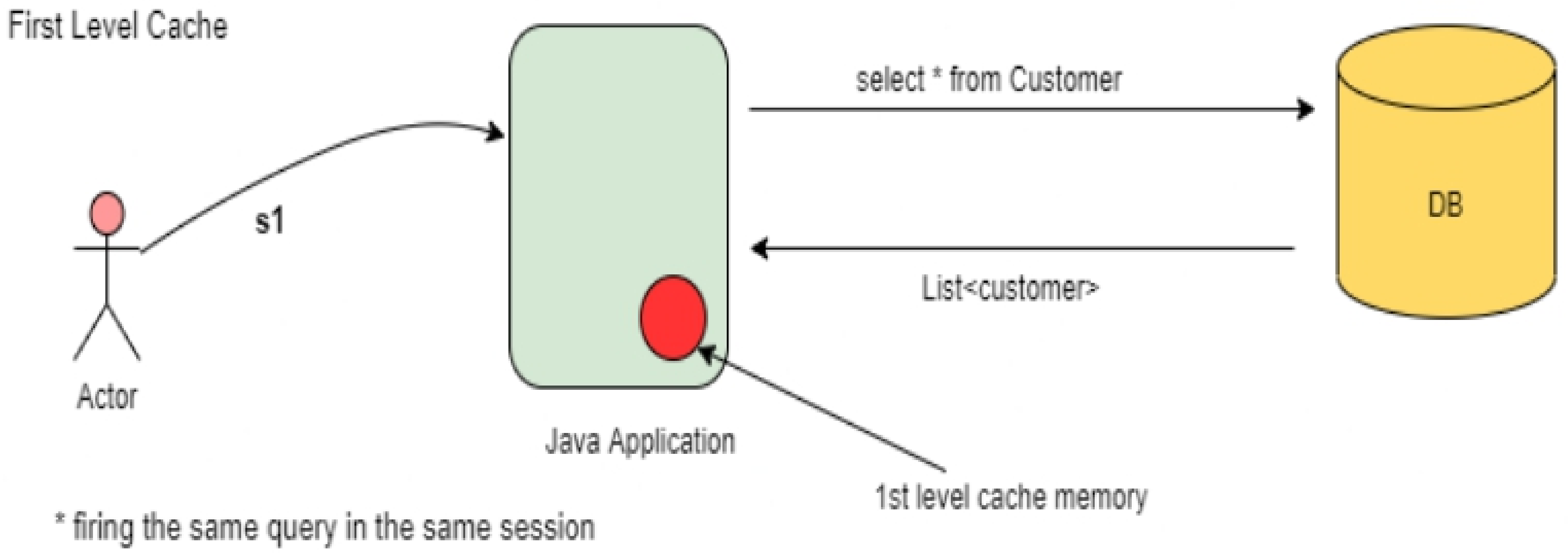
Using Named Parameters

```
String id = "C001";
String hql = "FROM Customer WHERE id = :customer_id";
Query query = session.createQuery(hql);
query.setParameter("customer_id", id);
List<Customer> result = query.list();
System.out.println(result);
```

Hibernate Caching

There are 2 types of hibernate caching levels that we can use in Hibernate. Which ar

1. First Level Caching
2. Second Level Caching



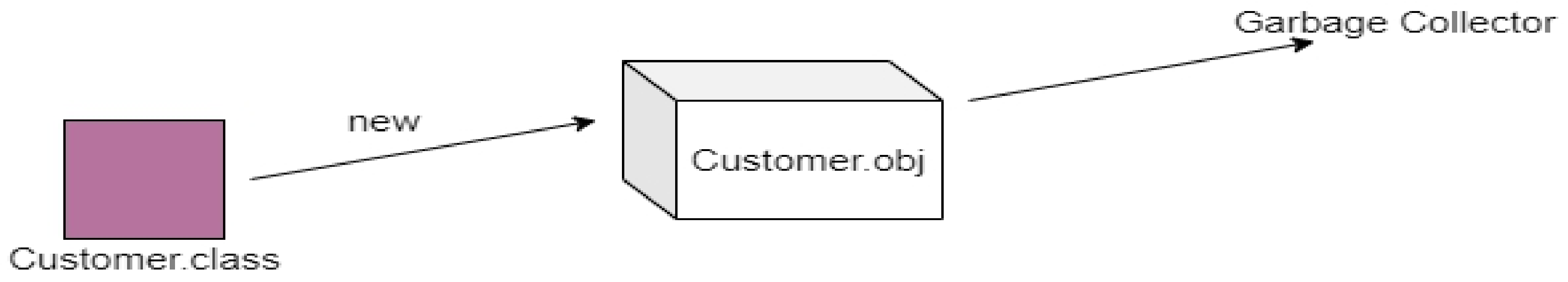
Second Level Caching

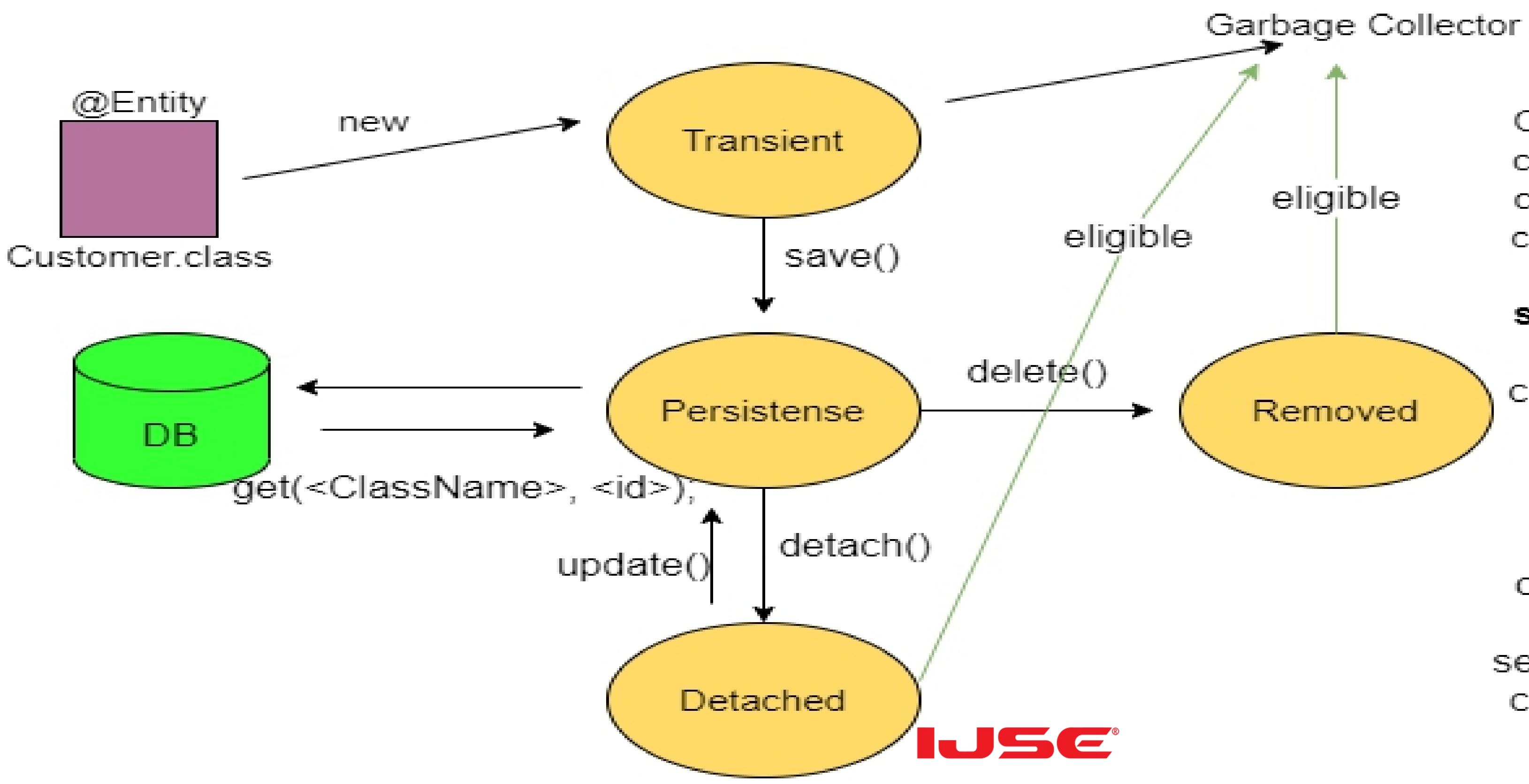
- This type of caching memory is not by default given by the Hibernate
- Therefore we need to get help of third party cache providers. Such as,
  - ehcache
  - swarm
  - os

- So, we have certian things to do before the Second Level Caching using.

1. Download the libraries that we need. (ehcache, hibernate-ehcache)
2. Configure out hibernate.cfg.xml file to allow second level cache.
3. Need to change our Entity as cacheble support by using annotation.
   - @cacheble
   - @Cache

Hibernate Object State /





```
Customer c = new Customer();
c.setId("C001");
c.setName("Danuja");
c.setAddress("Galle");

session.save(c);

c.setAddress("Panadura");

 session.detach(c);

c.setAddress("Colombo");

session.update(c);
 c.setAdddress("Colombo")
```