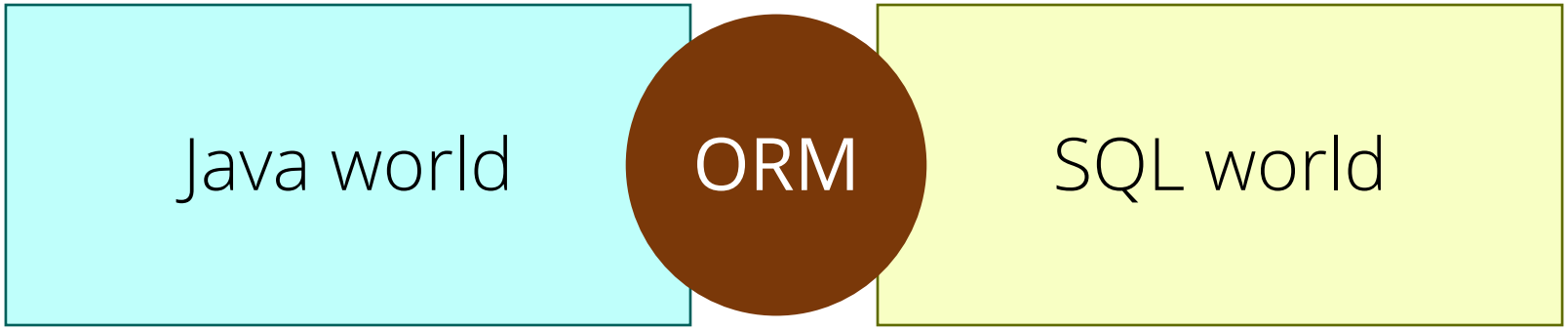
Hibernate

COMS10012 Software Tools

ORM



Classes
Objects
Types

Relations/Tables
Attributes/Columns
Types (different ones)

JPA

```
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Party {
    @Id public int id;
    public String name;
}
```

JPA

```
@Entity
public class Candidate {
    @Id public int id;
    public String name;

    @ManyToOne @JoinColumn(name="party")
    public Party party;
}
```

Hibernate

```
// import org.hibernate.CLASSNAMES


try (SessionFactory sf =
     new Configuration().configure().
     buildSessionFactory()) {

     // do stuff
}
```

Configuration

```
<!-- hibernate.cfg.xml -->

<property name="connection.url">jdbc:mariadb:...</property>
<property name="connection.username">vagrant</property>

<mapping class="org.example.Candidate" />
```

Sessions

```
try (Session session =
      sessionFactory.openSession()) {

    Party p = session.get(Party.class, 1);
}
```

Queries

```
TypedQuery<Ward> query = session.createQuery(
"FROM Ward WHERE name = :name", Ward.class);


query.setParameter("name", "Clifton");


List<Ward> wards = query.getResultList();
```

JOINs

By default, Hibernate does not JOIN, so if you ask for a Candidate then you won't get the party/ward objects loaded.

Instead, Hibernate gives you a Candidate *proxy object* that will load this information if you try and read it.

N+1 problem

```
List<Candidate> candidates =
  query.getResultList();


for (Candidate c : candidates) {
    System.out.println("Candidate " +
        c.getName() + " is in ward " +
        c.getWard().getName());
}
```

N+1 solution

```
session.createQuery(
  "FROM Candidate c JOIN FETCH c.ward",
  Candidate.class);
```

JOIN FETCH =
"I'm going to use these, so please preload them".