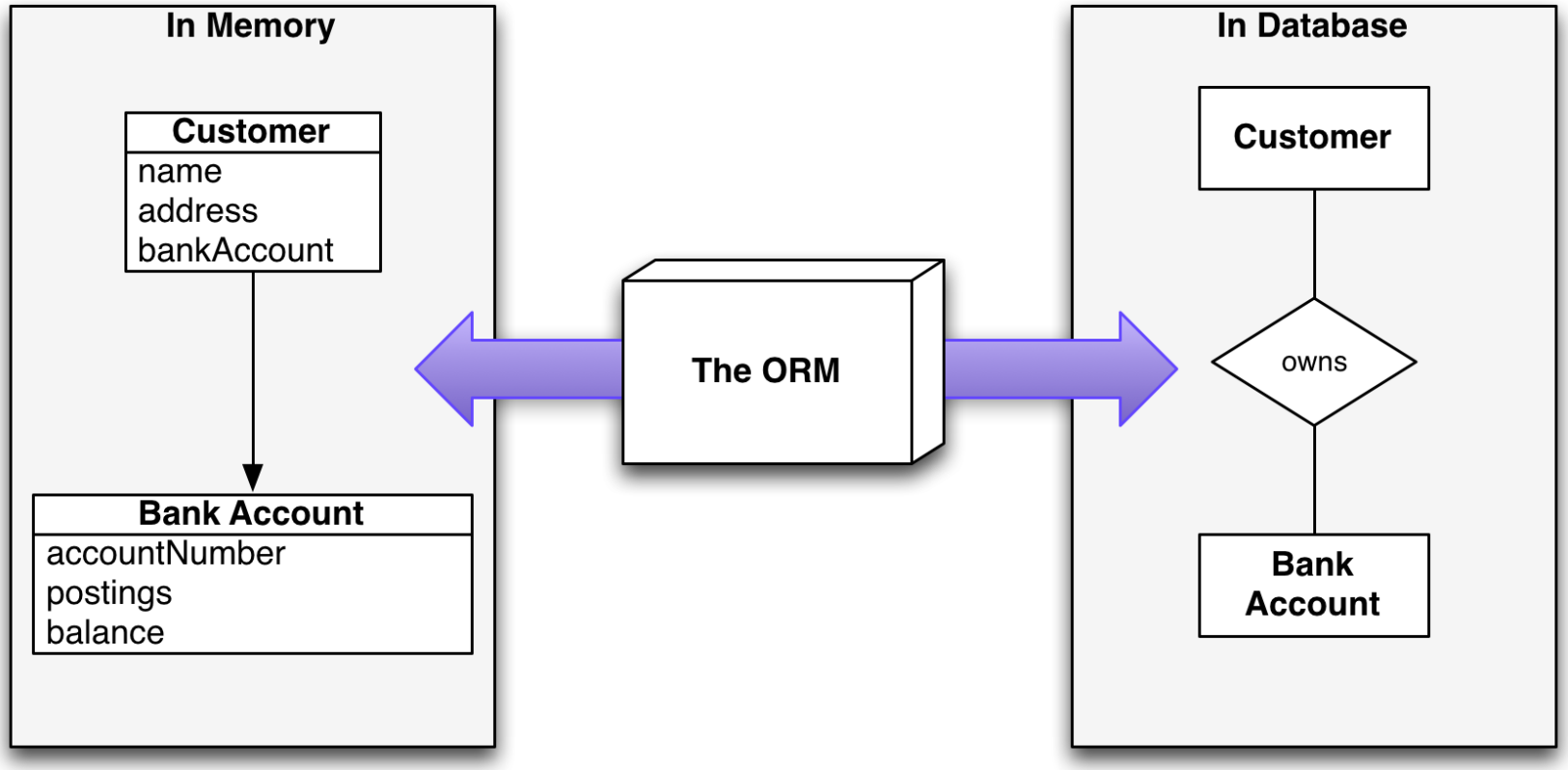
ORM

- Object Relational Mapping
- DataSet
- Anonymous classes and lambdas
- Pattern Executor
- Examples

Object Relational Mapping

ORM is technique for converting data between objects and tables in a relational database.



DataSet

DataSet is an object which stores result of selecting of one row from a table.

```
public class UsersDataSet {
  private long id;
  private String name;

  public UsersDataSet(long id, String name){
    this.id = id;
    this.name = name;
  }

  public String getName() {
    return name;
  }
  public long getId() {
    return id;
  }
}
```

DataSet is a part of ORM.

It represents a row of a table as an object.

Anonymous class

Anonymous classes enable you to declare and instantiate a class at the same time.

They are like local classes except that they do not have a name.

Example in the code

Lambda expression

Lambda expressions enable you to treat functionality as method argument.

Lambda expressions let you express instances of single-method classes without Anonymous classes.

Example in the code

Pattern Executor

Executor is an object which can process requests

It holds connection to DB and creates and manages statements

In case of update request it is relatively simple:

```
    public int execUpdate(Connection connection, String update) throws SQLException {
        try (Statement stmt = connection.createStatement()) {
            stmt.execute(update);
            return stmt.getUpdateCount();
        }
    }
```

How to select with help of Executor

The problem:

we can’t return ResultSet from Executor. ResultSet must be closed in the methods of Executor.

Let’s use generics and lambdas

The method

```
public <T> T execQuery(String query, TResultHandler<T> handler) throws SQLException {
  Statement stmt = connection.createStatement();
  stmt.execute(query);
  ResultSet result = stmt.getResultSet();
  T value = handler.handle(result);
  result.close();
  stmt.close();
  return value;
}
```

Usage

```
String name = execT.execQuery("select * from users where id=1", result -> {
  result.next();
  return result.getString(2);
});
```

Usage with UserDataSet

```
UserDataSet dataSet = execT.execQuery("select * from users where id=1", result -> {
  result.next();
  return new UserDataSet(result.getInteger(1), result.getString(2));
});
```