Using RowMappers

Make sure to import the **`RowMapper`** in the **`org.springframework.jdbc.core`** package!

Implements the **`RowMapper<T>`** interface. Make sure to specify the class type (**`State`** in this case).

```java
import com.techelevator.model.State;
import org.springframework.jdbc.core.RowMapper;

import java.sql.ResultSet;
import java.sql.SQLException;

public class StateRowMapper implements RowMapper<State> {

    @Override
    public State mapRow(ResultSet resultSet, int i) throws SQLException {
        State state = new State();
        state.setStateAbbreviation(resultSet.getString("state_abbreviation"));
        state.setStateName(resultSet.getString("state_name"));
        return state;
    }
}
```

Implement the required **`mapRow`** method. This will do what the **`mapRowToState`** method you wrote before did.

We can use the **`jdbcTemplate.query`** method instead of **`queryForRowSet`** which allows us to provide a **`RowMapper`**. This method will map the results to a **`List`** of the specified type and return the **`List`** without us needing to work with **`SqlRowSet`**.

```
@Override
public List<State> getStates() {
   String sql = "SELECT state abbreviation, state name FROM state" ;
   List<State> stateList = jdbcTemplate.query(sql, new StateRowMapper()) ;
   return stateList;
}

@Override
public State getStateByAbbreviation (String stateAbbreviation ) {
   String sql = "SELECT state abbreviation, state name FROM state  state WHERE state abbreviation = ? ";
   State theState = jdbcTemplate.queryForObject( sql, new StateRowMapper(), stateAbbreviation );
   return theState;
}
```

When using **`queryForObject`**, we can now provide a **`RowMapper`** to allow custom classes to be mapped.