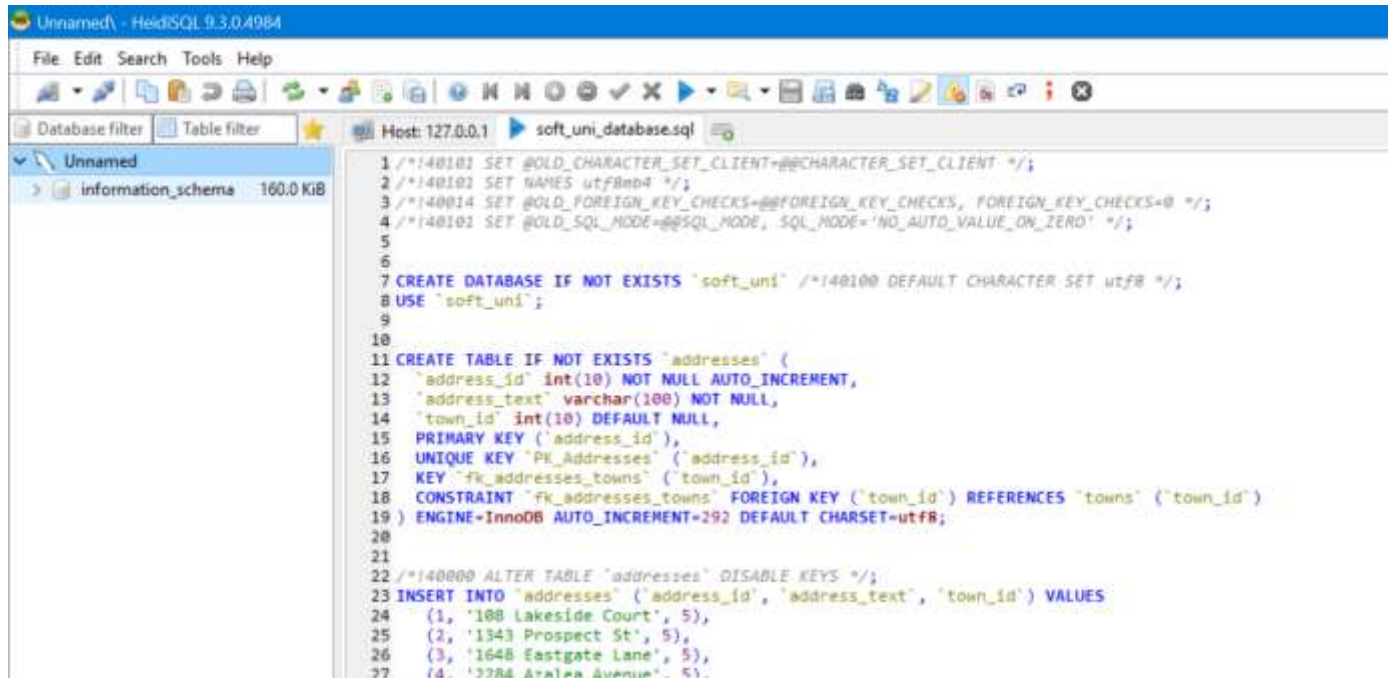


Exercises: Introduction to Hibernate

This document defines the **exercise assignments** for the "[Databases Advanced – Hibernate](#)" course @ [Software University](#). Please submit your solutions in [Judge](#).

1. Import the SoftUni Database

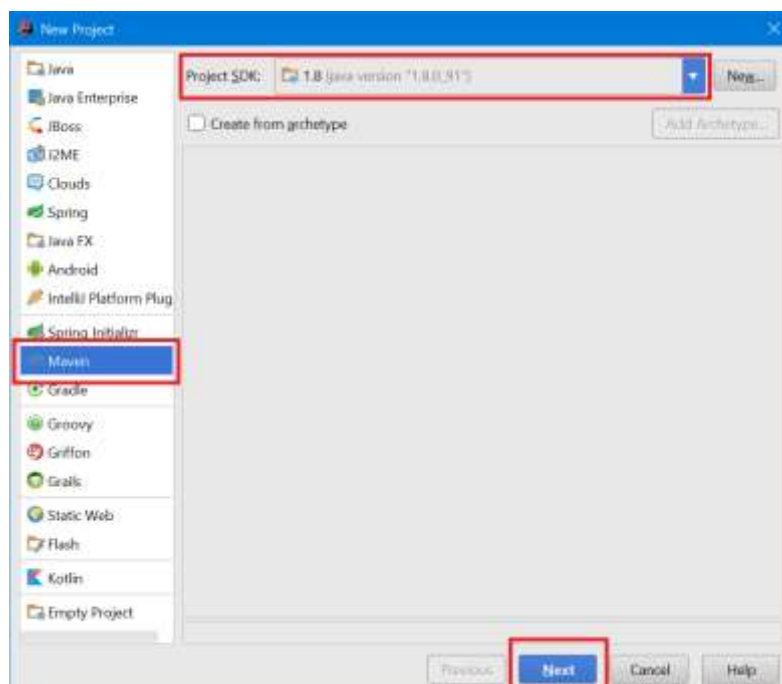
Import the **soft_uni** database into HeidiSQL by **executing** the provided **.sql** script.

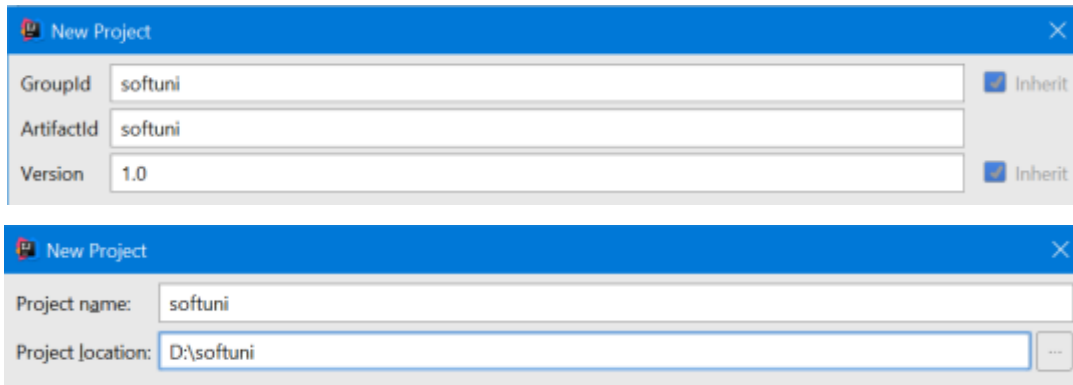


```
1 /*140101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
2 /*140102 SET NAMES utf8mb4 */;
3 /*140014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
4 /*140101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
5
6
7 CREATE DATABASE IF NOT EXISTS `soft_uni` /*!40100 DEFAULT CHARACTER SET utf8 */;
8 USE `soft_uni`;
9
10
11 CREATE TABLE IF NOT EXISTS `addresses` (
12   `address_id` int(10) NOT NULL AUTO_INCREMENT,
13   `address_text` varchar(100) NOT NULL,
14   `town_id` int(10) DEFAULT NULL,
15   PRIMARY KEY (`address_id`),
16   UNIQUE KEY `PK_addresses` (`address_id`),
17   KEY `fk_addresses_towns` (`town_id`),
18   CONSTRAINT `fk_addresses_towns` FOREIGN KEY (`town_id`) REFERENCES `towns` (`town_id`)
19 ) ENGINE=InnoDB AUTO_INCREMENT=292 DEFAULT CHARSET=utf8;
20
21
22 /*140000 ALTER TABLE `addresses` DISABLE KEYS */;
23 INSERT INTO `addresses` (`address_id`, `address_text`, `town_id`) VALUES
24 (1, '108 Lakeside Court', 5),
25 (2, '1343 Prospect St', 5),
26 (3, '1648 Eastgate Lane', 5),
27 (4, '2284 Avalon Avenue', 5),
```

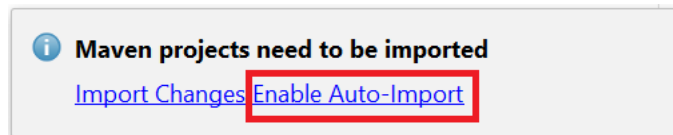
2. Persist Database

Create **new maven project** in IntelliJ IDEA.



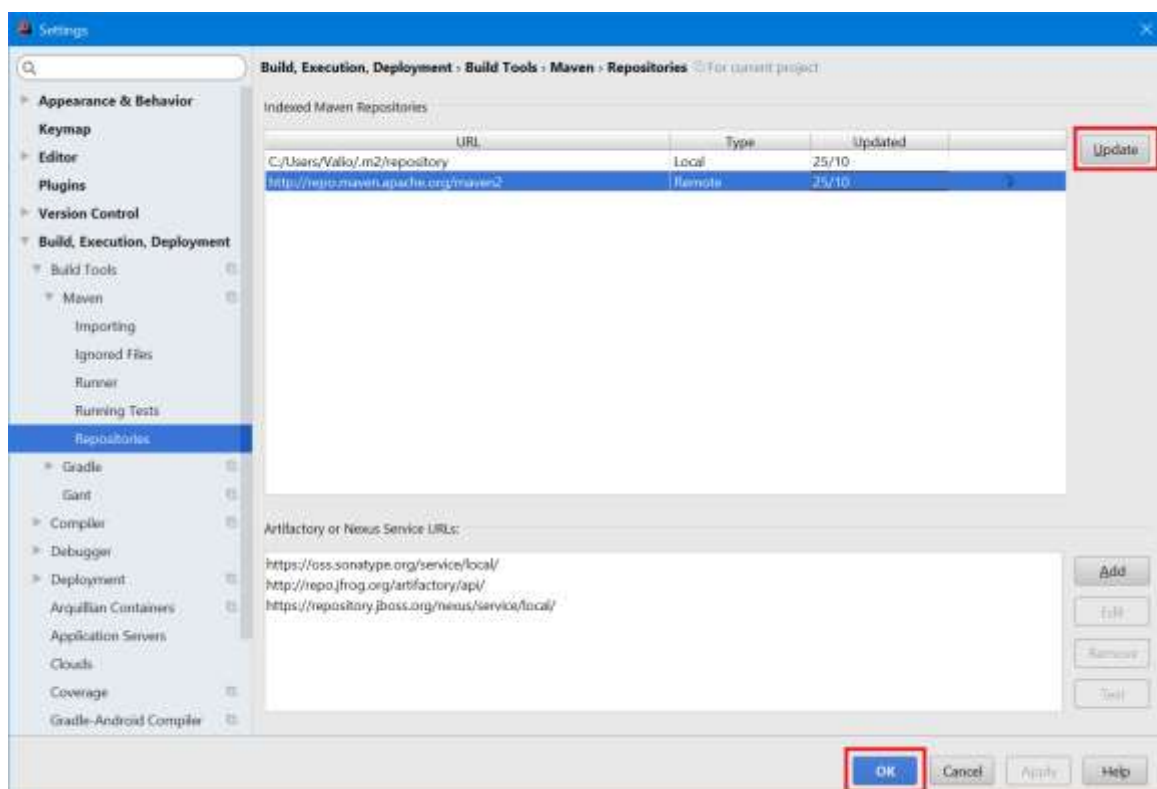


When the project loads a pop up appears that prompt you to enable auto import dependencies so dependencies would be automatically downloaded.

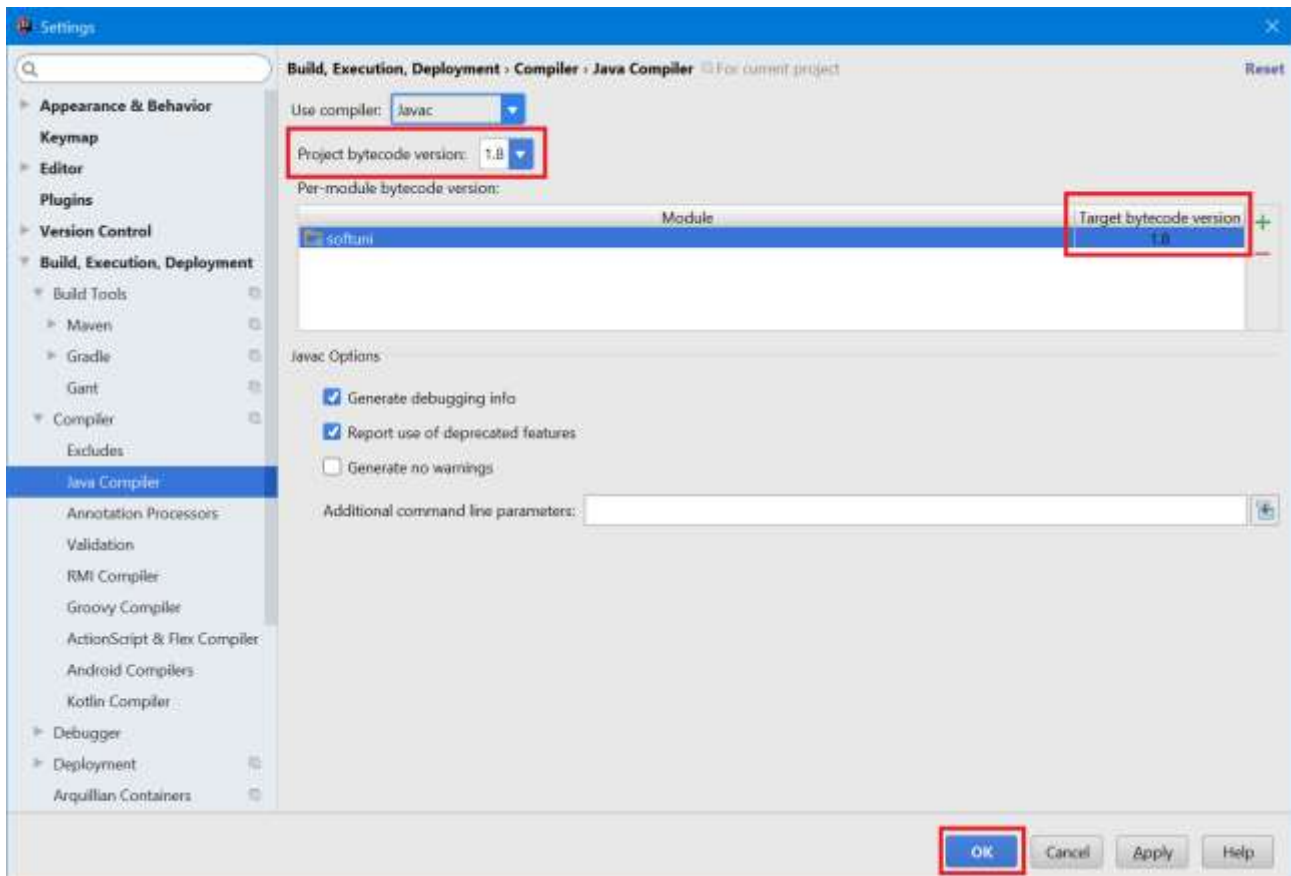


If that pop up did not show up you can turn them on from **File -> Settings -> Build, Execution, Deployment -> Maven -> Importing** and check “**Import Maven projects automatically**” then click **OK**.

The maven repositories should be updated so you can easily download and use any of the latest dependency projects you need to help you in your poroject. You can do than from **File -> Settings -> Build, Execution, Deployment -> Maven -> Repositories**. Then select every repository that you have in the list and click on **Update** button.

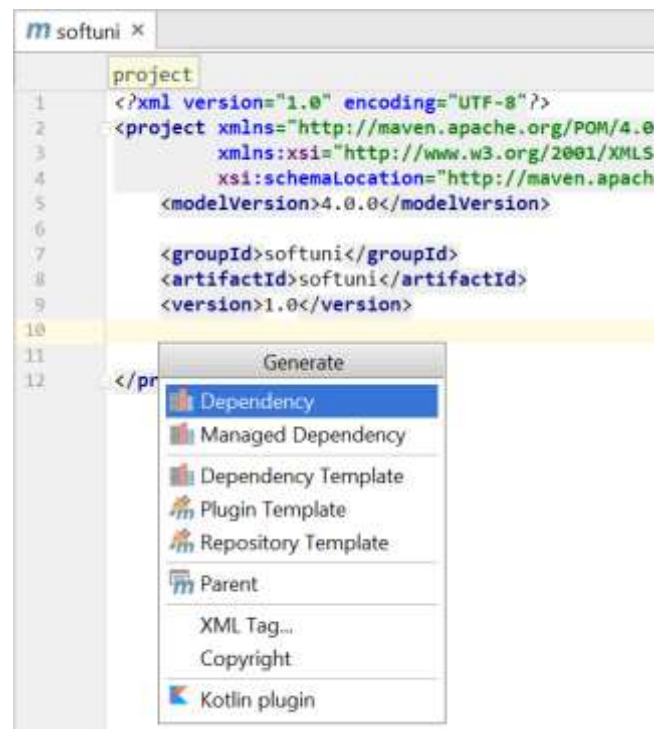


By default IntelliJ set build target of the project to be for Java 5 (pretty old). So we should change it to later version. Go to **File -> Settings -> Build, Execution, Deployment -> Compiler -> Java Compiler** and change **Project bytecode version** and **Target bytecode version** to **1.8**.



Configure maven – set hibernate dependency, set mysql connector dependency – provided templates

Now it is time to configure our dependencies. Open pom.xml file and use the shortcut **Alt + Insert** then select **Dependency**.



The dependencies we need are **hibernate-core 5.2.3.Final** and **mysql-connector-java 5.1.39**.

```

<dependencies>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.3.Final</version>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.39</version>
  </dependency>
</dependencies>

```

When the dependencies are downloaded create new folder META-INF inside resources folder. Inside it create file persistence.xml and put the following template in it. Change the database name, username and password accordingly to your database.

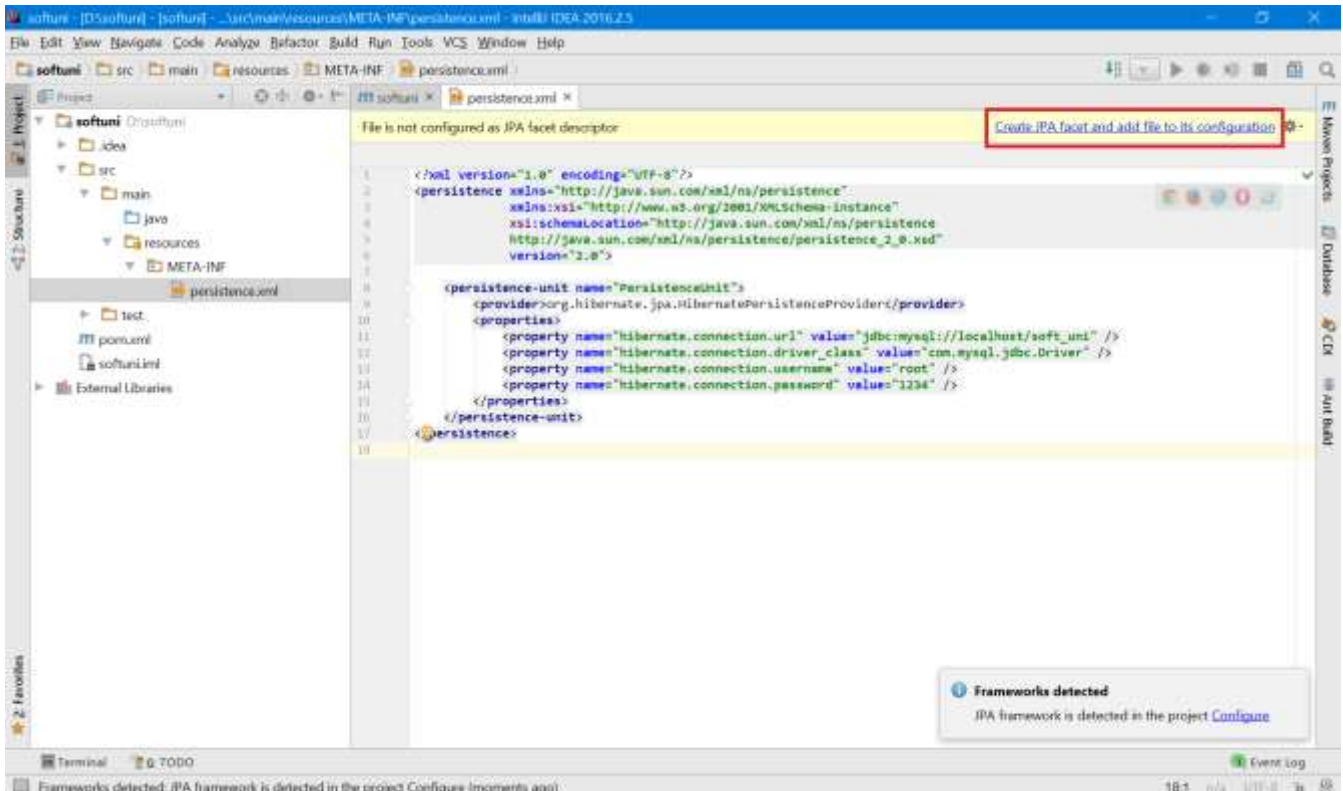
persistence.xml

```

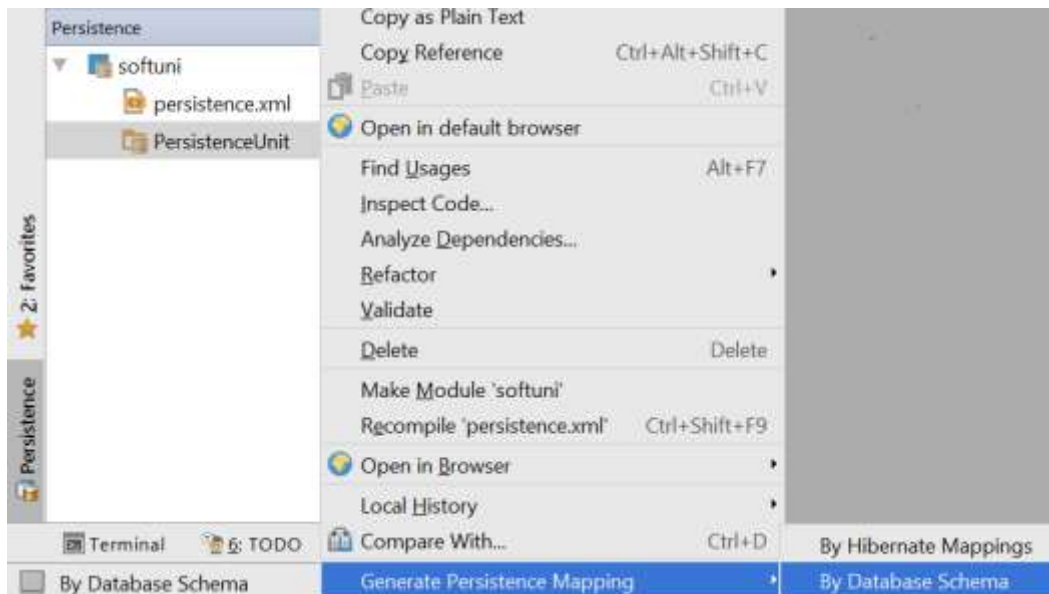
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">

  <persistence-unit name="PersistenceUnit">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="hibernate.connection.url" value="jdbc:mysql://localhost/DATABASE_NAME" />
      <property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver" />
      <property name="hibernate.connection.username" value="USERNAME_FOR_DATABASE" />
      <property name="hibernate.connection.password" value="PASSWORD_FOR_DATABASE_USER" />
    </properties>
  </persistence-unit>
</persistence>

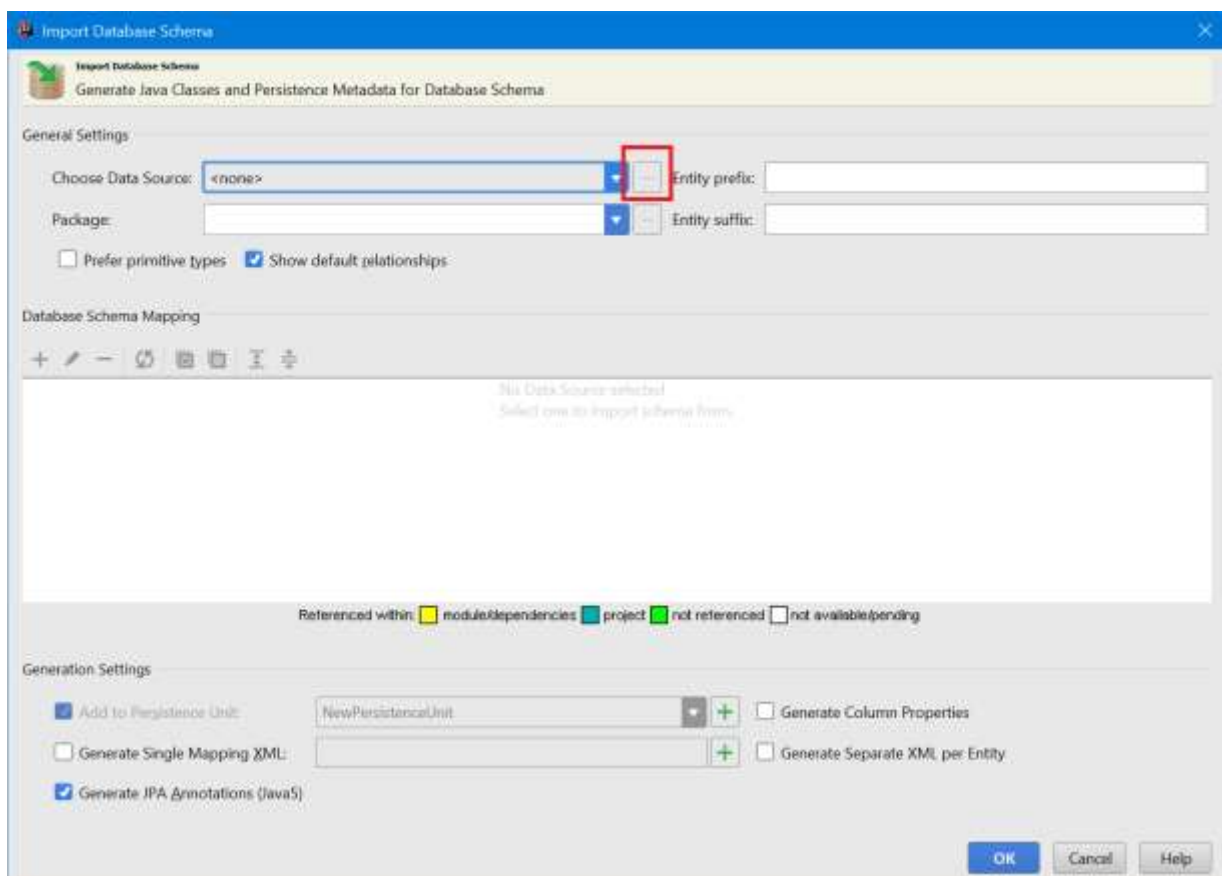
```



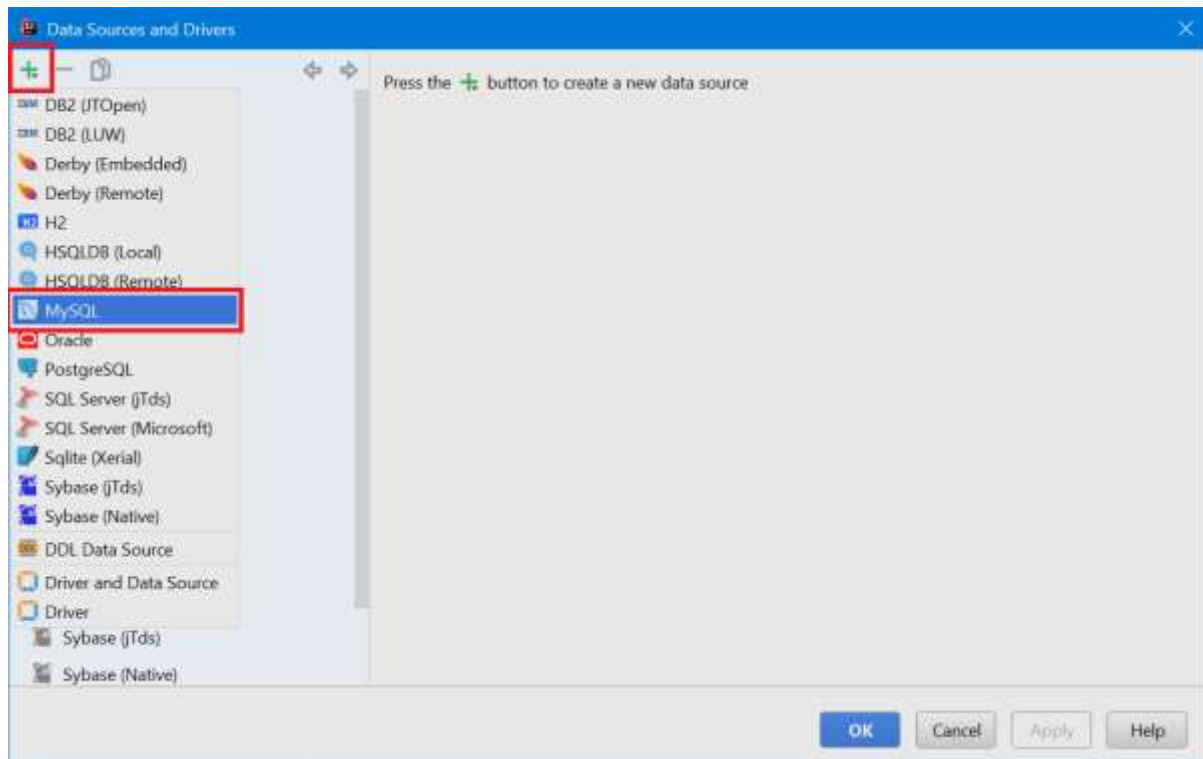
Now we can persist our classes from the database. Open the **Persistence** panel in IntelliJ and **right click on the Persistence unit** we created then select **Generate Persistence Mapping -> By Database Schema**



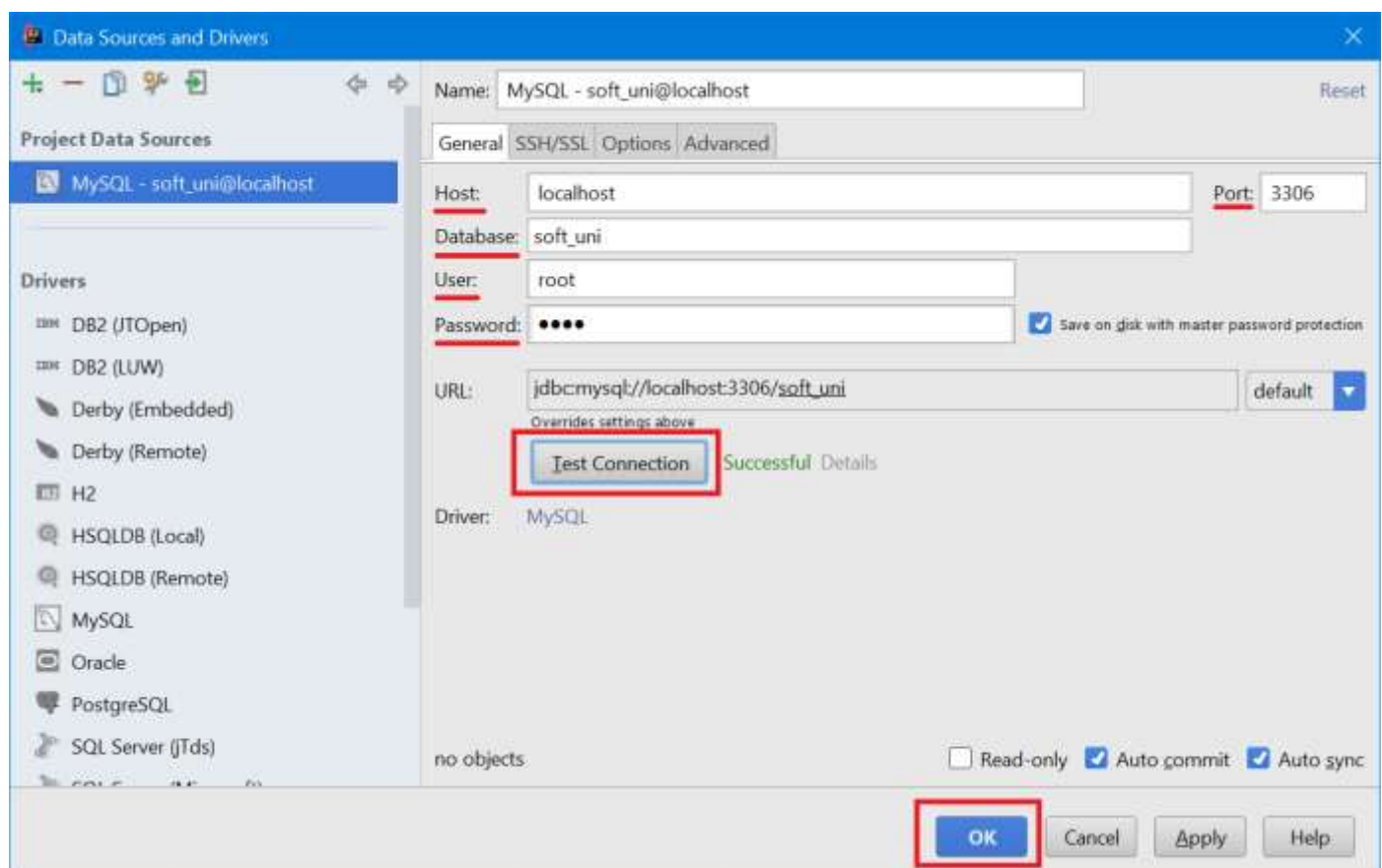
In this window, we can map the tables from the database to Java classes. First create and select **connection to the database**.



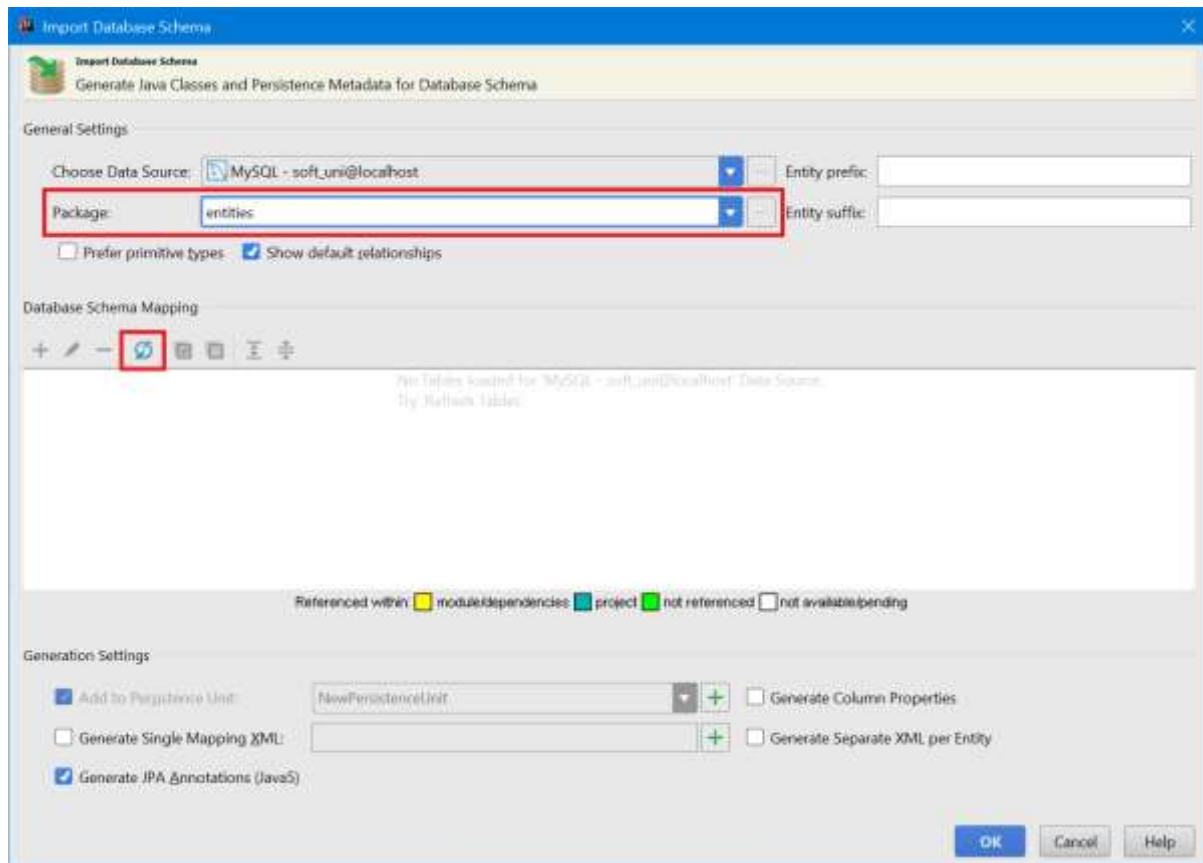
Choose **MySQL** as data source.



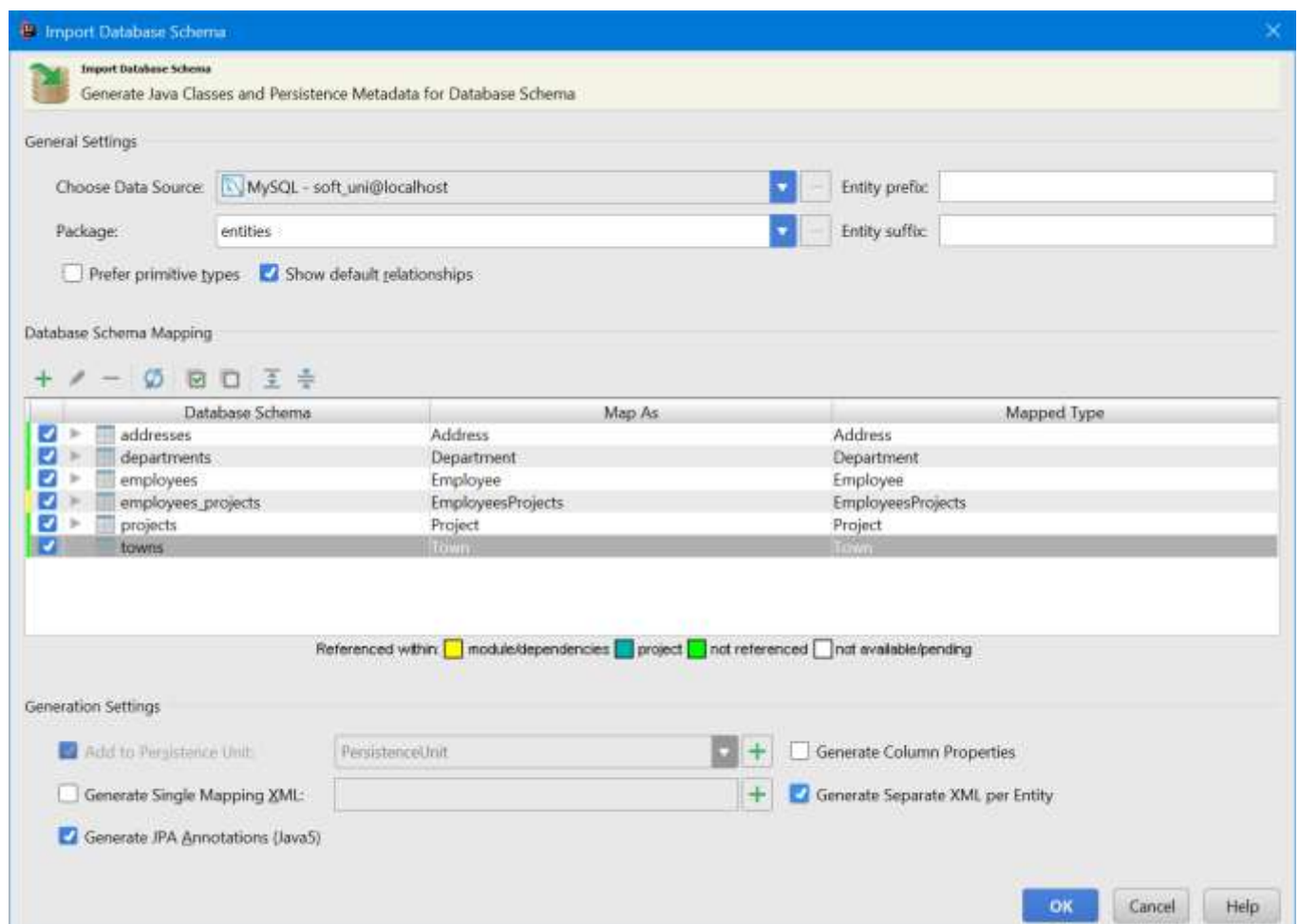
Fill the required information (such as **host**, **port**, **database**, **user** and **password**) to make connection to the database. **Test the connection** to make sure it is successful.



Choose a package to put our mapping classes (entities). Then click on the **refresh button** to see all the tables and information about how they would be mapped in our application.

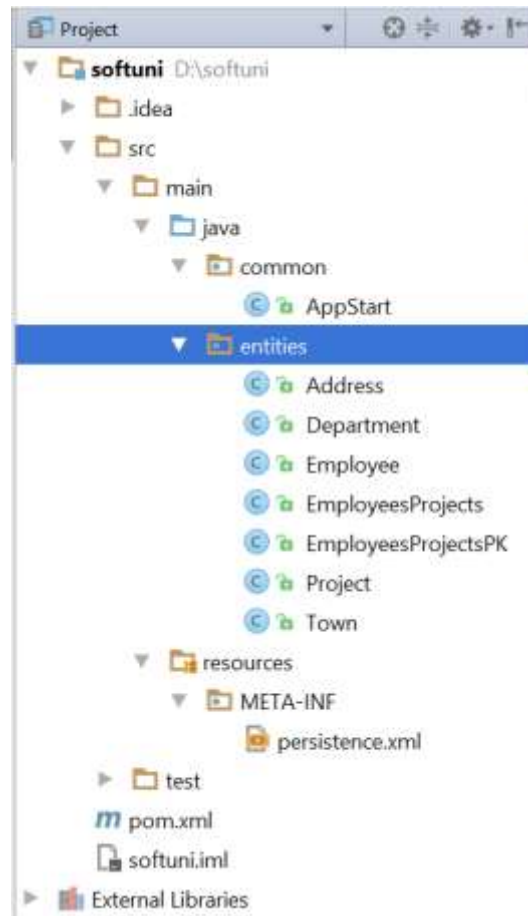


Make sure you selected every single one of the tables. Here you can see and control how every table, column and relationship between them would be named and mapped to our Java classes.



Last but not least create new Java class and put main method in it. You can now start manipulating the database through Java code.

Finally, we are done and we can see your classes in the entities package ready to be used in our application



3. Create Objects

Use the **soft_uni** database. Create new **Department Training**. Create new **Town Burgas**. Create **several new addresses** that would be **in Burgas**. Create **5 employees** and **assign them in the Training department** and make their addresses to be some of the addresses in Burgas. Make sure they are **all persisted** in the database.

4. Remove Objects

Use **soft_uni** database. Persist **all towns** from the database. Detach those whose name length is **more than 5 symbols**. Then transform the **name** of all attached towns **to lowercase** and **save them to the database**.

5. Contains Employee

Use **soft_uni** database. Write a program that check if given employee name as an input **is contained in the database**.

Example

Input	Output
Svetlin Nakov	Yes
John Doe	No

6. Data Refresh

Use **soft_uni** database. Get employee with id 4. Transform his first name to be uppercase. Then refresh the entity manager and persist that employee. Now print the first name of the employee. Was the uppercase transform successful?

7. Employee Queries

Step 1 - Employees with Salary Over 50 000

Let's start writing queries! Your first task is to extract **all employees** with **salary** over **50000**. Make sure the query returns **only the first names** of those employees.

```
public class Main {
    public static void main(String[] args) {
        EntityManagerFactory emfactory = Persistence.createEntityManagerFactory("PersistenceUnit");

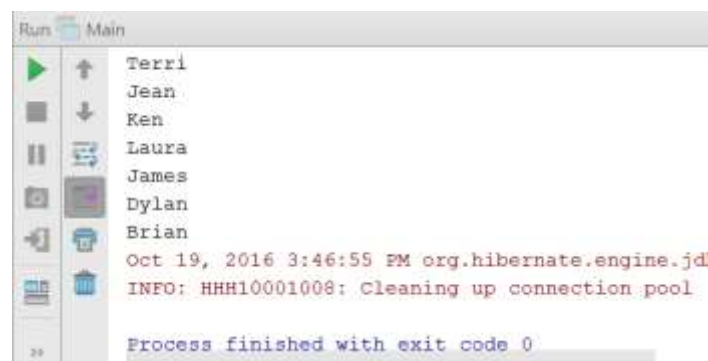
        EntityManager entitymanager = emfactory.createEntityManager();

        //TODO: Create query to select first name of employees with salary > 50000
        List<String> employeesNames = query.getResultList();

        for (String employeeName: employeesNames) {
            System.out.println(employeeName);
        }

        entitymanager.close();
        emfactory.close();
    }
}
```

Result on console:



```
Run Main
↑
↓
Terri
Jean
Ken
Laura
James
Dylan
Brian
Oct 19, 2016 3:46:55 PM org.hibernate.engine.jdt
INFO: HHH10001008: Cleaning up connection pool
Process finished with exit code 0
```

Step 2 - Employees from Seattle

Extract all employees from the **Research and Development** department. Order them by **salary** (in ascending order), then by **first name** (in descending order). Print only their **first name**, **last name**, **department name** and **salary**.

```

public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("PersistenceUnit");
    EntityManager entityManager = emf.createEntityManager();

    //TODO: get all employees from "Research and Development" department
    List<Employee> employees = query.getResultList();
    //TODO: order them by their salary (asc) then by first name (desc)

    for(Employee employee : employees){
        System.out.printf("%s %s from %s - $%.f2\n",
            employee.getFirstName(),
            employee.getLastName(),
            employee.getDepartment().getName());
    }
}

```

Result on console:

```

Run Main
Gigi Matthew from Research and Development - $40900.00
Diane Margheim from Research and Development - $40900.00
Michael Raheem from Research and Development - $42500.00
Svetlin Nakov from Research and Development - $48000.00
Martin Kulov from Research and Development - $48000.00
George Denchev from Research and Development - $48000.00
Dylan Miller from Research and Development - $50500.00

Process finished with exit code 0

```

8. Adding a New Address and Updating Employee

Create a new address with text "Vitoshka 15". Set that address to the employee with last name "Nakov"

```

public static void main(String[] args) {
    EntityManagerFactory emfactory = Persistence.createEntityManagerFactory("PersistenceUnit");

    EntityManager entitymanager = emfactory.createEntityManager();
    entitymanager.getTransaction().begin();
    Address address = new Address();
    address.setAddressText("Vitoshka 15");
    entitymanager.persist(address);

    Employee employee = null;
    //Create query to get employee with last name 'Nakov'
    //Set address to new address

    entitymanager.persist(employee);
    entitymanager.getTransaction().commit();
    entitymanager.close();
    emfactory.close();
}

```

The above code should successfully insert a new address in the database and set it as Nakov's new address.

```

Query query = entitymanager.createQuery("select e from Employee e where e.lastName = 'Nakov'");
Employee employee = (Employee) query.getSingleResult();

System.out.println(employee.getAddress().getAddressText());

```

```

Run Main
Vitoshka 15
Oct 20, 2016 9:56:09 AM org.hibernate
INFO: HHH10001008: Cleaning up connec
Process finished with exit code 0

```

9. Delete Project by Id

Use the entity manager to **delete the project** with **Id 2** from the database. Make sure there is no trace in the database after that.

10. Database Search Queries

Write the following queries:

- Find all employees who have **projects** started in the time period **2001 - 2003** (inclusive). Print each employee's **first name**, **last name** and **manager name** and each of their projects' **name**, **start date**, **end date**.
- Find all addresses, **ordered** by the **number of employees** who live there (descending), then by **town name** (ascending). Take only the **first 10 addresses** and print their **address text**, **town name** and **employee count**.

Query Materialization	Example Query Result
<pre> for (Address address : addresses) { System.out.printf("%s, %s - %d employees\n", address.getAddressText(), address.getTown().getName(), address.getEmployees().size()); } </pre>	163 Nishava Str, ent A, apt. 1, Sofia - 3 employees 7726 Driftwood Drive, Monroe - 2 employees 7902 Grammercy Lane, Bellevue - 1 employees 5678 Clear Court, Bellevue - 1 employees 2284 Azalea Avenue, Bellevue - 1 employees 9745 Bonita Ct., Bellevue - 1 employees 6448 Castle Court, Bellevue - 1 employees 3454 Bel Air Drive, Bellevue - 1 employees 7808 Brown St., Bellevue - 1 employees 4777 Rockne Drive, Bellevue - 1 employees

- Get an **employee by id** (e.g. 147). Print only his/her **first name**, **last name**, **job title** and **projects** (only their names). The projects should be **ordered by name** (ascending).
- Find **all departments** with more than **5 employees**. Order them by **employee count** (ascending). Print the **department name**, **manager name** and first name, last name, hire date and job title of every **employee**.

Query Materialization
<pre> System.out.println(departments.size()); for (Department department: departments) { System.out.printf("--%s - Manager: %s, Employees: %d\n", department.getName(), department.getManager().getLastName(), department.getEmployees().size()); } </pre>
Query Result

13

```
--Engineering - Manager: Duffy, Employees: 6
--Production Control - Manager: Krebs, Employees: 6
--Human Resources - Manager: Barreto de Mattos, Employees: 6
--Shipping and Receiving - Manager: Ackerman, Employees: 6
--Research and Development - Manager: Trenary, Employees: 7
--Quality Assurance - Manager: Word, Employees: 7
--Facilities and Maintenance - Manager: Altman, Employees: 7
--Marketing - Manager: Harnpadoungsataya, Employees: 8
--Finance - Manager: Tamburello, Employees: 10
--Information Services - Manager: Trenary, Employees: 10
--Purchasing - Manager: Bradley, Employees: 12
--Sales - Manager: Welcker, Employees: 18
--Production - Manager: Hamilton, Employees: 179
```

11. Concurrent Database Changes with Entity Manager

Open two different entity managers and perform concurrent changes on the **same records** in some database table.

1. Begin transaction in the first entity manager and make changes to a column
2. Begin transaction the second entity manager and make changes to a column
3. Consecutively call `getTransaction().commit()` on both entity managers

What will happen at `getTransaction().commit()`?

Change the **locking mode** of the objects to **pessimistic**. Run the code again and see if there are any differences.

12. Find Latest 10 Projects

Write a program that prints **last 10 started projects**. Print **their name, description, start and end date** and **sort them by name** lexicographically.

Output

Name	Description	Start Date	End Date
All-Purpose Bike Stand	Research, design and developm...	2005/09/01	(NULL)
Bike Wash	Research, design and developm...	2005/08/01	(NULL)
HL Touring Frame	Research, design and developm...	2005/05/16	(NULL)
...

13. Increase Salaries

Write a program that increase salaries of all employees that are in the **Engineering, Tool Design, Marketing or Information Services** department by **12%**. Then **print first name, last name and salary** for those employees whose salary was increased.

Example

Output
Kevin Brown (\$15120.00)
Roberto Tamburello (\$48496.00)
Rob Walters (\$33376.00)
...

14. Remove Towns

Write a program that **deletes town** which name is given as an input. Also **delete all addresses** that are in those towns. Print on the console the number addresses that were deleted.

Example

Input	Output
Sofia	1 address in Sofia was deleted
Seattle	44 addresses in Seattle were deleted

15. Find Employees by First Name

Write a program that finds all employees whose first name starts with pattern given as an input from the console. Print their first, last name, their job title and salary in the format given in the examples below.

Example

Input	Output
SA	Sariya Harnpadoungsataya - Marketing Specialist - (\$14400.00) Sandra Reategui Alayo - Production Technician - (\$9500.00) Sairaj Uddin - Scheduling Assistant - (\$16000.00) Samantha Smith - Production Technician - (\$14000.00) Sameer Tejani - Production Technician - (\$11000.00) Sandeep Kaliyath - Production Technician - (\$15000.00)

16. Employees Maximum Salaries

Write a program to find the **max salary** for each **department**. Filter those which have max salaries not in the range 30000 and 70000.

Example

Output
Tool Design - 29800.00
...

17. Deposits Sum for Ollivander Family

Use the **Gringotts database**. Write a program that print all **deposit groups** and its **total deposit sum** but only for the wizards who has their magic wand crafted by Ollivander family.

Output

Output
Human Pride - 188366.86
...

18. Deposits Filter

Use the **Gringotts database**. Write a program that print all **deposit groups** and its **total deposit sum** but only for the wizards who has their magic wand crafted by Ollivander family. After this filter total deposit amounts lower than 150000. Order by total deposit amount in descending order.

Example:

Output
Troll Chest - 126585.18
...

19. First Letter

Use the **Gringotts database**. Write a program that prints all unique wizard first letters of their **first names** only if they have deposit of type Troll Chest. Order them alphabetically. Use GROUP BY for uniqueness.

Example

Output
A
...