

Data Layer Documentation – companydata.jar

NOTE: When connecting to the remote DB, you must be on campus or using VPN.

Add **companydata.jar**, and **mysql-connector-java-8.0.27.jar** files to your project. You will need to add: `import companydata.*;` to the top of your class that wants to access the Data Layer.

To use it (see included TestDataLayer.java example):

```
DataLayer dl = null;

try {
    dl = new DataLayer("ritusername"); //e.g. kxmzgr
    //call the DataLayer method you want

} catch (Exception e) {
    //deal with the error - can be thrown by any of the methods
    below
} finally {
    dl.close();
}
```

Methods (all throw an exception if there is an error, such as the requested object doesn't exist):

//Delete all Departments, Employees and Timecards for a company

//Returns the number of rows deleted

public int deleteCompany(String companyName)

//Insert a department

//Returns the inserted Department

public Department insertDepartment(Department department)

//Get all Departments for a given company

//Returns a list of Department objects

public List<Department> getAllDepartment(String companyName)

//Get a given Department for a given company

//Returns the requested Department

public Department getDepartment(String companyName, int dept_id)

```
//Update a Department
//Returns the updated Department
public Department updateDepartment(Department department)
```

```
//Delete a given Department for a given company
//Returns the number of rows deleted
public int deleteDepartment(String company, int dept_id)
```

```
//Insert an Employee
//Returns the inserted Employee
public Employee insertEmployee(Employee employee)
```

```
//Get a list of Employees for a given company
//Returns the list of Employee objects
public List<Employee> getAllEmployee(String companyName)
```

```
//Get the requested Employee
//Returns the requested Employee
public Employee getEmployee(int emp_id)
```

```
//Update an Employee
//Returns the updated Employee object
public Employee updateEmployee(Employee employee)
```

```
//Delete the given Employee
//Returns the number of rows deleted
public int deleteEmployee(int emp_id)
```

```
//Insert a Timecard
//Returns the inserted Timecard
public Timecard insertTimecard(Timecard timecard)
```

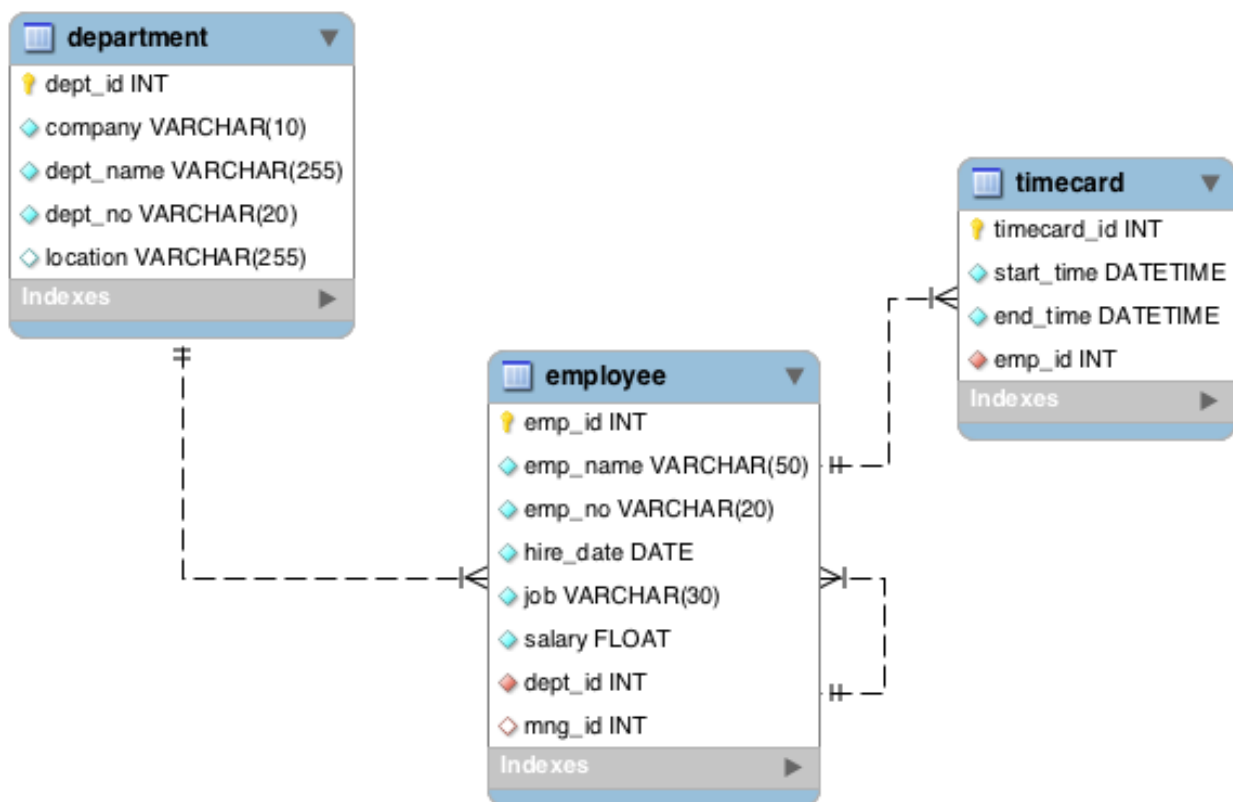
```
//Get all Timecards for a given Employee
//Returns the list of Timecard objects
public List<Timecard> getAllTimecard(int emp_id)
```

```
//Get the requested Timecard
//Returns the requested Timecard
public Timecard getTimecard(int timecard_id)
```

```
//Update a given Timecard
//Returns the updated Timecard
public Timecard updateTimecard(Timecard timecard)
```

```
//Delete the given Timecard
//Returns the number of rows deleted
public int deleteTimecard(int timecard_id)
```

```
//Close the connection
public void close()
```



The **employee** table has two foreign keys:

1. mng_id references **employee.emp_id**
2. dept_id references **department.dept_id**

The **employee** table has an additional unique index of emp_no besides the the primary key

The **timecard** table has one foreign key:

1. emp_id references **employee.emp_id**

The **department** table has one additional unique index of dept_no besides the primary key

Department	Employee	Timecard
<div><div>- String company</div><div>- String dept_name</div><div>- String dept_no</div><div>- String location</div><div>- int dept_id</div><div>+ String getCompany()</div><div>+ String getDeptName()</div><div>+ String getDeptNo()</div><div>+ String getLocation()</div><div>+ String toString()</div><div>+ int getId()</div><div>+ void setCompany(String)</div><div>+ void setDeptName(String)</div><div>+ void setDeptNo(String)</div><div>+ void setId()</div><div>+ void setLocation(String)</div></div>	<div><div>- Date hire_date</div><div>- Double salary</div><div>- String emp_name</div><div>- String emp_no</div><div>- String job</div><div>- int dept_id</div><div>- int emp_id</div><div>- int mng_id</div><div>+ Date getHireDate()</div><div>+ Double getSalary()</div><div>+ String getEmpName()</div><div>+ String getEmpNo()</div><div>+ String getJob()</div><div>+ int getDeptId()</div><div>+ int getId()</div><div>+ int getMngId()</div><div>+ void setDeptId(int)</div><div>+ void setEmpName(String)</div><div>+ void setEmpNo(String)</div><div>+ void setHireDate(Date)</div><div>+ void setId(int)</div><div>+ void setJob(String)</div><div>+ void setMngId(int)</div><div>+ void setSalary(Double)</div></div>	<div><div>- Timestamp end_time</div><div>- Timestamp start_time</div><div>- int emp_id</div><div>- int timecard_id</div><div>+ Timestamp getEndTime()</div><div>+ Timestamp getStartTime()</div><div>+ int getEmpId()</div><div>+ int getId()</div><div>+ void setEmpId(int)</div><div>+ void setEndTime(Timestamp)</div><div>+ void setId(int)</div><div>+ void setStartTime(Timestamp)</div></div>

Each class has constructors of:

Employee:

Employee(String emp_name, String emp_no, Date hire_date,
String job, Double salary, int dept_id, int mng_id)

Employee(int emp_id, String emp_name, String emp_no, Date hire_date,
String job, Double salary, int dept_id, int mng_id)

Department:

Department(String company, String dept_name, String dept_no,
String location)

Department(int dept_id, String company, String dept_name,
String dept_no, String location)

Department(String company, int dept_id)

Department()

Timecard:

Timecard(Timestamp start_time, Timestamp end_time, int emp_id)

Timecard(int timecard_id, Timestamp start_time, Timestamp end_time,
int emp_id)

Timecard()