Java SE 8 Programming Language

Training Assignment

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, Apr/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	18/Apr/2018	Add a new exam	Add	DieuNT1	VinhNV
2	26/Apr/2019	Update content for automatic build	Automatic build	DieuNT1	VinhNV
3	05/Jun/2019	Update from student feedback Fsoft template	Student feedback	DieuNT1	VinhNV

Contents

CODE:	JPL.Practice.T01
TYPE:	N/A
LOC:	N/A
DURATION:	180 MINUTES

Require 001: Working tools and Delivery requirements

- **Working tools**: Eclipse IDE for Java
- **Delivery**: Source code and test results in a compressed archive.

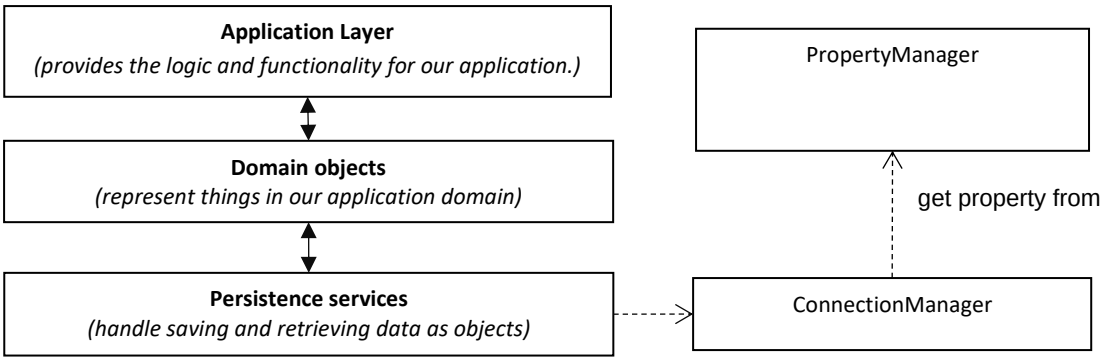
Require 002: Technologies

The product illustrates:

- OOP: Inheritance, Encapsulation, Polymorphims, Abstraction
- String, Java Collection (List, Set, Map)
- Lambda Expressions and Functional Interfaces
- Java I/O Fundamentals
- JDBC: Statement, PreprareStatement, CallableStatement, Batch
- Base Java Knowledge in the course.

Require 003: Technical Requirements

will be developed using the following architecture:



- The *Application Layer* consists of all functions described in the **functional requirements** section.
- The *domain layer* contains objects and logic related to our application. In this layer, you need to develop all the entity classes corresponding to your tables in database.
- The *persistence layer* contains data access objects (DAO) that provide services for accessing persistent data. DAO provide 4 basic services referred to as CRUD:
  - **Create**: save new object data to the database.
  - **Retrieve**: find object data in the database and recreate objects. There may be several methods for this service to enable different forms of object lookup.
  - **Update**: update data for an object already saved to the database.
  - **Delete**: delete object data from the database
- Each layer should be organized in a separated package.

Require 004: Technical Requirements

- Use Object-Oriented programming style.
- Follow the standard naming and coding convention.
- Add appropriate comments for each class, method, attribute, …
- Use console application template
- Create a new project and the appropriate packages
- Programming Java with JDBC.

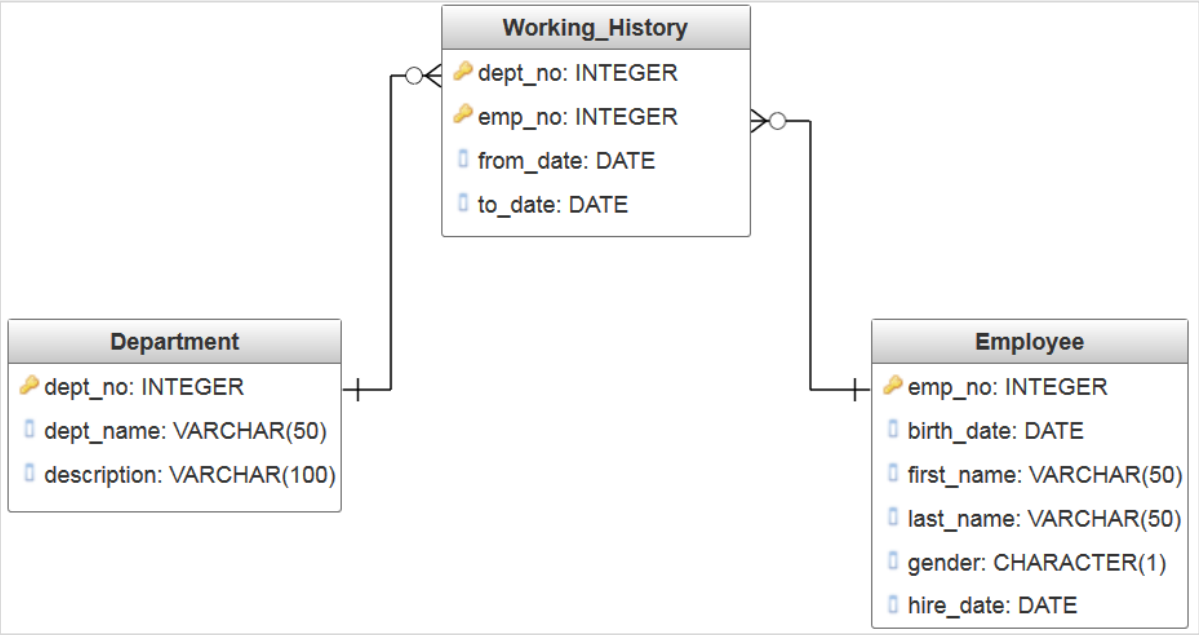
Create a project named **JPL.Practice.T01** to resolve the follow problems:

Specifications:

Students are required to develop a Java console application based on Java core and JDBC programming knowledge learned from the course to manage Employees (Employee Management System).

Database Requirements:

Create a new database schema named **JPL_TEST01** for this application that contains the table following:



Contraints:

- ✓ The *to_date* must be greater than *from_date*
- ✓ The *dept_name*, *first_name*, *last_name* should have a maximum of 50 characters.

Notice that, using default schema is 'dbo'.

Technical Requirements:

Create a new package named **fa.training.problem02** in **JPL.Practice.T01** project.

The trainee must create some appropriate sub-package to contain classes in this problem.

E.g

- ✓ *fa.training.problem02.entities* to manage entity classes**,**
- ✓ *fa.training.problem02.dao* to manage data access objects,
- ✓ *fa.training.utils* to manage the classes that process data constraint requirements, class utility classes, if need, etc.

Functional Requirements:

a. Write a function to create a new employee (*emp_no, birth_date, first_name, last_name, gender, hire_date*.) in database using a stored procedure (method name ***save***(Employee employee).

b. Write a function to list all the employees in the table, each employee with following information: *emp_no, birth_date, first_name, last_name, gender, hire_date (*method name List<Employee> ***findAll***()).

c. Write a function to update an employee info which includes *birth_date, first_name, last_name, gender, hire_date*. This function should use a stored procedure to do its work (method name ***update***(Employee employee)).

d. Write a function to find an employee by its *emp_no*. This function should return all the details (emp_no, birth_date, first_name, last_name, gender, hire_date.) of this employee (method name ***findById***(emp_no)).

e. Write a function to create a new department in database, using a stored procedure (method name ***save***(Department department)).

f. Write a function to add the working history of an employee to the **Working_History** table with the following info: *emp_no, dept_no, from_date, to_date* (method name ***save***(**Working_History** workingHistory)).

g. Write a function to find all the employees who were working in a department in a period of time, which is entered from user (method name List<Employee> ***findByWorkTime***(Date fromDate, Date toDate)).

Note*: all the functions you have implemented above must use entity classes, persistence classes in domain and persistence layers.*

User Interface Requirement

The program has a screen console for UI

The main screen allows selecting the functions for:

1. Employee management
    a. Add a new Employee
    b. Update a specific Employee
    c. Find an employee by emp_no
    d. Add the working history
    e. Find all the employees by working period of time
2. Department management
    a. Add a new department
3. Close program

Estimate time: 120 minutes

Mark scale :

- OO design/Class design Architecture : 15%;
- DB Design/Connection : 15%;
- Functional Requirement : 55%;
- User Interface Requirement : 15%

--THE END--