Java SE Programming Essentials

Training Exam

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

Hanoi, 06/2020

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	12/06/2020	Create a new Exam	Create new	DieuNT1	VinhNV

Contents

CODE	**:**	**JPE.Practice.T02**
TYPE	**:**	**Long**
LOC	**:**	**n/a**
DURATION	**:**	**180 minutes**

General Requirements

Require 01: Working tools and Delivery requirements

- **Working tools**: Eclipse IDE for Java, an appropriate Database (SQL Server, MySQL, Oracle, Derby 10.14) is downloaded and ready to use.
- **Delivery**: Source code and test results in a compressed archive.

Require 02: Technologies

The product illustrates:

- Base Java knowledge in the course.
- OOP: Inheritance, Encapsulation, Polymorphims, Abstraction
- String, Java Collections (List, Set, Map)
- JDBC: Statement, PreprareStatement, CallableStatement, Batch

Require 03: 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 **JPE.Practice.T02** to resolve the follow problems:

Problem 01. String, Java IO

Objectives:

- Understand basics of Java IO such as FileInputStream, BufferReader.

Problem Descriptions:

Given a file named "**course_register.txt**" that contains course registration data the following as:

```
1001_Nguyen Quang Anh_Java
1002_Nguyen Van Khoi_Java
1003_Le Thu Huong_FrontEnd
1004_Hoang Xuan Minh_NET
1005_Do Manh Truong_FrontEnd
1006_Vu Manh Phong_C++
<blank_line>
```

Write a program to read the file and count the number of students in each programming language.

Create a package named ***fa.training.problem01*** and class named **CourseRegister** that contains the following method to resolve the above problem:

```
public Map<String, Integer> countStudent(String filePath) {

}
```

Create a main() method to call countStudent() and show results the following as:

```
Java          2
FrontEnd      2
NET           1
C++           1
```

Estimate time: 30 minutes

Mark scale: 20%

- Create package, class, method: 10%;
- Problem solving : 70%;
- Main method : 20%;

Problem 02. Modulus/Divide, Control Flow

Objectives:

- Understand basics of controll flow statements, modulus/divide.

Problem Descriptions:

Write a program to returns the "reverse" of the input positive integer.

```
Enter a positive integer: 12345
The reverse is: 54321
```

Create a package named ***fa.training.problem02*** and class named **ReverseInt** that contains the following method to resolve the above problem:

```
public int reverseInt(int input) {

}
```

Create a JUnit test class named **ReverseIntTest**, and write five different test cases which exercise the method of the **ReverseInt** class (*notice that, students not must create main() function*).

Estimate time: 30 minutes

Mark scale: 20%

- Create package, class, method: 10%;
- Problem solving : 50%;
- Unit Test : 40%;

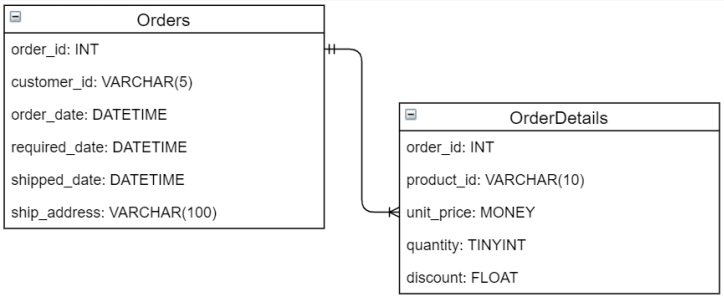
Problem 03. OOP, JDBC, Exception

Specifications:

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

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

Table: **Orders and OrderDetails**



Technical Requirements:

Create a new package named **fa.training.problem03** in **JPE.Practice.T02** project.

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

E.g

- ✓ *fa.training.problem03.models* to manage entity classes, ex: Order, OrderDetail
- ✓ *fa.training.problem03.dao* to manage data access objects, ex: OrderDao, OrderDaoImpl
- ✓ *fa.training.utils* to manage the classes that process data constraint requirements, class utility classes, if need, etc.

Functional Requirements:

a) The program has a method to create a new order with the required_date must greater than or equals order_date (method named *public String* **save**(Order order)). Returns "success" if a new record is added successfully into database table, otherwise will return "fail".

b) The program has a method to create a new order detail (method named *public String* **save**(OrderDetail orderDetail)). Returns "success" if the record update was successful, otherwise will return "fail".

c) Write a method returns the total amount of money each customer has ordered today (method named *public Map<String, Double> reportOfSale()*).

d) Write a method returns a list of all the orders of a customer that has id '12126' (method named *public List<Order> findOrderByCustomer(String customerId)*).

Console Screens:

Create a main() method to run the program.

1. Create a new order
2. Create a new order detail
3. Total money
4. List order

Estimate time: 120 minutes

Mark scale : 60%

– OO design/Class design	: 15%;	– Functional Requirement	: 55%;
– DB Design/Connection	: 15%;	– Main	: 15%;

-- THE END --