Refactoring Lab Exercises

Objective:

You will be provided with two applications archives.

- Application 1 – This is a swing based application.
- Application 2 – This is a web based application.

The objective of this lab exercise is you will be provided few problems, which you would have to solve using one or more refactoring techniques and find a solution.

Application 1:

Prerequisite:

Import the jar into workspace for swing Application:

- Open Your SDE 6.0.
- Go to File Menu, there click the import option.
- A window named import will be opened there click General and click "Existing Project into Workspace" and click the Next button (for select the root directory).
- There click the radio button named "Select the root directory" and with the help of Browse button select the location of the jar file.
- Then Click the next radio button named "Select the Archived File" and with the help of Browse button select the jar file click next and finish.
- The given jar file will be imported into your work space.

The application has the following problems apply refactor techniques and fix the issues.

Problem 1:

a. The names of some of the java files are spelt wrongly and are not adhering to the design specifications apply refactoring technique and rename the file name.

Wrong file Name	Design Specification
Lofic.java	SearchLogic.java
DadaConnect.java	DataConnect.java
SearghForm.java	SearchForm.java

b. The names of some variables are wrongly typed apply refactoring technique and rename the following variables.

File Name	Wrong Variable Name	Correct Variable Name
SearchLogic	rset	nameSet
DataConnect.java	con	connection
SearchForm.java	str	buttonCaption
ShowResult.java	slojic	searchLogic

c. The names of some methods are wrongly typed apply refactoring technique and rename the following methods.

File Name	Wrong Method Name	Correct Method Name
DataConnect.java	getConf	getConnection
SearchLogic.java	srcnam	searchNames

Problem 2:

The packages are wrong for few classes refactor and move it to the right packages.

Wrong Package Name	Design Specification
DataConnect.java & SearchLogic	Move to package "com.ccd.refactory.lab1.dataaccess, "
SearchForm .java and ShowResult.java	Move to Package "com.ccd.refactor.lab1.view"
MainApp.java	Move to Package "com.ccd.refactory.lab1.controller"

Problem 3:

The customer.java files has member variables apply refactoring technique and generate getter/setters for these.

Problem 4:

Some methods have been wrongly defined in the sub class , this needs to be moved to the parent class.

There are three classes Manager, Developer, & Tester , All the three classes have a common method printSalaryReciepts,applyLeave and getPFDetails. Since this is a common method repeated in three different place move this to a common super class named Employee. All the three classes should extend it.

Problem 5:

Make Changes in the return type and the parameters of the following method

a. Change the Return Types:

Java File Name	Method Name	Old Return Type	New Return Type
Employee	`getPfDetails`	Void	Long (return 1000 as default)
SearchLogic	searchNames	ArrayList	List (return the first name in the list)

b. Change the Parameters:

Java File Name	Method Name	Parameter Name	Old Parameter Type	New parameter Type
Customer	`getCustDetails`	Empid	String	Int
Customer	`getCustDetails`	Sal	Int	long

Problem 6:

Extract the methods, printSalaryReciepts,applyLeave and getPFDetails from the Employee class to an interface named IEmployee and Employee class should implement this interface

Problem 7:

In SearchLogic class there is a method named searchNames which is using ArrayList declaration, instead use the interface java.util.List . Refactor the code to use java.util.List declaration.

Problem 8:

In Customer.java a long literal named highSalary and a String literal “Manager” is used .Extract this to a constant so that it can be reused across the file.

Problem 9:

The method “weightCal ” in the Java file “SearchLogic ” complexity point is high. Extract some lines in the if else block to a private method and make it readable and maintainable. Also reduce the number of lines of this method from 50 to 30.

Application 2:

Prerequisite:

Import the war file into workspace for the web Application:

- Open Your SDE 6.0.
- Go to File Menu, there click the import option.
- A window named import will be opened there click Web and then click “war file” and click the Next button.
- Then give the project name, that is in which project you want to add the jar file, if the particular project does not exist create an web project and give the name of that web project there and select the corresponding server with the help of new button near to the “target Runtime” label .
- When you press the window a new window named “New Server Runtime Environment” will l be opened .select the corresponding server and click next
- There select the corresponding directory of the server and click finish and finally finish the import war file window.
- Now the war file will be added in the workspace.

This application has following problems

Problem 1:

Servlet should not have any business logic, the business logic implemented inside the Servlet “MyServlet” extract the method “getCon” and “getDetails” to a separate class DataConnect and EmpDetails respectively and create objects of these classes and invoke the methods.

Problem 2 :

The names of some variables are wrongly typed apply refactoring technique and rename the following variables.

File Name	Wrong Variable Name	Correct Variable Name
EmpDetails	q1	query
MyServlet.java	Rd	requestDispatch
DataConnect.java	Con	Connection
EmpDetails .java	Rs	empResultSet
EmpDetails .java	ab	employeeDetails

Problem 3:

The Servlet has a member variable registerNumber lets move this to a value object EmployeeVO and generate getters and setters. In the Servlet create a instance of the EmployeeVO and set the register number.

Problem 4:

Create new packages and put the following classes to it.

Java File	Old Package	New Package
MyServlet.java	com.ccd.refactor.lab1	com.ccd.refactor.lab1.controller
EmployeeVO.java	com.ccd.refactor.lab1	com.ccd.refactor.lab1.model
DataConnect.java	com.ccd.refactor.lab1	com.ccd.refactor.lab1.dataacess
EmpDetails	com.ccd.refactor.lab1	com.ccd.refactor.lab1.dataacess