Purpose of the Document:

The purpose of the document is to illustrate few examples which the trainer needs to demonstrate to the associates when conducting the refactoring sessions.

Theory Source Package Details:

Demo 1:

a. Renaming files:

Rename the files empt.java and main.java to Employee.java and MainApplication.java.

Files to be Renamed	Names of the Renamed Files
Empt.java	Employee.java
main.java	MainApplication.java

b. Renaming variables:

Rename the following variables in the above renamed file Employee.java

Names of The Renaming Variable	Names Of The Renamed Variables
empid	employeeId
ename	employeeName
eplace	employeePlace

Solution:

a. For renaming the file, click the file to be renamed and do the following steps

- Select the class file name and right click and in Refactor menu select "**Rename**".

Or

- Select the class file name in the source editor and press Alt+Shift+R.

a. For renaming the variable, do the following steps

- Select the variable to be renamed right click and in Refactor menu select "**Rename**".
- Rename the variable and press enter.

Or

Select the variable name in the source editor and press Alt+Shift+R.

Demo 2: **Moving files across packages:**

Lets move Employee.Java,Company.java and MainApplication.java into com.ccd.refactor.theory

Solution:

For moving the file to other package, select the file to be moved in package explorer and do the following steps

- Select the class name in the java source editor, right click and in Refactor menu select the move option.
- A Window named "**Move**" will be opened there select the package for moving the java class or create an new package with the help of create package button in the move window ,then tick the two check boxes below and then press the preview button and then Ok button and Finish

or

- Select the class file name in source editor and press Alt+Shift+V.
- A Window named "Move" will be opened there select the package for moving the file or create an new package with the help of "create package" button in the move window, then tick the two check boxes below and then press the preview button and then Ok button and Finish

Demo 3: **Change Method Signature:**

Change the parameter of constructor of Employee.java.

Old Signature of the Parameter	New Signature of the Parameter
long empId	int empId

Solution:

For Renaming the signature of a constructor or a method, select the method right click then do the following

- In refactor menu select “**Change Method Signature**”.
- A Window named “**Changed Method Signature**” will be opened there select the corresponding access modifier, return type. There you will be having two tabs Parameter and Exceptions.
- By clicking Parameters you will be getting type, name, and default value and you can add, remove and edit new parameters with the help of add, remove and edit button.
- By clicking Exceptions you will be getting add, remove button, by clicking add button list of all exceptions will be shown in a window, you select a particular exception you want to be thrown for the particular method and click preview and ok.

Or

- Select the method name and press Alt+Shift+C.
- A Window named “Changed Method Signature” will be opened there select the corresponding access modifier, return type. There you will be having two tabs Parameter and Exceptions.
- By clicking Parameters you will be getting type, name, default value and you can add, remove and edit new parameters with the help of add, remove and edit button.
- By clicking Exceptions you will be get add,remove button, by clicking add button list of all expetions will be shown in a window, you select a particular exception you want to be thrown for the particular method and click preview and ok.

Demo 4: Generalize Declaration Type

In MainApplication class, method getDetails() we have a ArrayList declared in main method change the declaration to List to make it implementation agnostic, Select the reference variable of the ArrayList and do the following.

Solution:

- Right click the declaration and in refactor menu click “**Generalize Declared Type**”
- A window name “Generalize Declared Type” will be opened there choose the generalize type as List click ok and finish.

Now the Code will be

```
List    names=new ArrayList ();
```

<u>Demo 5:</u> Encapsulate Field :

In The Employee.java for all the class variables generate private generate getters and setters.

Solution

For that select the variables which you want to generate getters and setters and right click, do the following

- Go to refactor there click “Encapsulate Fields”
- A window name “Encapsulate Fields” will be opened there give the setter name and getter name ,select the corresponding item from the combobox and choose the respective access specifiers and click ok.

<u>Demo 6:</u> Extract Interface from a class:

Car contains methods such as startEngine, applyBreak, honk with return type as void which can be moved to an interface named IVehicle and implement the interface with the class Car

Solution

For that right click Car.java and do the following

- Right Click the class which you want to move all the methods to the interface, go to “refactor” and there select “**Extract Interface**”.

- A window named “**Extract Interface**” will be opened, there give a name to the interface which is to be created and select the corresponding methods in the checkbox which you want to move from the class to interface and click preview or ok (by this all the methods from the class will be moved to the interface as abstract methods and the class will be automatically implemented with the interface).

<u>Demo 7:</u> Pull Up Member Variables/Method.

In Employee.java there is a method called getPfStatus,

Company.java is the parent class of the employee class
Pull up the getPfStatus method to Company Class.

Solution:

- Right Click the Employee class go to "Refactor" there click "Pull Up" menu.

- A Window named "Pull Up" will be opened there give the name of the super class and tick the corresponding checkboxes which shows the method names and field names you want to pull up . Click Next and then Click Finish.

Demo 8: Extract Constant:

The same constant "INDIA" has been used in multiple places in the same employee class file. Use the extract constant to extract it to a String member variable.

Solution:

Select the values "INDIA" in the Employee.java and do the following

- Right click the selected value go to refactor, in refactor menu select "**Extract Constant**".

- A window named "**Extract Constant**" will be opened, give the name of the constant and select the corresponding access specifiers and check the corresponding checkboxes .

- The selected string will be converted to constant final class variable.

Demo 9: Inline :

In Employee.java there are two static methods namely addProp and main method.For better performance Inline the addProp method into the main method.

Solution:

- Select the method call "addProp(props, str);".
- Right click and in refactor menu click inline and click Finish.

Demo 10: Extract Method

In Employee.java there is a method named calculateLeave, try to extract the codes mentioned below in a separate method “validateManagerLeave”

```
if (leavetaken < 10) {
  leaveeligble = 12;

} else if (leavetaken < 15) {
  leaveeligble = 8;

  } else {
   leaveeligble = 2;
      }
```

Extract the below switch case present in main method into a method “assignWeightage”

```
switch (condition) {
case 'D':
      posweightage = 010;
case 'T':
      posweightage = 12;
case 'P':
      posweightage = 13;
case 'M':

                }
```

Solution:

- Select the lines of the method which you want to place it in some other method.
- Right Click go to Refactor there click Extract Method
- A window named Extract Method will be opened there give the name of the method which you want to create give the Access Specifiers and tick the corresponding check boxes and click ok.The lines of the method which you had selected will be moved to the new method.