**Group Member:**

Aimen Tahir (181400179)

Arslan Ahmad (181400019)

Assignment: 2

**Refactoring in NetBeans IDE**

**Introduction:**

Refactoring isa technique to rebuild an already existing body of code. It is a controlled technique for improving the design of existing code. Every refactoring makes a small alteration, but a sequence of alterations produces a noteworthy refactoring. Refactoring is difficult to go wrong since each refactoring is very small.

**Technique of Refactoring:**

These are some ways to refactor your code:

1. **Rename:**

 We can change the name of packages, classes and variables to some meaningful names. The NetBeans IDE updates the names of all the elements in your source code.

To rename a variable, method or class name, consider the following steps:

* Click right on the variable/method/class name and select "Rename" from the Refactor menu.
* A window will appear, in "New Name" enter the name that you want to update or change.
* Then click on the "Refactor" button to finish the operation.
* You can even preview the effect of refactoring by clicking on the "Preview" button.

1. **Change the existing block of code with a method:**

We can just create a statement on the basis of some selected text and replace that selected text with a specific method. Use the following procedure to do that.

* Select the code you want to replace with a method and right-click on it, then select "Introduce Method" in the refactor menu.
* Now write the name of the method in the text field of the window appeared.
* NetBeans will do the rest of the work.

1. **Encapsulating the Fields:**

NetBeans has the ability to automatically generate the getter and setter method for any field. The following procedure describes how to encapsulate the fields.

* Right-click anywhere in your code and select "Encapsulate Field" from the "Refactor" menu.
* A window named "Encapsulate Field" will appear that contains the name of all the fields that can be encapsulated. You just need to select the field that you need to encapsulate, the rest is handled by the NetBeans IDE.
* Then click on the "Refactor" button, you can preview the result of refactoring by clicking on the "Preview" button.

1. **Change Method Parameters**

If you want to change a method's signature, you can use the IDE's Refactor Change Method Parameters command to update other code in your project that uses that method. Specifically, you can

* Add parameters.
* Change the order of parameters.
* Change the access modifier for the method.
* Remove unused parameters.

You cannot use the Change Method Parameters command to remove a parameter from a method if the parameter is used in your code.

To change a method's signature:

* Right-click the method in the Source Editor or the Projects window and choose Refactor | Change Method Parameters.
* Click Add if you want to add parameters to the method. Then edit the Name, Type, and optionally the Default Value cells for the parameter. You have to double-click a cell to make it editable.
* To switch the order of parameters, select a parameter in the Parameters table and click Move Up or Move Down.
* Select the preferred access modifier from the Access Modifier combo box.

Click Next.

If you have deselected the Preview All Changes checkbox, the changes are applied immediately.

1. **Extract super class**

You can use the Extract Superclass command to create such a superclass based on methods in one of the classes that you want to turn into a subclass. For each method that you add to the superclass, the Extract Superclass command enables you to choose between the following two options:

* Moving the whole method to the superclass
* Creating an abstract declaration for the method in the superclass and leaving the implementation in the original class
* To extract a new superclass:
* In the Source Editor or the Projects window, select the class that contains the methods that you want to be extracted into the new superclass.
* Choose Refactor | Extract Superclass.

In the Extract Superclass dialog box, select the checkbox for each method and field that you want to be moved to the new superclass. Private methods and private fields are not included.

1. **Safely delete**

Over time, your code might gather elements that have limited or no usefulness. To make the code easier to maintain, it is desirable to remove as much of this code as possible. However, it might be hard to immediately determine whether you can delete such code without causing errors elsewhere.