Group Assignment

Group Member:

Ammar Choudary (181400118)
Nadeem Sultan (181400126)
Salman Naseer (181400104)

Topic:

Refactoring

Course Title:

Software Re-engineering

Course Code:

CS-368

Section:

A

Assignment Date/Day:

April 9, 2022/Saturday

Submitted To:

Sir Fakhar Lodhi

PyCharm Refactoring Support:

Refactoring is a process of improving your source code without creating a new functionality. Refactoring helps you keep your code solid and easy to maintain.PyCharm supports the functionality of redesign as a Removal and Extract Variable strategy to enhance your code base within your editor.

Refactoring approaches in PyCharm:

1. Change Signature:

The Change Signature refactoring combines several different modifications that can be applied to a function signature. You can use this refactoring to:

- change the function name
- add, remove, and reorder parameters
- assign default values to the parameters

2. Convert from Python module to Python package:

A package typically is a directory that contains modules and initialization code. A module is a **.py** source file with Python definitions that can be imported to other modules.

Process of conversion:

- Select a **.py** file.
- Select **Refactor | Convert to Python Package.**
- Inspect the project: the package named as the converted module is created; the **__init__.py** file contains all code from the **.py** file.

With the package created, you can add more new modules to it, or you can use Move Refactorings to derive modules from the initial implementation. You can also modify the **__init__.py** file to put some initialization code for the package or list all the added modules by using the __all__ variable.

3. Extract/Introduce Refactoring:

3.1. Extract constant:

The Extract Constant refactoring makes your source code easier to read and maintain. It also helps you avoid using hard coded constants without any explanations about their values or purpose.

3.2. Extract field:

The Extract Field refactoring lets you declare a new field and initialize it with the selected expression. The original expression is replaced with the usage of the field.

3.3. Extract method:

The Extract Method refactoring lets you take a code fragment that can be grouped, move it into a separated method, and replace the old code with a call to the method.

3.4. Extract superclass

The Extract Superclass lets you either create a superclass based on an existing class or you can rename the original class so it becomes an implementation for the newly created superclass. In this case, PyCharm changes all original class usages to use a superclass where possible.

3.5. Extract/Introduce variable

If you come across an expression that is hard to understand or it is duplicated in several places throughout your code, the Extract Variable refactoring Ctrl+Alt+V can help you deal with those problems placing the result of such expression or its part into a separate variable that is less complex and easier to understand. Plus, it reduces the code duplication.

3.6. Extract parameter

The Extract Parameter refactoring is used to add a new parameter to a function declaration and to update the function calls accordingly.

4. Move Refactorings

The Move refactoring lets you move classes, functions, modules, files, and directories within a project.

The following Move refactorings are available:

- The Move File refactoring moves a file to another directory.
- The Move Directory refactoring moves a directory to another directory.
- The Move Module Members refactoring moves top-level symbols of a Python module.

5. Rename refactorings(Shift+F6)

Use the Rename refactoring to change names of symbols, files, and all the references to them throughout code.

Renaming local variables or private methods can be done easily inline since only the limited scope is affected. Renaming classes or public methods could potentially impact a lot of files. Preview potential changes before you refactor.