REPORT

Group Members:

Zunaira Sadia 181400005

Zainab Aslam 181400006

**What is Refactoring?**

Code refactoring is defined as the process of restructuring computer code without changing or adding to its external behavior and functionality.

There are many ways to go about refactoring, but it most often comprises applying a series of standardized, basic actions, sometimes known as micro-refactoring’s. The changes in existing source code preserve the software’s behavior and functionality because the changes are so tiny that they are unlikely to create or introduce any new errors.

**Refactoring in Eclipse:**

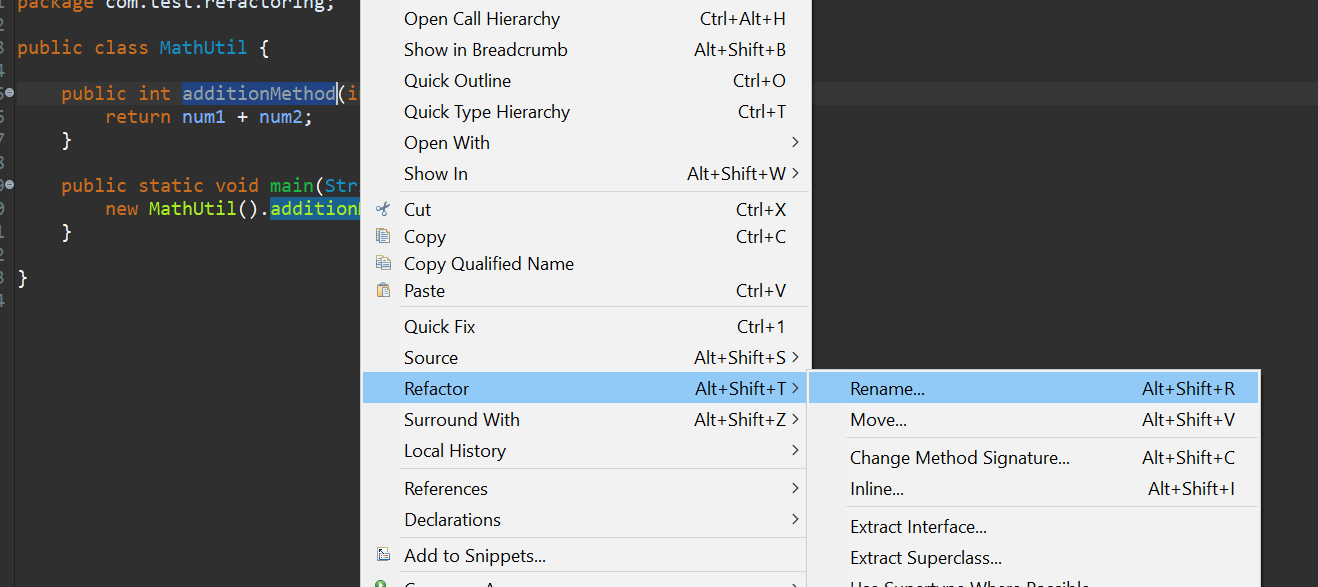
Eclipse support following refactoring:

**1. Renaming**

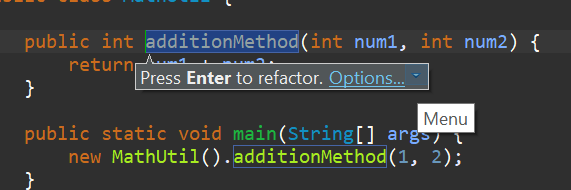
**1.1 Renaming Variables and Methods**

We can **rename variables and methods by following these simple steps**:

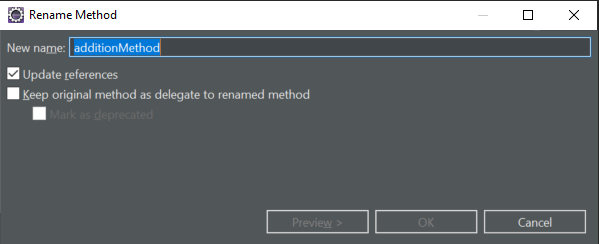
* + Select the element
  + Right-click the element
  + **Click the *Refactor > Rename* option**
* Type the new name
* Press *Enter*



* We can also perform the second and third steps by **using the shortcut key, *Alt+Shift+R***.
* When the above action is performed, Eclipse will find every usage of that element in that file and replace them all in place.
* We can also use an advanced feature to **update the reference in other classes** by hovering over the item when the refactor is on and clicking on *Options*:

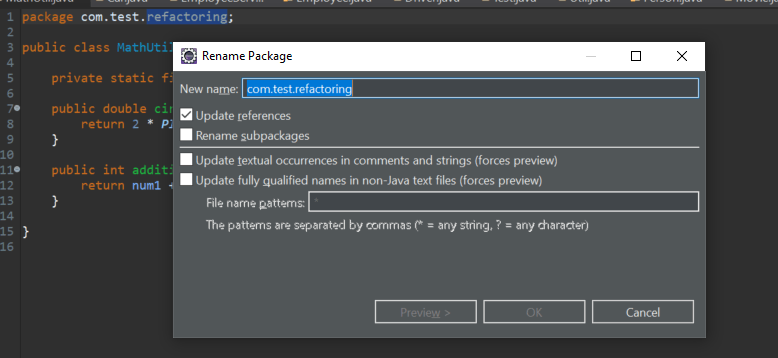


This will open up a pop-up where we can both rename the variable or method and have the option to update the reference in other classes:

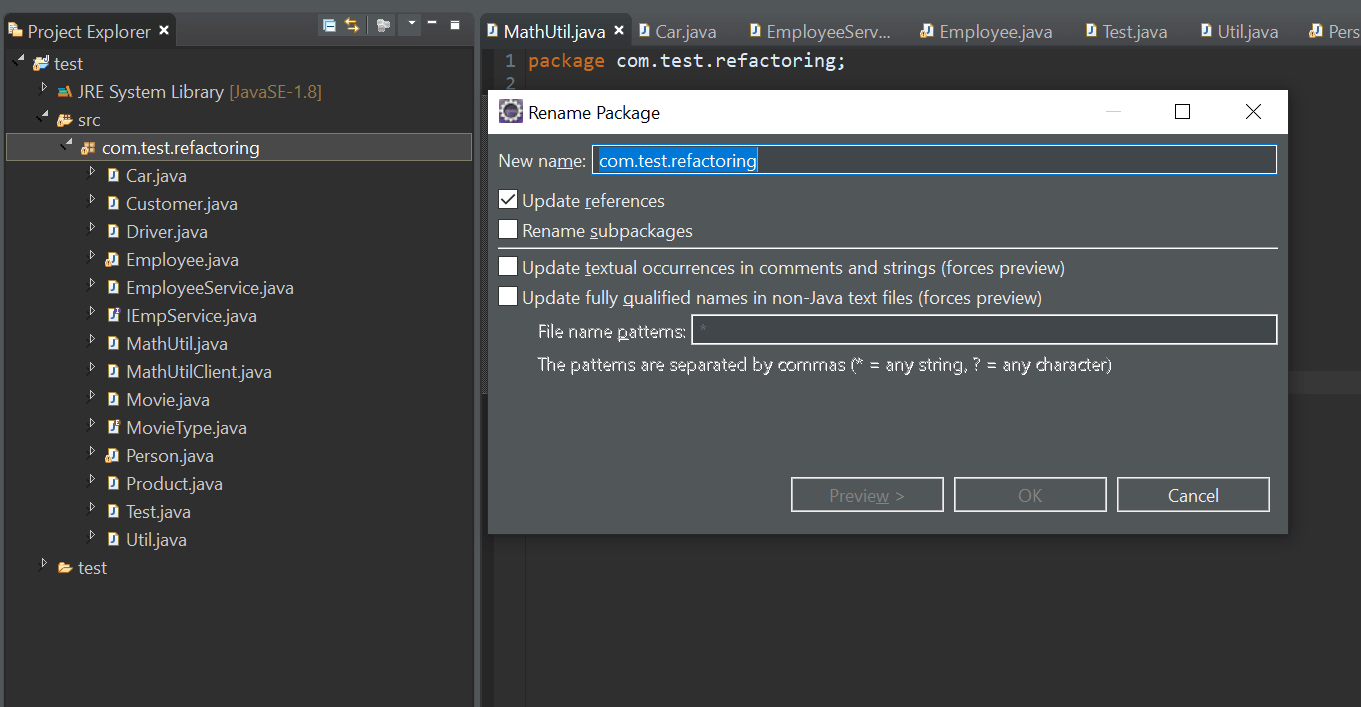


### 1.2. Renaming Packages

We can rename a package by selecting the package name and performing the same actions as in the previous example. A pop-up will appear right away where we can rename the package, with options like updating references and renaming subpackages.

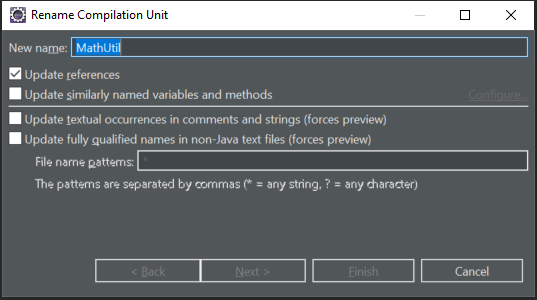


We can also rename the package **from the Project Explorer view by pressing F2**:



### 1.3. Renaming Classes and Interfaces

We can rename a class or interface by using the same actions or just by pressing F2 from Project Explorer. This will open up a pop-up with options to update references, along with a few advanced options:



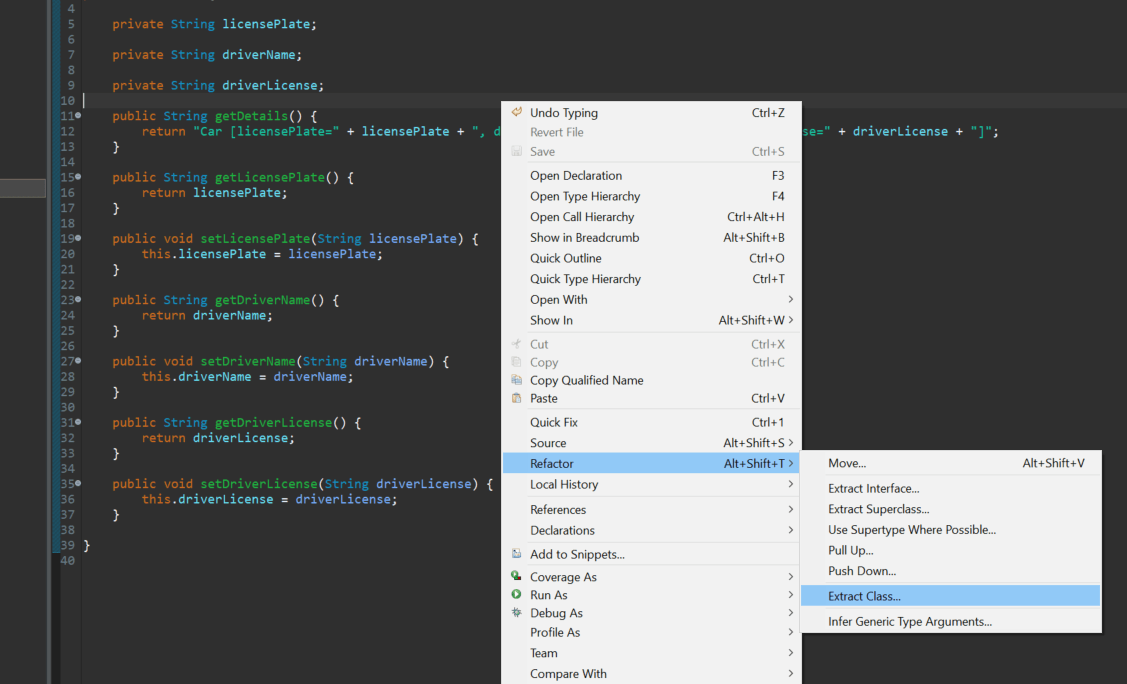
## 2. Extracting

Now, let's talk about extraction. Extracting code means **taking a piece of code and moving it.**

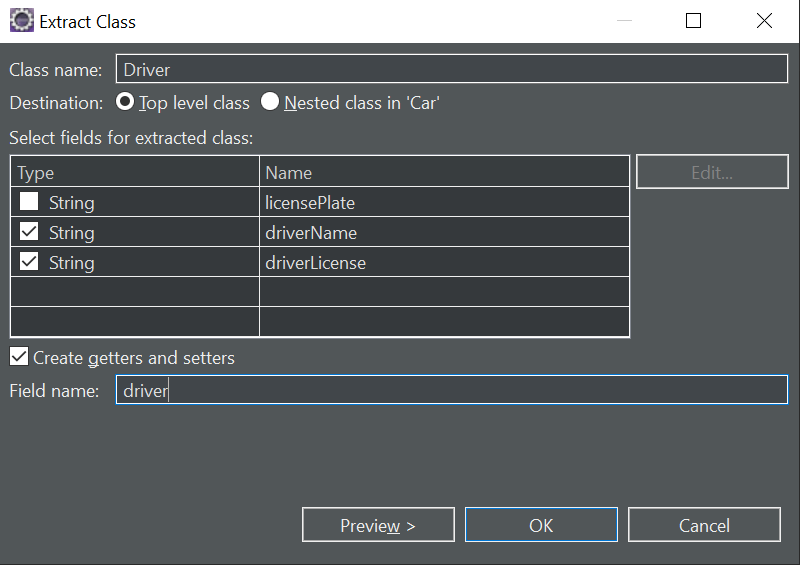
For example, we can extract code into a different class, superclass or interface. We could even extract code to a variable or method in the same class.

Eclipse provides a variety of ways to achieve extractions, which we'll demonstrate in the following sections.

### 2.1. Extract Class

We can do this by **right-clicking anywhere within the class and choosing the Refactor > Extract Class option**:

This will open up a pop-up where we can name the class and select which fields we want to move, along with few other options:

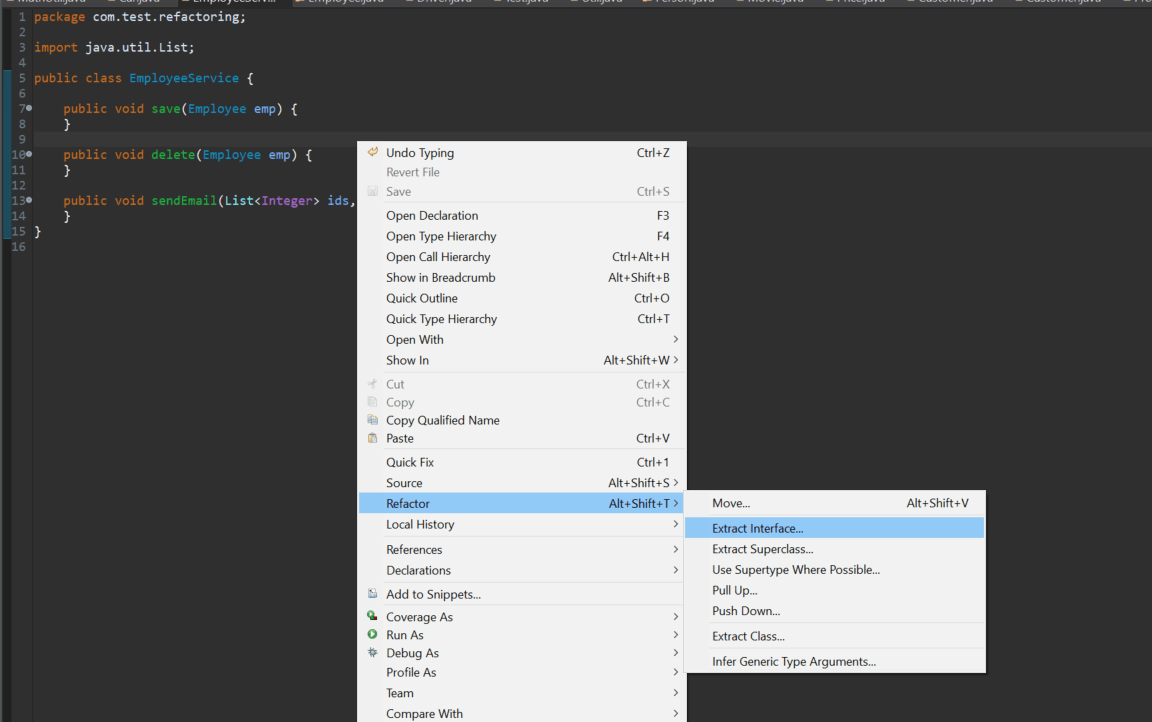


When we click OK, Eclipse will create a new class named Driver, and the previous code will be refactored.

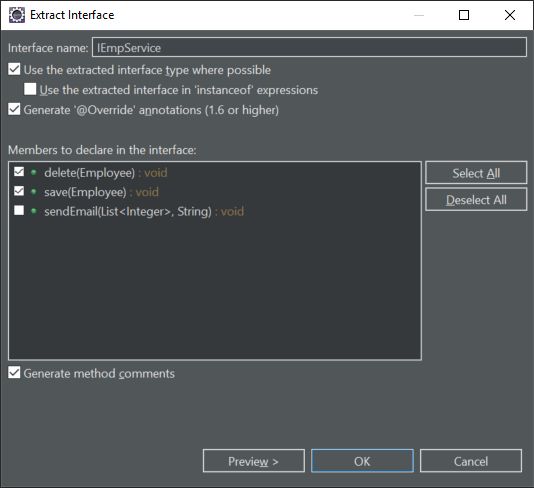
### 2.2. Extract Interface

We can also extract an interface in a similar fashion.

We can extract an interface by **right-clicking anywhere within the class and choosing the Refactor > Extract Interface option,**or we can use the Alt+Shift+T shortcut key command to bring up the menu directly:



This will open up a pop-up where we can enter the interface name and decide which members to declare in the interface:

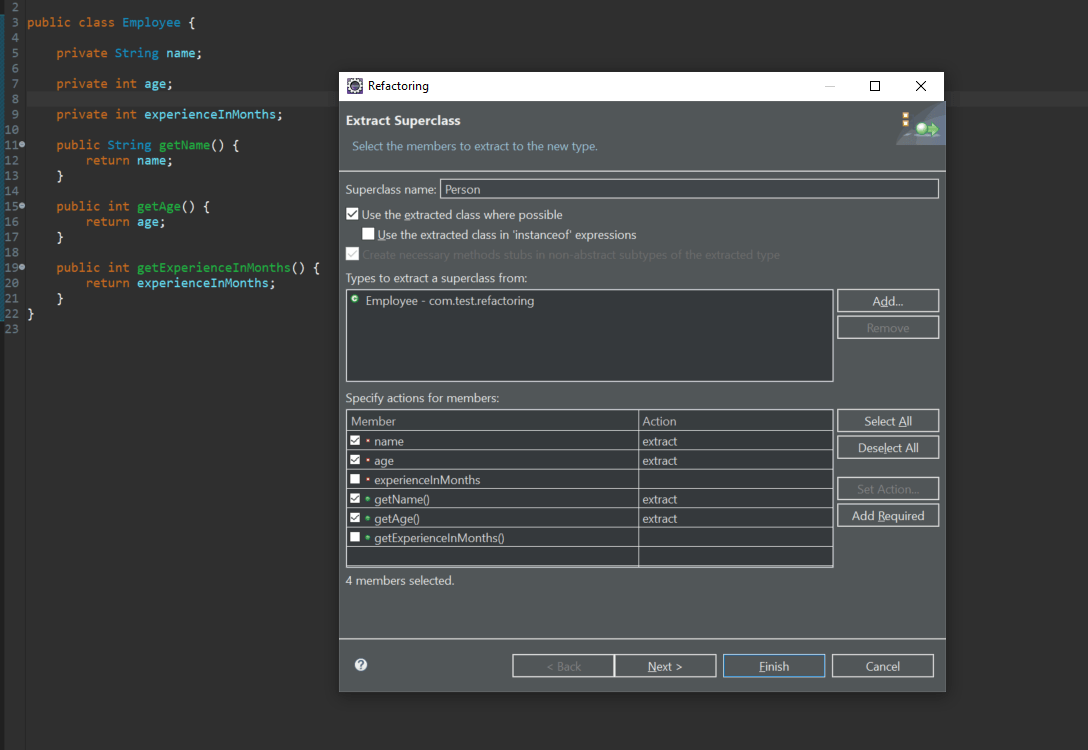


As a result of this refactoring, we'll have an interface IEmpService,

### 2.3. Extract Superclass

Suppose we have an Employee class containing several properties that aren't necessarily about the person's employment:

We may want to extract the non-employment-related properties to a Person superclass. To extract items to a superclass, we can  **right-click anywhere in the class and choose the Refactor > Extract Superclass option, or use Alt+Shift+T** to bring up the menu directly:



This will create a new Person class with our selected variables and method, and the Employee class will be refactored.

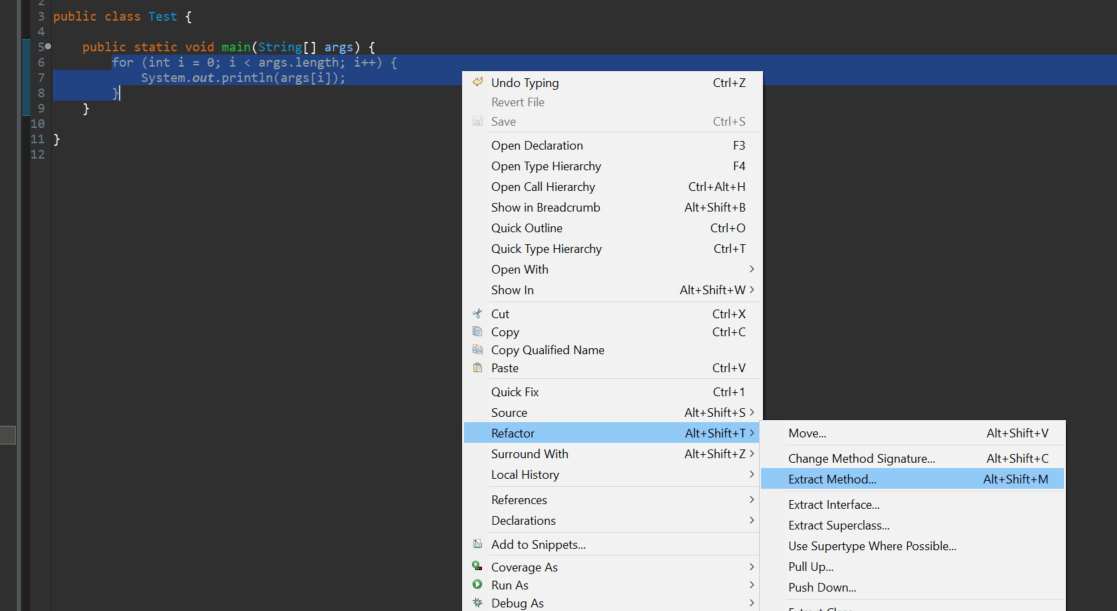
### 2.4. Extract Method

Sometimes, we might want to extract a certain piece of code inside our method to a different method to keep our code clean and easy to maintain.

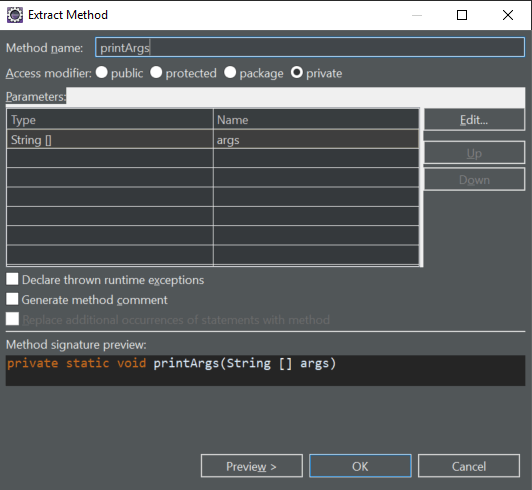
Let's say, for example, that we have a for loop embedded in our method:

To invoke the *Extract Method* wizard, we need to perform the following steps:

* Select the lines of code we want to extract
* Right-click the selected area
* Click the ***Refactor > Extract Method* option**



* **The last two steps can also be achieved by keyboard shortcut *Alt+Shift+M***. Let's see the *Extract Method* dialog:



### 2.5. Extract Local Variables

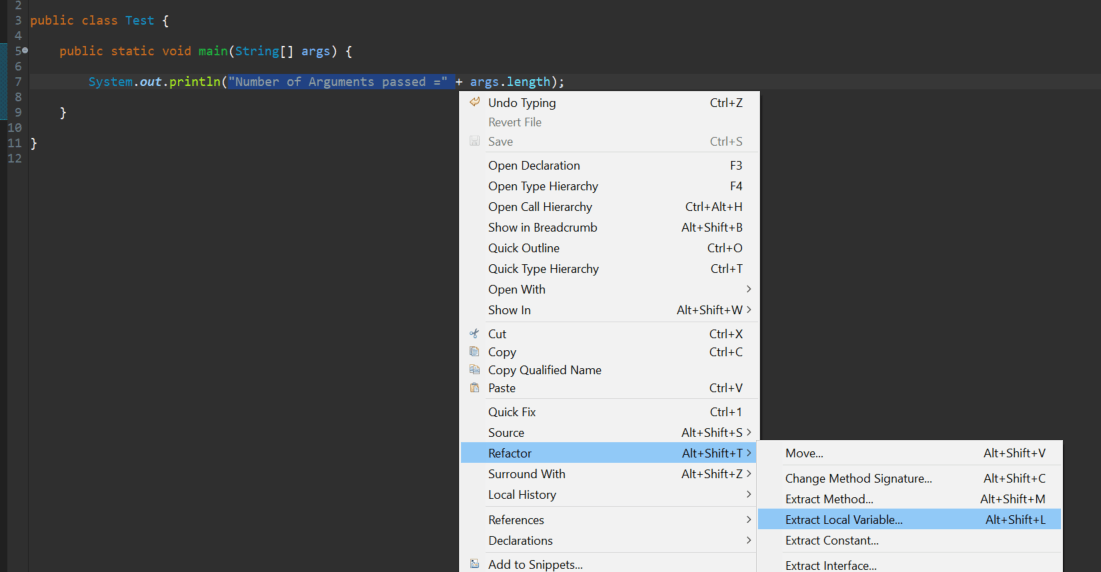
We can extract certain items as local variables to make our code more readable.

This is handy when we have a *String*literal:

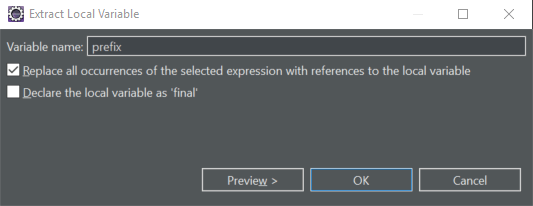
and we want to extract it to a local variable.

To do this, we need to:

* Select the item
* Right-click and choose ***Refactor > Extract Local Variable***



* **The last step can also be achieved by the keyboard shortcut Alt+Shift+L**. Now, we can extract our local variable:



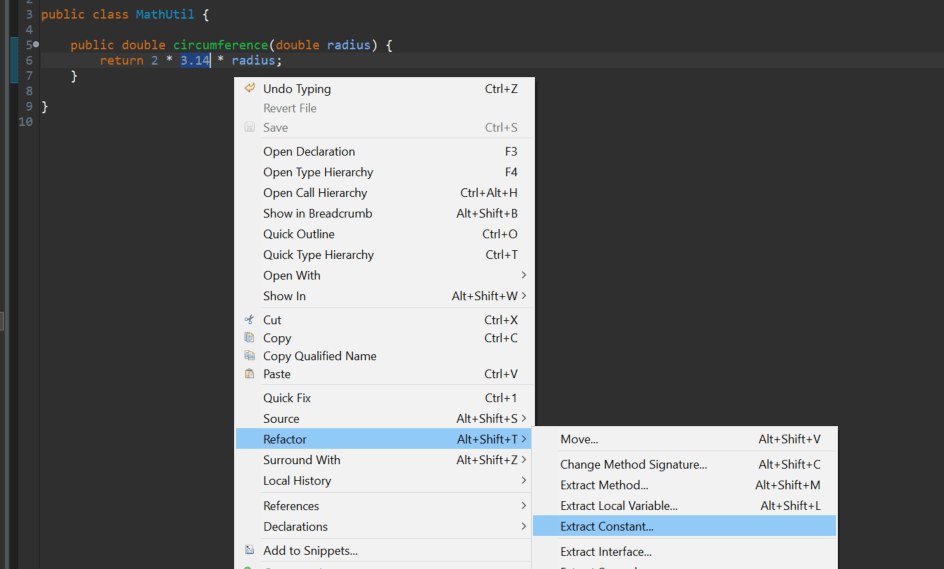
### 2.6. Extract Constant

Or, we can extract expressions and literal values to static final class attributes.

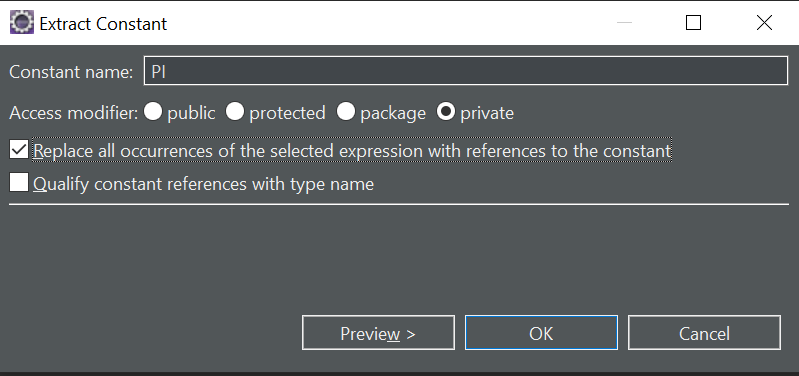
We could extract the 3.14 value into a local variable, as we just saw:

But, it might be better to extract it as a constant, for which we need to:

* Select the item
* Right-click and choose ***Refactor > Extract Constant***



This will open a dialog where we can give the constant a name and set its visibility, along with a couple of other options:



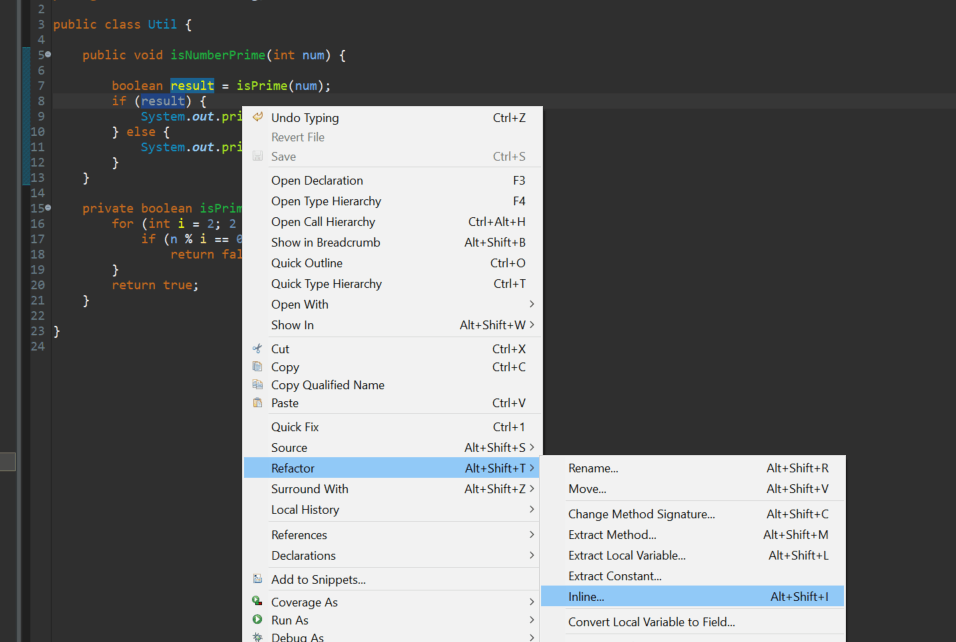
## 3. In lining

We can also go the other way and inline code.

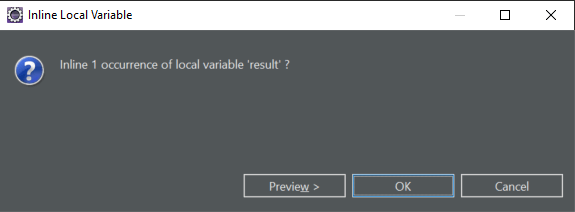
Consider a Util class that has a local variable that's only used once:

We want to remove the *result* local variable and inline the *isPrime* method call. To do this, we follow these steps:

* Select the item we want to inline
* Right-click and **choose the *Refactor > Inline* option**



* **The last step can also be achieved by keyboard shortcut Alt+Shift+I:**

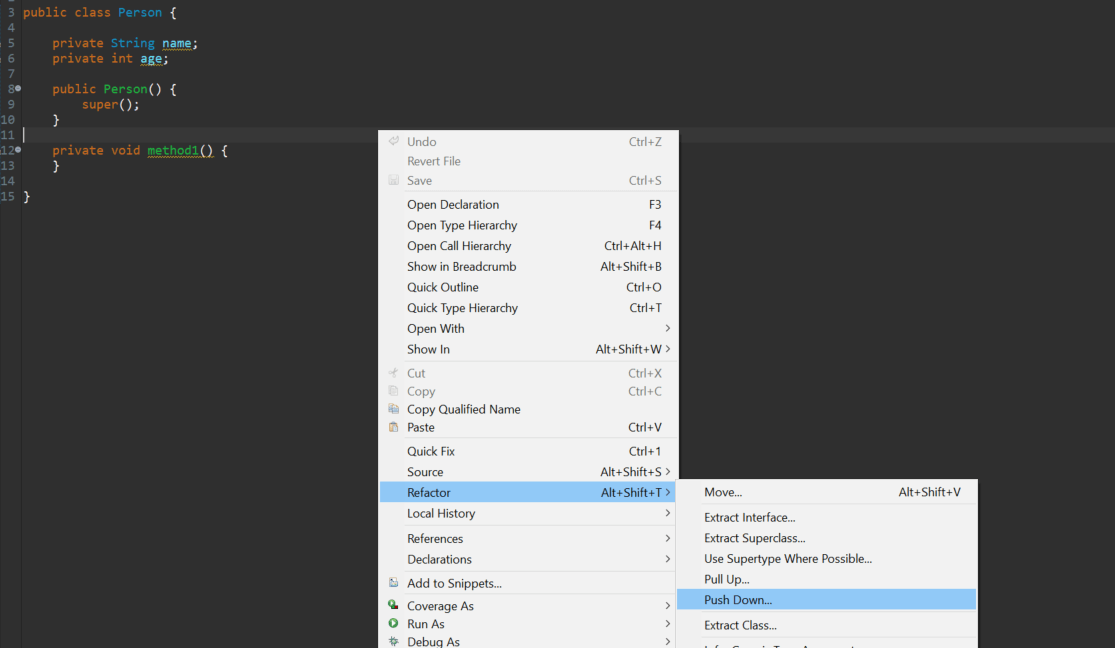


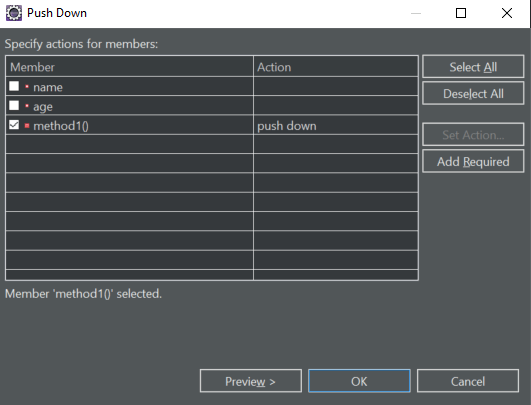
## 4. Push Down and Pull Up

If we have a parent-child relationship (like our previous Employee and Person example) between our classes, and we want to move certain methods or variables among them, we can use the push/pull options provided by Eclipse.

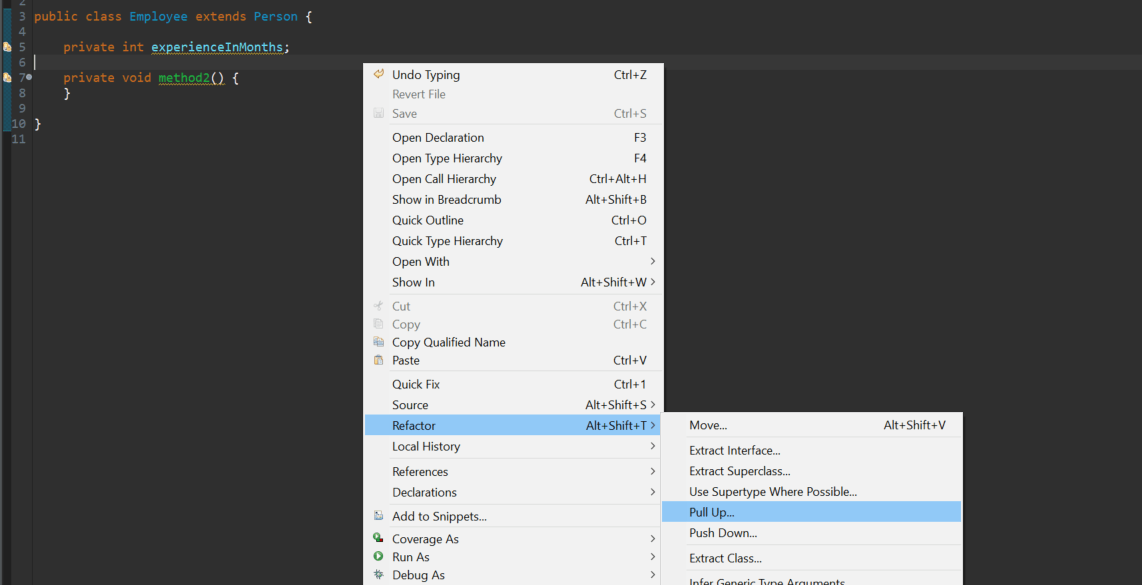
As the name suggests, the Push Down option moves methods and fields from a parent class to all child classes, while Pull Up moves methods and fields from a particular child class to parent, thus making that method available to all the child classes.

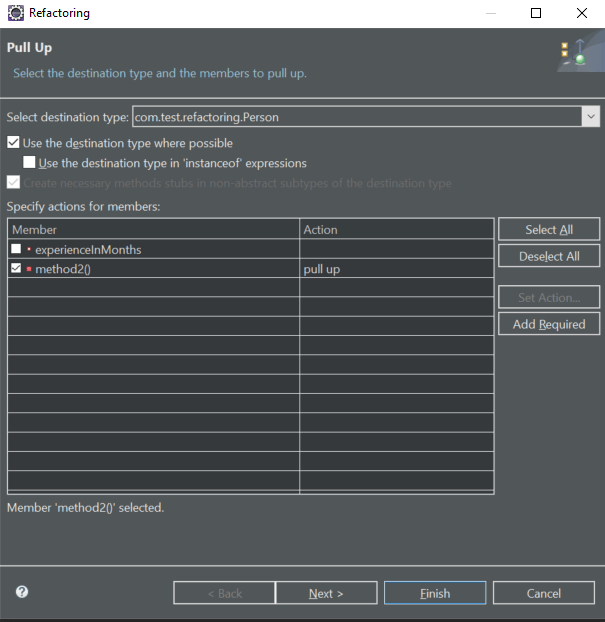
For moving methods down to child classes, we need to **right-click anywhere in the class and choose the Refactor > Push Down option**:



* This will open up a wizard where we can select items to push down:

* Similarly, for moving methods from a child class to parent class, we need to **right-click anywhere in the class and choose Refactor > Pull Up**:



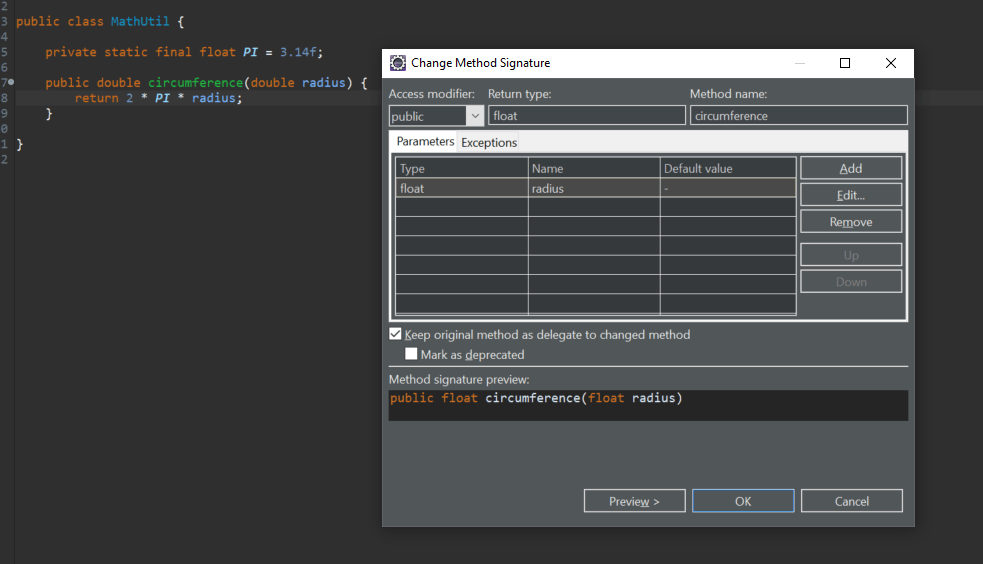
* This will open up a similar wizard where we can select items to pull up:

**5. Changing a Method Signature**

To change the method signature of an existing method, we can follow a few simple steps:

* Select the method or place the cursor somewhere inside
* **Right-click and choose *Refactor > Change Method Signature***

**The last step can also be achieved by keyboard shortcut*Alt+Shift+C.***

* This will open a popup where you can change the method signature accordingly:
* 

## 6. Moving

Sometimes, we simply want to **move methods to another existing class to make our code more object-oriented**.

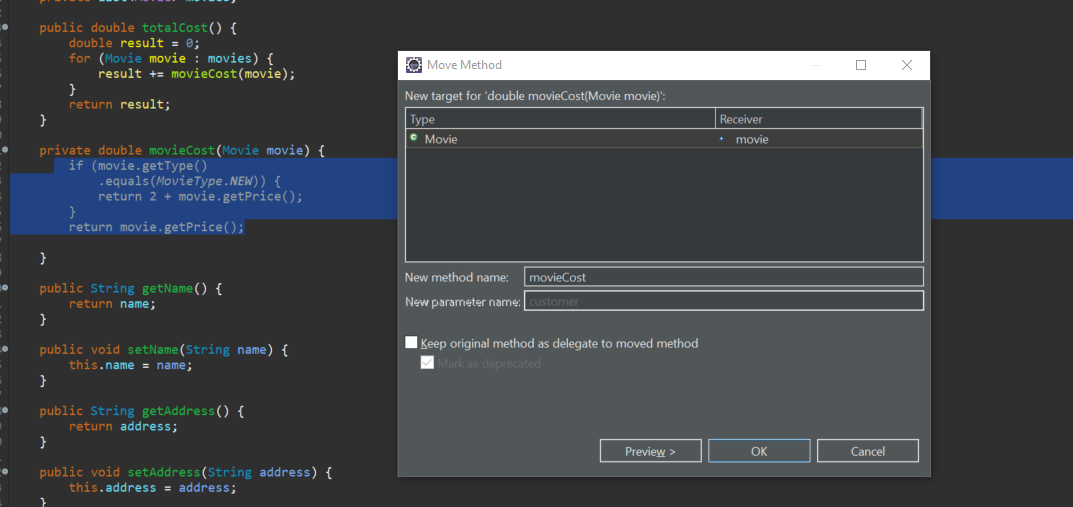
Consider the scenario where we have a Movie class, And MovieType is a simple enum:

* Suppose also that we have a requirement that if a Customer rents a movie that is NEW, it will be charged two dollars more, and that our Customer class has the following logic to calculate the totalCost()

Clearly, the calculation of the movie cost based on the *MovieType* would be more appropriately placed in the *Movie* class and not the *Customer* class. We can easily move this calculation logic in Eclipse:

* Select the lines you want to move
* **Right-click and choose the *Refactor > Move* option**

**The last step can also be achieved by keyboard shortcut *Alt+Shift+V*:**



Eclipse is smart enough to realize that this logic should be in our *Movie* class. We can change the method name if we want, along with other advanced options.

The final Customer class code will be refactored.