**Replace Nested Conditional with Guard Clauses**

**Motivation**

Code that has been nested a large amount of times can be difficult to read. It is hard to determine what code will execute at what conditional. This pattern helps to consolidate the conditionals into a series of individual steps.

**Methods**

Try to rid the code of side effects – **Separate Query from Modifier** may be helpful for the purpose. This solution will be necessary for the reshuffling described below.

1. Isolate all guard clauses that lead to calling an exception or immediate return of a value from the method. Place these conditions at the beginning of the method.
2. After rearrangement is complete and all tests are successfully completed, see whether you can use **Consolidate Conditional Expression** for guard clauses that lead to the same exceptions or returned values.

**Sample code to refactor**

The government wants to figure out how much money to pay in government assistance. The status of a worker affects the amount of money they receive, such as if they are retired or divorced. They work out the correct amount to pay using the following method.

**public** **double** **getPayAmount**() {

**double** result;

**if** (isDead){

result = deadAmount();

}

**else** {

**if** (isSeparated){

result = separatedAmount();

}

**else** {

**if** (isRetired){

result = retiredAmount();

}

**else**{

result = normalPayAmount();

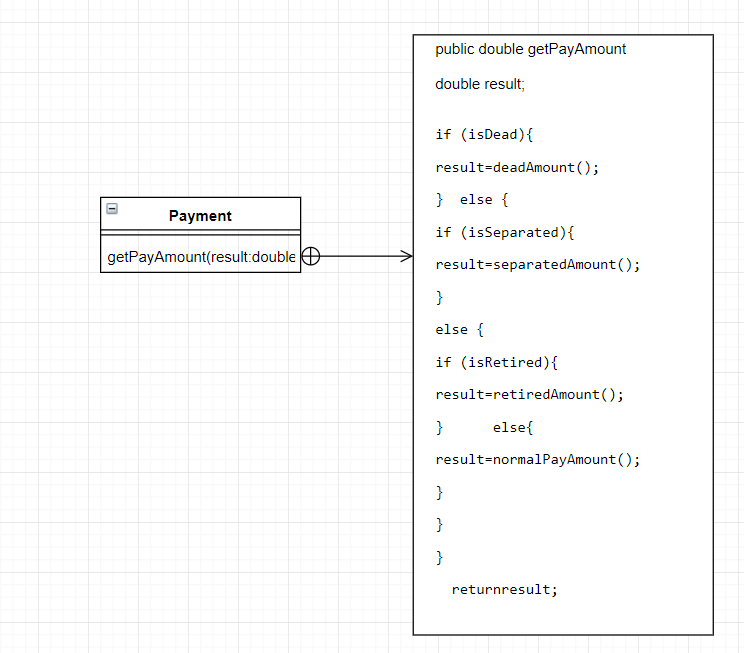
}

}

}

**return** result;

}



**Risks**

* The refactoring is very tedious depending on how much nesting has occurred
* The programmer may become lost and put the wrong functionality into the guard clause