Tyler Sulsenti

HW06

Bad Smell 1

Method: fillIndiList

Bad Smell: Unnecessary Complexity

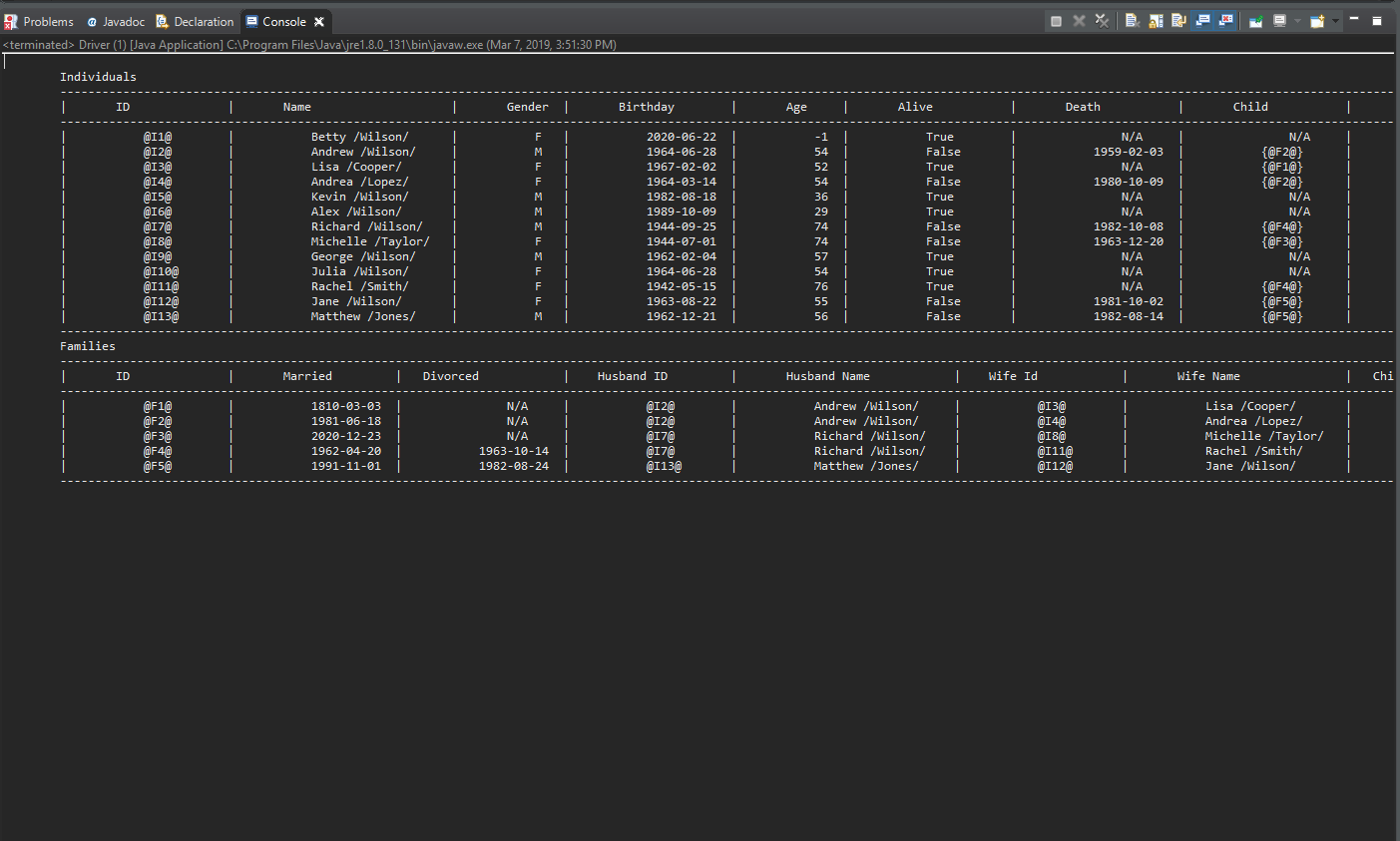
Code Lines: 182-201

Before Refactor:



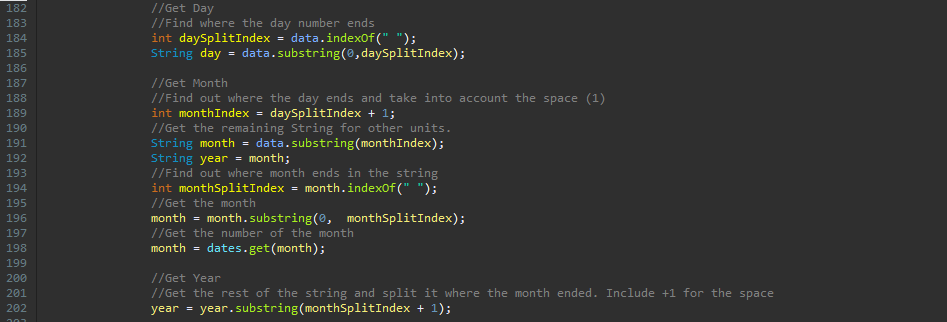
Test Case Pass:

Test Case Lines: 299-495 in Project.java in the folder marked Original. Methods CheckFDates and CheckIDates and helper function getComponents



After:

Code Lines: 182-202



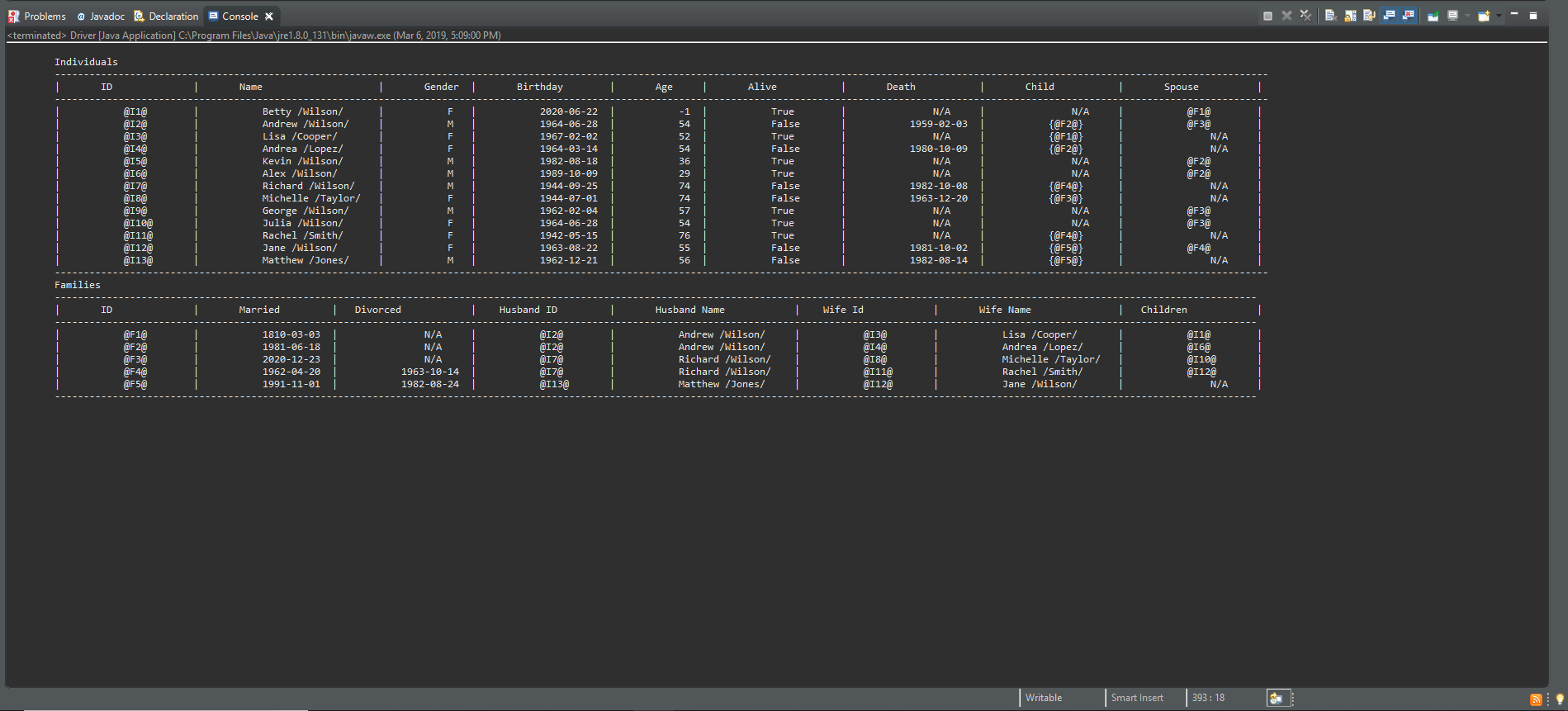
Refactor: While operational, the original code was extremely complex, using many nested functions of the String class to perform String manipulation to extract the required data. The original method performed manipulation on the same original string. The String was a date that would always be in the format

DD MMM YYYY. So 06 JUN 2020 for example. Using the always known locations of the spaces between the units, this was complicated. Not to mention that because it relied on some original master string, each individual unit string (month, day, year) used data that was reused in some other one. This resulted in duplicate code. The other issue with this block of code is the units are out of order. The program finds the units in month-> day-> year from the original master string but the master string is formatted as day -> month -> year.

When I refactored this code I took into account that each unit used code that was used in some previous unit to find it in the master string. Instead of relying on spaces and a complex set of substrings, I chose to have the strings rely on the positions of the previous unit in the String, since this can be always assumed to be the same order. By doing this and including descriptive comments and variable names, I have significantly reduced the complexity and increased the understandability of this code.

Test Case Pass:

Test Case Lines: Automated Test Code: Lines 334-413 in Project.java in the Refactored folder



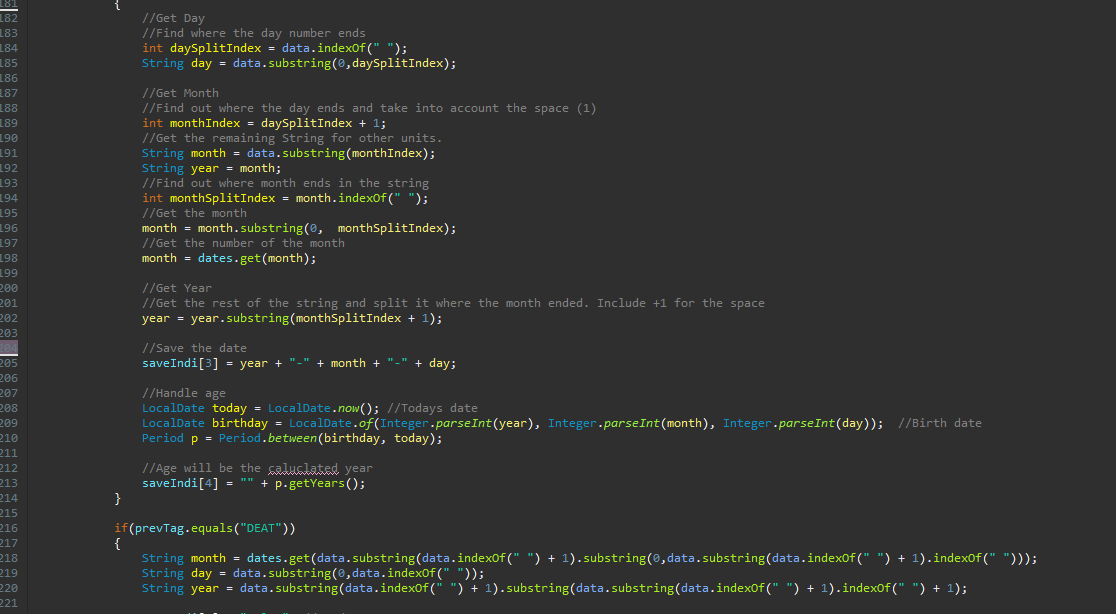
Bad Smell 2

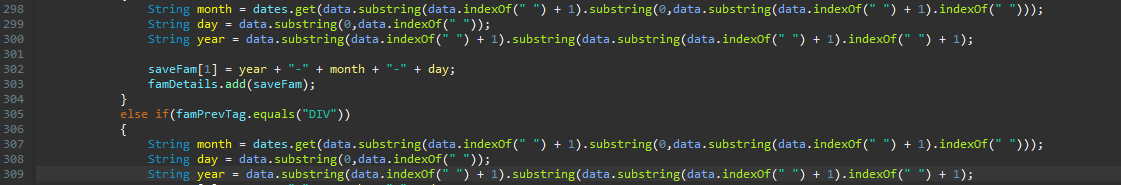
Methods: fillIndiList, fillFamList

Bad Smell: Duplicate Code

Code Lines: 182-222 and 298-309

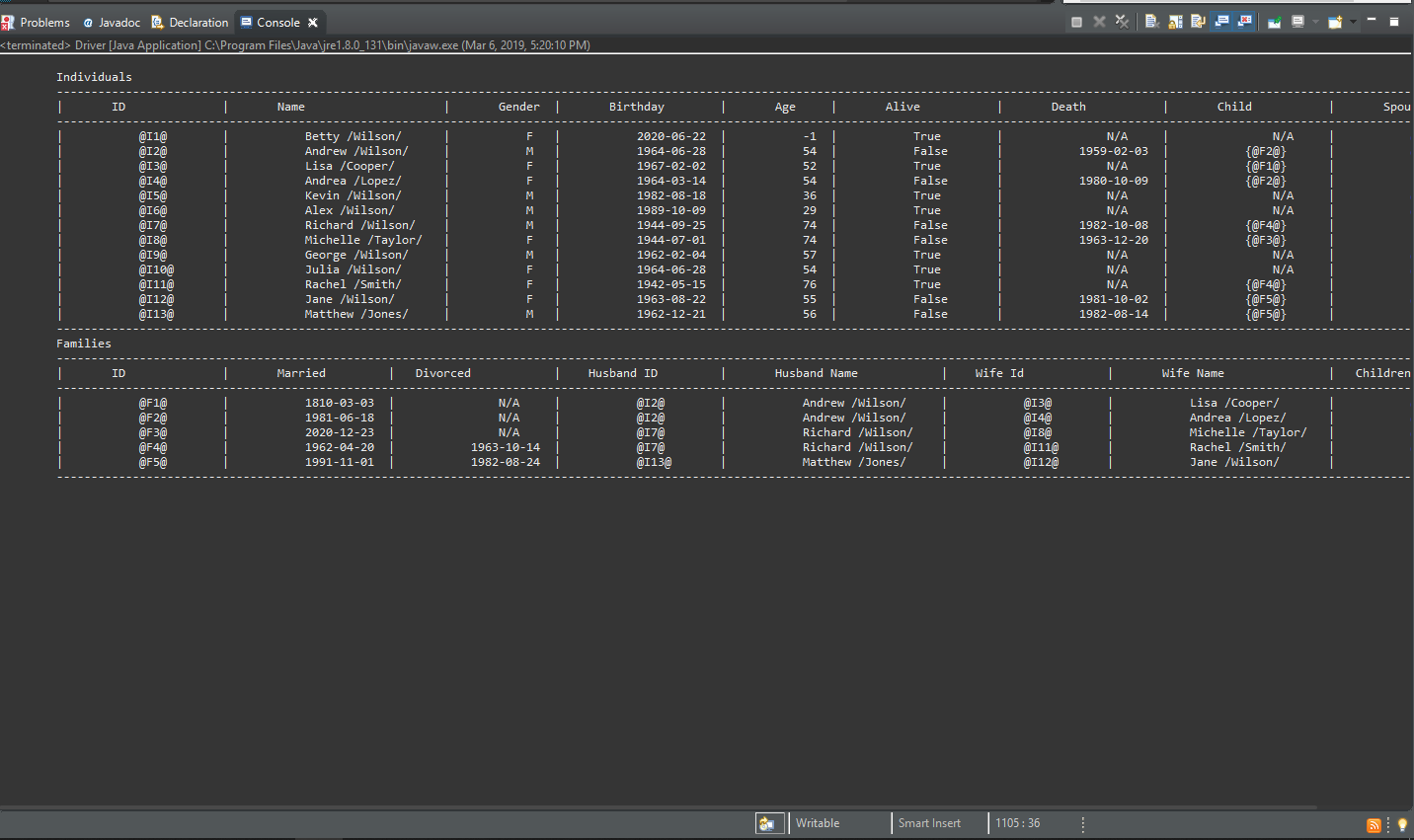
Before:





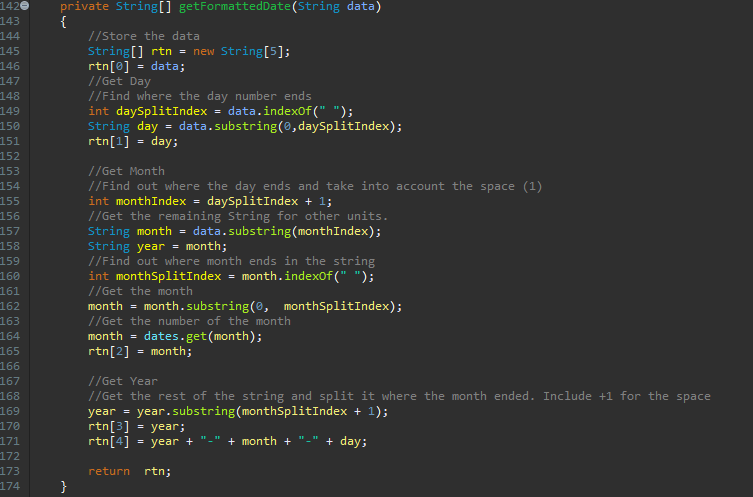
Test Case Pass:

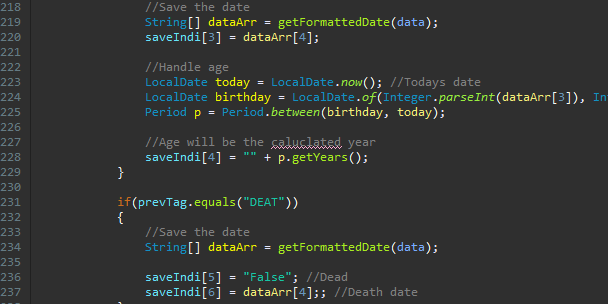
Test Case Lines: 299-367 in Project.java in the folder marked Original

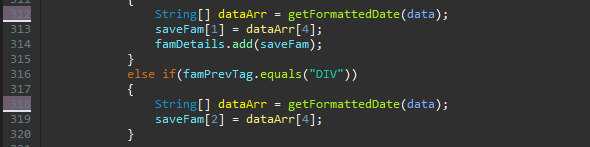


After Refactor:

Code Lines 142-173, 219-220, 236-237, 312-313, 318-319



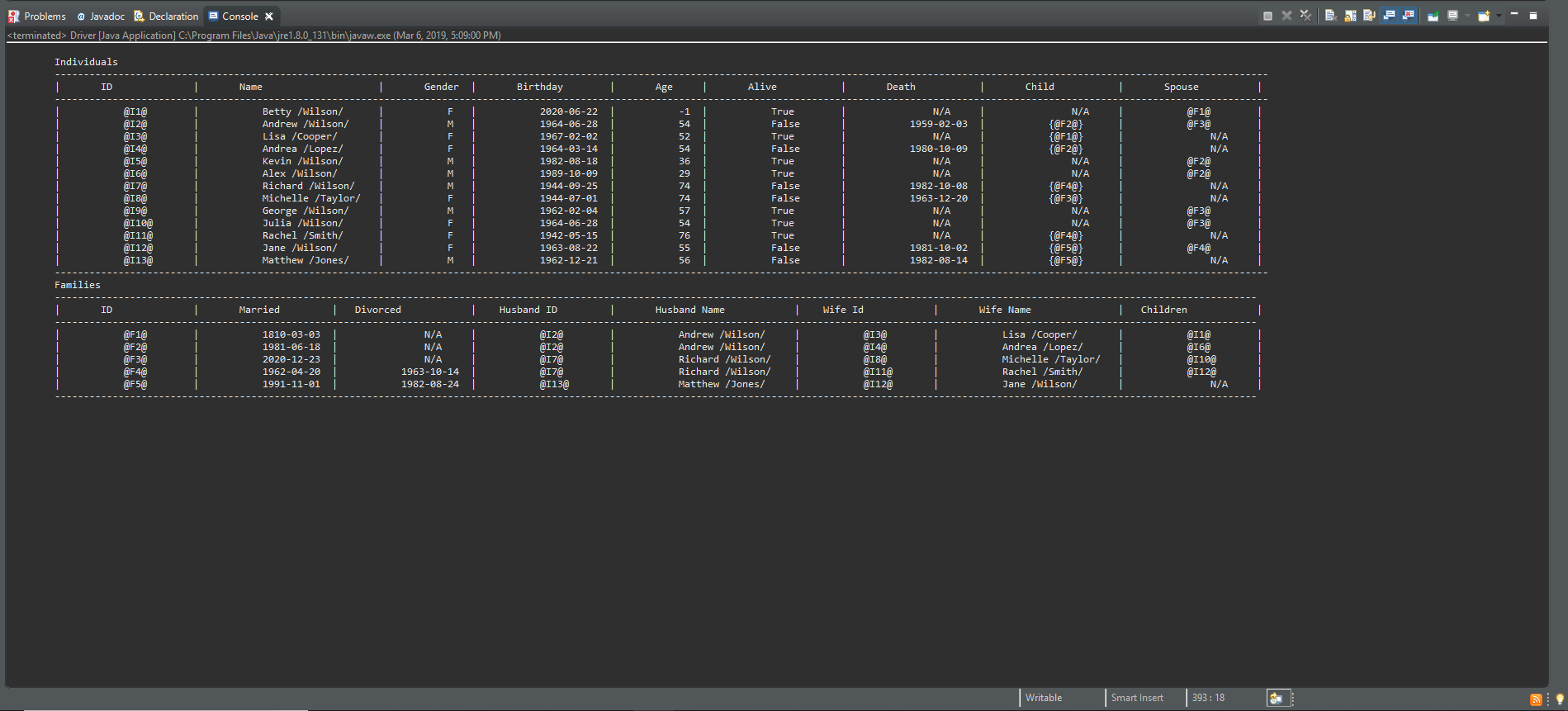




Refactor: These are the same lines affected by the previous refactor. They all were meant to reformat given dates to the required terms. After the first set was refactored, obviously the other 3 sets needed to be as well. But, since these were duplicates, that is another Bad Smell in code. To fix this issue, I created a method called getFormattedDate and sent in the working data string. I added the return type of String[] to store the various data collected in the String manipulation. Now, instead of refactoring every duplicate, we can eliminate all duplicate code by making a call to this method that contains the original refactored code. This truly makes the code neat, easy to read and understand and not unnecessarily complex.

Test Case Pass:

Test Case Lines: Automated Test Code: Lines 334-413 in Project.java in the Refactored folder



Both smells and smells test cases are in Original/Project.java and both refactors and refactor test cases are in Refactor/Project.java.

The test method checkDates checks both refactors and uses the helper method getComponents