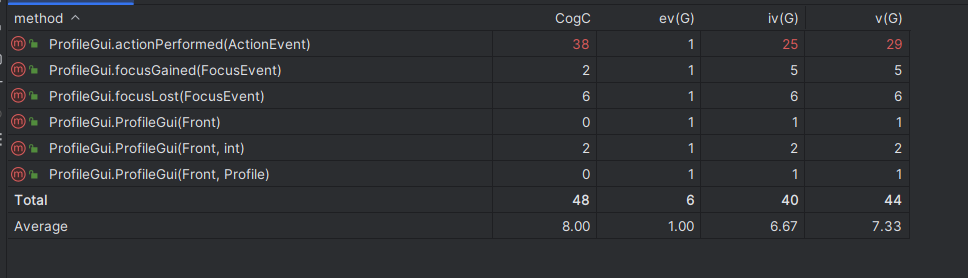
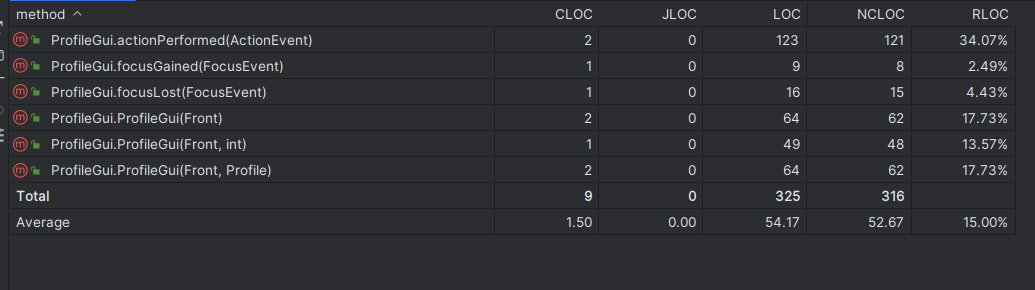
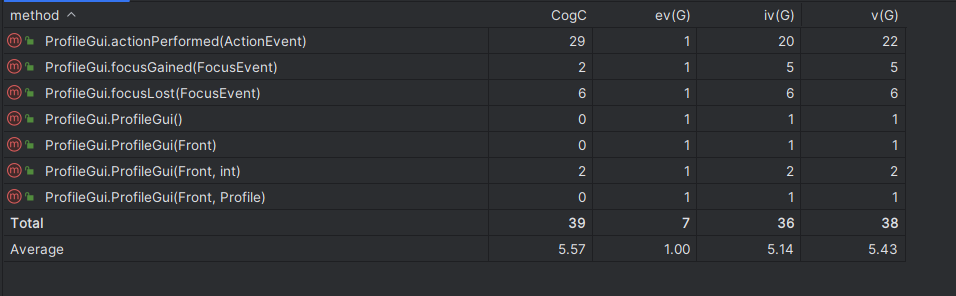
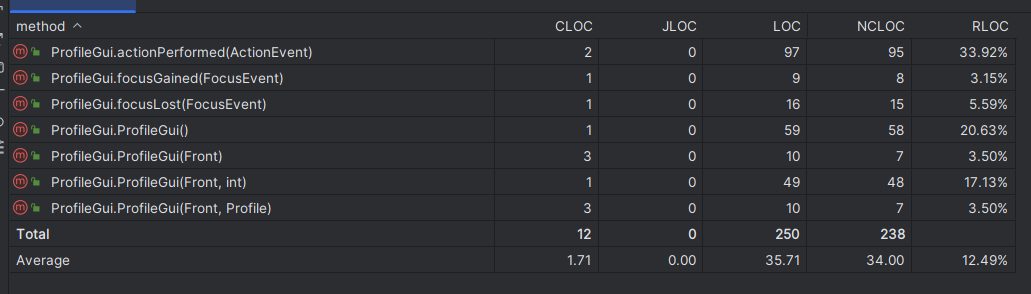
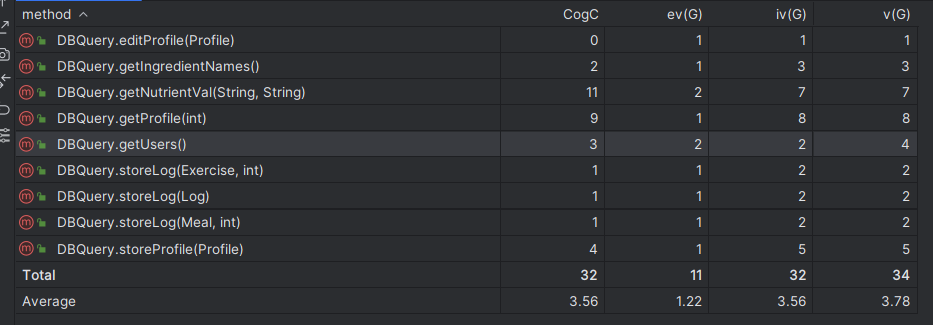
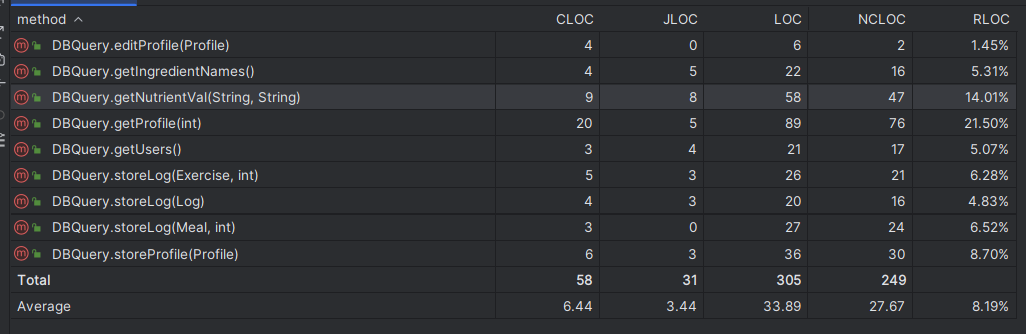
**Use case 1 + Database Queries:**

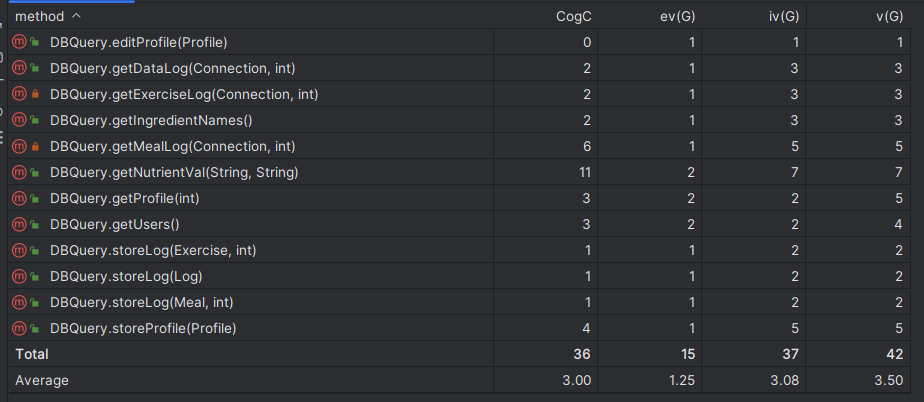
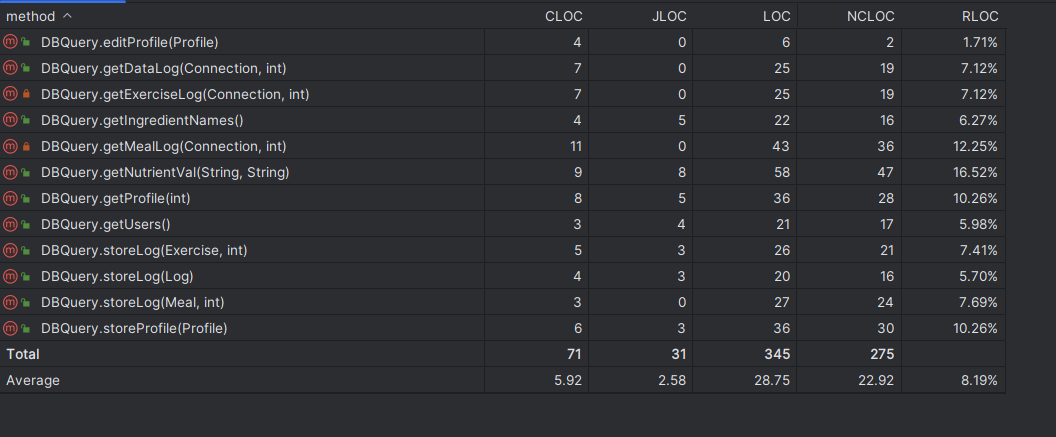
Initial metrics:  
  


After refactoring:  
  


Code Smell: Duplicated Code

* The constructors are used to display the different GUIs associated with the user profile. Of the three, ProfileGui(Front) and ProfileGui(Front, Profile) contain very similar codes.
* The method actionPerformed(ActionEvent) is essential for the functionality of the ProfileGui.java. As such, the size and cyclomatic complexity of the code is expected to be much higher than other methods in the class. However, examining the code shows that there are duplicated code within the method that needed adjustments.
* Solution:
  + The constructors are redefined to be chained from a common constructor instead.
  + The duplicated portions are combined as much as possible to remove the code smell.

Initial metrics:  
  


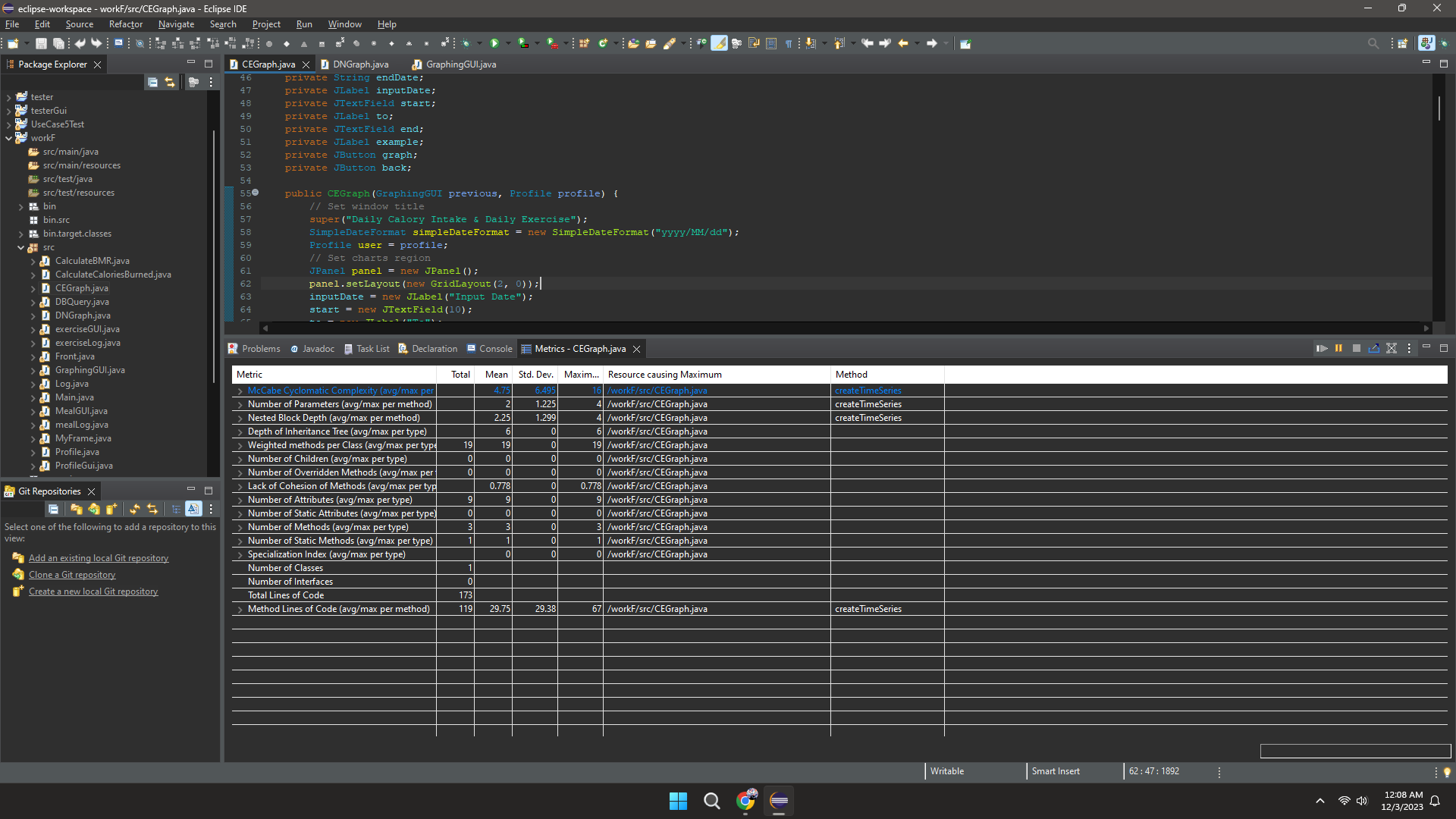
After refactoring:  
  


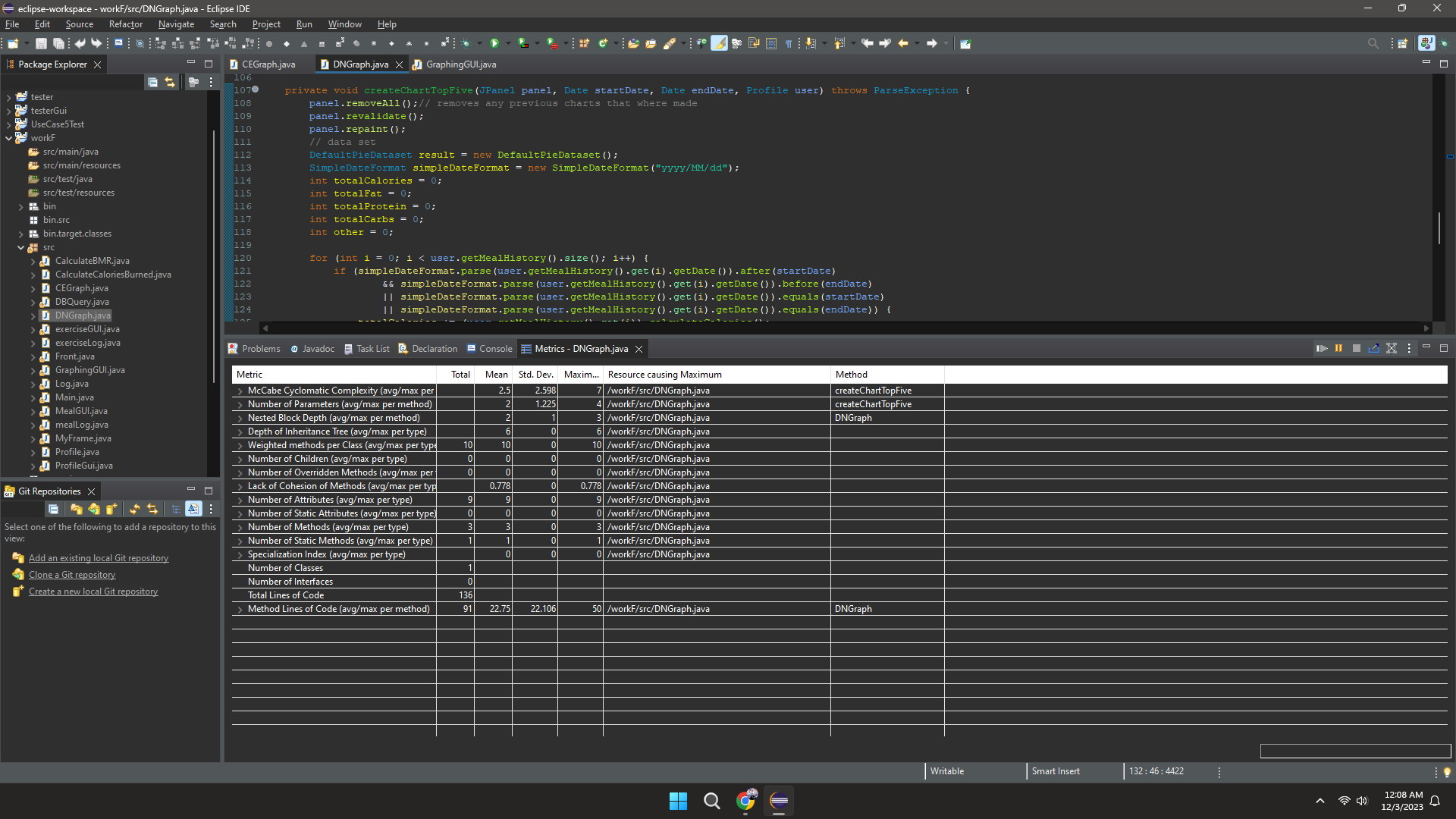
Code Smell: Long Method:

* The method getProfile(int) has the highest value of cyclomatic complexity as well as accounting for 21.50% of the code within the class. Further examinations show that the method also violates SRP, being responsible for creating a Profile object from the database but also creating the necessary histories of the user’s profile at the same time, making it confusing for other developers to understand.
* Solution: Extract the creation of each history field into individual methods, preserving SRP as well as adding more reusability to the class.
  + Cyclomatic complexity on average is lower at the cost of increasing the total of the class.
  + The lines of code per method in the class is reduced.

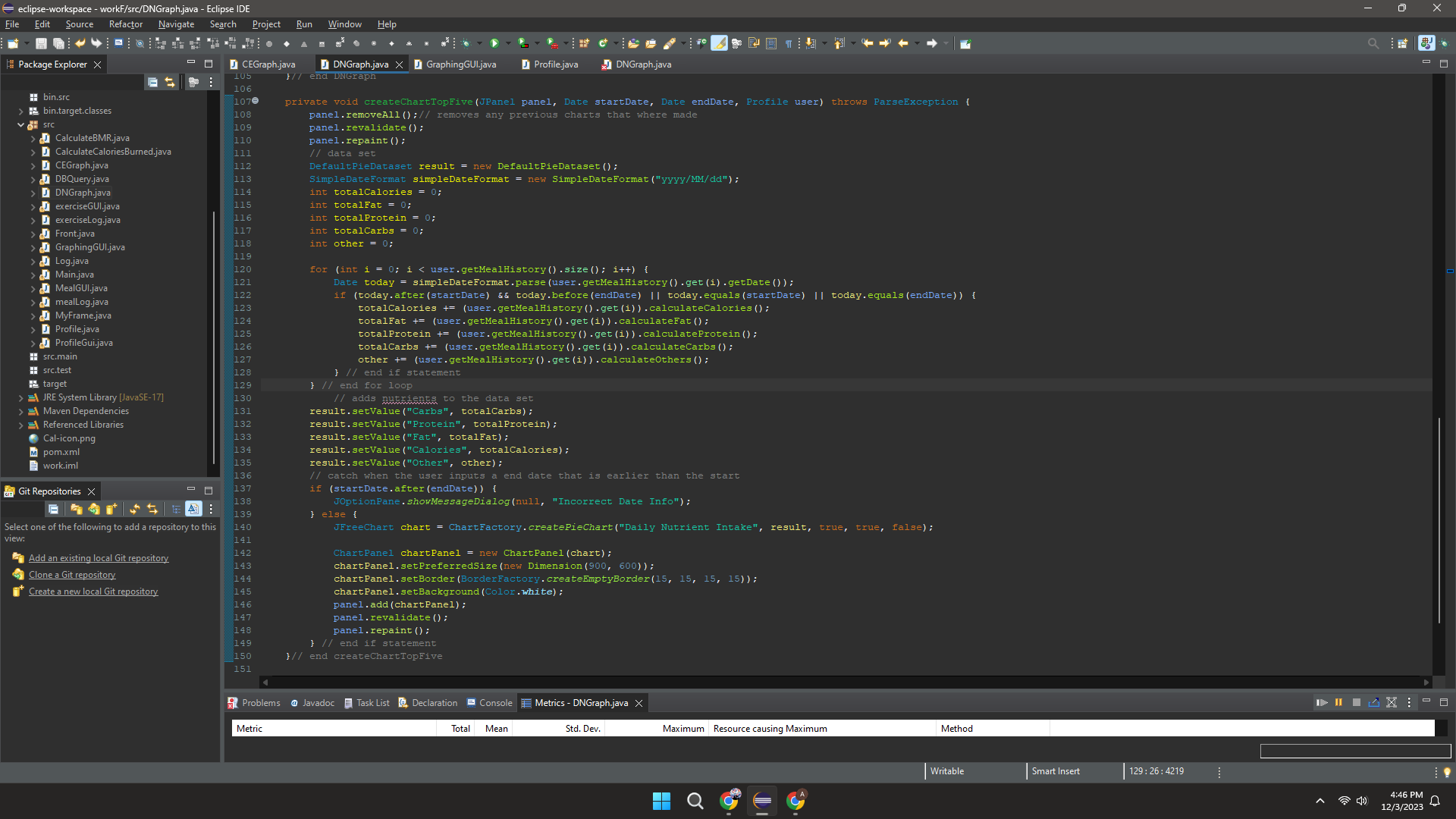
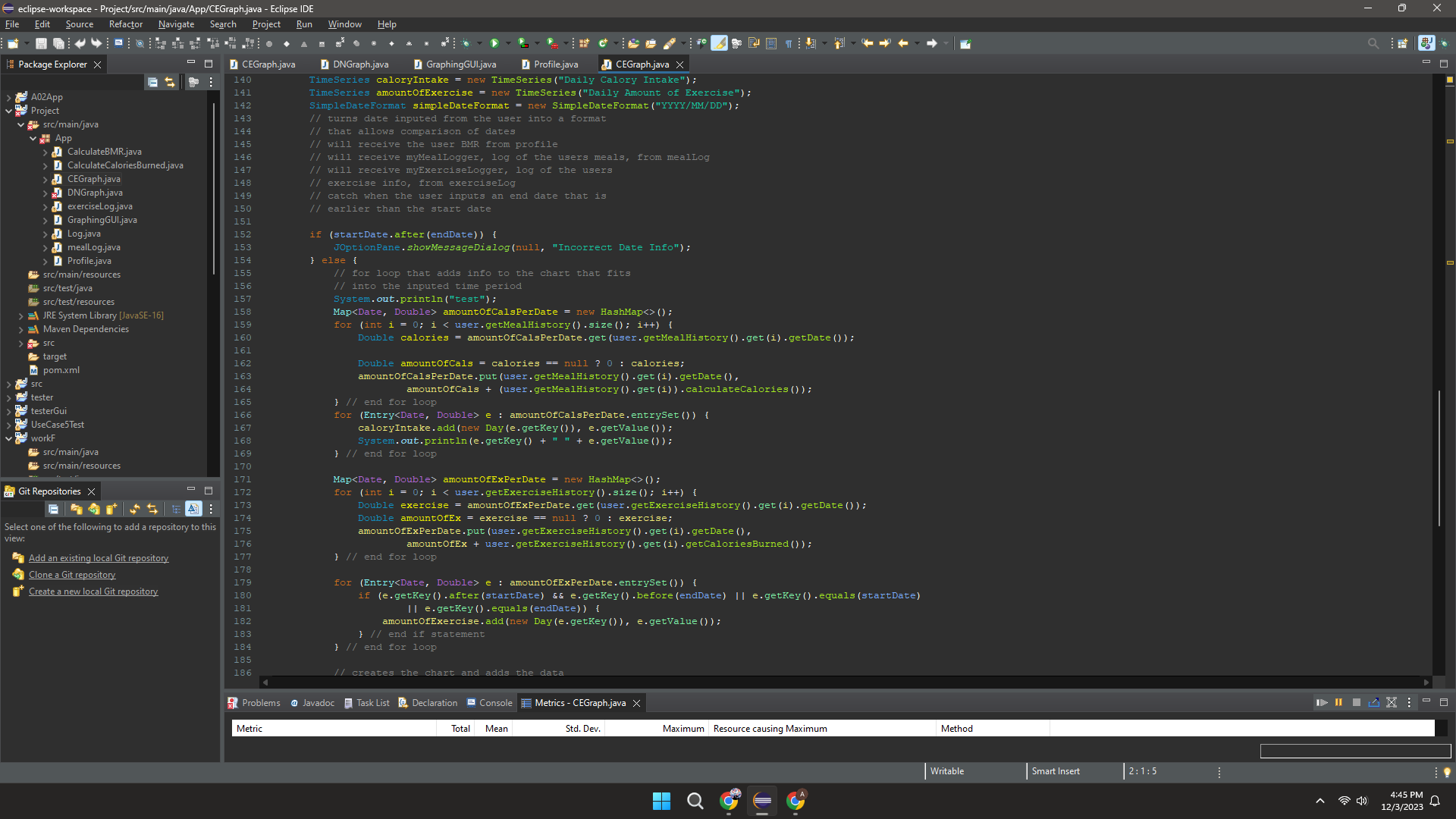
**Use case 2-3:**

**Use case 4-5:**

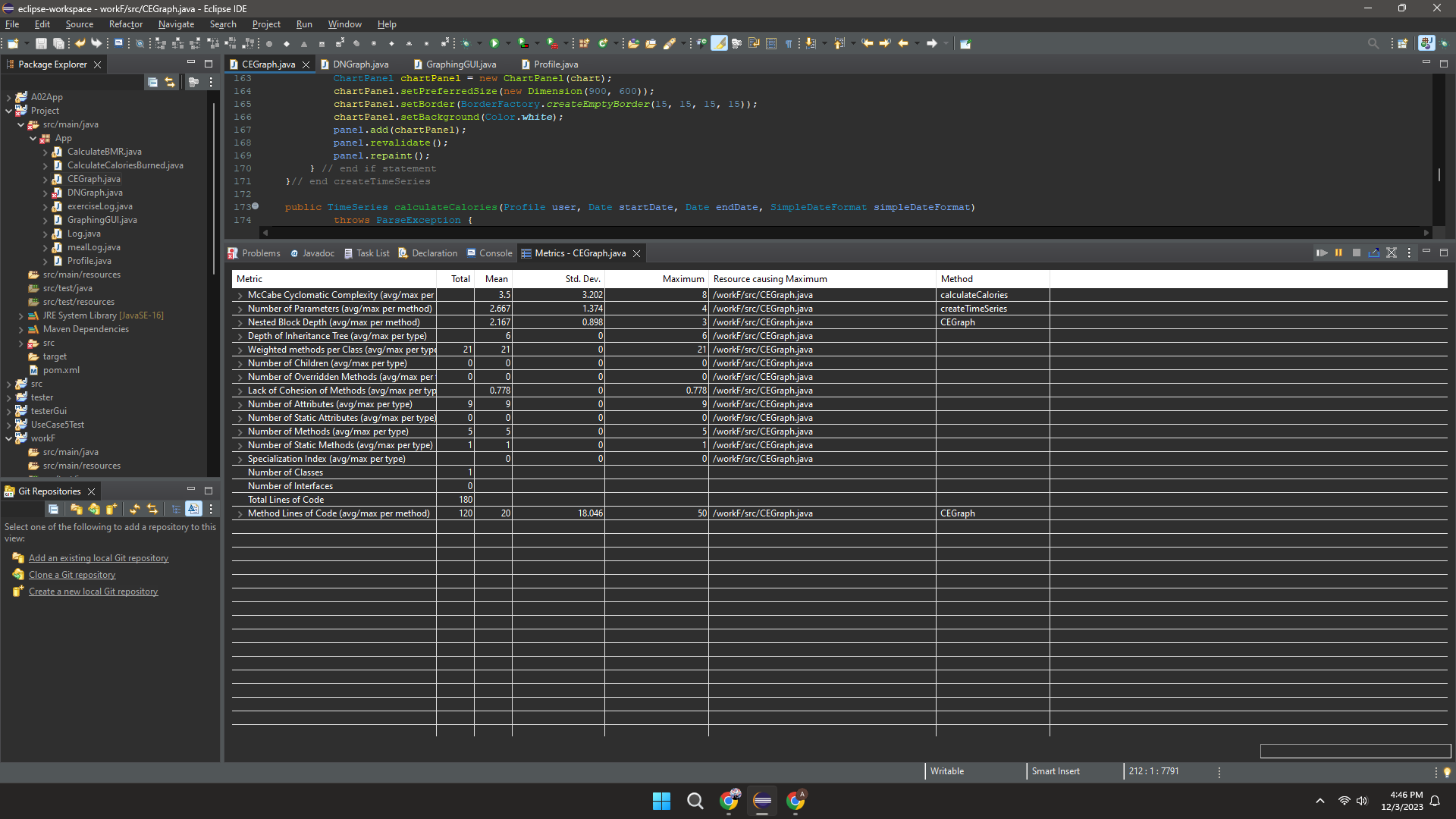
Initial metrics

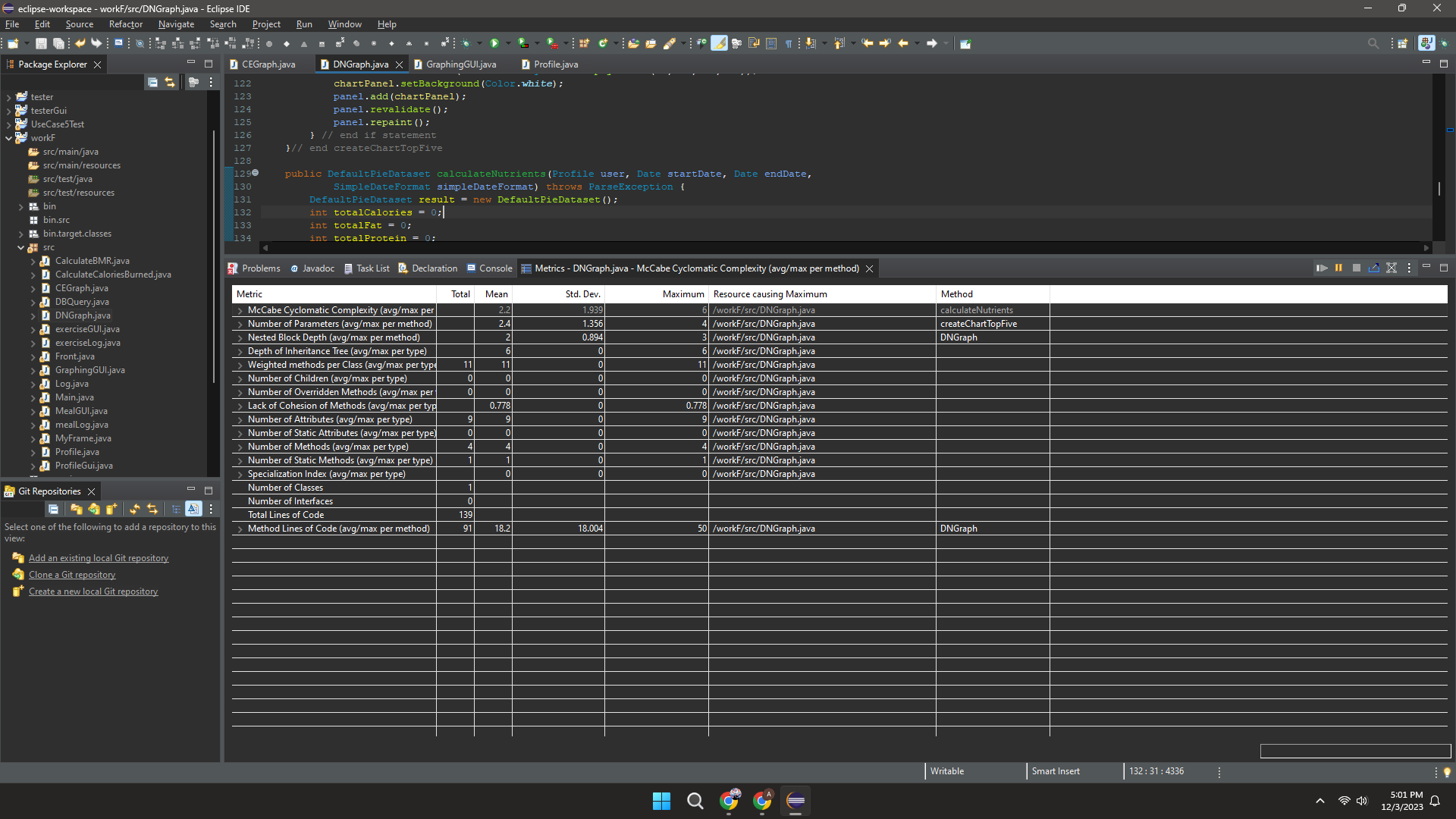


Initial Code

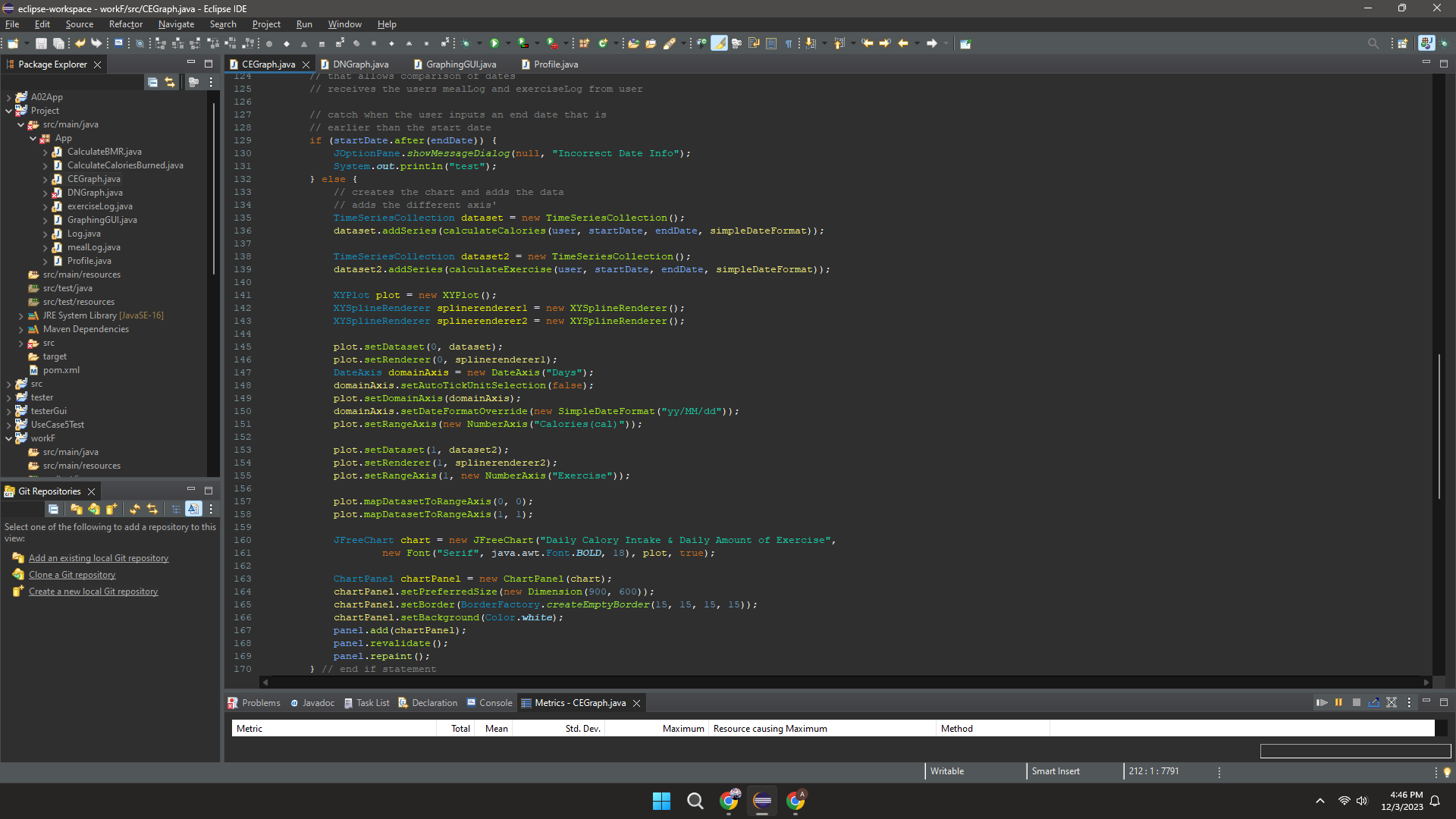


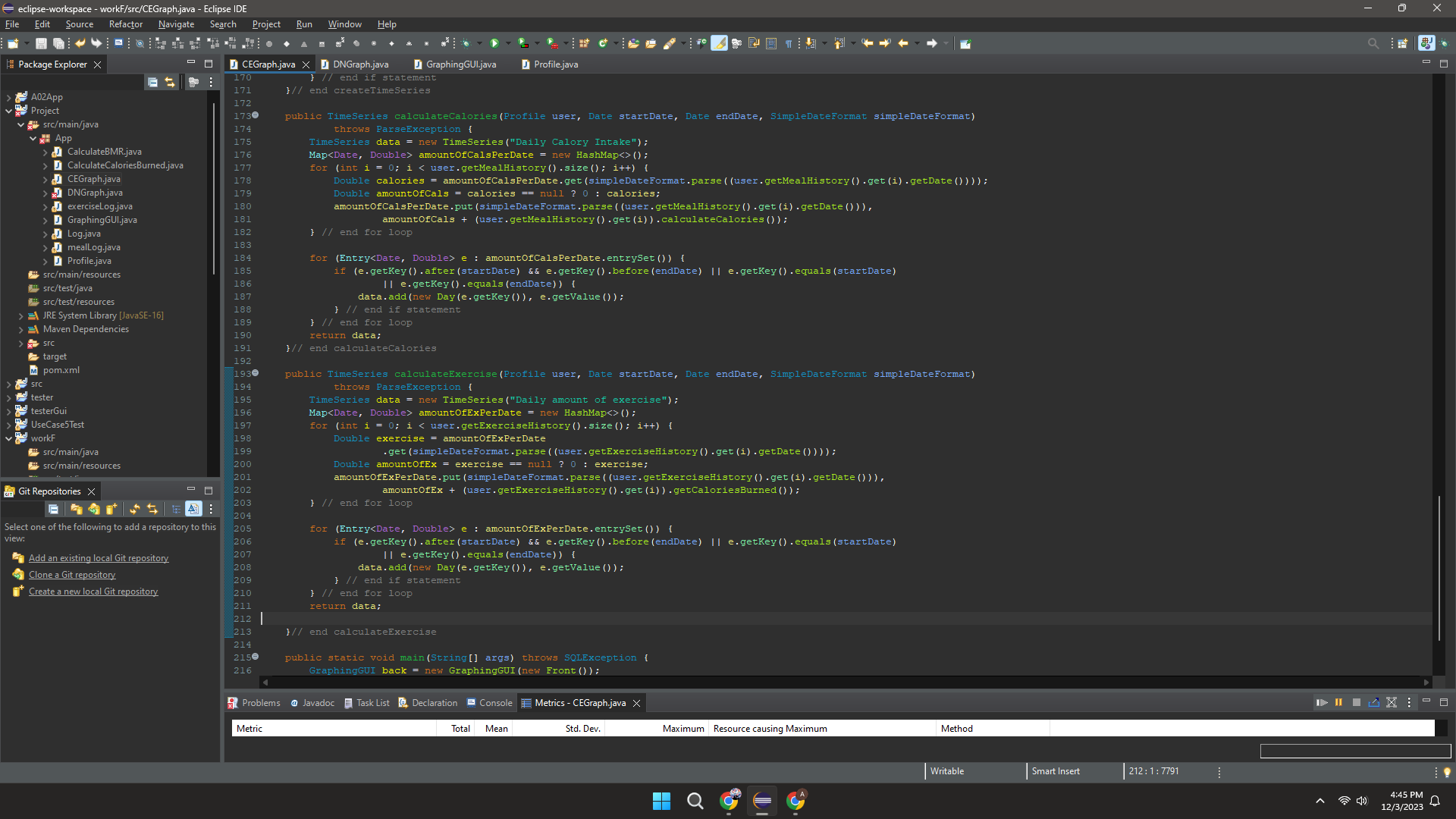
New Metrics

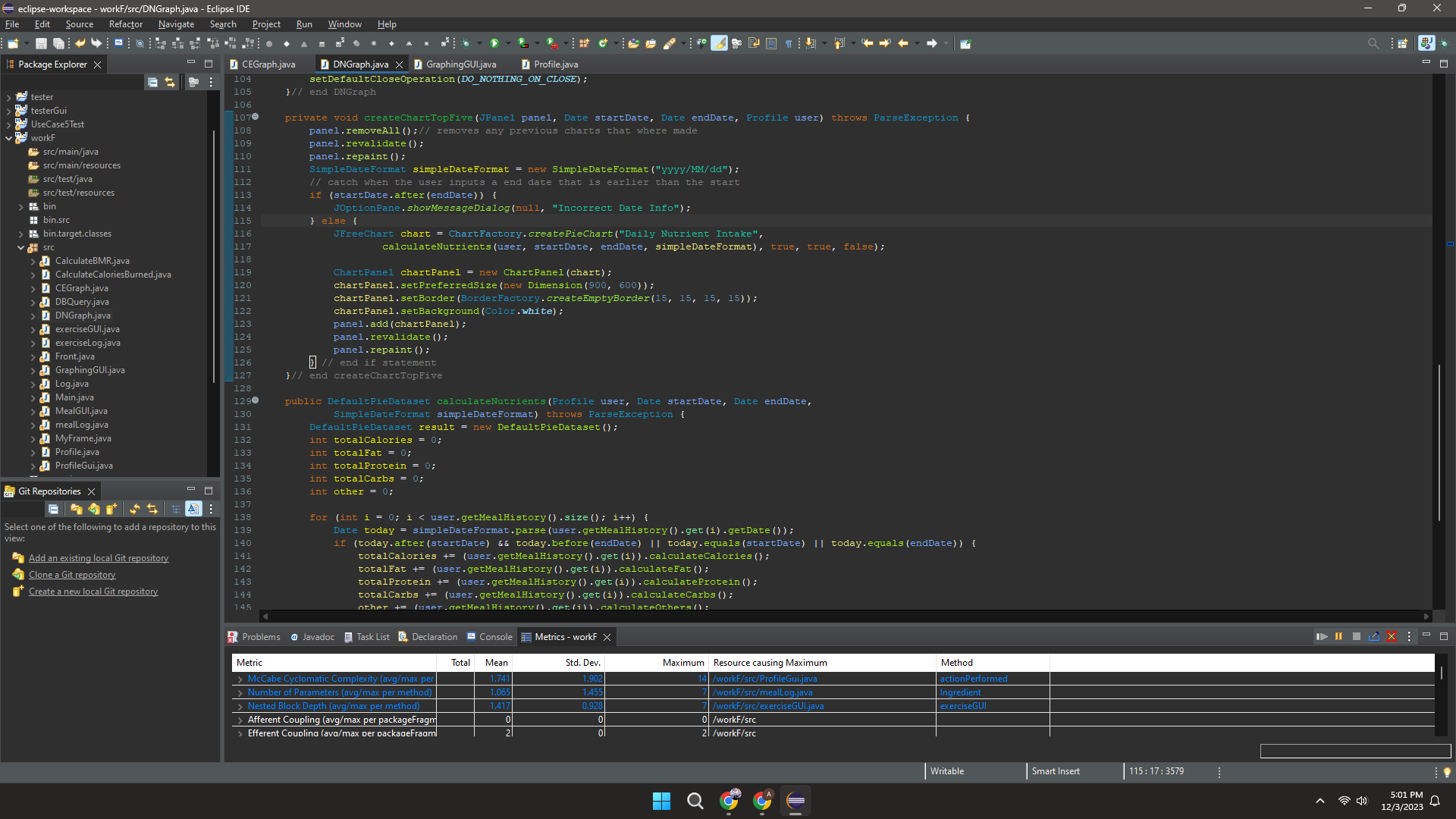
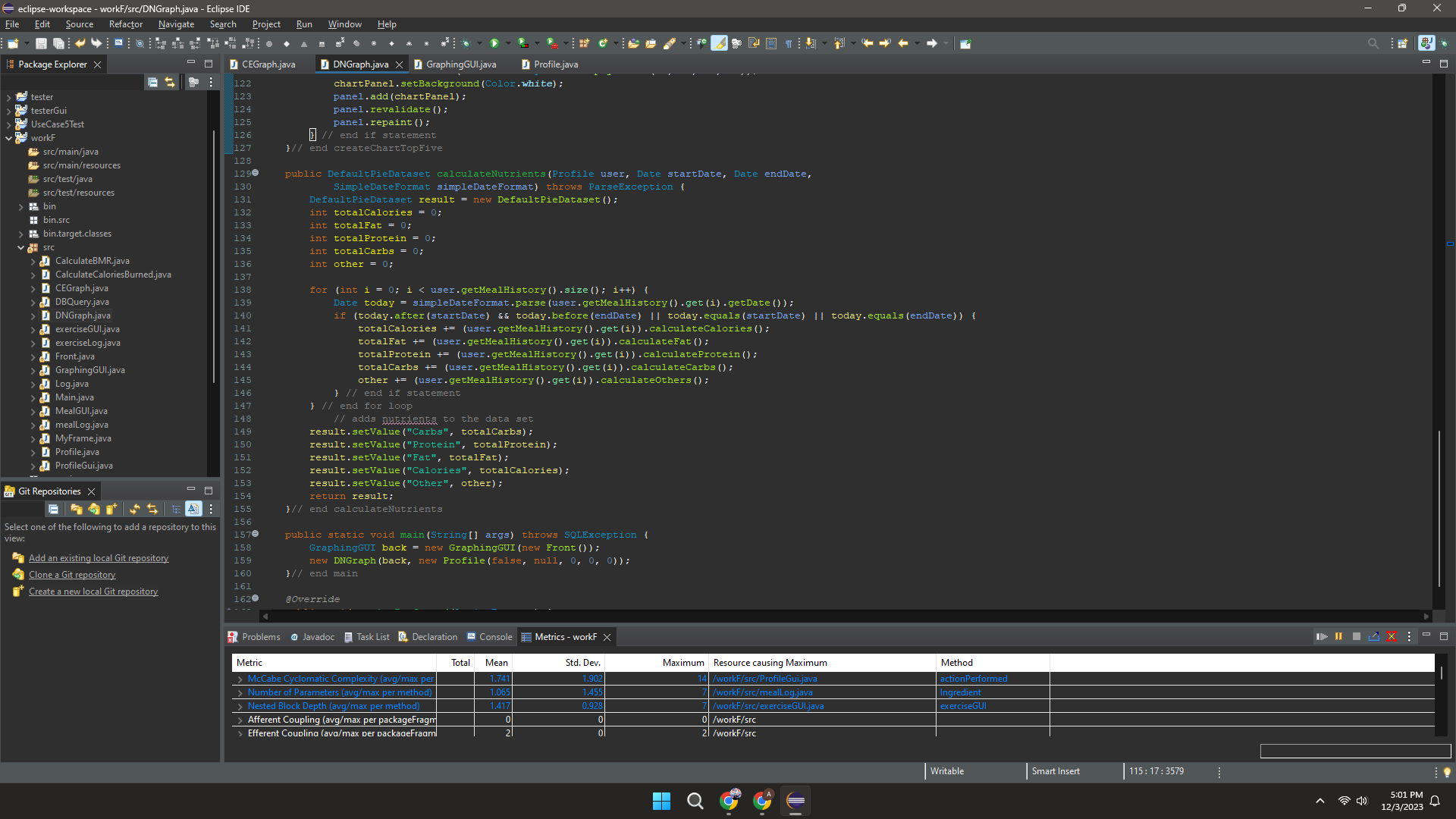




New Code







Code smells

Duplicated Code in CEGraph -> extract method

Long Method for both -> extract method

CEGraph’s method createChart repeats and calculates data in multiple places in the system so it is duplicated code. This can make the code harder to read for others and can mean that it is difficult to alter the code in the future. I used the extract method to separate the createChart method and create separate methods to calculate calorie intake and exercise.

DNGraph’s method createPieChart has a lot of lines of code and high complexity so it is a long method. Again this makes the code harder to read and update in the future and can cause future maintenance problems. I used the extract method to be able to separate the methods. One creates the chart and calls the other method to calculate the data and add it to the dataset.

With the new adjustments the complexity of both CEGraph and DNGraph has decreased.

**Use case 6-7:**