# **Lab 0**

**Preet Tailor 094790 BBIT EC**

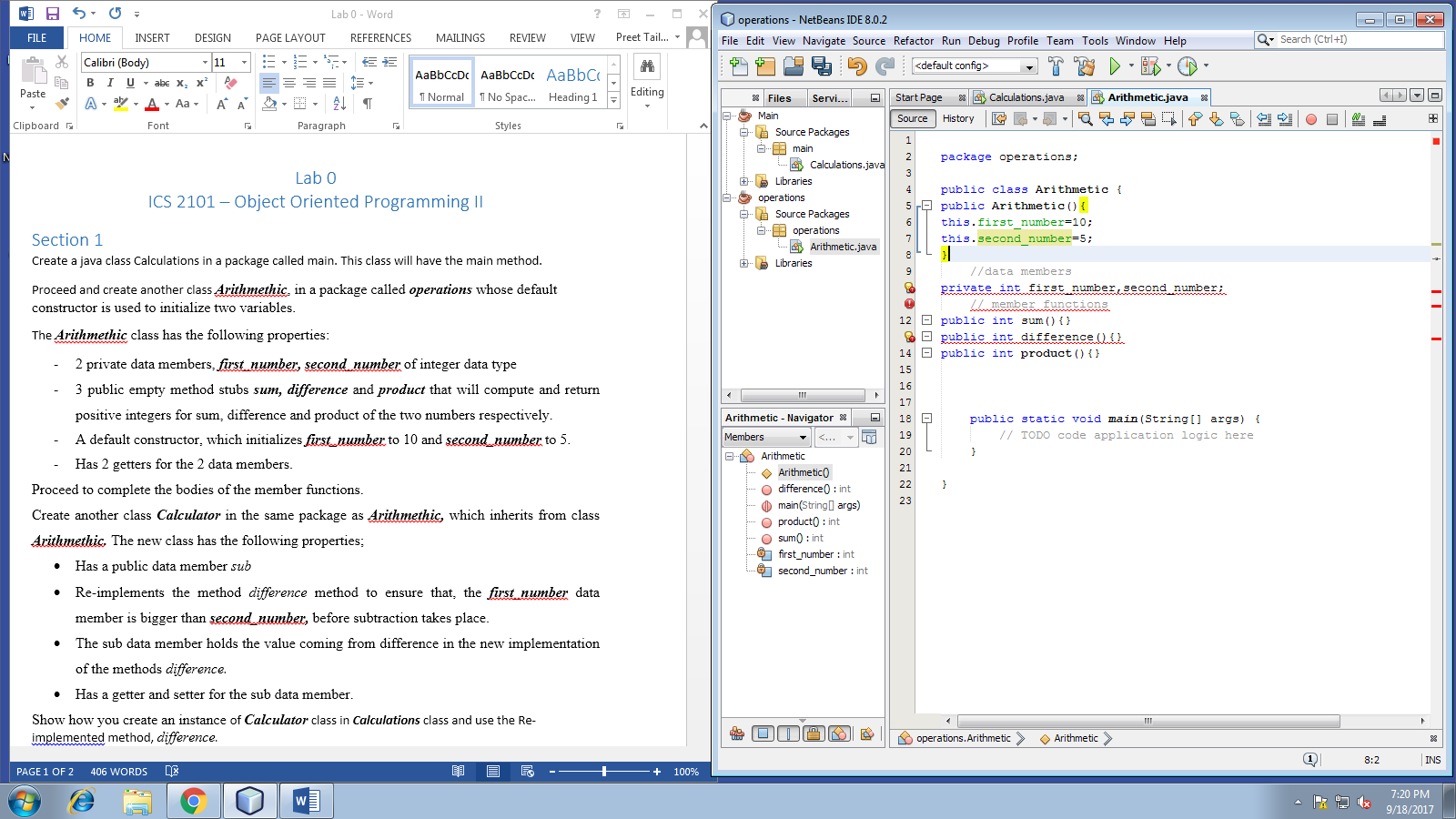
# **Object Oriented Programming II**

………………………………………………………………………………………………………………………………………………………………….

# Section 1

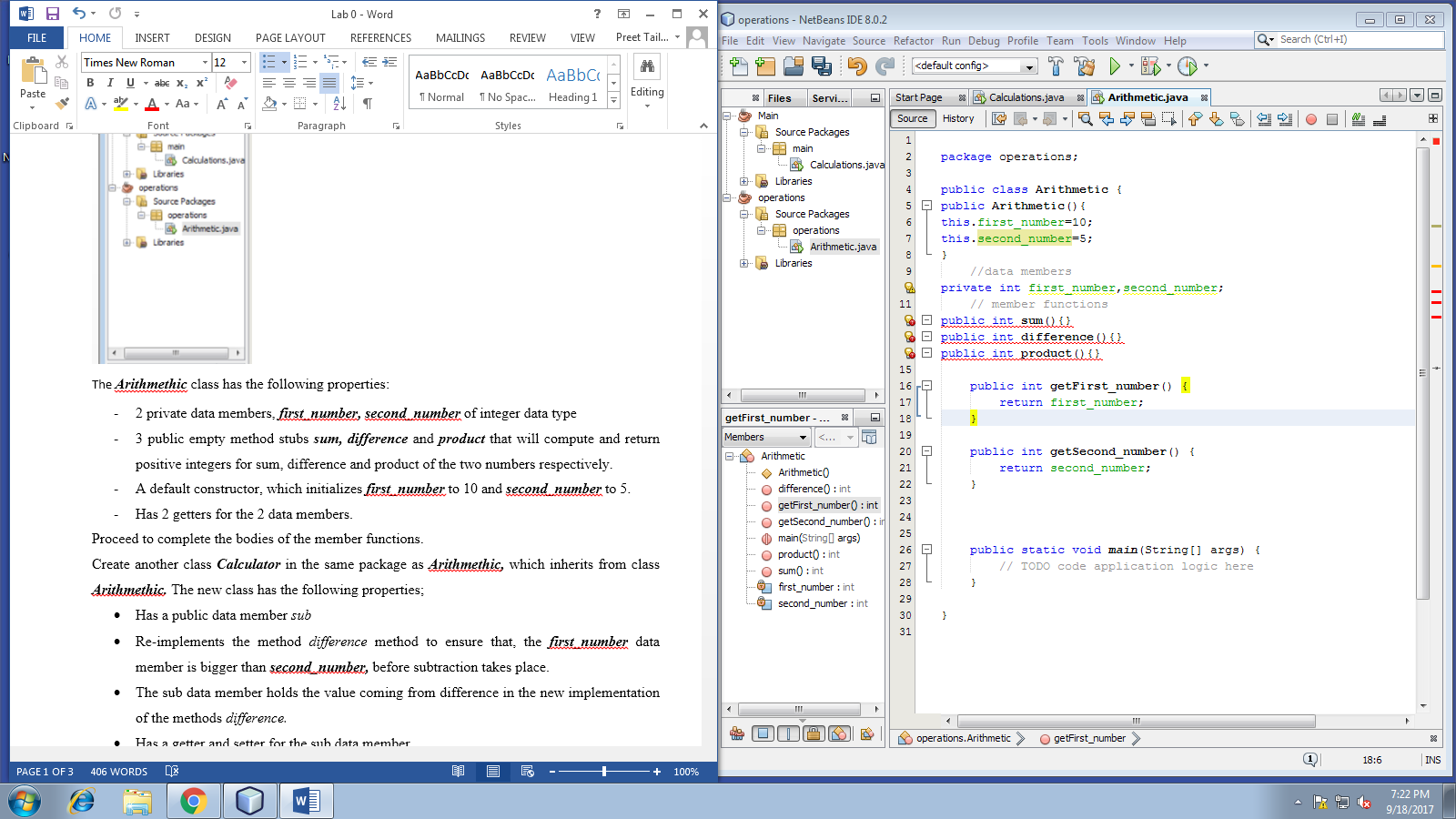
Create a java class Calculations in a package called main. This class will have the main method.

Proceed and create another class ***Arithmethic***, in a package called ***operations***whose default constructor is used to initialize two variables.

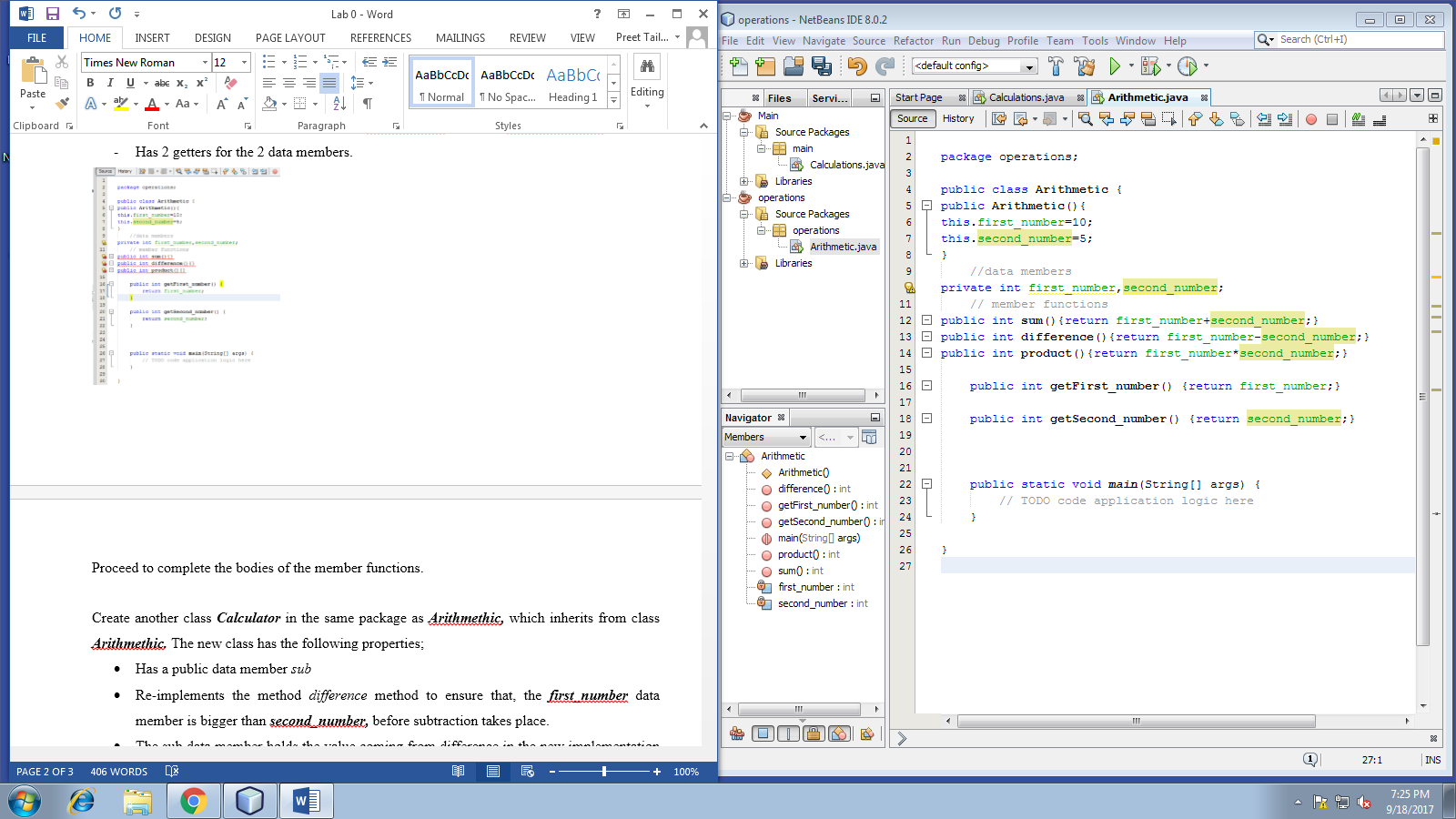


The ***Arithmethic***class has the following properties:

* 2 private data members, ***first\_number, second\_number*** of integer data type
* 3 public empty method stubs ***sum, difference*** and ***product*** that will compute and return positive integers for sum, difference and product of the two numbers respectively.
* A default constructor, which initializes ***first\_number***to 10 and ***second\_number***to 5.
* Has 2 getters for the 2 data members.

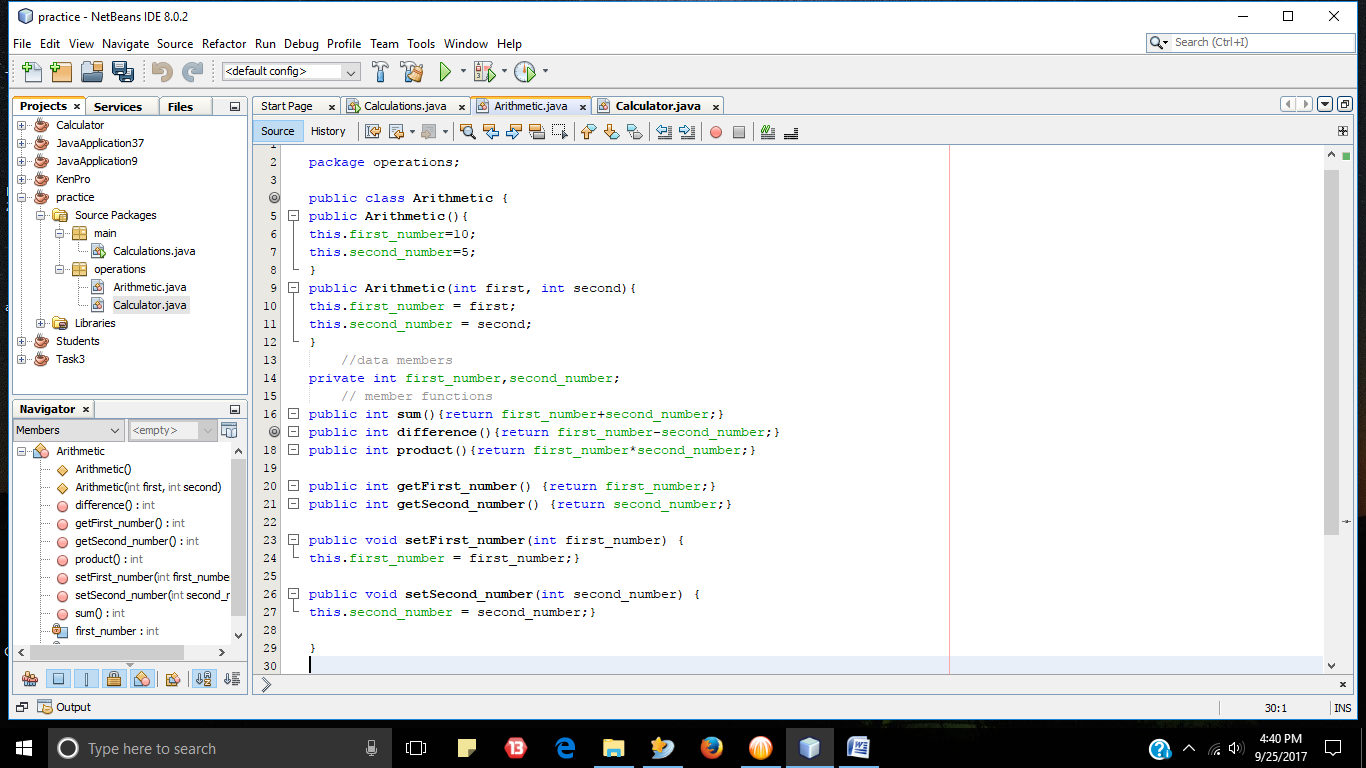


Proceed to complete the bodies of the member functions.

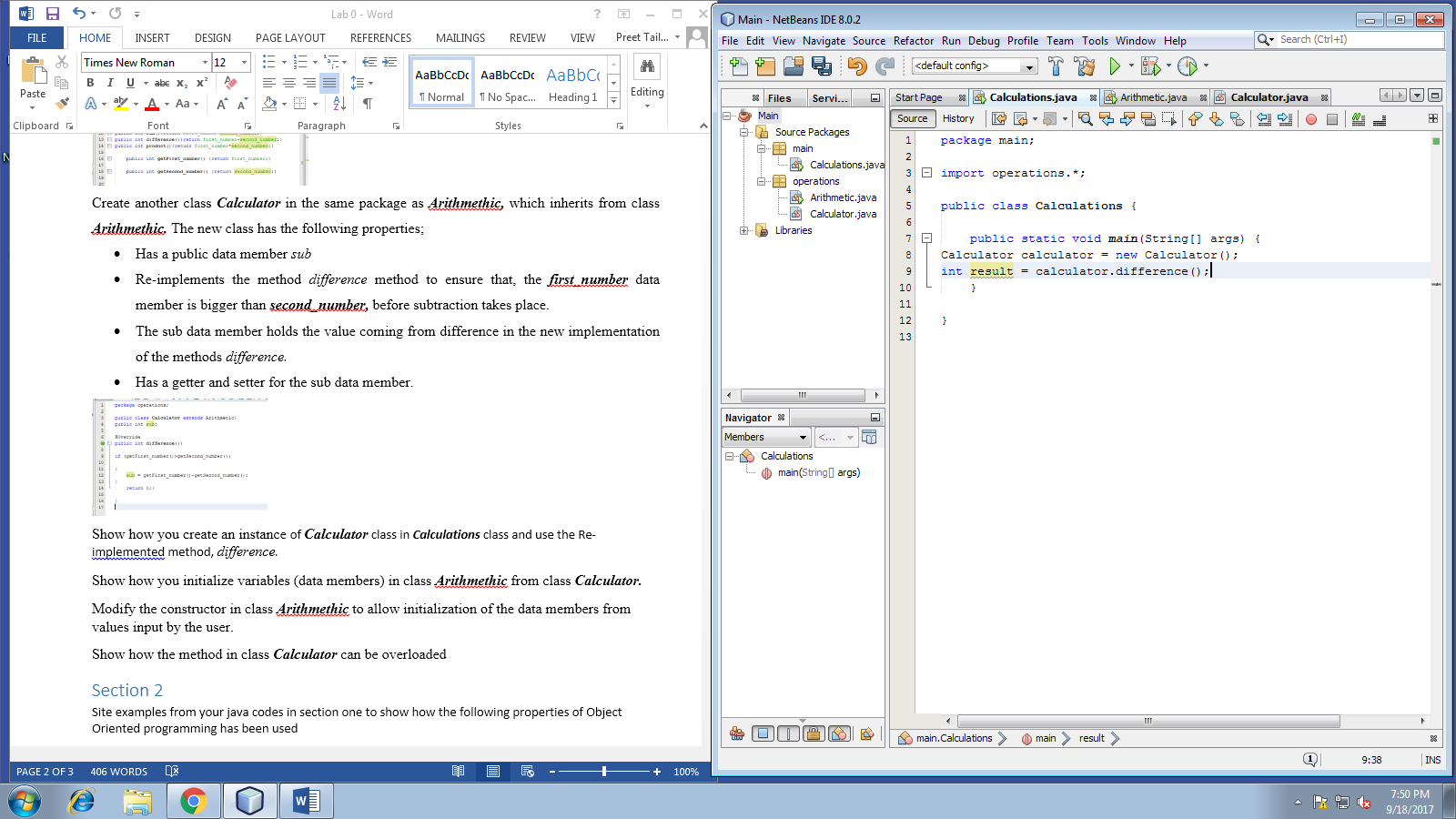


Create another class ***Calculator*** in the same package as ***Arithmethic,*** which inherits from class ***Arithmethic.*** The new class has the following properties;

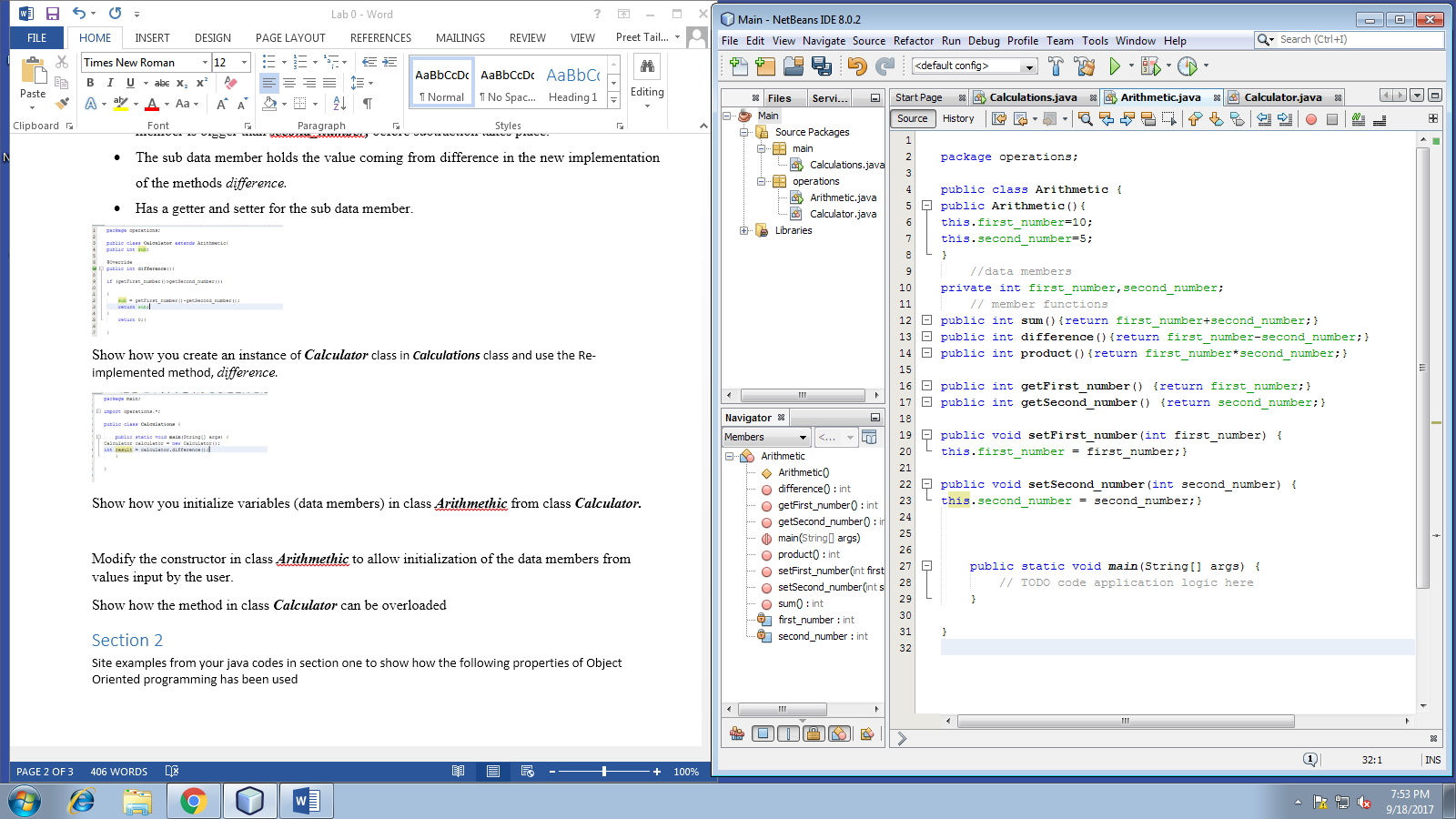
* Has a public data member *sub*
* Re-implementsthe method *difference* method to ensure that, the ***first\_number*** data member is bigger than ***second\_number,*** before subtraction takes place.
* The sub data member holds the value coming from difference in the new implementation of the methods *difference.*
* Has a getter and setter for the sub data member.



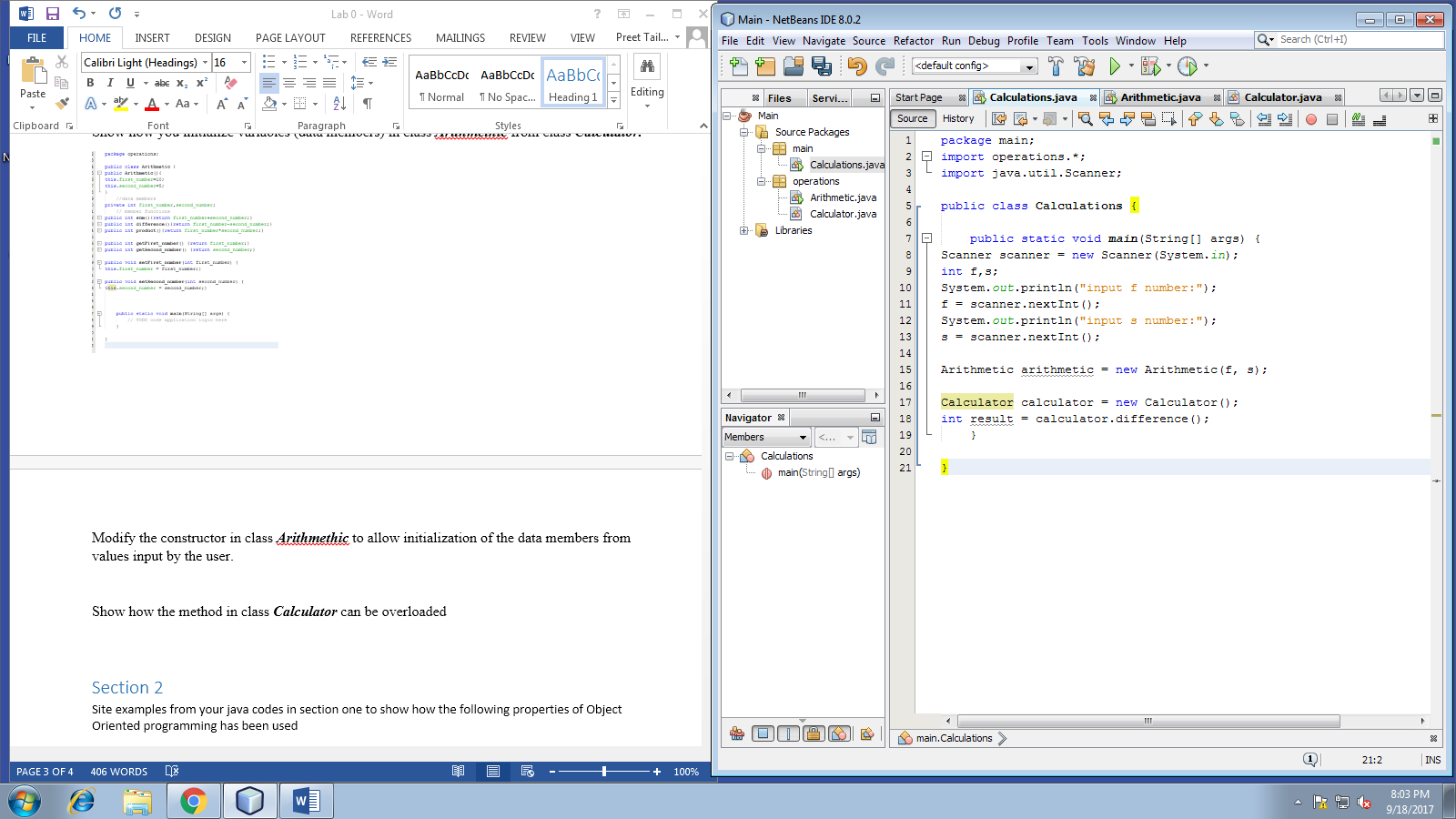
Show how you create an instance of ***Calculator*** class in ***Calculations*** class and use the Re-implemented method, *difference.*



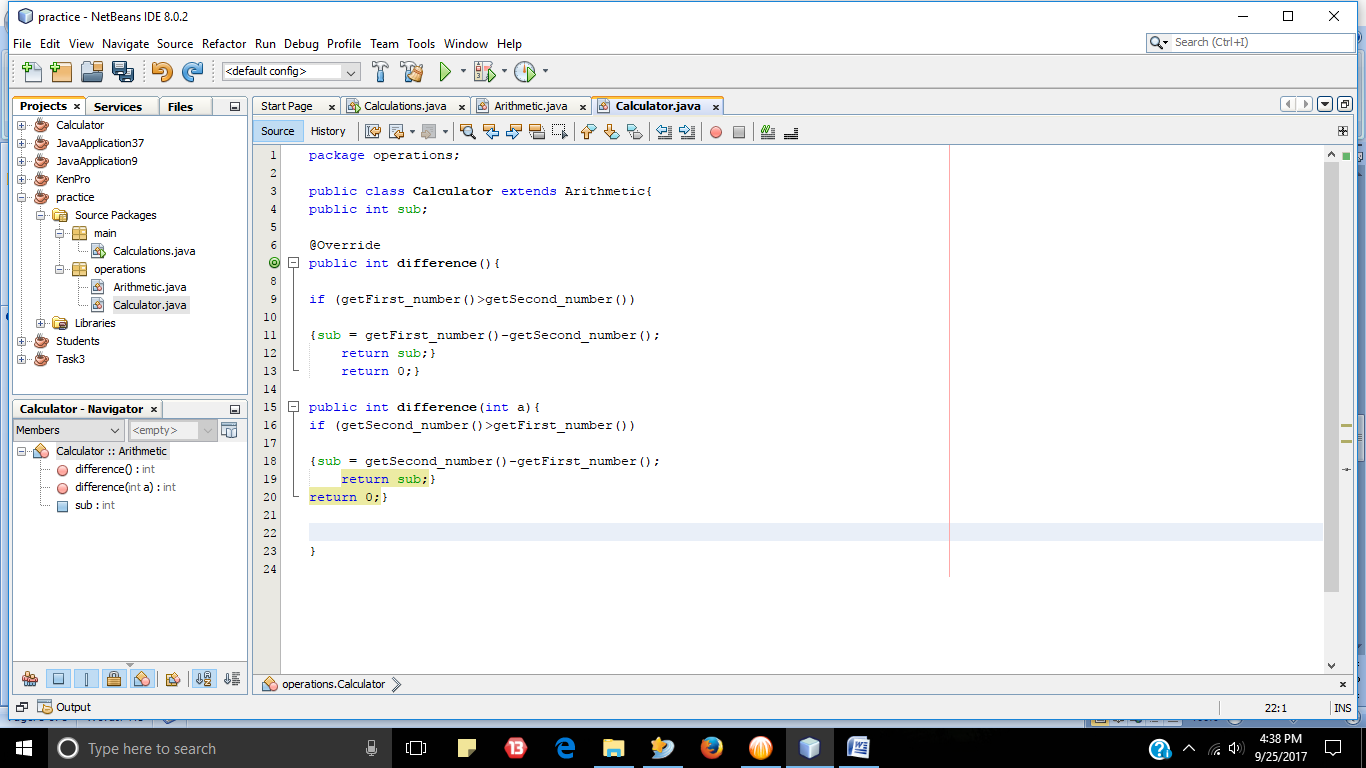
Show how you initialize variables (data members) in class ***Arithmethic*** from class ***Calculator.***



Modify the constructor in class ***Arithmethic*** to allow initialization of the data members from values input by the user.



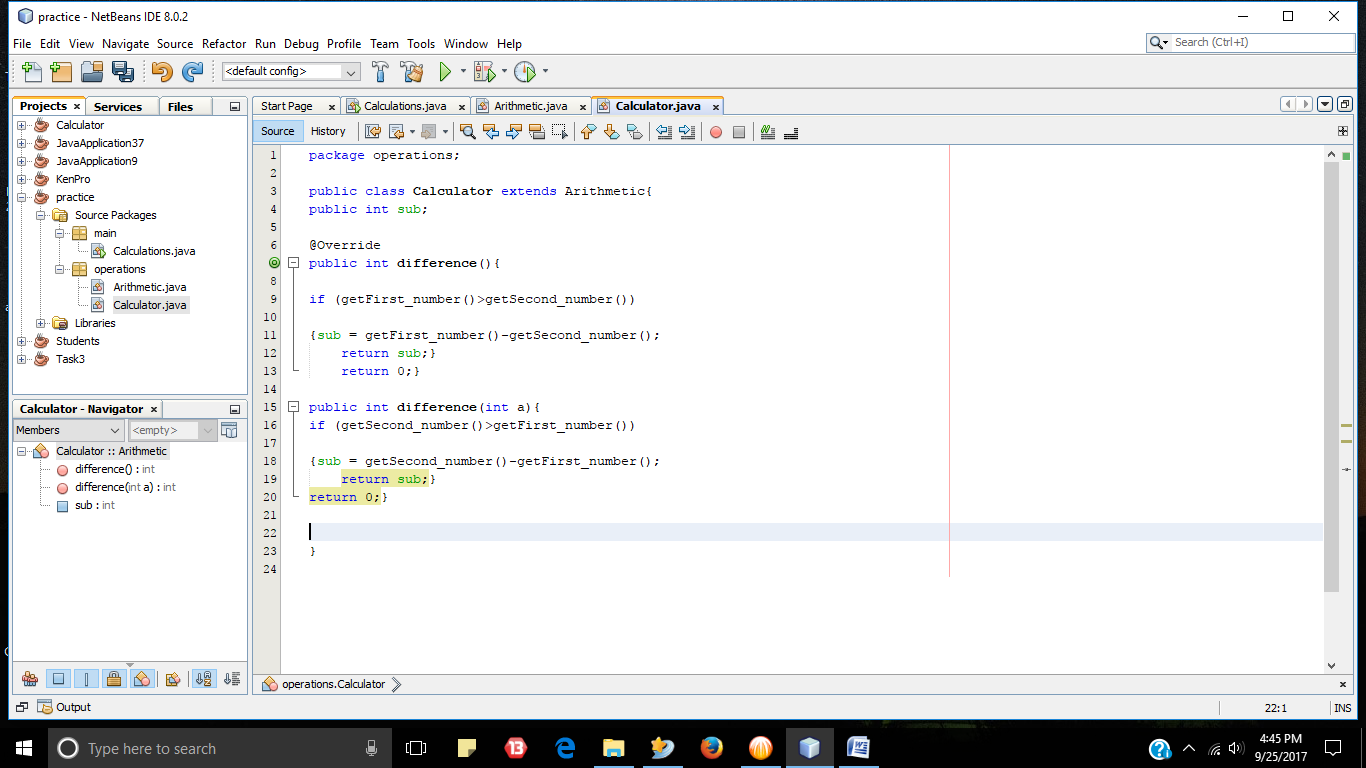
Show how the method in class ***Calculator*** can be overloaded



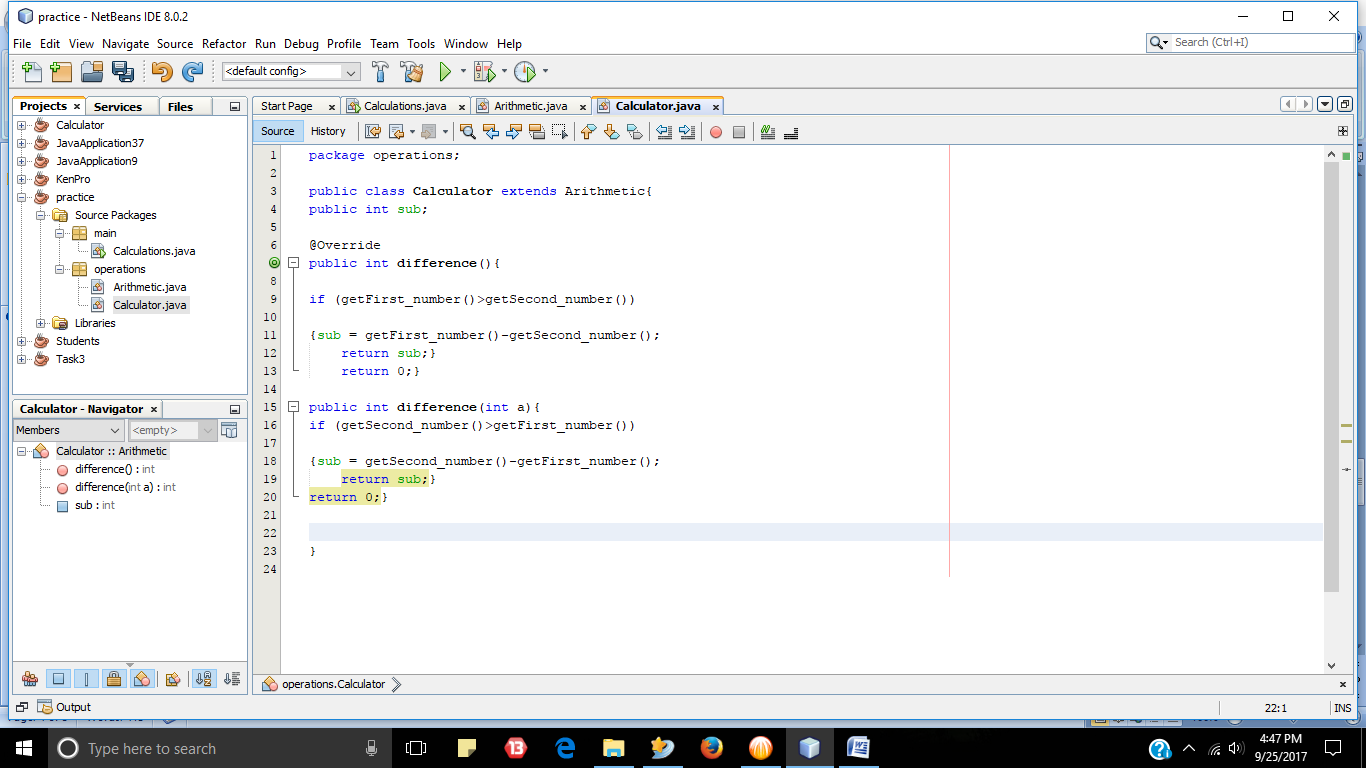
# Section 2

Site examples from your java codes in section one to show how the following properties of Object Oriented programming has been used

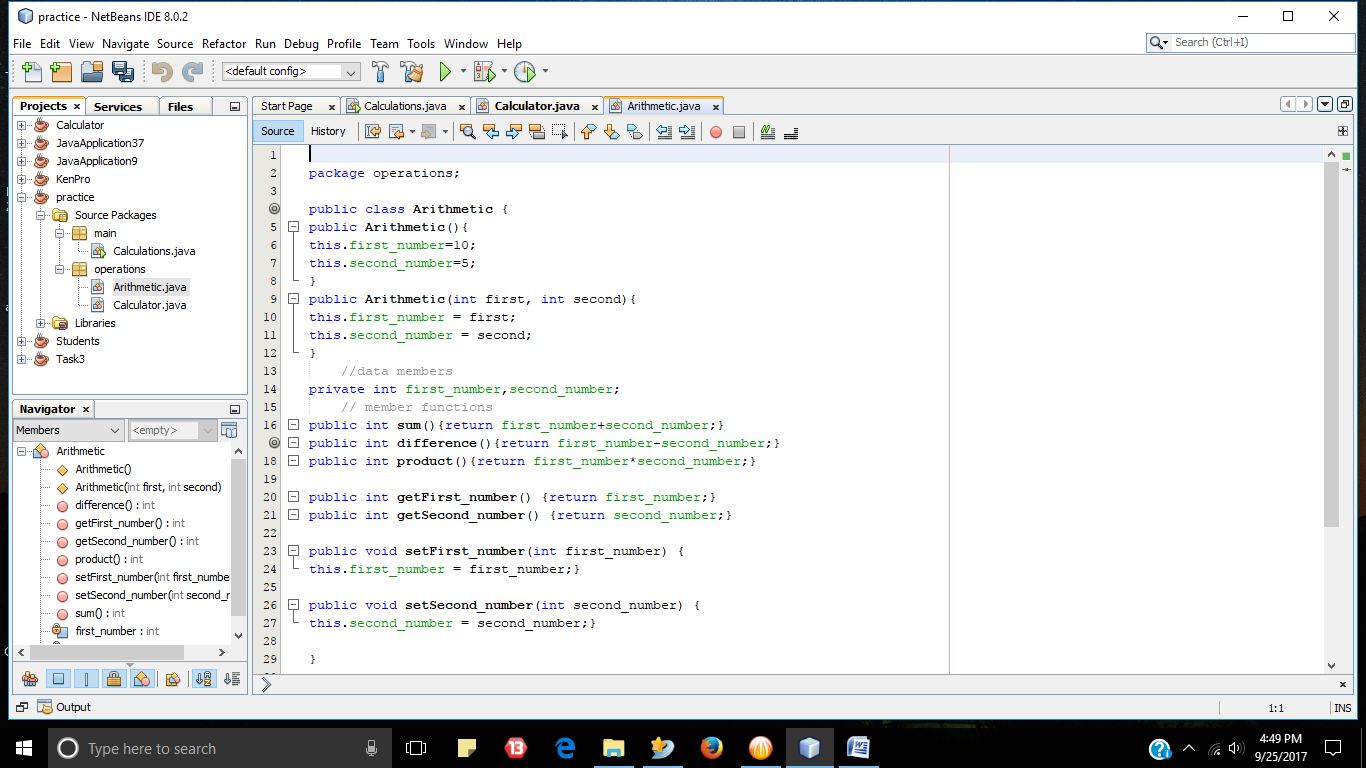
1. Inheritance



1. Polymorphism



1. Encapsulation



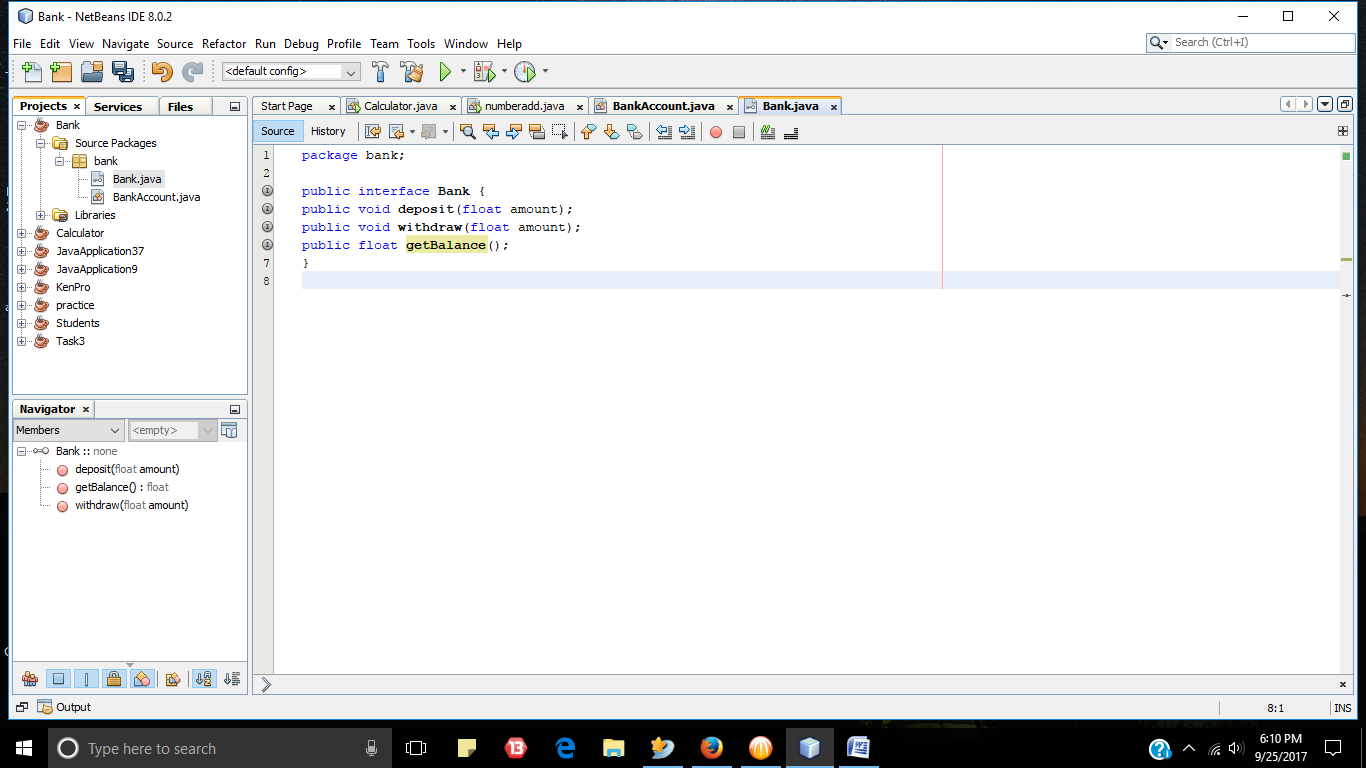
# Section 3

Create an interface Bank with the following unimplemented methods

Deposit – take in one parameter, *amount,* it is void

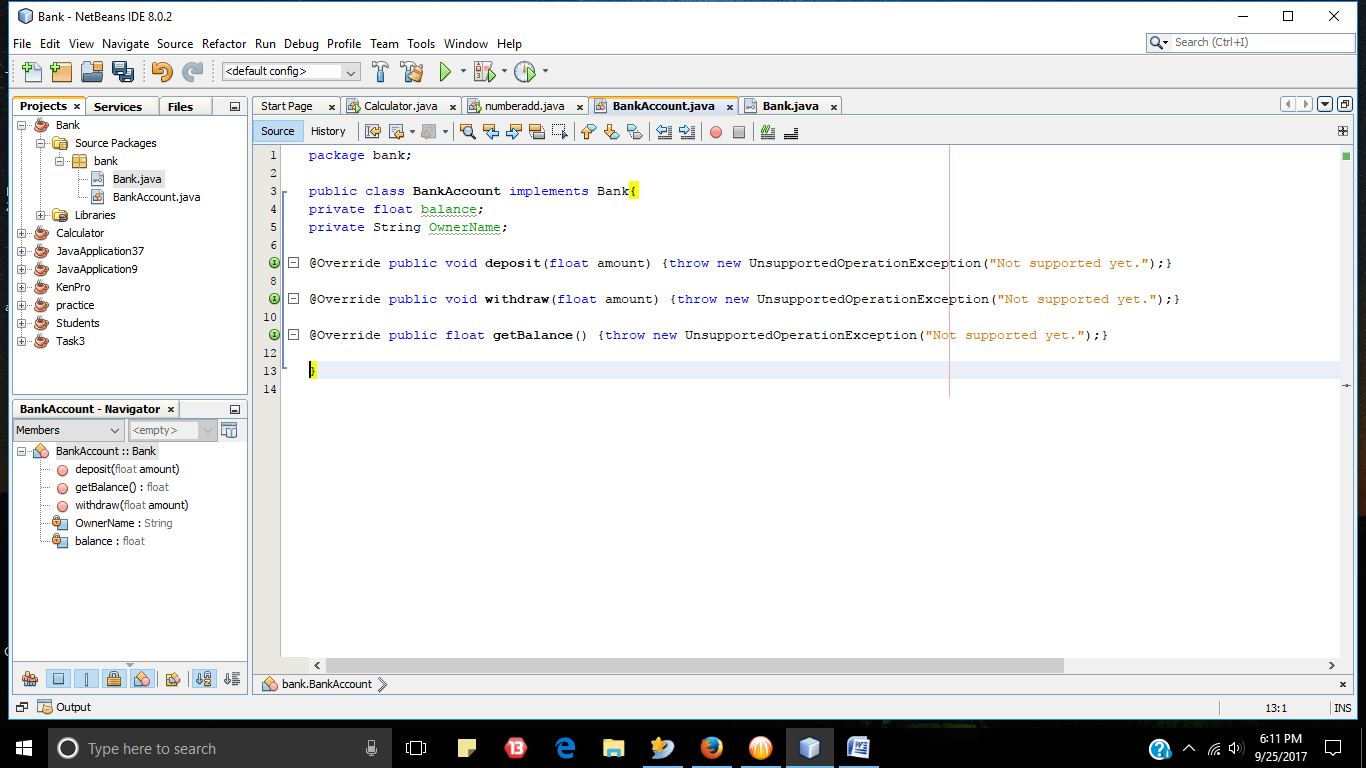
Withdraw - take in one parameter, *amount,* it is void

getBalance – no parameter, but returns float

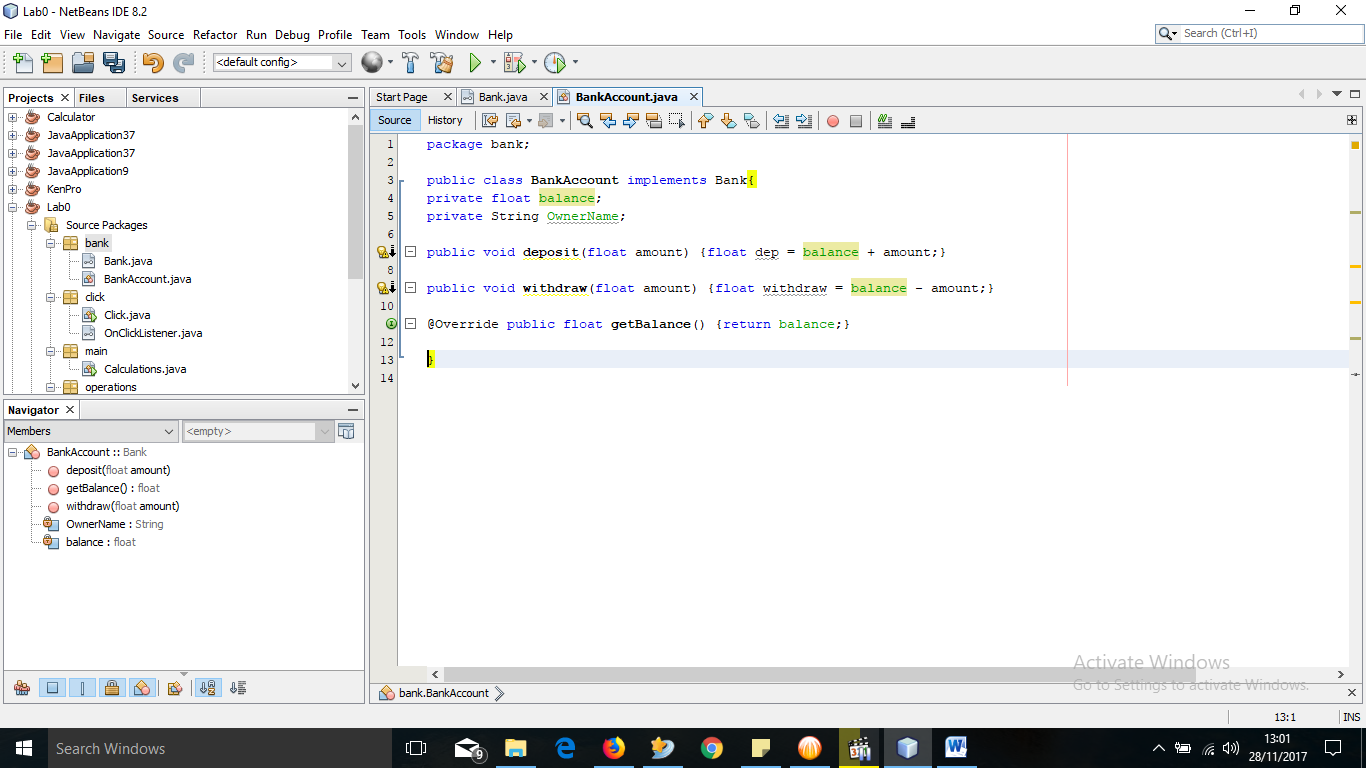


Show how you implement the interface in a class called BankAccount, which has the following data members;

* Balance – the amount of money in the bank
* OwnerName – the full names of account holder

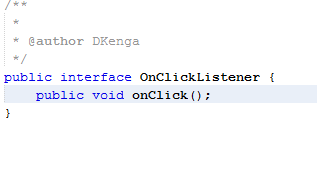


Feel free to write method bodies in your own way, but at least your method bodies must make sense in relation to the method sname.



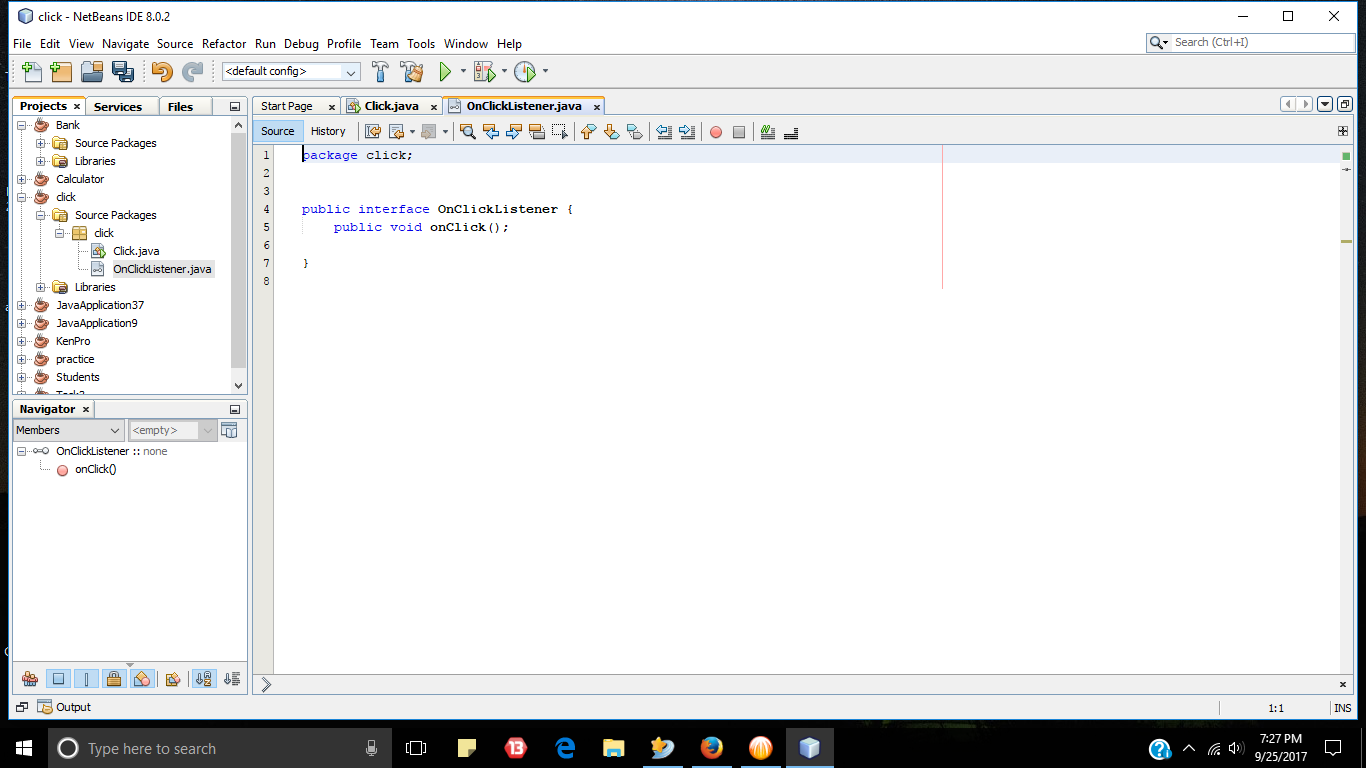
# Section 4

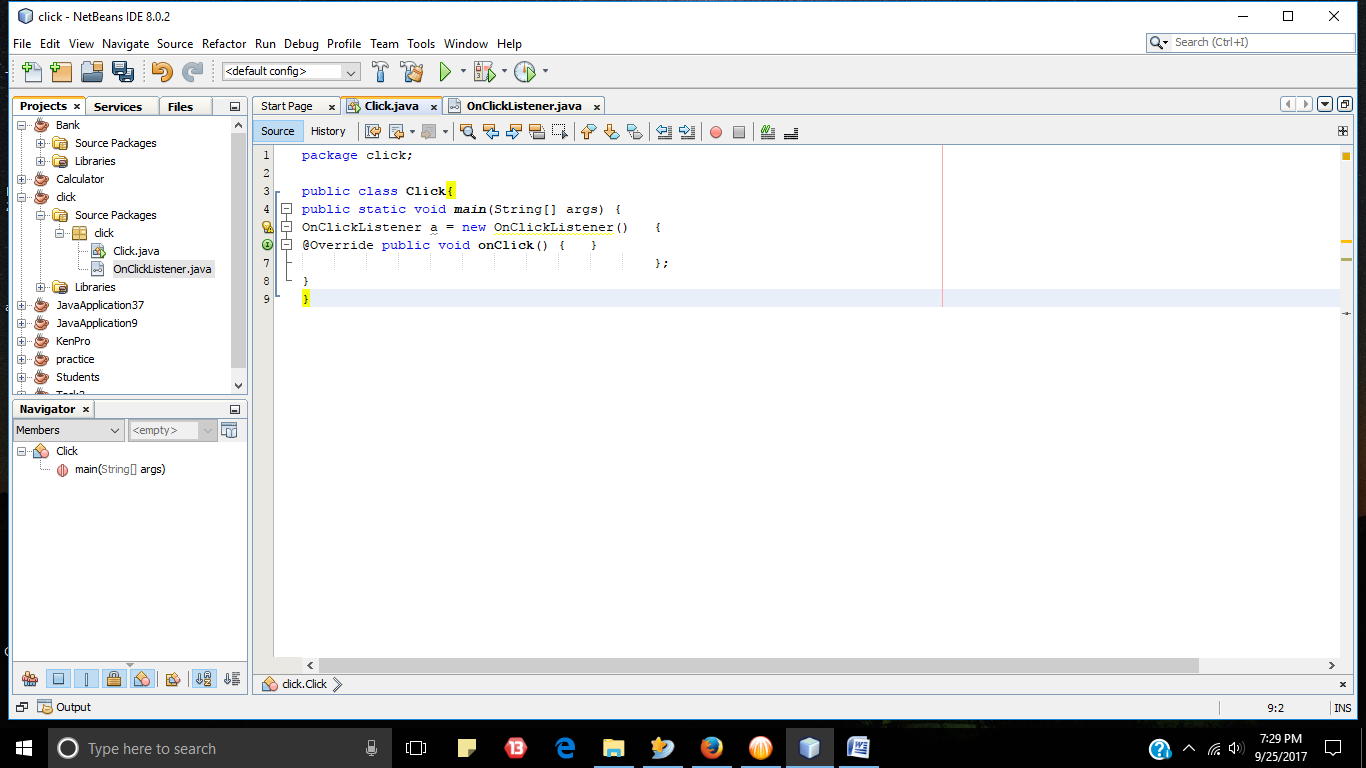
Consider an interface shown below



Show how you implement the above interface using

1. A named class -
2. Anonymous inner class.





***-This lab is complete-***