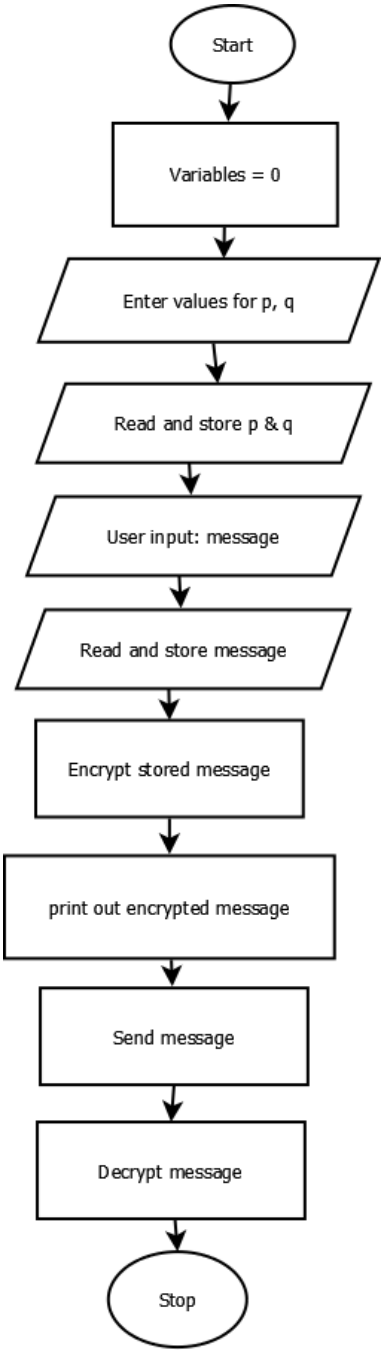
Matt Sellman (msell010)

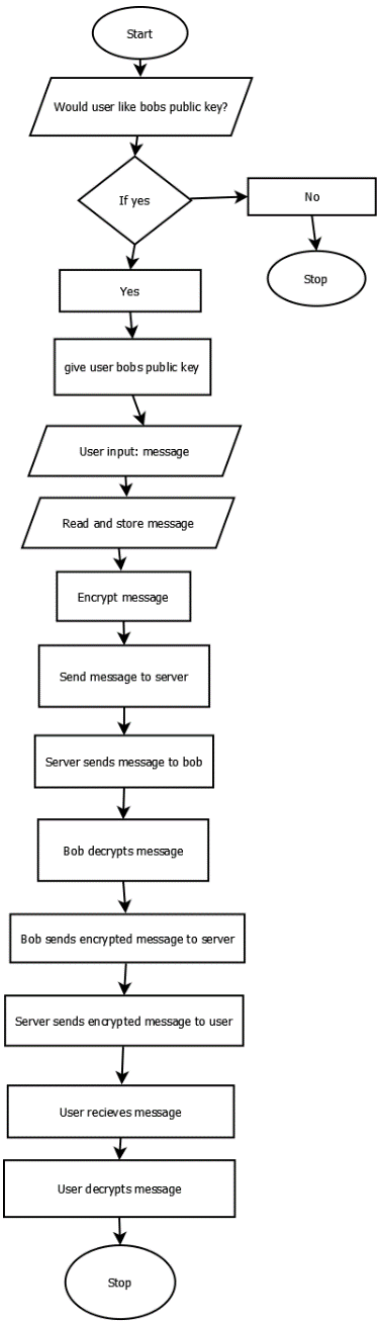
My Report:

(a) Algorithms (in flow-chart):

In my code I used the RSA algorithm, I did this by first creating a key pair, namely, e and n. To make the values needed for the encrypt I needed a few numbers, the first of which was n. Which was made by multiplying p and q together. Next I needed to make phi, or ϕ by (ϕ = ϕ(n) = (p-1).(q-1). While e was made by being the co-prime of phi and being less than n.

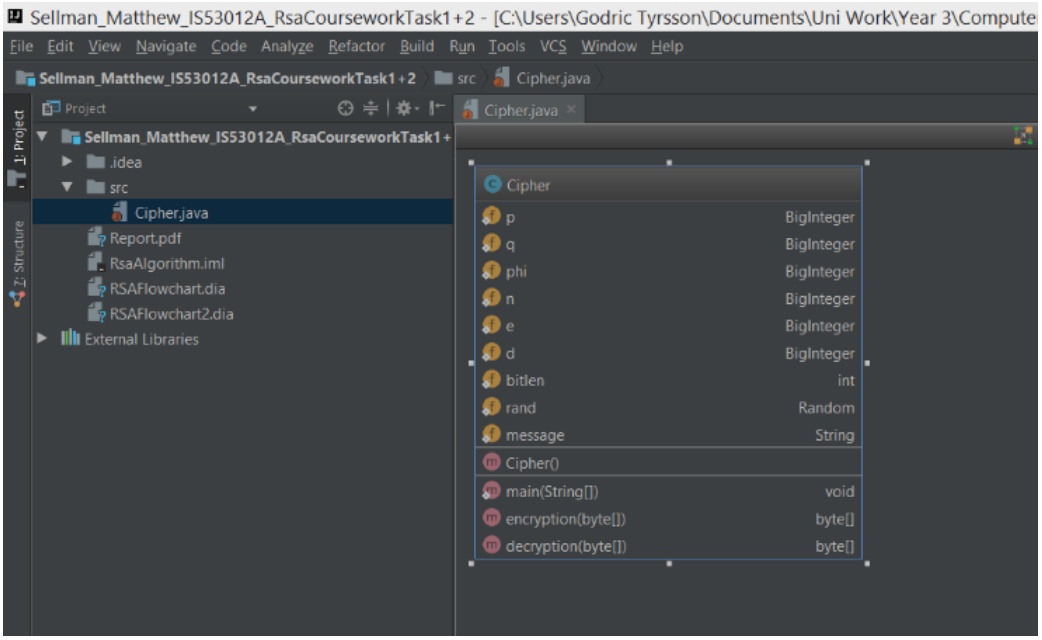
These are then used to encrypt a message by turning that message into bytecode, then multiplying the bytecode by the key pair. This would then be sent to the other user, who would decrypt it using their private key, which would be made from d and n. I made d by using the inverse of e modulo ϕ: e. So d mod ϕ = 1. So the flowchart will look more like:





Matt Sellman (msell010)

(b) Design (in block diagram or class-diagram in UML)

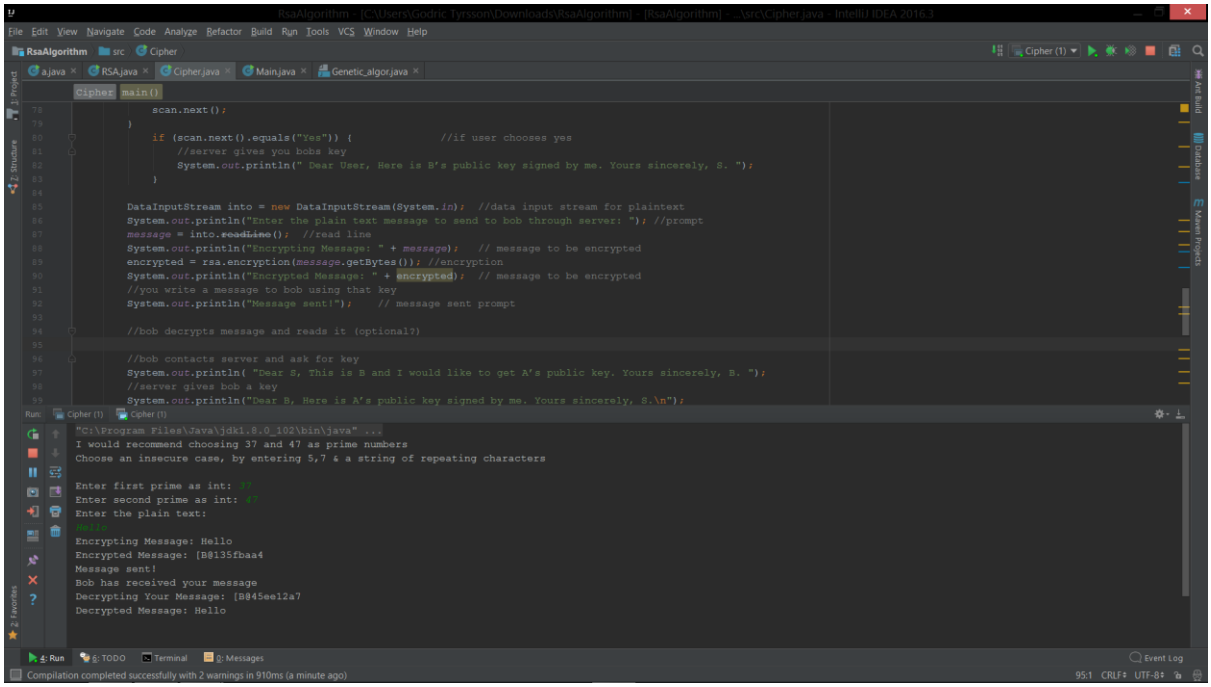


The main class is the cipher class, with encryption and decryption being two separate objects that interact with the main method. There are also several big integers, one int, one random variable and one string.

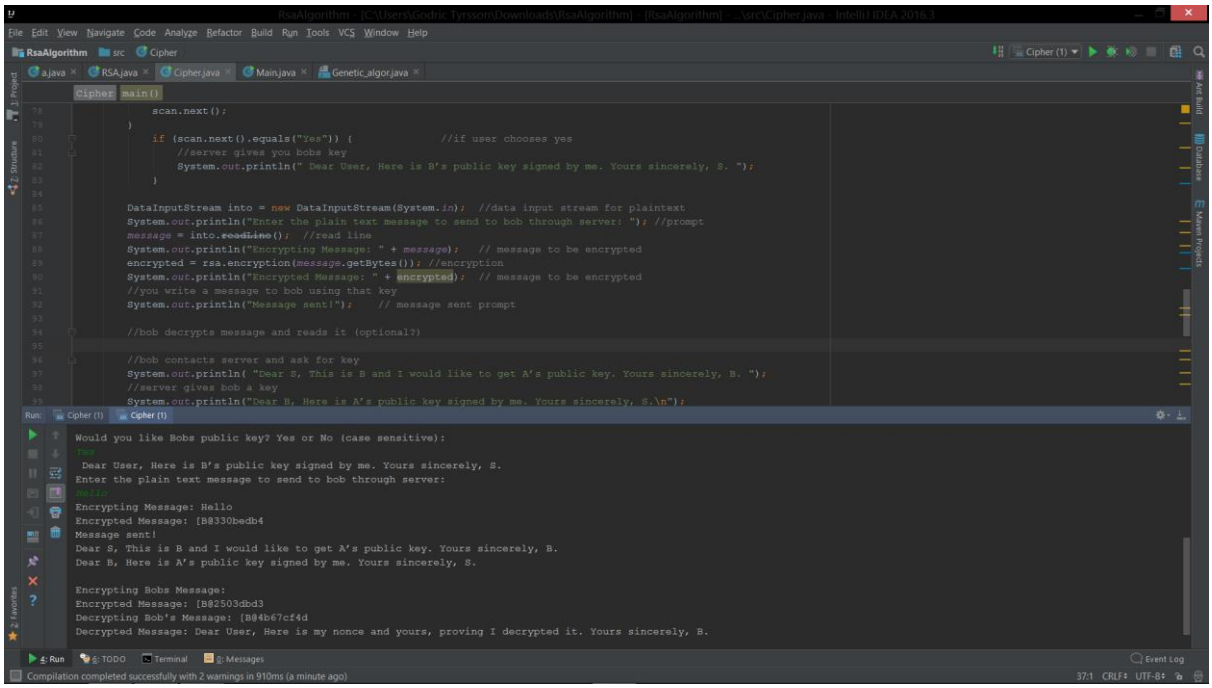
Matt Sellman (msell010)

(c) Demonstration (in 5 best screen-shots)

Code functioning normally (task 1)

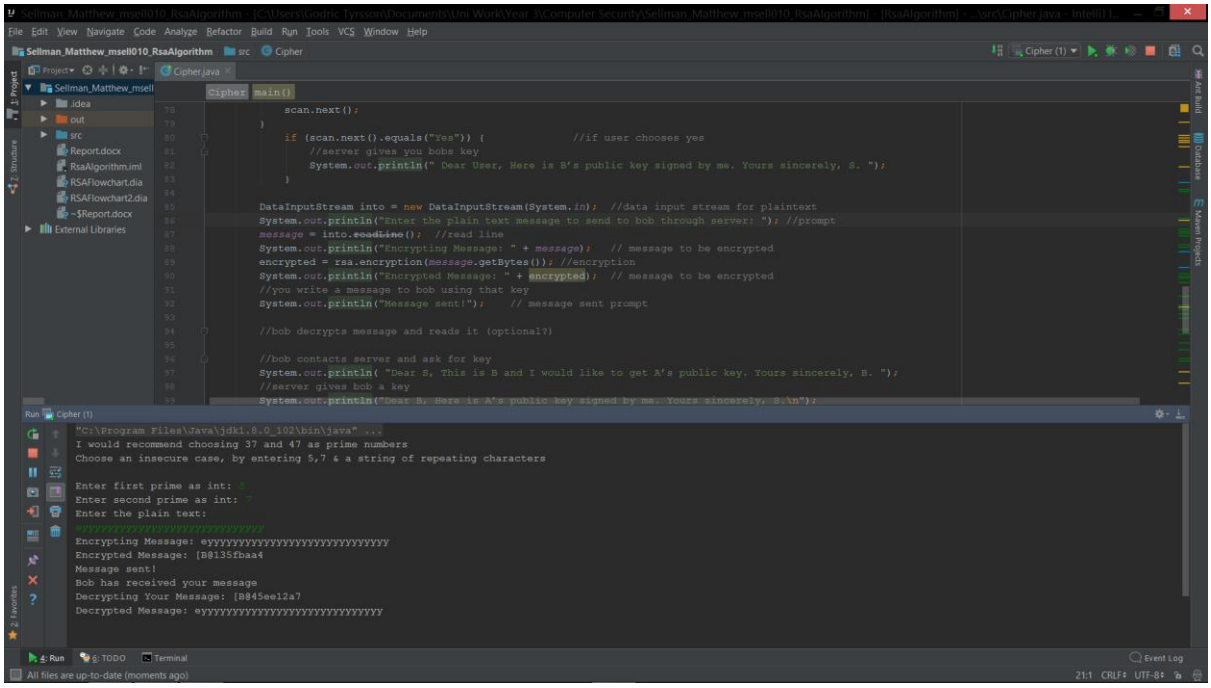


Code functioning normally (task 2)

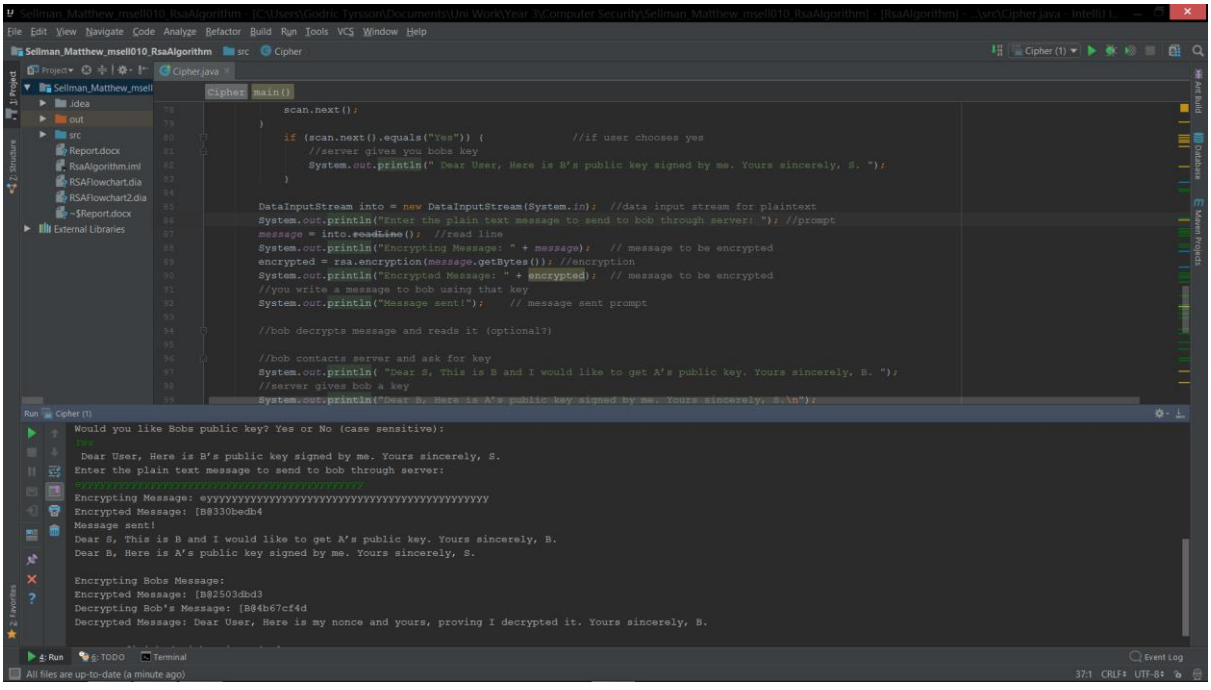


Matt Sellman (msell010)

An insecure case (task 1) repeating characters mean that the bytecode and easily be interpreted and the small prime numbers mean that the keys are not large numbers. This means that their origin would only be a small range of numbers and therefore the private and public keys could be computed. (task 1)

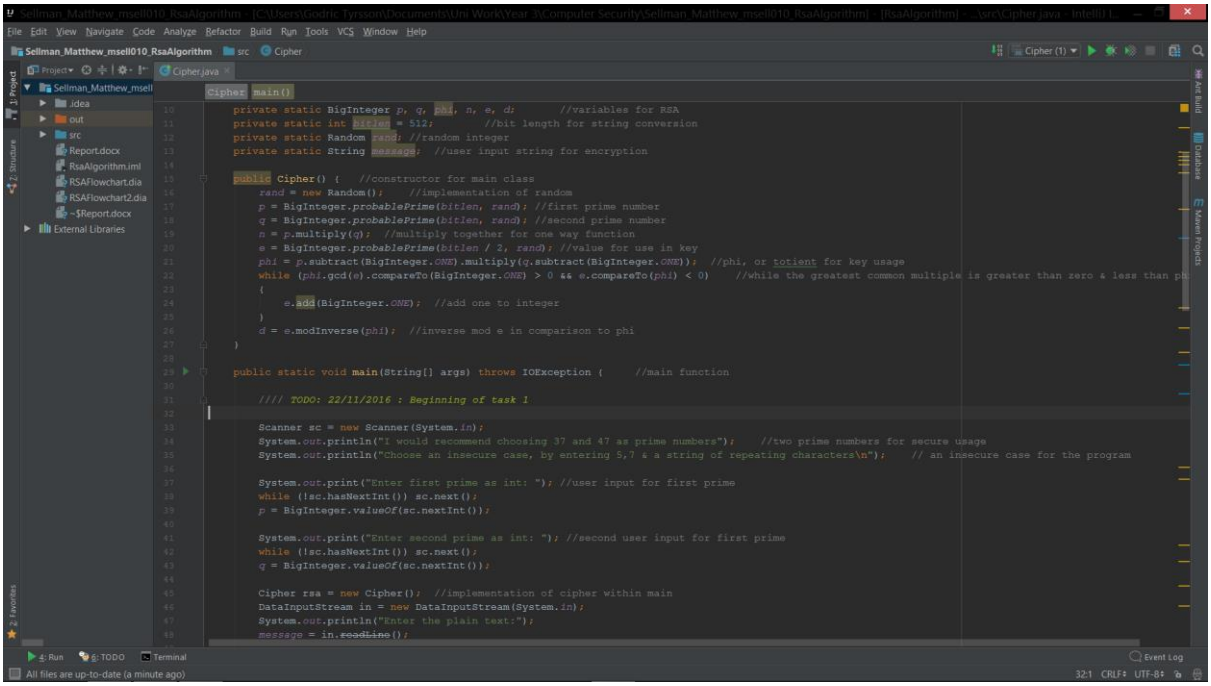


Below is the insecure case for task 2, the same as above applies, but this time, it is the server that is also in danger, since the composition of a public and private key pair could be used to forge messages that could ask for the public keys and therefore the private keys of individuals.



Matt Sellman (msell010)

Below is a sample of the code I have made, without any processes running.



(d) Discussion (including answers to any questions/problems in the Coursework assignment, your experience in attempt of the coursework, and full bibliography)

My main problem in the coursework was deciding how I would encrypt the message string given by the user. First I tried to convert the string into the respective ascii codes and try to encrypt those numbers. However, to do this I needed to convert the string to chars and then to numbers and then encrypt. This was too slow and clunky for what I needed. Therefore I decided to convert the string straight into bytecode and from there encrypt the numbers. This proved to be far more successful and much easier to integrate into my program.

Another issue that I had was that the encryption and decryption were the same no matter what input I put into the program. I fixed this by making sure the message variable being encrypted was the user input, rather than a different variable put in at an earlier time. This made the encryption different each time and the bug was removed from the program.

Another interesting problem I came across was the mathematics itself, the problem stemmed from the complication of the mathmatical integers used. When I was using ints, longs and doubles, I had to try to add, subtract and multiply values that did not correspond to one another, in order to bypass this I had to turn the value into Bigintegers, this allowed me much more flexibility when it came to computing the individual values given to be computed.

Matt Sellman (msell010)

Total hours spent:

Total Number of Hours Spent	60 hours
Hours Spent for Algorithm Design	5 hours
Hours Spent for Programming	20 hours
Hours Spent for Writing Report	15 hours
Hours Spent for Testing	20 hours
Note for the examiner (if any):	My optimal mark would be 75% or above

Bibliography: -

Code done with help from http://www.sanfoundry.com/java-program-implement-rsa-algorithm/

Everything else if my own work, as is most of the code.