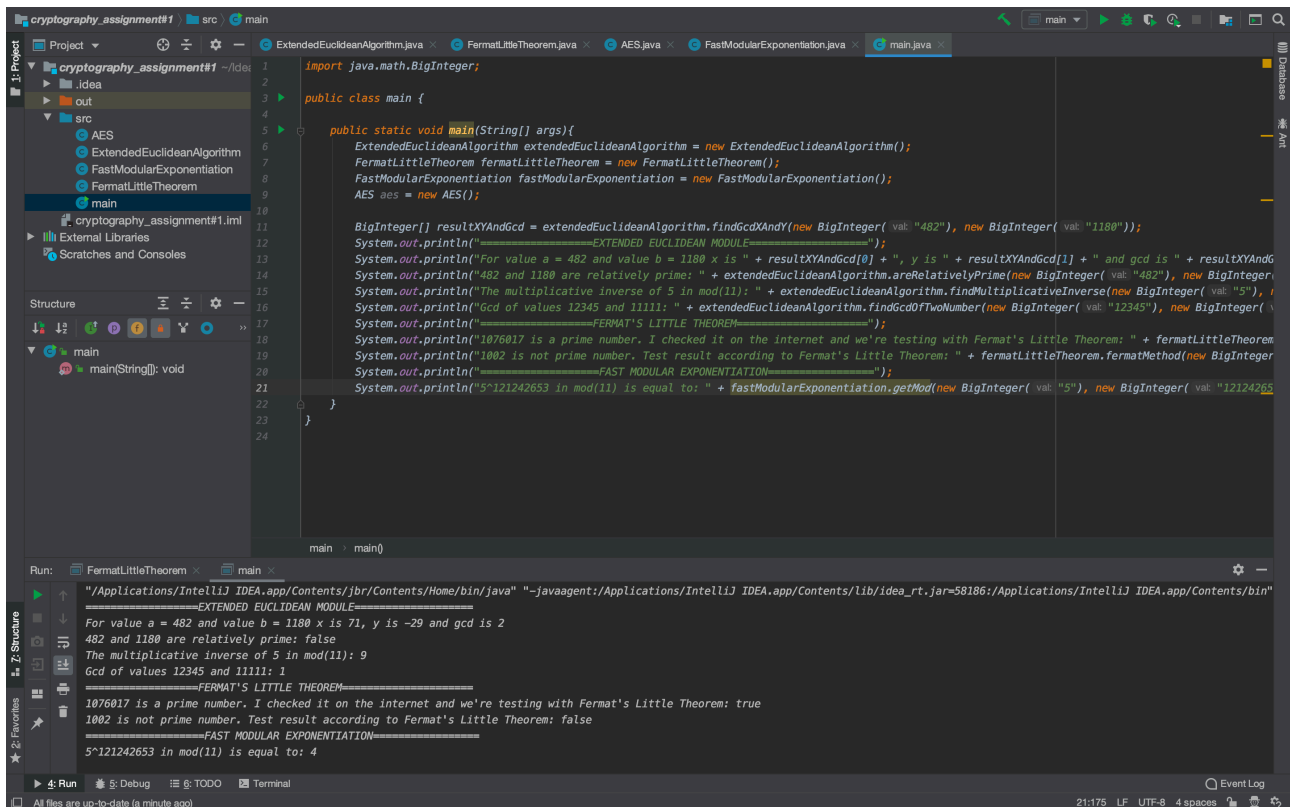


# Cryptography Homework #1 Report

I tested all the algorithm in main class. I'll share all the test values next part as screenshots.

- Used IntelliJ IDEA as an IDE.
- Used JAVA language for the implementation.
- Used BigInteger class in the implementation. This class in Java used for mathematical operation which involves ver big integer calculations that are outside the limit of all available primitive data types.
- ExtendedEuclideanAlgorithm class has 4 methods. findGcdXAndY method gives us gcd, x and y using values a and b according to  $ax + by = \text{gcd}(a,b)$  findGcdOfTwoNumber method gives us only gcd. We can check two numbers are prime using areRelativelyPrime method. We can find Multiplicative inverse of a number in modular arithmetic using findMultiplicativeInverse method.
- FermatLittleTheoremClass has two methods. These methods are generateRandomCoPrimeNumber and fermatMethod. The one of the important thing in here is iteration number. If iteration number is increased, we'll get more certain result. However, execution time can be increased.
- FastModularExponentiation class contains 4 methods. Thanks to this algorithm, we can compute  $a^b \% p$ .
- AES class implements AES algorithm for encryption and decryption operations. This class contains 3 important methods. RandomKeyGenerator generates random 256-bit key for every encryption&&decryption pair. Encrypt method uses initial vector because of cipher block chaining. From the initial vector, we created IvParameterSpec which is required when creating the cipher.
- In main class, I created objects in order to test algorithms calling the methods. I'll share my screenshots. In addition, I wrote comments in every classes so that can be understand easily.

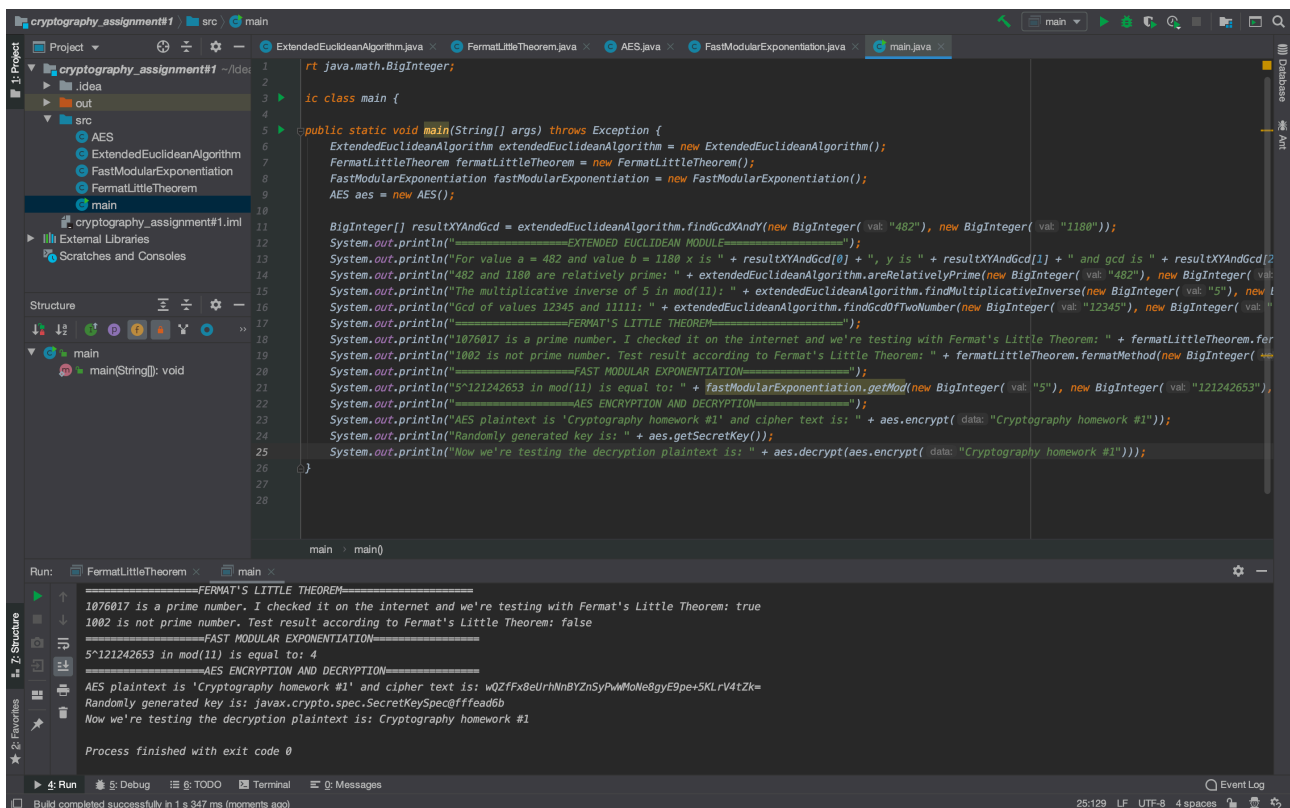
# SCREENSHOTS



```
1 import java.math.BigInteger;
2
3 public class main {
4
5     public static void main(String[] args){
6         ExtendedEuclideanAlgorithm extendedEuclideanAlgorithm = new ExtendedEuclideanAlgorithm();
7         FermatLittleTheorem fermatLittleTheorem = new FermatLittleTheorem();
8         FastModularExponentiation fastModularExponentiation = new FastModularExponentiation();
9         AES aes = new AES();
10
11         BigInteger[] resultXYAndGcd = extendedEuclideanAlgorithm.findGcdXY(new BigInteger( val: "482"), new BigInteger( val: "1180"));
12         System.out.println("=====EXTENDED EUCLIDEAN MODULE=====");
13         System.out.println("For value a = 482 and value b = 1180 x is " + resultXYAndGcd[0] + ", y is " + resultXYAndGcd[1] + " and gcd is " + resultXYAndGcd[2]);
14         System.out.println("482 and 1180 are relatively prime: " + extendedEuclideanAlgorithm.areRelativelyPrime(new BigInteger( val: "482"), new BigInteger( val: "1180")));
15         System.out.println("The multiplicative inverse of 5 in mod(11): " + extendedEuclideanAlgorithm.findMultiplicativeInverse(new BigInteger( val: "5"), new BigInteger( val: "11")));
16         System.out.println("Gcd of values 12345 and 11111: " + extendedEuclideanAlgorithm.findGcdOfTwoNumber(new BigInteger( val: "12345"), new BigInteger( val: "11111")));
17         System.out.println("=====FERMAT'S LITTLE THEOREM=====");
18         System.out.println("1076017 is a prime number. I checked it on the internet and we're testing with Fermat's Little Theorem: " + fermatLittleTheorem.isPrime(1076017));
19         System.out.println("1002 is not prime number. Test result according to Fermat's Little Theorem: " + fermatLittleTheorem.isPrime(1002));
20         System.out.println("=====FAST MODULAR EXPONENTIATION=====");
21         System.out.println("5^121242653 in mod(11) is equal to: " + fastModularExponentiation.getMod(new BigInteger( val: "5"), new BigInteger( val: "121242653")));
22     }
23 }
24
```

Run: FermatLittleTheorem × main ×

```
"/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" "-javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58186:/Applications/IntelliJ IDEA.app/Contents/bin"
=====EXTENDED EUCLIDEAN MODULE=====
For value a = 482 and value b = 1180 x is 71, y is -29 and gcd is 2
482 and 1180 are relatively prime: false
The multiplicative inverse of 5 in mod(11): 9
Gcd of values 12345 and 11111: 1
=====FERMAT'S LITTLE THEOREM=====
1076017 is a prime number. I checked it on the internet and we're testing with Fermat's Little Theorem: true
1002 is not prime number. Test result according to Fermat's Little Theorem: false
=====FAST MODULAR EXPONENTIATION=====
5^121242653 in mod(11) is equal to: 4
```



```
1 import java.math.BigInteger;
2
3 public class main {
4
5     public static void main(String[] args) throws Exception {
6         ExtendedEuclideanAlgorithm extendedEuclideanAlgorithm = new ExtendedEuclideanAlgorithm();
7         FermatLittleTheorem fermatLittleTheorem = new FermatLittleTheorem();
8         FastModularExponentiation fastModularExponentiation = new FastModularExponentiation();
9         AES aes = new AES();
10
11         BigInteger[] resultXYAndGcd = extendedEuclideanAlgorithm.findGcdXY(new BigInteger( val: "482"), new BigInteger( val: "1180"));
12         System.out.println("=====EXTENDED EUCLIDEAN MODULE=====");
13         System.out.println("For value a = 482 and value b = 1180 x is " + resultXYAndGcd[0] + ", y is " + resultXYAndGcd[1] + " and gcd is " + resultXYAndGcd[2]);
14         System.out.println("482 and 1180 are relatively prime: " + extendedEuclideanAlgorithm.areRelativelyPrime(new BigInteger( val: "482"), new BigInteger( val: "1180")));
15         System.out.println("The multiplicative inverse of 5 in mod(11): " + extendedEuclideanAlgorithm.findMultiplicativeInverse(new BigInteger( val: "5"), new BigInteger( val: "11")));
16         System.out.println("Gcd of values 12345 and 11111: " + extendedEuclideanAlgorithm.findGcdOfTwoNumber(new BigInteger( val: "12345"), new BigInteger( val: "11111")));
17         System.out.println("=====FERMAT'S LITTLE THEOREM=====");
18         System.out.println("1076017 is a prime number. I checked it on the internet and we're testing with Fermat's Little Theorem: " + fermatLittleTheorem.isPrime(1076017));
19         System.out.println("1002 is not prime number. Test result according to Fermat's Little Theorem: " + fermatLittleTheorem.isPrime(1002));
20         System.out.println("=====FAST MODULAR EXPONENTIATION=====");
21         System.out.println("5^121242653 in mod(11) is equal to: " + fastModularExponentiation.getMod(new BigInteger( val: "5"), new BigInteger( val: "121242653")));
22         System.out.println("=====AES ENCRYPTION AND DECRYPTION=====");
23         System.out.println("AES plaintext is 'Cryptography homework #1' and cipher text is: " + aes.encrypt( data: "Cryptography homework #1"));
24         System.out.println("Randomly generated key is: " + aes.getSecretKey());
25         System.out.println("Now we're testing the decryption plaintext is: " + aes.decrypt(aes.encrypt( data: "Cryptography homework #1")));
26     }
27 }
28
```

Run: FermatLittleTheorem × main ×

```
=====FERMAT'S LITTLE THEOREM=====
1076017 is a prime number. I checked it on the internet and we're testing with Fermat's Little Theorem: true
1002 is not prime number. Test result according to Fermat's Little Theorem: false
=====FAST MODULAR EXPONENTIATION=====
5^121242653 in mod(11) is equal to: 4
=====AES ENCRYPTION AND DECRYPTION=====
AES plaintext is 'Cryptography homework #1' and cipher text is: wQZfX8eUrhNnBV2nSyPwM0Me8gyE9pe+5KLrV4tZk=
Randomly generated key is: javax.crypto.spec.SecretKeySpec$ffead6b
Now we're testing the decryption plaintext is: Cryptography homework #1
Process finished with exit code 0
```