Java Programming 1

PROFESSOR: CÂI FILIAULT

Topics for this week

String manipulation

Typecasting

The && and || Operator

Introduction to OOP

SpyTek

NEW EMAIL & NEW CLIENT

Email

Anonymous

Hello,

I would like for you to build a program that is able to encrypt and decrypt email messages. I would like you to use the Caesar Cipher for the encryption. The software should ask the user if they wish to encrypt or decrypt a message. The software should then ask the user to input the message. The user then needs to know and enter the correct key to decrypt the message

-Thanks,

Anonymous

Plan of Action

Encrypting and decrypting text sounds like a useful task that we may be able to reuse so we should build a function that is able to do that.

The function we build should be able to:

- Decrypt or encrypt text on request:
- Take text, key and mode as a parameter.

We should then build a piece of software that is able to utilize this function:

- The software should ask the user whether they want to encrypt or decrypt a message
- The software should then prompt the user to enter the message
- The software should then ask the user to enter a key to encrypt or decrypt the message with
- Finally the software should output the encrypted/decrypted text to the screen.

Key Information

Before we can continue:

- What is encrypting and decrypting?
- What is a Caesar Cipher
- How does a Caesar Cipher work?

Encryption

- Encryption is the process of encoding a message so that only the people the message is intended for can read it.
- Encryption is the process of taking regular text and transforming it into ciphertext
- Ciphertext can only be read if the cipher text is decrypted back into regular text.
- Decryption is the process of taking cipher text and transforming it back into regular text.

Caesar Cipher

Caesar Cipher:

- One of the earliest ciphers ever used and one of the simplest.
- The cipher is named after Julius Caesar, who's army originally used the cipher
- The Caesar cipher is what's known as a substitution cipher.
- A substitution cipher is a cipher in which each letter in the alphabet is substituted for another letter in the alphabet.
- The entire message is based upon a key.
- The key is how many letters down the alphabet we shift
- If I choose the letter a and the key 5

Caesar Cipher

Caesar Cipher:

- The letter “a” shifted 5 letters down the alphabet is the letter “f”
- We should follow this same method for the entire message
- The opposite is true for the cipher (to decrypt the message shift back 5 letters)
- Decipher the following: “FGQJ BFX N JWJ N XFB JQGF”

Caesar Cipher

Cipher Text Translated:

The famous palindrome said by Napoleon Bonaparte when he was exiled to Elba. “Able was I ere I saw Elba”

Book

Suggested Reading:

The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography

- ISBN-10: 0385495323
- ISBN-13: 978-0385495325

This book is a great resource for learning many different cipher techniques and algorithms

It explores some of the most ancient and cryptic cipher algorithms to ever exist

It gives sample exercises so you can try to decipher each encoded text.

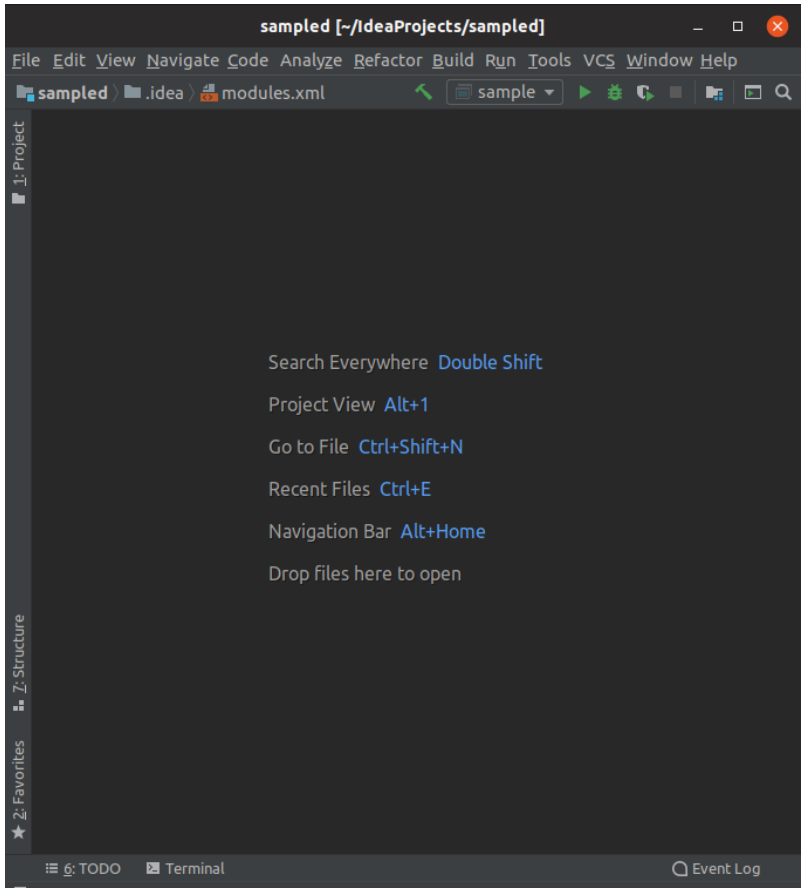
It also provides some very useful statistics for breaking any home brew encryption algorithm

From the most used letters in the English language to the most used words of a given length

Creating a new project

Start by opening IntelliJ and selecting the following:

File> New> Project



New Project

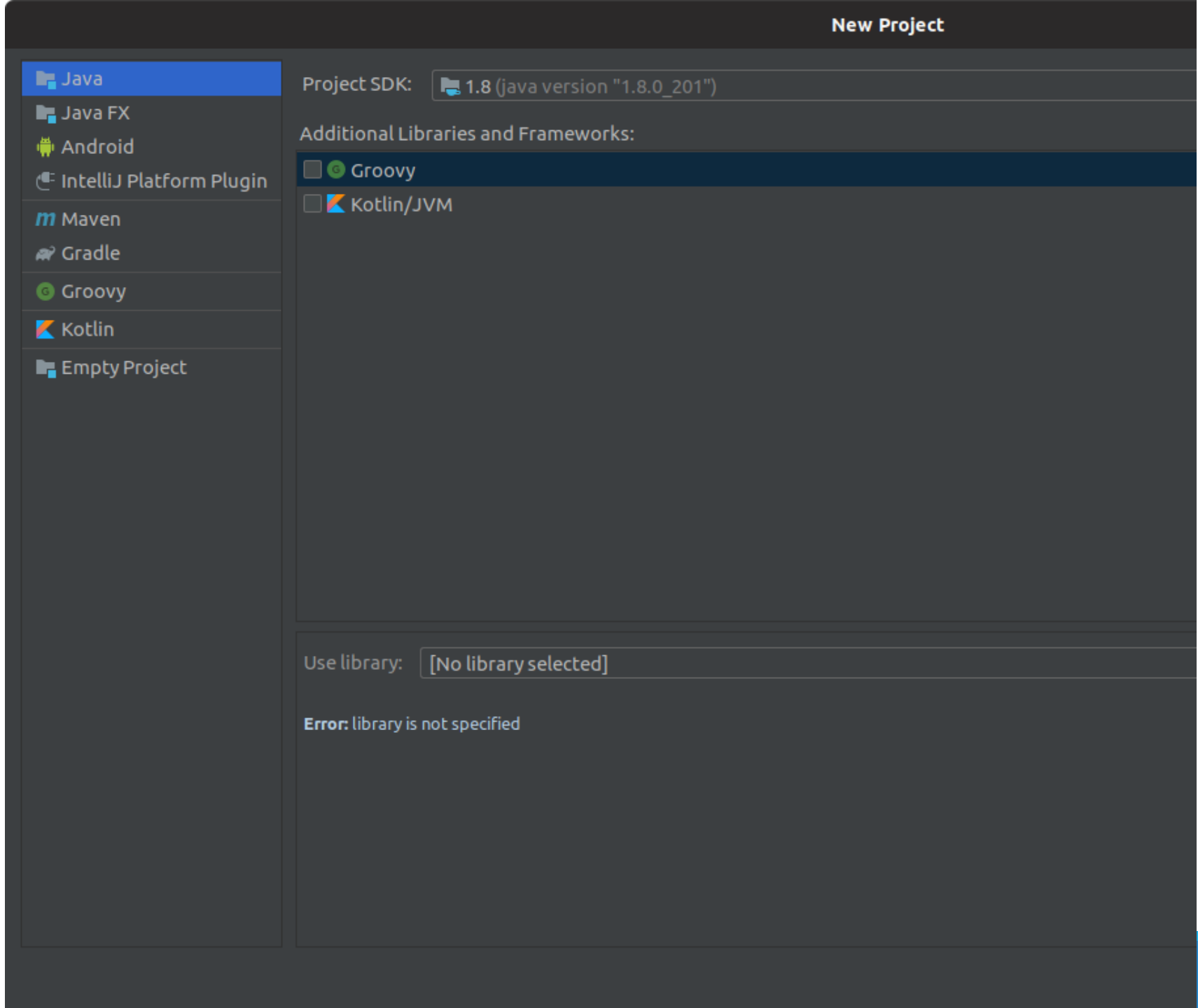
You will now see a dialog box appear *(It may be a little different than the one that I have shown)*

Starting on the left we will select a new Java Project

We will now want to select the project SDK of 1.8

SDK stands for Software Development Kit and it dictates which version of Java we are using.

At this point we can hit the next button twice



New Project

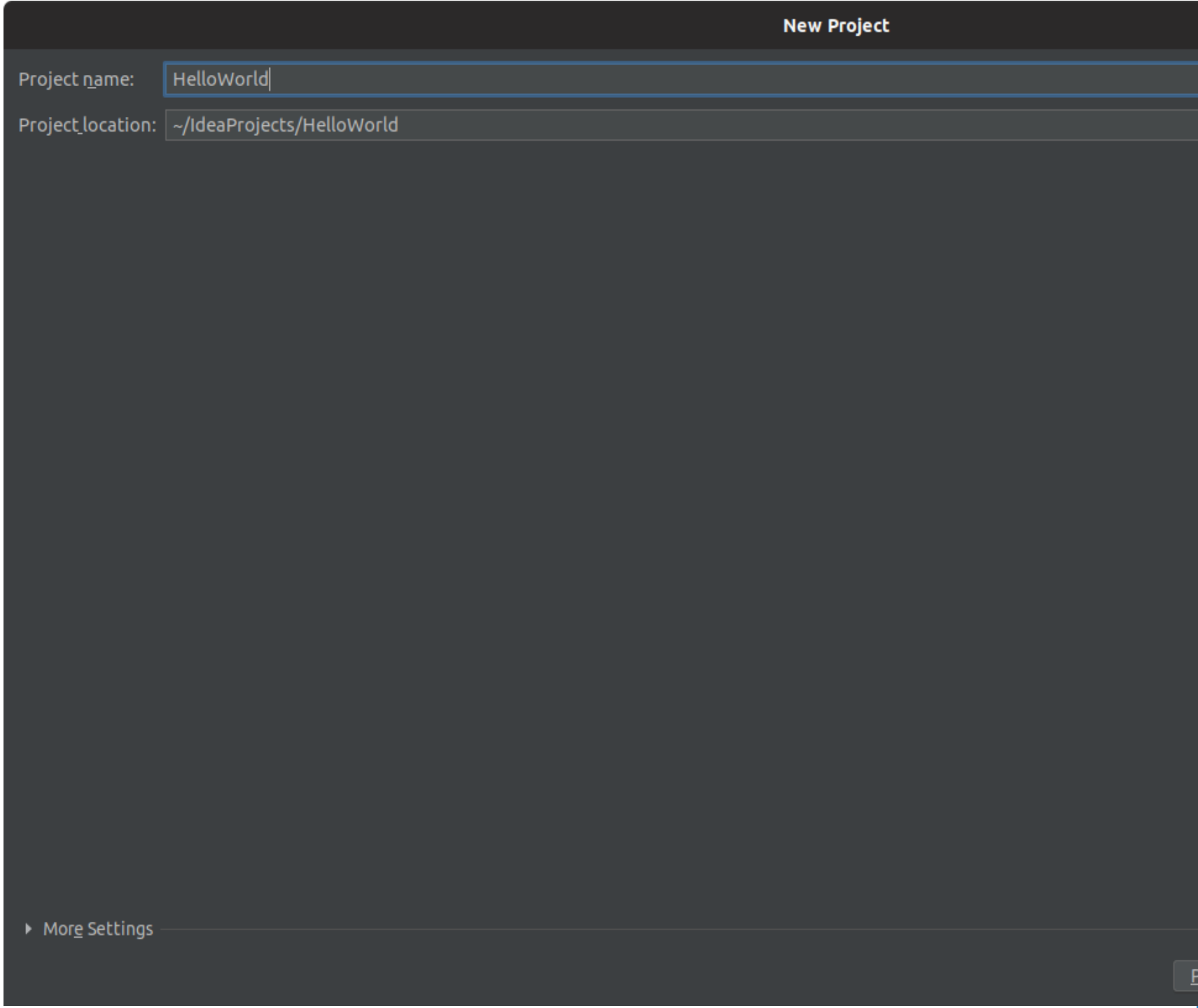
Give the project a name of:

CaesarCipher

At this time you can change the project location to any convenient place you would like

Tip:

Keeping your programming projects on a flash/jump drive is perfectly fine. However you will want to avoid from directly working on the drive.

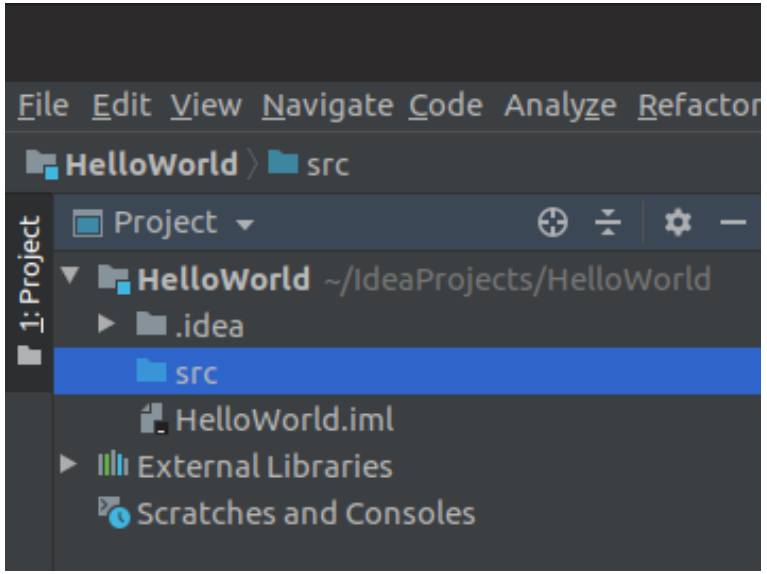


Creating a new project

You will now see in the top left-hand corner of your IDE a package explorer.

Within the package explorer you should see the **CaesarCipher** project.

Expand the **CaesarCipher** project to see a folder named **src**.



Creating a new class file

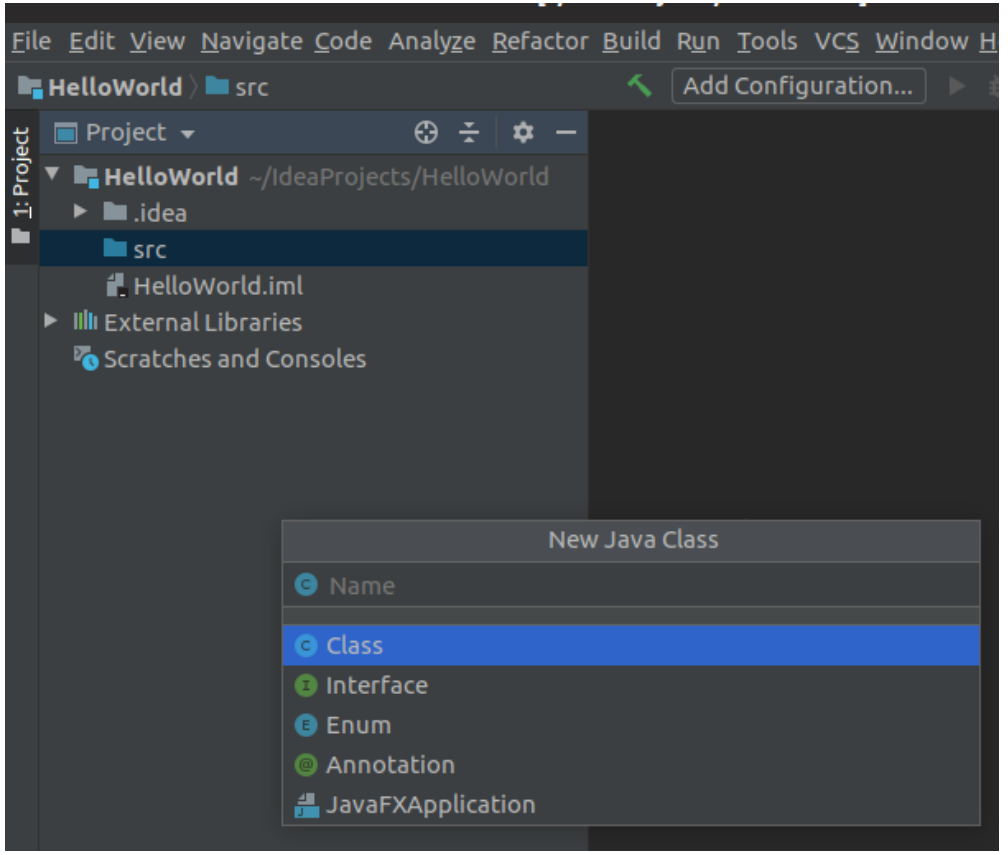
Next, we will need to create a new class file.

Right click on the **src** folder and select:

New> Java Class

Creating a new class is like creating a new program

Name the class **CaesarCipher** and hit **enter**



Writing the program

```
2  public class CesarCipher {
3
4      //build a function that is able to encrypt and decrypt text
5
6      public static void main(String[] args) {
7
8      // The software should ask the user whether they want to encrypt or decrypt a message
9      // The software should then prompt the user to enter the message
10     // The software should then ask the user to enter a key to encrypt or decrypt the message with
11     // Finally the software should output the encrypted/decrypted text to the screen.
12         
13     }
14
15 }
```

Writing the program

- Let's start to create the program by declaring a function within the class declaration.
- The function will take 3 parameters
  - The text
  - The mode in which we wish to run the function (encryption or decryption)
- The key used during the encryption or decryption

```
2  public class CesarCipher {
3
4      //build a function that is able to encrypt and decrypt text
5      public static String cipher(String text, int mode , int key){
6
7          return "";
8      }
9
10     public static void main(String[] args) {
11
12     // The software should ask the user whether they want to encrypt or decrypt a message
13     // The software should then prompt the user to enter the message
14     // The software should then ask the user to enter a key to encrypt or decrypt the message with
15     // Finally the software should output the encrypted/decrypted text to the screen.
16
17     }
18
19 }
```

Writing the program

- Because we asked the user to input a number on line 17, we need to clear the input buffer.
- When a user enters a number into the scanner the user follows up the number entry with the enter key
- The enter key stores a carriage return that needs to be escaped.
- To escape a carriage return we can use the input.nextLine() command which will read input from the scanner class until it encounter the carriage return.
- Because a carriage return is still residing in the buffer it will grab that carriage return and continue in the code.
- the user will never even know they were prompted for a string

```
15      System.out.println("Welcome to the top secrect encoding system");
16      System.out.println("Would you like to encrypt(1) or decrypt(2) a message: (Enter 1 or 2)");
17      int choice = input.nextInt();
18      input.nextLine();
19      // The software should then prompt the user to enter the message
20      System.out.println("Enter the message: ");
21      String message = input.nextLine();
22      // The software should then ask the user to enter a key to encrypt or decrypt the message with
23      System.out.println("Enter they key: (1 - 26)");
24      int key = input.nextInt();
25      // Finally the software should output the encrypted/decrypted text to the screen.
26
27      }
28
29  }
30
```

Writing the program

- Line number 20 and 21 asks the user to enter a message. Please keep in mind this message uses a carriage return delimiter. Once the enter key is pressed the message is complete. If you want to insert large amounts of text you can do so just don't hit the enter key.
- We store the message the user entered in a variable named message
- We next ask the user to enter a key (1 – 26)

```
15      System.out.println("Welcome to the top secrect encoding system");
16      System.out.println("Would you like to encrypt(1) or decrypt(2) a message: (Enter 1 or 2)");
17      int choice = input.nextInt();
18      input.nextLine();
19      // The software should then prompt the user to enter the message
20      System.out.println("Enter the message: ");
21      String message = input.nextLine();
22      // The software should then ask the user to enter a key to encrypt or decrypt the message with
23      System.out.println("Enter they key: (1 - 26)");
24      int key = input.nextInt();
25      // Finally the software should output the encrypted/decrypted text to the screen.
26
27      }
28
29  }
30
```

Writing the program

- The key recorded on line 23 and 24 will be used by the cipher function to determine how many letters down the alphabet the cipher is to shift.

```
15      System.out.println("Welcome to the top secrect encoding system");
16      System.out.println("Would you like to encrypt(1) or decrypt(2) a message: (Enter 1 or 2)");
17      int choice = input.nextInt();
18      input.nextLine();
19      // The software should then prompt the user to enter the message
20      System.out.println("Enter the message: ");
21      String message = input.nextLine();
22      // The software should then ask the user to enter a key to encrypt or decrypt the message with
23      System.out.println("Enter they key: (1 - 26)");
24      int key = input.nextInt();
25      // Finally the software should output the encrypted/decrypted text to the screen.
26
27      }
28
29  }
30
```

Writing the program

Important concepts:

We are going to shift each letter in a message down the alphabet by a certain number of characters (the key).

This is normally a complicated process but can be completed easily with a few concepts understood:

- String tokenization
- Type casting
- String manipulation
- The ASCII table

Writing the program

String Tokenization:

- String tokenization is the process of breaking a string into multiple sub strings using some form of delimiter.
- a delimiter is something that separates values within a data set.
  - The delimiter for a sentence is a period
  - The delimiter for a paragraph is a carriage return
  - The delimiter for a word is a space
- In a csv document the delimiter is a comma
- String tokenization is the process of taking a string and using a delimiter like the one posted above for separating values.
- A string tokenizer can extract all the words in a sentence, sentences in a paragraph, or paragraphs inside a block of text.
- This is a very useful tool

Writing the program

In java string tokenization is possible through creating a new tokenizer object

StringTokenizer tokenizerName = new StringTokenizer(“string”, “delimiter”);

After we have created the StringTokenizer we can then utilize any of the functions offered by the object. The two methods we will be using today are as follows:

- tokenizerName.hasMoreTokens()
- tokenizerName.nextToken()

There are many other methods that exist within the StringTokenizer class.

You can explore the methods in your text or on the java website.

Writing the program

Typecasting:

- Typcasting is a concept that allows us to convert a value that is one data type to another data type
- Imagine we had a character that we wanted to store inside of an integer
- In some cases this is possible
- In Java typecasting can be classified in two different types
  - Implicit casting
  - Explicit casting

Writing the program

Implicit casting:

- Is automatically done in java without any extra code from the programmer
- This is the process of moving forward. (an int can store inside a double)

Writing the program

Explicit casting:

- Requires the programmer to put the type the user wishes to cast into in brackets before the value (int) etc.
- Explicit casting is needed when going backwards down the chart
- Let's do a quick example of this:

Writing the program

String Manipulation:

- StringTokenizer is not the only form of string manipulation tools that exist.
- We are also able to do all the following tasks:
  - substring(begin index, end index) grabs a substring of the calling string
  - toUpperCase() converts the string to uppercase
  - toLowerCase() converts the string to lowercase
  - replace(old characters, new characters) replaces all old characters with the new ones
  - charAt(index location) returns the character at the index location
  - .length() returns the length of the string
  - toCharacterArray()
- More Examples

Writing the program

ASCII (American Standard Code for Information Interchange)

- Earlier in the year we discussed ASCII and how ASCII was a table that is used at a core level of the computer to know:
  - What characters are stored in memory
  - What character the user entered in the keyboard
- ASCII is also used by java.
- Let's look at the ASCII table and how java utilizes it.

Note:

The following slides are another take on how this program can be completed. The method used in class are different but the concepts used are the same.

In class we break the cipher method down into encypt() and decrypt() methods. Please refer to the video and posted code for the other method.

Writing the program

- Using our newfound knowledge we need to do the following:
- Line number 14 ensures that the loop will run until we have no more tokens to convert
- Line number 16 grabs the next token in the machine so that we may process it.
- If the mode was set to 1, we want to encrypt the text
- We can use a for loop to look at each character inside of the string and then shift that character through the alphabet.
- We store the shifted character inside of the cipherWord variable.

```
 8    public static String cipher(String text, int mode , int key){
 9        String cipherText = "";
10        String cipherWord = "";
11        String token = "";
12        int amount;
13        StringTokenizer machine = new StringTokenizer(text, " ");
14        while(machine.hasMoreTokens()){
15            cipherWord = "";
16            token = machine.nextToken();
17            if(mode == 1){
18                for(int i = 0; i < token.length(); i++){
19                    if(((int) token.charAt(i) + key) > 122){
20                        amount = ((int) token.charAt(i) + key) - 122;
21                        cipherWord += (char) (96 + amount);
22                    }else{
23                        cipherWord += (char)((int) token.charAt(i) + key);
24                    }
25                }
26                cipherText += cipherWord + " ";
27            }
28        }
29        return "";
30    }
```

Writing the program

- The only trick to this process is understanding that the key can be a value from 1 – 26.
- Keeping this in mind we know that the ASCII table spans from 96 – 122
- (for lower case values)
- We must build a formula that will be able to cycle the user back through that range if they pass it in either direction

```
8    public static String cipher(String text, int mode , int key){
9        String cipherText = "";
10       String cipherWord = "";
11       String token = "";
12       int amount;
13       StringTokenizer machine = new StringTokenizer(text, " ");
14       while(machine.hasMoreTokens()){
15           cipherWord = "";
16           token = machine.nextToken();
17           if(mode == 1){
18               for(int i = 0; i < token.length(); i++){
19                   if(((int) token.charAt(i) + key) > 122){
20                       amount = ((int) token.charAt(i) + key) - 122;
21                       cipherWord += (char) (96 + amount);
22                   }else{
23                       cipherWord += (char)((int) token.charAt(i) + key);
24                   }
25               }
26               cipherText += cipherWord + " ";
27           }
28       }
29       return "";
30   }
```

Writing the program

- Line number 27 checks that the user entered mode 2, in which case the program will be translating the value from cipher text to plain text.
- The formula used is like the one seen in the initial for loop that ciphers the text.
- The only difference is we are subtracting instead of adding to the text because we wish to move backwards in the alphabet.
- Finally we save each cipherWord to the cipher text with a space.
- Please note we have not accounted for any special characters
- Once finished we return the cipherText
- Either encrypted or decrypted

```
14          while(machine.hasMoreTokens()){
15              cipherWord = "";
16              token = machine.nextToken();
17              if(mode == 1){
18                  for(int i = 0; i < token.length(); i++){
19                      if(((int) token.charAt(i) + key) > 122){
20                          amount = ((int) token.charAt(i) + key) - 122;
21                          cipherWord += (char) (96 + amount);
22                      }else{
23                          cipherWord += (char)((int) token.charAt(i) + key);
24                      }
25                  }
26              }
27              else if(mode == 2){
28                      for(int i = 0; i < token.length(); i++){
29                          if(((int) token.charAt(i) - key) < 97){
30                              amount = 96 - ((int) token.charAt(i) - key);
31                              cipherWord += (char) (122 - amount);
32                          }else{
33                              cipherWord += (char)(((int) token.charAt(i)) - key)
34                          }
35                      }
36              }
37              cipherText += cipherWord + " ";
38          }
39
40          return cipherText;
41      }
```

Writing the program

- The final step of the software is using the function we just created.
- Line 57 shows the function used to transform the message using the choice the user selected and the key the user selected
- Finally if the key was 1 output the encrypted message otherwise output the decrypted message

```
54      System.out.println("Enter they key: (1 - 26)");
55      int key = input.nextInt();
56      // Finally the software should output the encrypted/decrypted text to the screen.
57      String transformedText = cipher(message, choice, key);
58      if(key == 1){
59          System.out.println("The encrypted message is: " + transformedText);
60      }
61      else{
62          System.out.println("The decrypted message is: " + transformedText);
63      }
64
65      input.close();
66      }
67
68  }
69
```

Test your software

Take a moment to test your software

Homework

Read Chapter 7 of your textbook

Next Week

Topics for next week:

OOP concepts

Building simple objects

Manipulating simple objects