

CSC110AB

MinilabFileIO

Background: In Java, chars are stored as ints. chars and ints are mapped to each other using the ANSI table that can be shown in Textpad (View -> Clip Library). Java actually uses UNICODE, which handles many more characters, but the ANSI table is a subset of UNICODE and contains the original ASCII table (up to char #127) as well as ANSI (up to char #255).

Therefore, chars and ints can be typecast to each other:

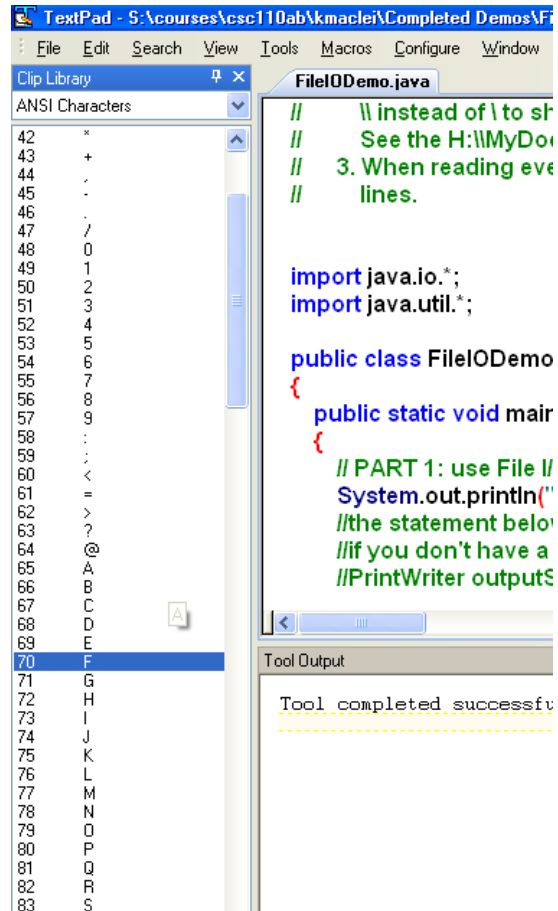
(int)'F' is 70
(char)70 is 'F'

You can even add chars and ints together. The result is an int:

'F' + 5 is 75

Putting these concepts together, you can generate the char 5 larger than 'F' by:

(char)('F' + 5) is 'K'



The screenshot shows the TextPad application with the 'Clip Library' window open, displaying the 'ANSI Characters' table. The table lists characters from 42 to 83, with 'F' highlighted at row 70. The main window shows the code for 'FileIODemo.java', which includes imports for java.io.* and java.util.*, a public class FileIODemo, and a public static void main method. The main method contains comments and code for printing output. The 'Tool Output' window at the bottom shows the message 'Tool completed successfully'.

```
// \\ instead of \ to sl
// See the H:\\MyDoc
// 3. When reading eve
// lines.

import java.io.*;
import java.util.*;

public class FileIODemo
{
    public static void main
    {
        // PART 1: use File I/
        System.out.println("
        //the statement below
        //if you don't have a
        //PrintWriter outputS
    }
}
```

Tool Output

Tool completed successfully

This Minilab: You are to write a program called Encrypt.java. It will “encrypt” the contents of a file by writing its contents to another file, but changing all of its characters by a certain amount. Your program should:

- Ask the user to input the name of the input file, name of the output file, and key (integer)
- Open the input file for reading and the output file for writing (but not append)
- Read every line of the input file; every time you read a line, start a new (empty) String to hold the its encryption. Then go through all its chars one at a time and convert them to a different char by adding or subtracting the key. After you have created an “encrypted” character, put it at the end of your encrypted String. After you have finished processing the original line, write the “encrypted” String to the output file.
- When all lines from the original file have been processed, close both files.

The results:

If the user enters:

```
infile.txt  
codefile.txt  
4
```

Then your program should write the “encrypted” contents of infile.txt into codefile.txt (without changing the contents of infile.txt). If you ran your program again, with the user entering:

```
codefile.txt  
thirdfile.txt  
-4
```

Then the contents of thirdfile.txt should be “unencrypted” and exactly match the contents of infile.txt.

Note: It is possible that the key could be large enough that it could go past the ends of the conversion tables and it is likely that your program would crash. You could avoid this problem by incorporating a % operation into the addition and subtraction; this would force it to go to the other end of the table if the result got too big or too small. However, the purpose of this Minilab is to do file input and output, so you don’t have to worry about this problem occurring.

Please submit: your Encrypt.java via Canvas