Java 6 is almost ready to release, and includes an API for implementing command-line programs that go beyond the basic functionality available in prior examples by using `System.in` and `System.out` . The `java.io.Console` class is a simple extension of the command-line, and an instance is available through the `System.console()` method. Note that any operations through this API should consider the possibility that it can return null:

```
Console console = System.console();
if(console != null) {
 // [...]
}
```

Using the new console API to output formatted content is much simpler thanks to the integration with the formatter API. You can call either `console.format(String fmt, Object... args)` or `console.printf(String fmt, Object... args)` to print formatted data to the UI:

```
String formatString = "%1$4s %2$10s %3$10s%n";
console.printf(formatString, "Idx", "A", "B");
console.printf(formatString, "1", "10", "100");
console.printf(formatString, "2", "20", "200");
console.printf(formatString, "3", "30", "300");
console.printf(formatString, "4", "40", "400");
```

The output of this block of code would look something like this:

```
 Idx          A          B
   1         10        100
   2         20        200
   3         30        300
   4         40        400
```

You can read the Formatter API for more detail on how to use format strings.

The key methods for handling user-input are the `readLine()` methods. There are two; one to simply read data, and another to provide formatted text in conjunction with the input for the user. Here is an example of prompting for input using the second method:

```
String name = console.readLine("[Please Provide
```

This method (`readLine(String fmt, Object... args)`) is a combination of formatted output with user input. The method uses the `java.util.Formatter` syntax for the first argument, and applies the results of the args array as it would to a normal call to `Formatter.format` . The console input is then accepted at the end of the formatted output (including any new-lines embedded in the format string).

There is also, for the first time, support for accepting passwords through the console. Passwords can be requested using the `readPassword(String fmt, Object... args)` method (or the corresponding no-arg version). There are two primary differences with the password reading methods: 1. they do not echo the user keypresses back to the console and 2. the input is returned as a char[] as opposed to a String.

Retrieving a password via the console is very similar:

```
char[] passdata = console.readPassword("[Please
if(passdata != null) {
        for(int i=0; i<passdata.length; i++) {
         // validate password.
        }
}
```

The idea behind the character array (as opposed to a String or other container) is that a primitive array is one of the few things in Java that can be deterministically cleared from memory (there-by minimizing the time that the data is active in the application memory). This is mentioned in the documentation for the console class, and they provide this example on how to clear the data out of memory in a timely fashion:

```
char[] passdata = console.readPassword("[Please
if(passdata != null) {
        for(int i=0; i<passdata.length; i++) {
         // validate password.
        }
}
Arrays.fill(passdata, ' '); // re-sets all data
```

You can hook the Scanner API up to the console for reading input as well (if readLine doesn't suffice); you simply need to request a `reader()` from the console:

```
Scanner scanner = new Scanner(console.reader());
int value = 0;
while(value != 99)
{
        console.printf("Please input a value bet
        value = scanner.nextInt();
}
```

This example basically keeps requesting an integer until the user inputs 99. Now, this isn't exactly a robust implementation (it will error-out when faced with any other value aside from an int), but it should give you some idea of the flow of the reader.

Likewise, if you don't want to deal with the formatter-centric output methods, you can obtain a print writer for your own use:

```
PrintWriter out = console.writer();
out.println("Test regular writing!");
```

Here is a full source example:

```
import java.io.Console;
import java.io.PrintWriter;
import java.util.Scanner;

public class Main
{
        public static void main(String[] args)
        {
                Console console = System.console
                if(console != null)
                {
                        String formatString = "
                        console.printf(formatSt
                        console.printf(formatSt
                        console.printf(formatSt
                        console.printf(formatSt
                        console.printf(formatSt
                        String name = console.r
                        char[] passdata = conso
                        Scanner scanner = new S
                        int value = 0;
                        while(value != 99)
                        {
                                console.printf("
                                value = scanner.
                        }

                        PrintWriter out = conso
                        out.println("Test regul
                }
                else
                {
                        throw new RuntimeExcept
                }
        }
}
```