printf
Printing to the Screen

This is easy, using the System Output Stream `out`. There are several related methods:

- `System.out.print( String s);`
- `System.out.println( String s);`
- `System.out.printf(String s, args );`

`println` appends a newline character '\n' to the string. Both print and println allow you to print objects of any class by calling the object's `toString()` method. All objects have a `toString()` method but the default method just returns a String version of the object’s location in memory, which is fairly useless. Overwrite `toString()` if you want a useful method.
One situation that arises frequently in programming is that you need to print a statement about data that is contained in variables. For example you might be walking around a grid and need to print a statement

“I am at row 3, column 5.”

where the 3 and the 5 are in variables $r$ and $c$. There are lots of awkward ways to do this, but Java gives us one elegant way: the printf( ) statement.
printf( ) takes as it arguments a *format string* and some values. A format string is just like any other string only it is allowed to have *placeholders*. There are three different placeholders: %d for integer values, %f for floats, and %s for strings. There must be one value supplied for each placeholder in the format string.
For example, the row-column statement from the previous slide could be printed from

printf( "I am at row %d, column %d.\n", r, c);

If you want to end the line like a println( ) statement would just include \n in the format string:

printf( "I am at row %d, column %d. \n", r, c);
A single printf statement can use several kinds of placeholders. A value can be an expression that computes a value.

For example if we have variables

```java
String richGuy = "Bill Gates";
String poorGuy = "bob"
int money = 500
```

then

```java
printf( "Tell %s to give %s $%d." , richGuy, poorGuy, 10*money );
```

will print the statement

“Tell Bill Gates to give bob $5000."
The placeholders can also take *fieldwidths*:  
\%5d says to format the int using at least 5 spaces, padded with blanks on the left  
\%-5d is the same, only padded on the right.  
\%5s pads a string with blanks to take at least 5 spaces.  
\%7.3f says to pad the float to 7 spaces, using exactly 3 decimal places, as in 123.456

The main use for fieldwidths is getting data to come out in nice columns for tables.
For example, one of the first example programs we looked at printed a table of prime numbers:

```java
for (int i = first; i <= last; i++) {
    if (isPrime(i)) {
        System.out.printf( "%6d", i );
        lineCount += 1;
        if (lineCount == LINE_SIZE) {
            System.out.println();
            lineCount = 0;
        }
    }
}
```

Because the primes are printed with format “%6d” they come in columns 6 spaces wide: