**Formatted Output**

The following code segment uses the method Math.pow to display the first ten powers of 2 on separate lines in a text area named output:

int expo, data;

expo = 1;

while (expo <= 10){

data = Math.pow (2, expo);

output.append (data + "\n");  
    expo++;

}

The output this code produces would be

2

4

8

16

32

64

128

256

512

1024

Consider the ***justification*** of this list. These numbers are left justified, that is, aligned to the left. Now suppose we would like them to be right justified, that is, aligned to the right. To accomplish this, we must somehow insert the correct number of blank spaces as padding in front of each number. The largest number in our data set occupies four columns of output area. Thus, we would want three blanks inserted in front of single-digit numbers, two blanks in front of two-digit numbers, and one blank in front of three-digit numbers, as shown in the following output:

2

4

8

16

32

64

128

256

512

1024

The BreezyGUI package provides a Format class that allows the programmer to format output that is left justified, right justified, or centered within a given number of columns. The Format class defines four class methods to format data as shown in the table:

| **Method** | **What it does** |
| --- | --- |
| String justify  (char alignment,  String data, int width) | Returns the string data aligned to the left ('l'), center ('c'), or right ('r') in the columns specified by width. |
| String justify  (char alignment,  char data, int width) | Returns the character data aligned to the left ('l'), center ('c'), or right ('r') in the columns specified by width. |
| String justify  (char alignment,  long data, int width) | Returns the string representation of data aligned to the left, center, or right in the columns specified by width. |
| String justify  (char alignment,  double data, int width,  int precision) | Returns the string representation of data aligned to the left, center, or right in the columns specified by width.precision specifies the number of places to the right of the decimal point. |

Some example of the justify method are shown in the following table:

| **Example** | **What it does** |
| --- | --- |
| Format.justify('l', "Hello", 7) | Returns the string "Hello ". |
| Format.justify('c', "Hello", 7) | Returns the string " Hello ". |
| Format.justify('r', "Hello", 7) | Returns the string " Hello". |
| Format.justify('r', "Hi", 7) | Returns the string " Hi". |
| Format.justify('r', 'H', 7) | Returns the string " H". |
| Format.justify('r', 237, 7) | Returns the string " 237". |
| Format.justify('r', 2.37, 7, 1) | Returns the string " 2.4". |

Note that in the last example, the decimal point occupies one column of the string's width, and the number is rounded to the nearest tenth. Thus, the following code segment would output the first 10 powers of 2 in right-justified format:

int expo, data;

expo = 1;

while (expo <= 10){

data = Math.pow (2, expo);

output.append (Format.justify ('r', data, 4) + "\n");  
    expo++

}