WRITING YOU OWN JAVA METHODS

A method is simply a chunk of code that does a particular job. But methods are set out in a certain way. You have a method header, and a method body. The header is where you tell Java what value type, if any, the method will return (an int value, a double value, a string value, etc.). As well as the return type, you need a name for your method, which also goes in header. You can pass values over to your methods, and these go between a pair of round brackets. The method body is where your code goes.

```
int total (int aNumber) {
      int a_Value= aNumber +10;
      return a_Value;
}
```

The method's return type goes first, which is an `int` type in the code above. After the method type, you need a space followed by the name of your method. We've called the one above "`total`." In between a pair of round brackets we've told Java that we will be

handing the method a variable called `aNumber`, and that it will be an integer.

To separate this method from any other code, you need a pair of curly brackets. Your code for the method goes between the curly brackets. Note the word "`return`" in the method after your code has been executed. It must be of the same type as the return type in the method header. Thus, the return value can't be a string if you started the method with `int total`.

Sometimes you don't want Java to return anything at all. Think of Trim method. You may only want your method to do a job and not return any value. In such case, we use the keyword "void" to set up such kind of method. For example:

```
void print_text (String someText) {
     System.out.println("Some Text Here");
}
```

All the method above does is to print some text. It can just get on with its job, so we've set it as a void method. There's no return value.

Methods don't need to have values passed to them all the time. You can just execute some code. For example:

```
void print_text () {
     System.out.println (“Some text here);
}
```