# 9.2 The Yes-No Example

Recall that we've seen some code that repeatedly asks a user to enter a response until they say yes or no. You can imagine that you could need to do that a lot in a program, meaning it's an ideal choice for a method:

```
static String yesOrNo() {
     Scanner in = new Scanner(System.in);
     String inValue;

     do {
          System.out.print("Enter yes or no: ");
          inValue = in.nextLine();
     } while(!inValue.equalsIgnoreCase("yes") &&
          !inValue.equalsIgnoreCase("no"));
     return inValue;
}
```

Notice this one – as well as being much more complex than `square` – has *no* parameters. This is OK – *but we still need to put the brackets* in even though there's nothing between them. The way we call this is, for example:

```
String result = YesOrNo();
```

or maybe:

```
do {
     //some stuff
} while(YesOrNo().equals("yes"));
```

Notice that our principle of D.R.Y. – Don't Repeat Yourself – is being violated here and we should have constants (final variables) for 'yes' and 'no'. In fact a better version might be one that uses parameters for the value for 'yes' and 'no':

```
static String yesOrNo(String stopVal, String contVal) {
     Scanner in = new Scanner(System.in);
     String inValue;

     do {
          System.out.print("Enter " + stopVal + " or " +
                contVal + ": ");
          inValue = in.nextLine();
     } while(!inValue.equalsIgnoreCase(contVal) &&
          !inValue.equalsIgnoreCase(stopVal));
     return inValue;
}
```

and then do something like this when we call it:

```
final String CONT_VAL = "yes"; //or "ja", or "ναί", or...
final String STOP_VAL = "no";  //or "nein", or "όχι", or...
do {
     //some stuff
} while(YesOrNo(STOP_VAL, CONT_VAL).equalsCONT_VAL));
```

which means we can change languages just by changing the value of the two `final` variables. (Of course to really make sense we should also print out the messages in the same language - that's left as an exercise.)

Notice also that this version has *two* arguments – and they are written like this:

`typeName argumentName, typeName argumentName,…`

the name of the type, then the name of the argument, then a “,” - for as many arguments (of different types, or the same types) as you need.

# 9.2.1 User Experience...

We **mentioned earlier (https://canvas.swansea.ac.uk/courses/44525/pages/7-dot-1-scanners-and-reading-data)** that code like this isn't a good *User Experience (UX)* because it doesn't give users an opportunity to 'back up' and change their mind. They *have* to type yes or no - they can't go "sorry, didn't mean that". The obvious and very simple way to do something about this is to add a third option - in this case, let them type "back":

```
    static String yesOrNo() {
        Scanner in = new Scanner(System.in);
        String inValue;

        do {
            System.out.print("Enter yes, no or back: ");
            inValue = in.nextLine();
            System.out.println(inValue);
        } while (!inValue.equalsIgnoreCase("yes") &&
                !inValue.equalsIgnoreCase("no") &&
                !inValue.equalsIgnoreCase("back"));
        return inValue;
    }
```

and of course it doesn't have to be 'back' - it could be, say, a blank line instead. You might reasonably say that this doesn't actually solve the problem - we still need to actually do something with the user's choice. We'll get back to that at the **end of the course (https://canvas.swansea.ac.uk/courses/44525/pages/12-dot-4-a-menu-class-for-bank)** .