

Version 1: Print messages and pause after each, asking the user if they are ready to continue

```
class MethodDemoV1
{
    public static void main (String[] args)
    {
        char ready = 'n';

        // placeholder for a much larger section of code
        System.out.println("Welcome to the program");

        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getChar();
        } while (ready != 'y' && ready != 'Y');

        // placeholder for a much larger section of code
        System.out.println("The program is still running...");

        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getChar();
        } while (ready != 'y' && ready != 'Y');

        // placeholder for a much larger section of code
        System.out.println("Over yet?  Nope, still running!");

        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getChar();
        } while (ready != 'y' && ready != 'Y');

        // placeholder for a much larger section of code
        System.out.println("Program terminated");
    }
}
```

This program does not do much (printing messages and waiting for the user to indicate they are ready to continue). You might notice, however, that the code that waits for the user is the same in several locations.

Repeated code provides a good opportunity to use methods. A single method can contain the repeated code, and then we call the method whenever we want to run that code.

Version 2: Create a method called `readyToContinue` and move the repeated code there

In version 2 of the code, the **repeated code** has been moved into a method called `readyToContinue`. Whenever we want to use the code in our program, we use the simple command "`readyToContinue()`;" instead.

```
class MethodDemoV2
{
    public static void readyToContinue()
    {
        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getChar();
        } while (ready != 'y' && ready != 'Y');
    }

    public static void main (String[] args)
    {
        char ready = 'n';

        // placeholder for a much larger section of code
        System.out.println("Welcome to the program");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("The program is still running...");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Over yet? Nope, still running!");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Program terminated");
    }
}
```

Unfortunately, this program does not work. The problem is the variable `ready`, which was declared in the `main` method, but is used in the `readyToContinue` method.

A variable must be declared inside the block of code where it is used. In this case, the block of code is the `readyToContinue` method. We will fix this problem in version 3.

Version 3: Move the variable ready into the method readyToContinue

The variable `ready` has been removed from the main method and placed into the `readyToContinue` method instead. This new version will compile and run correctly.

```
class MethodDemoV3
{
    public static void readyToContinue()
    {
        char ready = 'n';

        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getChar();
        } while (ready != 'y' && ready != 'Y');
    }

    public static void main (String[] args)
    {
        // placeholder for a much larger section of code
        System.out.println("Welcome to the program");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("The program is still running...");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Over yet?  Nope, still running!");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Program terminated");
    }
}
```

Another advantage of methods is the ability to change code more easily. If we decided to change the way the user indicates they are ready to continue, we can make our changes in the method `readyToContinue`. In version 1 of our program, we would have to make changes throughout the program. Not only is this time consuming, but there is a greater chance of making a mistake as well.

Version 4: Change the user response from a character (char) to a string (String)

```
class MethodDemoV4
{
    public static void readyToContinue()
    {
        String ready = "no";

        do
        {
            System.out.print("Ready to continue? (Y/N) ");
            ready = In.getString();

        } while (!ready.equals("yes") && !ready.equals("y") &&
                !ready.equals("YES") && !ready.equals("Y") &&
                !ready.equals("ya");
        )
    }

    public static void main (String[] args)
    {
        // placeholder for a much larger section of code
        System.out.println("Welcome to the program");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("The program is still running...");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Over yet? Nope, still running!");

        readyToContinue();

        // placeholder for a much larger section of code
        System.out.println("Program terminated");
    }
}
```