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

```
%%% MAIN PROGRAM BELOW %%%

var ready : string
ready := "n"

put "Welcome to the program"

% check if user is ready to continue
loop
   put "Ready to continue? (Y/N) " ..
   get ready
   exit when (ready = "y") or (ready = "Y")
end loop

put "The program is still running..."

% check if user is ready to continue
loop
   put "Ready to continue? (Y/N) " ..
   get ready
   exit when (ready = "y") or (ready = "Y")
end loop

put "Over yet?  Nope, still running..."

% check if user is ready to continue
loop
   put "Ready to continue? (Y/N) " ..
   get ready
   exit when (ready = "y") or (ready = "Y")
end loop

put "Program terminated... finally."
```

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 procedure called readyToContinue and move the repeated code there

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

```
procedure readyToContinue()
   % check if user is ready to continue
   loop
      put "Ready to continue? (Y/N) " ..
      get ready
      exit when (ready = "y") or (ready = "Y")
   end loop
end readyToContinue

%%% MAIN PROGRAM BELOW %%%

var ready : string
ready := "n"

put "Welcome to the program"
readyToContinue()

put "The program is still running..."
readyToContinue()

put "Over yet?  Nope, still running..."
readyToContinue()

put "Program terminated... finally."
```

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.

```
procedure readyToContinue()

   var ready : string
   ready := "n"

   % check if user is ready to continue
   loop
      put "Ready to continue? (Y/N) " ..
      get ready
      exit when (ready = "y") or (ready = "Y")
   end loop

end readyToContinue

put "Welcome to the program"
readyToContinue()

put "The program is still running..."
readyToContinue()

put "Over yet?  Nope, still running..."
readyToContinue()

put "Program terminated... finally."
```

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.