

# 20

*Flow Control*

## Self-review Questions

- 20.1** Run the programs `While1` and `While2` and determine what the introduced bug is. Why do we get this behaviour? What can we do to fix it (apart from using the original, correctly behaved program)?
- 20.2** The repetition of the Boolean expression in `Do3` can be avoided. How is this possible?
- 20.3** Compare and contrast the four programs, iterative and recursive implementation of factorial numbers and Fibonacci series programs, using the factors of time to implement, execution speed and memory usage.

See Exercise ??

## Programming Exercises

- 20.1** Re-code the program `Break1` without using the `break` statement. Which version of the program do you prefer? Why?
- 20.2** Implement the factorial function without using recursion, i.e. implementing the iteration using iteration statements. Have you tested your factorial implementation? If you have, you will have noticed a serious flaw in the implementation. What is it and why is it there?

We came up with the function:

```
public static int factorial ( final int n ) {
    int product = 1 ;
    for ( int i = 2 ; i <= n ; ++i ) { product *= i ; }
    return product ;
}
```

Simply outputting the first 20 values, highlights the problem:

```
0: 1
1: 1
2: 2
3: 6
4: 24
5: 120
6: 720
7: 5040
8: 40320
9: 362880
```

```
10: 3628800
11: 39916800
12: 479001600
13: 1932053504
14: 1278945280
15: 2004310016
16: 2004189184
17: -288522240
18: -898433024
19: 109641728
```

`factorial ( 14 )` is smaller than `factorial ( 13 )` according to this function, but that is clearly wrong. The problem is that an integer is a 32-bit binary quantity and the value of `factorial(14)` is too large to be represented in an `int`. So overflow occurs and no expression evaluation is correct. The factorial function creates very large values very rapidly, the only way of working with numbers such as these is to use the `BigDecimal` class for expression evaluation.

**20.3** *Implement the Fibonacci function using recursion rather than implementing the iteration using iteration statements.*

An example recursive method is:

```
public static int fibonacci ( final int n ) {
    if ( n < 2 ) { return 1 ; }
    else { return fibonacci ( n - 1 ) + fibonacci ( n - 2 ) ; }
}
```

For an explanation of how this method was written see Exercise ??