

**1. basic while loop**

This conditional loop is useful when you don't know exactly how many times a loop is going to have to repeat.

```
// establish condition that will have value of true
// or false, and then...

while ( <boolean condition> )
{
    // do these
    // statements
}
```

**2. while loop as a counter**

The while-loop *can* be used as a simple counting loop, and works perfectly well that way. More often, a for-loop is used for that purpose, however.

```
int i = 0;
while (i < finalValue)
{
    // statements
    i++;
}
```

**3. basic for (counting) loop**

When you know how many times a loop is going to repeat, a for-loop is usually the best choice.

```
for (int i = 0; i < finalValue; i++)
{
    // do these
    // statements
}
```

**4. Sentinel loop looking for a signal to end the loop**

A "sentinel value" signifies the end of the looping.

```
System.out.print("Enter a value, or 'Q' to quit: ");
String input = in.next();
while (!input.equalsIgnoreCase("q"))
{
    double x = Double.parseDouble(input); // converts input to double
    // do something with the value in x
    System.out.print("Enter a value, or 'Q' to quit: ");
    String input = in.next();
}
```

**5. Error checking loop (using a break statement to exit the loop body)**

This kind of loop is usually considered bad form.

```
while(true) // infinite loop unless we break out of it!
{
    System.out.println("Enter a number greater than 0: ");
    double input = in.nextDouble();
    if (input > 0) break;
}
```

## 6. Nested loops (using **for** as an example)

```
for (int row = 0; row < height; row++)
{
    for (int col = 0; col < width; col++)
    {
        // do something with
        // data at data[row][col]
    }
}
```

### **TASKS**

Typical things you might be asked to do include:

1. Identify the differences between **while** loops and **for** loops, and when each type of loop might be most appropriate.
2. Know what an off-by-one error is, and give examples.
3. Know how to use nested **for** loops.
4. Know the different ways that a loop can be ended: a condition being met, a **break**, a **return**...
5. Write a loop that counts things (like vowels, odd numbers, etc.).
6. Write a loop that sums things (like values entered).

### **EXERCISES**

1. Write a **while** loop that prints the numbers from 1 to 20, as well as their squares, in this format:  
1 squared = 1  
2 squared = 4  
3 squared = 9  
.  
.  
.
2. Write a **for** loop that counts from 0 to 100 and prints out each number.
3. Write a **while** loop that asks the user to enter a series of positive numbers that will be added. The loop stops accepting input when the user enters a 0. Then print out the sum of those numbers.
4. Write a **for** loop that prints out the numbers 1, 4, 7, 10, 13, ... , 298, 301.
5. Write a **while** loop that prints out the numbers 0, 4, 8, 12, ... , 96, 100.
6. Write an infinite loop that has the user repeatedly enter passwords until he/she enters the correct password, a password of your choosing. Once the password is entered, **break** out of the infinite loop.
7. Write a loop that displays the Fibonacci sequence. The first two numbers in the Fibonacci sequence are 0 and 1. Subsequent numbers are found by adding the previous two numbers, so the sequence begins 0, 1, 1, 2, 3, 5, 8, 13, ...
8. Write a “prime finder” loop that determines whether a given number  $n$  is prime or not. Any integer  $n > 2$  is prime if no number between 2 and  $\sqrt{n}$  (inclusive) evenly divides into  $n$ . The loop should return **true** if  $n$  is prime and **false** if  $n$  is not prime.