Recursion:

Recursion is a programming technique in which a method calls itself in order to complete a problem. A method that does this is called a recursive method. The purpose of a recursive method is to create a simpler way to solve a certain problem. Here is an example of a recursive method that sums the individual digits of a number, for example 1263, would return 12:

```
public int sumDigits(int n) {
  if (n<10)
     return n;
  else
     return n%10+sumDigits(n/10);

}
```

A key thing to note about this method is that mod (%) by 10 yields the rightmost digit (1263 % 10 is 3), while divide (/) by 10 removes the rightmost digit (1263 / 10 is 126). Every recursive method needs an end condition, otherwise it will continue to run infinitely. The end condition of this method occurs when the inputted number is less than ten. This case is called the base case. It is the simplest possible case of the recursive definition: the case in which the method does not call itself. All other cases are called recursive cases. Most of the time, recursive methods are comprised of an if-else statements that allows the base case to return one value and any non-base case to recursively call the same method with a smaller set of data. Every recursive method needs to have three things, a base case, a way of getting the problem closer to the base case, and a recursive call that passes that simpler problem back into the method. The "if" part of the statement usually returns the base case. The "else" part of the statement does the job of getting the problem closer to the base case and then recursively calling the method. The following is what it looks like when you trace through the sumDigits() method with the number 1263:

```
sumDigits(1263)
3 + sumDigits(126)
3 + 6 + sumDigits(12)
3 + 6 + 2 + sumDigits(1)
3 + 6 + 2 + 1
12
```

The final return value of the method is 12.

Tail recursion is a type of recursion that occurs when the recursive call is at the very end of the recursive method, and nothing is done after that call. The above example is not tail recursive because although the method is called in the final line of code, it is not called alone.

The biggest problem that can occur with recursive methods, besides running infinitely, is running out of space. If a recursive methods calls itself too many times in a row, it can cause a StackOverflowError to happen. Recursive methods are also, on average, slightly slower than iterative methods. This is usually not that significant of a difference, but it can be problematic. Some advantages to recursive methods are that they are simple to write, often easy to understand, and are usually very short. The decision to use a recursive or an iterative method really depends on the problem that needs to be solved and on the programmer who is trying to solve that problem.

Sources: Horstmann (Chapter 13), http://danzig.jct.ac.il/java_class/recursion.html

Katherine Lee, Nate Bonshire, Max Kelly, Brian Ahmann