Java Recursion

Recursion is where a method calls itself. The method must have some way to get out of constantly calling itself.

A Fibonacci sequence is a set of numbers based on adding the 2 previous numbers

0 1 1 2 3 5 8 13 21 34 55 89

Here is a looping method to work out a number in the Fibonacci sequence:

```
public int fibLoop(int input)
     {//run as loop
          int last = 0;
          int pos1 = 0;
          int pos2 = 1;

          if(input <= 1)
               return 0;
          else
          {
               for (int x = 0 ; x < input ; x ++)
               {
                    last = pos1;
                    pos1=pos2;
                    pos2=last +pos2;
                    System.out.print(pos2 + " : ");
               }
               return pos1;
          }

     }
```

The counter x is never output although it is used to run through the sequence and increment pos2 and last the 2 final numbers that are added.

The same result can be achieved by a recursive method

```
     public int fib( int input )
     {
          if (input == 1 || input == 2)
          {
               return 1; //this is a stopping condition
          }
          else
          {
               return fib(input -1) + fib(input -2);
//recursion
          }
```

```
}
```

A similar series to Fibonacci is the factorial. This is made of numbers multiplied together. Factorial 3 is 3x2x1

```
public int fact( int input )
{
     if (input <= 1 )
     {
          return 1; //this is a stopping condition
     }
     else
     {
          return input * fact(input -1); //recursion
     }
}
```

Here is QuickSort; a sorting algorithm that uses recursion.

```
int partition(int arr[], int left, int right)
     { //used by quickSort
          int i = left, j = right;
          int tmp;
          int pivot = arr[(left + right) / 2];
          while (i <= j) {
               while (arr[i] < pivot)
                    i++;
               while (arr[j] > pivot)
                    j--;
               if (i <= j) {
                    tmp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = tmp;
                    i++;
                    j--;
               }
          };
          return i;
     }

     void quickSort(int arr[], int left, int right) {
          int index = partition(arr, left, right);
          if (left < index - 1)
               quickSort(arr, left, index - 1);
          if (index < right)
               quickSort(arr, index, right);
     }
```

Recursion is also used for searching. A sequential search goes through a set from the beginning to the end looking for the matching item. The recursive binary search breaks up the set into smaller sets and searches through each of these using recursion.

```java
public  boolean binarySearch(int[] a, int b) {
              if (a.length == 0)
                return false;
              int low = 0;
              int high = a.length-1;
              while(low <= high) {
                int middle = (low+high) /2;
                if (b> a[middle]){
                  low = middle +1;
                } else if (b< a[middle]){
                  high = middle -1;
                } else { // The element has been found
                  return true;
                }
              }
              return false;
          }
```