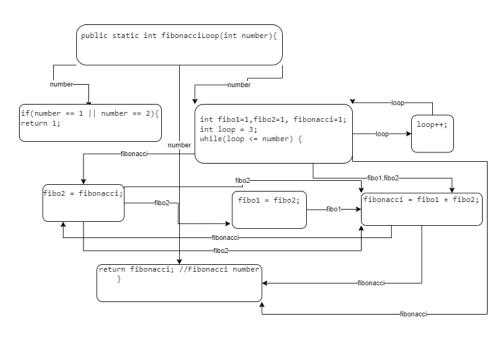
DATA DEPENDENCE GRAPH:

```
public static int indexOf(int[] a, int key) {
    if (a.length == 0)
        return -1;
    if (a.length == 1 && a[0] == key)
        return 0;
    int lo = 0;
    int hi = a.length - 1;
    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        if (key > a[mid])
            lo = mid + 1;
        else if (key < a[mid])
            hi = mid - 1;
        else return mid;
    }
    return -1;
}</pre>
```

```
public static int fibonacciloop(int number){
    if(number == 1 || number == 2){
        return 1;
    }
    int fibo1=1,fibo2=1, fibonacci=1;
    int loop = 3;
    while(loop <= number) {
        fibonacci = fibo1 + fibo2;
        fibo1 = fibo2;
        fibo2 = fibonacci;
        loop++;
    }
    return fibonacci; //Fibonacci number
}</pre>
```



CONTROL DEPENDENCE GRAPH:

```
public static int indexOf(int[] a, int key) {
         if (a.length == 0)
return -1;
          if (a.length == 1 && a[0] == key)
                return 0;
          return 0;
int lo = 0;
int hi = a.length - 1;
while (lo <= hi) {
   int mid = lo + (hi - lo) / 2;
   if (key > a[mid])
                                                                              public static int indexOf(int[] a, int key) {
                     lo = mid + 1;
               else if (key < a[mid])
hi = mid - 1;
               else return mid;
          return -1;
                                                                                                                              return -1;
                                                                if (a.length == 0)
}
                                                             if (a.length == 1 && a[0] == key)
                   return 0;
                                                             int lo = 0;
int hi = a.length - 1;
                                                                  while (lo <= hi) {
                                                                                                                          return -1;
                                                                  int mid = lo + (hi - lo) / 2;
                                if (key > a[mid])
                                                                                                lo = mid + 1;
       else if (key < a[mid])
                                                                hi = mid - 1;
public static int fibonacciLoop(int number){
                                                                            public static int fibonacciLoop(int number){
         if(number == 1 || number == 2){
              return 1;
                                                                            if(number == 1 \mid \mid number == 2)\{
          int fibo1=1,fibo2=1, fibonacci=1;
         int loop = 3;
         while(loop <= number) {
   fibonacci = fibo1 + fibo2;</pre>
              fibo1 = fibo2;
fibo2 = fibonacci;
                                                                                                              int fibo1=1,fibo2=1, fibonacci=1;
   int loop = 3;
   while(loop <= number) {</pre>
                                                                      return1:
              loop++;
                                                                        }
          return fibonacci; //Fibonacci number
                                                                             fibonacci = fibo1 + fibo2;
                                                                                            fibo1 = fibo2;
fibo2 = fibonacci;
                                                                                                                                        return fibonacci; //Fibonacci number
                                                                                            loop++;
                                                                                       }
```