

```
1
2  abstract class Fib {
3      /*
4       * produces the n'th fibonacci number
5       */
6      public abstract long fib(int n);
7
8      /* for profiling */
9
10     private static long count = 0;
11     public static void resetCount() { count = 0; }
12     public static void incrCount() { count++; }
13     public static long getCount() { return count; }
14 }
15
16
17 /* 1 */
18
19 class FibRec extends Fib {
20     public long fib(int n) {
21         incrCount();
22         if (n <= 0) { return 0; }
23         else if (n == 1) { return 1; }
24         else { return fib(n-1) + fib(n-2); }
25     }
26 }
27
28
29 /* 2 */
30
31 class FibMemo extends Fib {
32     private long M[] = {};
33
34     public long fib(int n) {
35         if (M.length <= n) {
36             M = new long[n+1];
37         }
38         return memoFib(n);
39     }
40
41     // memoized
42     private long memoFib(int n) {
43         incrCount();
44         if (n <= 0) { return 0; }
45         else if (n == 1) { return 1; }
46         else {
47             if (M[n] == 0) { // hasn't been filled in
48                 M[n] = memoFib(n-1) + memoFib(n-2);
49             }
50             return M[n];
51         }
52     }
53 }
```

```
54
55
56  /* 3 */
57
58  class FibDP extends Fib {
59      private long M[] = {};
60
61      public long fib(int n) {
62          if (M.length <= n) {
63              M = new long[n+1];
64
65              M[0] = 0;
66              M[1] = 1;
67              for (int i = 2; i <= n; i++) {
68                  M[i] = M[i-1] + M[i-2];
69              }
70          }
71
72          return M[n];
73      }
74  }
75
76
77  /* 4 */
78
79  class FibIter extends Fib {
80      public long fib(int n) {
81          long M2 = 0;    // i-2
82          long M1 = 1;    // i-1
83
84          for (int i = 0; i < n; i++) {
85              long sum = M1 + M2;
86              M2 = M1;
87              M1 = sum;
88          }
89
90          return M2;
91      }
92  }
93
```