Euclidean Algorithm

The Euclidean Algorithm is an ancient algorithm used to find the greatest common divisor (gcd) of two integers. The idea is straightforward and fairly simple. First, you must accept that it is easier to find common divisors of small numbers than of large numbers. Next, observe that if $d$ is a divisor of two integers $a$ and $b$ (with $a > b$), and the division algorithm allows us to write $a = bq + r$, then $d$ must also be a divisor of $r$. It follows that every common divisor of the numbers $a$ and $b$ is also a common divisor of $b$ and $r$. Since $b < a$ and $r < b$, we have reduced the problem of finding common divisors of $a$ and $b$ to finding the common divisors of two smaller numbers $b$ and $r$. We repeat the process until the greatest common divisor is obvious! (This happens when the smaller of the two is zero, since every number divides zero.)

Rather than write an entire method to apply the Euclidean algorithm, I’ll provide an outline and let you fill in the details. (Fill in the three blanks below!)

```java
public class Euclid {
    public static long euclid(long a, long b){
        for( ; b > 0 ; ){ // Continue while b is not zero.
            System.out.println(a + " " + b);
            long r = ____ ; // Compute the remainder.
            a = ______ ; // What is the new ‘a’?
            b = ______ ; // What is the new ‘b’?
        }
        return(a); // If we get here, b must be zero.
    }

    public static void main(String[] args){
        System.out.println(euclid(Long.parseLong(args[0]),
                                Long.parseLong(args[1])));
    }
}
```

Try the following experiments.

1. Type in and complete the Euclidean Algorithm program above. Does it work?
2. Can you change it to compute the least common multiple of two integers?

3. Modify the program to compute the number of steps it takes to compute the greatest common divisor of two integers. Can you find any pairs of integers that take a lot of steps?