Euclid.java

From Robert Sedgewick’s notes at Princeton.

/*************************************************************************
  * Compilation:  javac Euclid.java
  * Execution:    java Euclid p q
  *
  * Reads two commandline parameters p and q and computes the greatest
  * common divisor of p and q using Euclid's algorithm.
  *
  *************************************************************************/

public class Euclid {
    public static int gcd(int p, int q) {
        if (q == 0) return p;
        else return gcd(q, p % q);
    }

    public static void main(String[] args) {
        int p = Integer.parseInt(args[0]);
        int q = Integer.parseInt(args[1]);
        int d = gcd(p, q);
        System.out.println("gcd(" + p + ", " + q + ") = " + d);
    }
}
public class IterativeEuclid {

    public static int gcd(int a, int b) {
        while (b > 0) {
            // assert: b > 0, gcd(a,b) does not change in the loop
            int q = a / b;
            int r = a - q * b;
            a = b;
            b = r;
        }
        // the loop exits with (a,b) = (a,0) and a = gcd of original a and b
        return a;
    }

    public static void main(String[] args) {
        int p = Integer.parseInt(args[0]);
        int q = Integer.parseInt(args[1]);
        int d = gcd(p, q);
        System.out.println("gcd(" + p + ", " + q + ") = " + d);
    }
}
BinaryGCD.java

From Wikipedia's entry on the binary GCD algorithm:

The binary GCD algorithm is an algorithm which computes the greatest common divisor of two nonnegative integers. It gains a measure of efficiency over the ancient Euclidean algorithm by replacing divisions and multiplications with shifts, which are cheaper when operating on the binary representation used by modern computers. This is particularly critical on embedded platforms that have no direct processor support for division. While the algorithm was first published in modern times by Josef Stein in 1961, it may have been known in first-century China (Knuth, 1998).

/********************************************************************************
* Compilation:   javac BinaryGCD.java
* Execution:     java BinaryGCD p q
*                
* Reads two commandline parameters p and q and computes the greatest
* common divisor of p and q using the binary gcd algorithm.
* 
* ******************************************************************************/

public class BinaryGCD {

    public static int gcd(int p, int q) {
        if (q == 0) return p;
        if (p == 0) return q;

        // p and q even
        if ((p & 1) == 0 && (q & 1) == 0) return gcd(p >> 1, q >> 1) << 1;

        // p is even, q is odd
        else if ((p & 1) == 0) return gcd(p >> 1, q);

        // p is odd, q is even
        else if ((q & 1) == 0) return gcd(p, q >> 1);

        // p and q odd, p >= q
        else if (p >= q) return gcd((p-q) >> 1, q);

        // p and q odd, p < q
        else return gcd(p, (q-p) >> 1);
    }

    public static void main(String[] args) {
        int p = Integer.parseInt(args[0]);
        int q = Integer.parseInt(args[1]);
        System.out.println("gcd(" + p + ", " + q + ") = " + gcd(p, q));
    }
}
Exercise 1:

Explain in terms of integers and divisors the effect of the following Euclid-like function.

```java
public static boolean gcdlike(int p, int q) {
    if (q == 0) return (p == 1);
    return gcdlike(q, p % q);
}
```

*Answer.* Returns whether \( p \) and \( q \) are relatively prime.

Exercise 2:

Verify that `Euclid.java` is correct.

Exercise 3:

Verify that `IterativeEuclid.java` is correct.

Exercise 4:

Verify that `BinaryGCD.java` is correct.