Java code to compute gcd of non-negative numbers:

```java
public static int gcd(int a, int b) {
    while (a != 0 && b != 0) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return (a > b) ? a : b;
}
```

Let's prove it with KeY-System.
We first need a specification.

**Definition (GCD)**

Let $a$ and $b$ be natural numbers. A number $d$ is the greatest common divisor (GCD) of $a$ and $b$ iff

1. $d \mid a$ and $d \mid b$
2. If $c \mid a$ and $c \mid b$, then $c \mid d$.

$d \mid a$ means $d$ divides $a$.

$d \mid a :\Leftrightarrow \exists q. d \times q = a$
The specification can be converted to JML:
/*@
   @ requires a >= 0 && b >= 0;
   @ ensures \result >= 0;
   @ ensures (\exists int q; \result * q == a) &&
            (\exists int q; \result * q == b) &&
            (\forall int c; \exists int q; c * q == a) && \exists int q; c * q == \result));
@*/
public static int gcd(int a, int b)

So lets start proving ...
Loop-Invariant

What is the loop invariant?

The algorithm changes $a$ and $b$, but the gcd of $a$ and $b$ should stay the same.

In fact the set of common divisors of $a$ and $b$ never changes.
This suggests the following invariant:

$$\forall d. (d | \text{old}(a) \land d | \text{old}(b) \iff d | a \land d | b)$$

In JML this can be specified as:

```jml
/*@ loop_invariant a >= 0 && b >= 0 &&
  @ (\forall int d; true;
  @ (\exists int q; \text{old}(a) == q*d)
  @ && (\exists int q; \text{old}(b) == q*d)
  @ <==> (\exists int q; a == q*d) && (\exists int q; b == q*d)
  @ );
  @ assignable a, b;
  @ decreases a+b;
  @*/
```