for $a, b \in \mathbb{R}$. This type of operator or library method was not available in the standard Java API until recently.[2] The following Java method implements the mod operation according to the definition in Eqn. (F.1):[3]

```
int Mod(int a, int b) {
  if (b == 0)
    return a;
  if (a * b >= 0 || a % b == 0) {   ← error fixed!
    return a - b * (a / b);
  else
    return a - b * (a / b - 1);
}
```

Note that the *remainder* operator `%`, defined as

$$a \,\%\, b \;\equiv\; a - b \cdot \text{truncate}(a/b), \qquad \text{for } b \neq 0, \tag{F.2}$$

is often used in this context, but yields the same results only for *positive* operands $a \geq 0$ and $b > 0$. For example,

$$\begin{array}{rcrcr} 13 & \text{mod} & 4 & = & 1 \\ 13 & \text{mod} & -4 & = & -3 \\ -13 & \text{mod} & 4 & = & 3 \\ -13 & \text{mod} & -4 & = & -1 \end{array} \quad \text{vs.} \quad \begin{array}{rcrcr} 13 & \% & 4 & = & 1 \\ 13 & \% & -4 & = & 1 \\ -13 & \% & 4 & = & -1 \\ -13 & \% & -4 & = & -1 \end{array}$$

F.1.3 Unsigned Byte Data

Most grayscale and indexed images in Java and ImageJ are composed of pixels of type `byte`, and the same holds for the individual components of most color images. A single byte consists of eight bits and can thus represent $2^8 = 256$ different bit patterns or values, usually mapped to the numeric range $0, \ldots, 255$. Unfortunately, Java (unlike C and C++) does *not* provide a suitable "unsigned" 8-bit data type. The primitive Java type `byte` is "signed", using one of its eight bits for the $\pm$ sign, and is intended to hold values in the range $-128, \ldots, +127$.

Java's `byte` data can still be used to represent the values 0 to 255, but conversions must take place to perform proper arithmetic computations. For example, after execution of the statements

```
int  a = 200;
byte b = (byte) p;
```

the variables `a` (32-bit `int`) and `b` (8-bit `byte`) contain the binary patterns

```
a = 00000000000000000000000011001000
b =                         11001000
```

Interpreted as a (signed) `byte` value, with the leftmost bit[4] as the sign bit, the variable `b` has the decimal value -56. Thus after the statement

[2] Starting with Java version 1.8 the mod operation (as defined in Eqn. (F.1)) is implemented by the standard method `Math.floorMod(a, b)`.
[3] The definition in Eqn. (F.1) is not restricted to integer operands.
[4] Java uses the standard "2s-complement" representation, where a sign bit = 1 stands for a negative value.

Burger, Burge: Digital Image Processing – An Algorithmic Introduction Using Java, 2nd edition, © Springer, 2016. www.imagingbook.com