**TABLE 10.6:** *Selecting an arithmetic operator*

| Operator | Type | | | |
|---|---|---|---|---|
| | int | long | float | double |
| + | iadd | ladd | fadd | dadd |
| - | isub | lsub | fsub | dsub |
| / | idiv | ldiv | fdiv | ddiv |
| * | imul | lmul | fmul | dmul |
| % | irem | lrem | frem | drem |
| unary - | ineg | lneg | fneg | dneg |
| & | iand | land | - | - |
| \| | ior | lor | - | - |
| ^ | ixor | lxor | - | - |

that both subexpressions have the same type. If the expressions have different types, then one must be coerced to have the same type as the other.

### 10.8.1 Numeric Coercions

Coercion is the process of converting a value of one type into a value of a different type. Although the two values are completely different to the JVM, they mean something similar to the user. The number `1.0` (the floating-point number 1) is completely different from the number `1` (the integer 1); arithmetic instructions that apply to one of them do not apply to the other. However, it is clear that `1.0` and `1` are corresponding values in the different domains of numbers.

For example, consider this Java expression:

```
1.0 + 1
```

According to *The Java Language Specification,* the result should be the floating-point number `2.0`. The naïve transformation into bytecodes is this:

```
fconst_1            ; Push 1.0
iconst_1            ; Push 1
fadd                ; ERROR! Can't add a float to an int
```

In order to make these two values have the same type, it is necessary to convert one of them to have the same type as the other. The primary goal is to preserve the magnitude of the number, and the secondary goal is to preserve the precision.