Special Topic 5.5

De Morgan's Law

De Morgan's law shows how to simplify expressions in which the not operator (!) is applied to terms joined by the && or || operators.

In the preceding section, we programmed a test to see whether `amount` was between `0` and `1000`. Let's find out whether the opposite is true:

```
if (!(0 < amount && amount < 1000)) . . .
```

This test is a little bit complicated, and you have to think carefully through the logic. "When it is *not* true that `0 < amount` and `amount < 1000` . . . " Huh? It is not true that some people won't be confused by this code.

The computer doesn't care, but humans generally have a hard time comprehending logical conditions with *not* operators applied to *and*/*or* expressions. De Morgan's law, named after the mathematician Augustus de Morgan (1806–1871), can be used to simplify these Boolean expressions. De Morgan's law has two forms: one for the negation of an *and* expression and one for the negation of an *or* expression:

`!(A && B)` is the same as `!A || !B`

`!(A || B)` is the same as `!A && !B`

Pay particular attention to the fact that the *and* and *or* operators are *reversed* by moving the *not* inwards. For example, the negation of "the input is S or the input is M",

```
!(input.equals("S") || input.equals("M"))
```

is "the input is not S *and* the input is not M"

```
!input.equals("S") && !input.equals("M")
```

Let us apply the law to the negation of "the amount is between 0 and 1000":

```
!(0 < amount && amount < 1000)
```

is equivalent to

```
!(0 < amount) || !(amount < 1000)
```

which can be further simplified to

```
0 >= amount || amount >= 1000
```

Note that the opposite of < is >=, not >!