Decisions in Java

Boolean Variables & Operations

NOTICE
You must be as Tall as this Sign to attack the city

```
how tall are you?

if you are as tall as this sign,
    attackAllowed = true

if you are not,
    attackAllowed = false
```

Boolean Variables

A <u>boolean</u> value is either <u>true</u> or <u>false</u>. A boolean variable must contain a boolean value.

Naming of boolean variables is particularly important. It must be clear what is means when the variable is true, and when it is false.

```
boolean doorOpen;
boolean isRaining;
boolean gameOver;
```

Naming Boolean Variables

Good Names	Poor Names
`isRaining`	`weather`
`doorOpen`	`door`
`gameOver`	`game`
`oldEnough`	`age`

Boolean Variables – If/Else

A boolean variable can be used as part of a selection statement (such as the if/else statement).

```
if (age >= 18)
```

could be replaced by

```
boolean canVote = (age >= 18);
if (canVote)
```

Obviously, this isn't always an improvement!

Comparing Values

Relational Operator	Meaning	Example	Result
`==`	is equal to	`5 == 5`	`true`
`!=`	is not equal to	`5 != 6`	`true`
`<`	is less than	`3 < 7`	`true`
`<=`	is less than or equal to	`4 <= 4`	`true`
`>`	is greater than	`3 > 7`	`false`
`>=`	is greater than or equal to	`7 >= 3`	`true`

Complex Boolean Expressions

Boolean can be useful even with simple decisions, but they become more useful with complex decisions.

Recall: A boolean expression is a comparison between two values.

It is possible to combine multiple comparisons into a single expression.

Boolean Operators

- boolean operators allow us to combine multiple conditions into a single statement
- code can be made shorter (more efficient)
- in some ways, these conditions are more like our natural way of thinking

- there are two ways of combining comparisons
  - AND (all conditions must be true)
  - OR (at least one condition must be true)

Boolean Operators – AND

- when using AND, we require that all conditions be true at the same time
- this is the "picky" boolean operator
- for example:
  "I like movies that have action and comedy"
- to a computer, this person only likes movies that include both action and comedy

```
likeMovie = (movie == action) and (movie == comedy)
```

Boolean Operators – OR

- when using OR, we only require that a single condition be true; the others can be anything
- this is the "easy" boolean operator
- for example:
  "I like movies that have action <u>or</u> comedy"
- to a computer, this person likes movies that have action, or comedy, or both

```
likeMovie = (movie == action) or (movie == comedy)
```

Boolean Operators

`p`	`q`	`p && q` (p and q)	`p || q` (p or q)
`true`	`true`	`true`	`true`
`true`	`false`	`false`	`true`
`false`	`true`	`false`	`true`
`false`	`false`	`false`	`false`

Boolean Expressions – OR

Suppose you have programmed a game and want to know when the game is over. The game is over if <u>either</u> of the following conditions are met.

```
numLives <= 0
timeLeft <= 0
```

These could be combined as:

```
gameOver = (numLives <=0) || (timeLeft <= 0)
```

Boolean Expressions – AND

Suppose you have programmed a game and want to know when the game has been won. The game is won if both of the following conditions are met.

```
numLives > 0
levelsDone >= 10
```

These could be combined as:

```
winGame = (numLives > 0) && (levelsDone >= 10)
```

Boolean Operators - NOT

The "not" operator reverses any boolean value. True becomes false, and false becomes true.

`p`	`!p` `(not p)`
`true`	`false`
`false`	`true`