boolean & if Statements

AP Computer Science

Credit: Slides are modified with permission from Barry Wittman at Elizabethtown College

The `boolean` Type

The `boolean` type

- The **`boolean`** type keeps track of whether something is **`true`** or **`false`**
- Declaration of a **`boolean`** variable is like:

```
boolean value;
```

Storage for a boolean

```
boolean value;
```

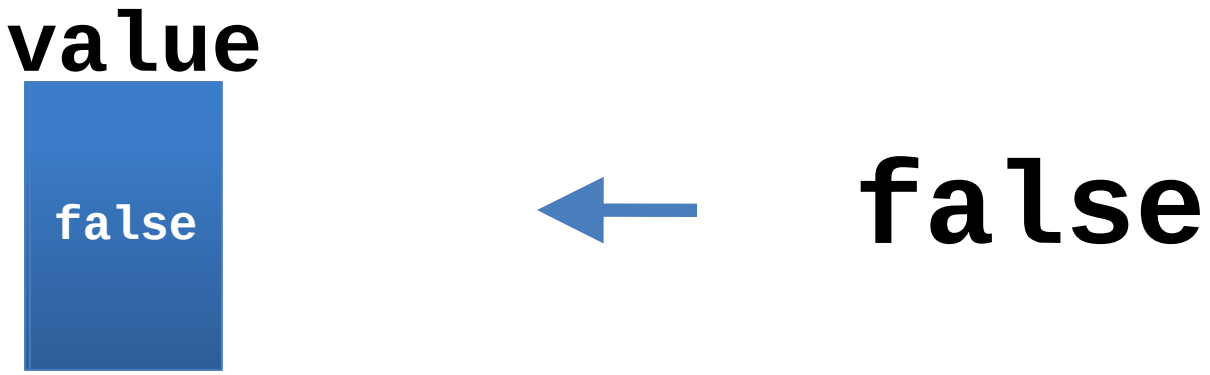
- This line of code creates a box named **`value`** designed only to hold **`boolean`**s
- It cannot be used to store numbers

`value`

Assignment for a boolean

```
value = false;
```

- This line of code **stores `false`** into **`value`**
- Remember the **=** operator is like an arrow pointing left



Operations on `boolean`s

Operations on `boolean`s

- Why would we want to do operations on booleans?
- Like numerical types, we can combine **`boolean`**s in various ways.
- You might be familiar with these operations if you have taken a course in logic.

The ! Operator

- The **NOT** operator
- Changes a **`true`** into a **`false`** or a **`false`** into a **`true`**

x	!x
`true`	`false`
`false`	`true`

Combining `boolean`s

- We can combine statements in logic together to make other interesting statements
- The way we combine them makes a difference, e.g.
  - Politicians lie
    (True)
  - Cast iron sinks
    (True)
  - Politicians lie in cast iron sinks.
    (Absurd)

The && Operator

- The **AND** operator
- It gives back **`true`** only if both things being combined are **`true`**
- If I can swim **AND** the pool is not filled with acid, then I will survive

x	y	x && y
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

The || Operator

- The **OR** operator
- It gives back **`true`** if either or both things being combined are **`true`**
- If I get punched in the face **OR** kicked in the stomach, then I will be in pain

x	y	x || y
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

Quick Check

```
(!true && (false ||(false || true)))
```

- Is this expression **`true`** or **`false`**?
- It is **`false`**

Short circuit evaluation

- In some circumstances, Java does not check the whole expression:
- **`(true ||`** (some complicated expression)**`)`**
  - Ignores everything after **`||`** and gives back **`true`**
- **`(false &&`** (some complicated expression)**`)`**
  - Ignores everything after **`&&`** and gives back **`false`**

Laws of Boolean Algebra

- **Absorption Law**
  - A || (A && B) = A
  - A && (A || B) = A
- **Distributive Law**
  - A && (B || C) = A && B || A && C
  - A || (B && C) = (A || B) && (A || C)

For more rules: http://mathworld.wolfram.com/BooleanAlgebra.html

DeMorgan's Law

- DeMorgan was a British mathematician who showed the importance of several logic rules.
- Two of these:
  - !(A && B) is equivalent to !A || !B
  - !(A || B) is equivalent to !A && !B
- These come in very handy and are often tested on the AP Exam

Precedence of Operators

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* !
multiplicative	* / %
additive	+ -
relational	< > <= >=
equality	== !=
logical AND	&&
logical OR	||
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Conditional Execution

Conditional execution

- So far we have only considered **Java** programs that do one thing after another, in sequence
- Our programs have not had the ability to choose between different possibilities
- Now, they will!

Behold!

- The **`if`**-statement:

```
int x = 4;

if( x < 5 )
    System.out.println("x is small!");
```

- **`x is small!`** will only print out if **`x`** is less than 5
- In this case, we know it is, but **`x`** could come from user input

Anatomy of an `if`

The **`if`** part

Any **`boolean`** expression

```
if( condition )
  statement;
```

Any single executable statement

The idea of an `if`

- It is simply a decision
- A very natural if-then sort of relationship
- If the condition is true, then do something
- For example:
  - If I win a million dollars,
  - Then I will yodel like an insane Swiss monkey

Multiple Statements and Nesting

What if you need to do several things conditionally?

- Use braces to treat a group of statements like a single statement
- I would encourage you to use this style all the time!

```
if( x == 4 )
{
    System.out.println("I dislike 4");
    System.out.println("Let us change
x.");
    x = 10;
}
```

An `if` with multiple statements

```
if( condition )
{
    statement1;
    statement2;
    …
    statementN;
}
```

Multiple statements

Conditions

Conditions in the `if`

- Any statement that evaluates to a **`boolean`** is legal
- Examples:
  - `x == y`
  - `true`
  - `(1 + 2) < 5`
  - `s.equals("Help me!") && (z < 4)`

Comparison

- The most common condition you will find is a comparison between two things
- In **Java**, that comparison can be:
  - **==** equals
  - **!=** does not equal
  - **<** less than
  - **<=** less than or equal to
  - **>** greater than
  - **>=** greater than or equal to

Equals

- You can use the **`==`** operator to compare any two things of the same type
- Different numerical types can be compared as well (**`3 == 3.0`**)
- Be careful with **`double`** types, **`0.33333333`** is not equal to **`0.33333332`**

```
int x = 3;
if( x == 4 )
     System.out.println("Does this print?");
```

Not Equals

- Any place you can use the **==** operator, you can use the **!=** operator
- If **==** gives true, the **!=** operator will always give false, and vice versa
- If you want to negate a condition, you can always use the **!** as a not

```
if( x != 4 )
```

is the same as

```
if( !(x == 4) )
```

= != ==

- Remember, a single equal sign (**=**) is the assignment operator (think of a left-pointing arrow)
- A double equals (**==**) is a comparison operator

```
int y = 10;
if( y = 6 ) //compiler error!
```

```
boolean b = false;
if( b = false ) //no error but confusing
```

Less Than (or Equal To)

- Inequality is very important in programming
- You may want to take an action as long as a value is below a certain threshold
- For example, you might want to keep bidding at an auction until the price is greater than what you can afford

```
if( x <= 4 )
    System.out.println("x is less than 5");
```

- Watch for strict inequality (**<**) vs. non-strict inequality (**<=**)

Greater Than (or Equal To)

- Just like less than or equal to, except the opposite
- Note that the opposite of **<=** is **>** and the opposite of **>=** is **<**
- Thus,
  - **! (x <= y)** is equivalent to **(x > y)**
  - **! (x >= y)** is equivalent to **(x < y)**