# **Boolean Data Type**

15-110 Summer 2010 Margaret Reid-Miller

#### **Boolean values**

- Named after George Boole (1815-1864), who invented mathematical logic and defined Boolean algebra.
- A variable of the primitive data type boolean can have two values: true and false (Boolean literals).
- Boolean variables are used to indicate whether a condition is true or not, or to represent two states, such as a light being on or off.

```
e.g.,
boolean hasLicense;
boolean isDone = false;
boolean isTurnedOn = true;
```

#### **Expressions**

- Up to now we have seen
  - arithmetic expressions that use the operators
     + \* / % ++ --
  - assignment expressions that use the operators
     = += -= ...
- Boolean expressions use relational and logical operators.
- The result of a Boolean expression is either true or false.
- Boolean expressions allow us to write programs that decide whether to execute some code or not.
- These decisions changes the flow of the program execution.

### **Relational operators**

 Relational operators compare two arithmetic expressions and evaluate to a boolean result.

Operator	Name
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Careful: Do not confuse the assignment operator = with the equality operator ==.

#### Relational operators

- The relational operators determine the relative ordering between values.
- The relational operators may be use to compare expressions that evaluate to numeric and char data.
- These relational operators have lower precedence than the arithmetic operators.
  - Thus, arithmetic expressions are evaluated first, then the resulting Boolean expressions.
  - That is, Java does the "math" first, then the comparison.

#### Relational operators

#### **Examples:**

```
int x = 15;
int y = 100;
System.out.println(x > y);
System.out.println(x < 15);
System.out.println(x <= 15)
System.out.println(x == y);
System.out.println(x != 5);
System.out.println(x * -y > 0);
boolean isBigger = x > y;
```

• Logical operators combine boolean values and evaluate to a boolean result.

Operator	Name	Example	Result
1	Logical NOT	!a	true if a is false, false if a is true
&&	Logical AND	a && b	true if both a and b are true, false otherwise
	Logical OR	a    b	true if a or b, or both are true, false otherwise

#### **Truth Tables**

• *Truth tables* list all possible combination of values for the variables in an expression.

a	b	a && b	a    b	!a
true	true	true true		false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

#### Example:

age > 26	hasLicense	(age > 26) && hasLicense

15-110 Summer 2010

boolean canRentCar = (age > 26) && hasLicense;

#### Example:

age > 26	hasLicense	(age > 26) && hasLicense			
true	true	true			
true	false	false			
false	true	false			
false	false	false			

```
int age = 16;
boolean hasLicense = true;
boolean canRentCar = (age > 26) && hasLicense;
```

#### **Examples:**

#### Logical operators: Exercise 1

- It is time to buy a new phone when at least one of the following situations occurs:
  - the phone breaks
  - the phone is at least 3 years old

#### Logical operators: Exercise 1

- It is time to buy a new phone when at least one of the following situations occurs:
  - the phone breaks
  - the phone is at least 3 years old

#### **Logical Operators: Exercise 2**

 Assume x, y, and z are int variables that have been initialized.

```
boolean areAllEqual = _____
```

#### **Logical Operators: Exercise 2**

Assume x, y, and z are int variables that have been initialized.

```
boolean are All Equal = (x == y) \&\& (y == z);
```

#### **Boolean Algebra**

Double negative: !!a = a

de Morgan's Law:

```
!(a \&\& b) \equiv !a || !b
!(a || b) \equiv !a \&\& !b
```

## de Morgan's Law (version 1)

$$!(a \&\& b) == (!a | | !b)$$

а	b	a && b	!(a && b)	!a	!b	!a    !b
Т	Т					
Т	F					
F	Т					
F	F					

## de Morgan's Law (version 1)

! ( &	equal (!a    !b)						
	a	b	a && b	!(a && b)	!a	! b	!a    !b
	Т	Т	Т	F	F	F	F
	Т	F	F	Т	F	Т	Т
	F	Т	F	Т	Т	F	Т
	F	F	F	Т	Т	Т	Т

## de Morgan's Law (version 2)

$$!(a \mid | b) == (!a \&\& !b)$$

a	b	a    b	!(a    b)	!a	!b	!a && !b
Т	Т					
Т	F					
F	Т					
F	F					

### de Morgan's Law (version 2)

! (a	a    b) == (!a && !b)						
	a	b	a    b	!(a    b)	!a	!b	!a && !b
	Т	Т	Т	F	F	F	F
	Т	F	Т	F	F	Т	F
	F	Т	Т	F	Т	F	F
	F	F	F	Т	Т	Т	Т

#### de Morgan's Law

In Java:

```
!((age < 12) | | (age >= 65))
```

In English: It is not the case that age less than 12 or age greater than or equal to 65. !!!?

Simplify using de Morgan's Law:

```
!(age < 12) && !(age >= 65)
```

The reverse the meaning of the relational expressions:

```
(age >= 12) \&\& (age < 65)
```

That is, when age is at least 12 and less than 65.

### de Morgan's Law

#### In English:

Words neither rhyme nor alliterate.

#### In Java:

!wordsRhyme && !wordsAlliterate

Words don't rhyme and they don't alliterate

#### Apply de Morgan's Law:

! (wordsRhyme | | wordsAlliterate)
It's not the case words rhyme or alliterate.