

# Boolean Values, conditions, algebra

Conditions, AND, OR, NOT, if statements



Produced Ms. Mairead Meagher,  
by: Ms. Siobhán Roche.

# Topics list

---

1. Boolean values
2. Boolean conditions and variables
3. Compound conditions / Boolean Algebra
  - i. Logical operators
  - ii. Bringing conditions together

# Boolean values

---

- A **boolean** is a data type that can only have two possible values:
  - true
  - false
- Think of it like a light switch - it's either ON or OFF. There's no in-between!
- Booleans are named after George Boole, a mathematician who developed Boolean algebra.

# Why do we need Booleans?

---

- In programming, we constantly need to make decisions:
  - Is the user old enough to vote?
  - Does the student have enough credits to graduate?
  - Is there enough money in the account for this purchase?
  - Has the password been entered correctly?
- All of these questions have
  - YES/NO or
  - TRUE/FALSE answers –
- that's where booleans come in!

# Topics list

---

1. Boolean values
2. Boolean conditions and variables
3. Compound conditions / Boolean Algebra
  - i. Logical operators
  - ii. Bringing conditions together

# Boolean Conditions

---

- A boolean condition is an expression that evaluates to either
  - true or
  - false.
- Example Boolean Conditions:

```
age >= 18           // Is age greater than or equal to 18?  
temperature < 0    // Is temperature less than 0?  
score == 100        // Is score equal to 100?  
balance != 0        // Is balance not equal to 0?
```

# Java Relational Operators

---

Operator	Use	Returns true if...
>	<code>op1 &gt; op2</code>	op1 is <b>greater than</b> op2
<b>&gt;=</b>	<code>op1 &gt;= op2</code>	op1 is <b>greater than or equal</b> to op2
<	<code>op1 &lt; op2</code>	op1 is <b>less than</b> to op2
<b>&lt;=</b>	<code>op1 &lt;= op2</code>	op1 is <b>less than or equal</b> to op2
<b>==</b>	<code>op1 == op2</code>	op1 and op2 are <b>equal</b>
<b>!=</b>	<code>op1 != op2</code>	op1 and op2 are <b>not equal</b>

**BEWARE** = is an assignment operator.

It doesn't test for equality. Use == to test for equality in primitive types

# Boolean Variables - use in if statement

---

- A boolean variable stores a true or false value.
- They can be used to make your code more readable and lets you reuse conditions.

```
boolean canVote;      // Is age greater than or equal to 18?  
  
canVote = age >= 18;  
  
if (canVote) System.out.println("You can vote!");
```

```
boolean found = false; //checking are vars equal to a lookingFor val  
:    // some code  
if (val == lookingFor) {  
    found = true;  
}
```

# Simple conditions – one condition only

---

```
if (age >= 18 ) { // Is age greater than 18  
    System.out.println("I'm an adult!");  
}
```

```
if (price < 0) { // invalid price  
    System.out.println("Too cheap!");  
}
```

# Topics list

---

1. Boolean values
2. Boolean conditions and variables
3. Compound conditions / Boolean Algebra
  - i. Logical operators
  - ii. Bringing conditions together

# Compound conditions – using Boolean algebra

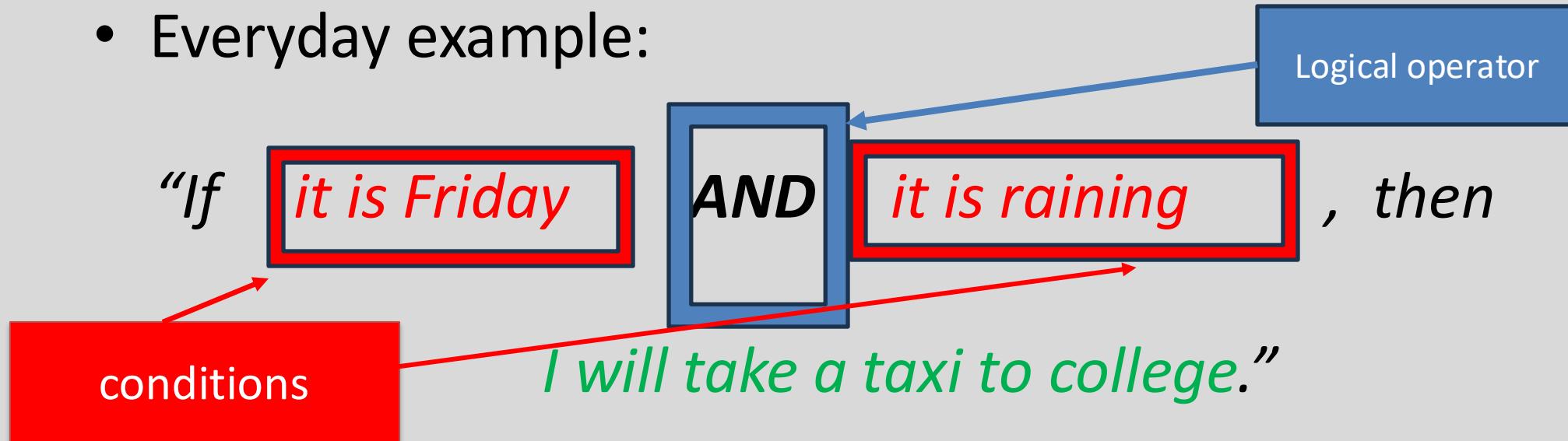
---

- When we use a combination of conditions we call it a compound condition.
- This is done by taking simple compound statements and putting it with another condition with either an
  - and (`&&`) or
  - or(`||`)
  - not (`!`)

# Boolean Conditions with AND (&&)

---

- An AND condition requires **both tests** to be true.
- Everyday example:



*The two conditions are brought together to make one condition.*

# Boolean Conditions with AND (&&)

---

*“If    it is Friday    AND    it is raining    , then  
I will take a taxi to college.”*

Only if both conditions are true will I take a taxi.

I will **NOT** take a taxi if either (or both) of the following is **False**

‘it is Friday’ is false (e.g. it’s Thursday)

‘it is raining’ is false (i.e. it is dry)

# Code example using AND (&&)

---

```
if ( (mark >= 0) && ( mark <= 100 ){  
    System.out.println("This is a valid % mark.");  
}
```

Question : Will the following values of mark lead the condition to be true or false:

mark has value 0 ?

-1 ?

100 ?

101 ?

# Boolean Conditions with OR (||)

---

*“If    it is Friday    OR    it is raining    , then  
I will take a taxi to college.”*

if either or both conditions are true will I take a taxi.

The only way I won't take a taxi is if both of the conditions are false.

# Code example using AND (&&)

---

```
if ( (mark <0 ) || ( mark > 100 ){  
    System.out.println("This is NOT a valid % mark.");  
}
```

Question : Will the following values of mark lead the condition to be true or false:

mark has value 0 ?

-1 ?

100 ?

101 ?

# Boolean Conditions with NOT (!)

---

*“If NOT( it is Friday) , then  
I will take the bike to college.”*

If it is NOT Friday I will  
take the bike to college

NOT True is False

$\text{! True} \rightarrow \text{False}$

NOT False is True

$\text{! False} \rightarrow \text{True}$

# Code example using NOT (!)

---

```
boolean isEligible = false;  
if ( !isEligible ){  
    System.out.println("Sorry you cannot vote.");  
}
```

**Question :** Will the following values of ineligible lead to “Sorry you cannot vote.” being printed:

ineligible has value true ?  
false ?

# Questions?

---

