Conditionals \((if)\)
Overview

- Like humans, programs should be able to make decisions based on conditions.
- In this module, we will learn how to express conditions in a program.
- The program will decide to execute some code if a condition is true and another part if it is false.
- Example:
  - if the light is green *walk* else *stop*
Learning Objectives

- To be able to create and evaluate boolean expressions
- To be able to use if statements to control the flow of a program
- To be able to use if-else statements to control the flow of a program
Boolean expressions

- Boolean expressions evaluate to `true` and `false`
- **Relational operators** (less than, equals to, greater than, etc.) are used in boolean expressions
- Relational operators compare numeric values or arithmetic expressions
- `compareTo()` and `equals()` methods are used to compare String variables
Relational Operators

- Relational operators are **binary operators** (they take two operands)
  
  \[ \text{operand1} \ \text{operator} \ \text{operand2} \]

- \(<\) Less Than
  
  \[ 5 \ < \ 7 \rightarrow \text{true} \]

- \(>\) Greater Than
  
  \[ 6 \ > \ 1 \rightarrow \text{true} \]

- \(\leq\) Less than or equal to
  
  \[ 6 \ \leq \ 1 \rightarrow \text{false} \]

- \(\geq\) Greater than or equal to
  
  \[ 3 \ \geq \ 5 \rightarrow \text{??} \]

- \(==\) Equals
  
  \[ 4 \ == \ 4 \rightarrow \text{??} \]

- \(!=\) Does not equal
  
  \[ 0 \ !\neq \ 10 \rightarrow \text{??} \]
Logical expressions

- Return a boolean value
- Uses **logical operators**
  - **&&** logical conjunction (**and**)
    - both expressions must be true for conjunction to be true
  - **||** logical disjunction (**or**)
    - either expression must be true for disjunction to be true
  - **!** logical negation (**not**)
    - True $\rightarrow$ false,  false $\rightarrow$ true
## Truth table

| P   | Q   | P && Q | P || Q | !P |
|-----|-----|--------|--------|----|
| true| true| true   | true   | false |
| false| true| false  | true   | true |
| true| false| false  | true   | false |
| false| false| false  | false  | true |
Conditionals

- Programs execute one statement after another
- Conditionals allow us to control the flow of a program
- If statement is a flow control structure
If statement

• The if statement
  – Evaluate a boolean expression
  – If true, execute some statements
  – If false, skip those statements
If statement flowchart

Figure 2: The order that statements execute in a conditional
// a single if statement
if (boolean expression) {
    doStatement1;
    doStatement2;
    doStatement3;
    ...;
    doStatementN;
}
Boolean expressions and if statement

- Boolean expressions use relational operators (==, !=, <, >)
- What gets printed when we run the program?

```java
public static void main(String[] args) {
    int age = 16;
    if (age > 17) {
        System.out.println("Eligible to vote");
    }
}
```
Boolean expressions and `if` statement

- **Boolean expressions** use **relational operators** (`==`, `!=`, `<`, `>`) 
- What gets printed when we run the program?

```java
public static void main(String[] args) {
    // note: this spacing works, but don't use it, please!
    if (true)  { System.out.println("Apple"); }
    if (10 > 10)  { System.out.println("Banana"); }
    if (10 >= 10) { System.out.println("Orange"); }
}
```
Compound Boolean Expressions

public static void main(String[] args) {
    int score = 10;
    if (score < 0 || score > 100) {
        System.out.println("Score has an illegal value.");
    }
    if (score >= 0 && score <= 100) {
        System.out.println("Score is in the valid range.")
    }
}
Compound Boolean Expressions

```java
public static void main(String[] args) {
    int score = 10;
    if (score < 0 || score > 100) { // false
        System.out.println("Score has an illegal value. ");
    } // false
    if (score >= 0 && score <= 100) { // true
        System.out.println("Score is in the valid range.");
    }
}
```
If-else statement

- A conditional with two options

The order that statements execute in a conditional with 2 options: if and else
Structure of If-else statement

```java
if (boolean expression) {
    doStatement1;
    doStatement2;
} else {
    doOtherStatement1;
    doOtherStatement2;
}
```
Can you go to see your friend at the park?

boolean isNearby = true;
boolean haveHomework = true;
if(isNearby && !haveHomework) {
    System.out.println("Yes!");
} else {
    System.out.println("No.");
}

What values of these variables cause the program to print “yes?”
Nested if statements

```java
if (boolean expression) {
    if (other boolean expr) {
        do when both true;
    }
    do only when first true;
} else {
    do only when neither true;
}
```

Follow the curly braces to figure out which “if” the “else” belongs to!

(in this case, it’s the first one)
Can you go to see your friend at the park?

boolean isNearby = true;
boolean haveHomework = true;

if(isNearby) {
    if (!haveHomework) {
        System.out.println("Yes!");
    } else {
        System.out.println("No, do HW.");
    }
} else {
    System.out.println("No, too far.");
}
If-else-if statement

- A conditional with three or more options

The order that statements execute in a conditional with 3 options: if, else if, and else
Structure of `if-else-if` statement

```plaintext
if (boolean expression) {
    doSomething;
} else if (other boolean expr) {
    doOtherThing;
} else if (yet another bool expr) {
    doAnotherThing;
} else {
    doDefaultThing;
}
```
Can you go to see your friend at the park?

boolean isNearby = true;
boolean haveHomework = true;
if(!isNearby) {
    System.out.println("no, too far");
} else if (haveHomework) {
    System.out.println("no, do HW");
} else {
    System.out.println("yes, go see them");
}

Same as the previous nested version.
Comparing Strings

- The `equals` method is used to compare two strings letter by letter.
- The operator `==` is used to compare if two variables refer to the same object.
- Two variables that refer to the same object are called *aliases* for the same object.
- Always use `equals()` to compare strings.

```java
String a = new String("hi");
String b = new String("bye");
String c = b;   // c is now an alias for b
```

Figure 1: String aliases
Live Coding: Parking Sign